Descriptive Statistic I

Data Types (Quantitative vs. Categorical)

> Quantitative data takes on numeric values that allow us to perform mathematical operations (like the number of dogs).
>
> Categorical is used to label a group or set of items (like dog breeds - Collies, Labs, Poodles, etc.).

Categorical Ordinal vs. Categorical Nominal

Categorical Ordinal data take on a ranked ordering (like a ranked interaction on a scale from Very Poor to Very Good with the dogs).

Categorical Nominal data do not have an order or ranking (like the breeds of the dog).

Data Types (Continuous vs. Discrete)

Continuous data can be split into smaller and smaller units, and still a smaller unit exists. An example of this is the age of the dog - we can measure the units of the age in years, months, days, hours, seconds, but there are still smaller units that could be associated with the age.

Discrete data only takes on countable values. The number of dogs we interact with is an example of a discrete data type.

Summary

Data Types

> Quantitative:
>
> > Continuous: Height, Age, Income
> >
> > Discrete: Pages in a Book, Trees in Yard, Dogs at a Coffee Shop
>
> Categorical:
>
> > Ordinal:Letter Grade, Survey Rating
> >
> > Nominal: Gender, Marital Status, Breakfast Items

Analyzing Quantitative Data

Four Aspects for Quantitative Data

There are four main aspects to analyzing Quantitative data.

```
1-Measures of Center
2-Measures of Spread
3-The Shape of the data.
4-Outliers
```

Measures of Center

> The Mean : The mean is often called the average or the expected value in mathematics. We calculate the mean by adding all of our values together and dividing by the number of values in our dataset.

> The Median : splits our data so that 50% of our values are lower and 50% are higher
>
> > Median for Odd Values :If we have an odd number of observations, the median is simply the number in the direct middle
> >
> > Median for Even Values :If we have an even number of observations, the median is the average of the two values in the middle

> The Mode :The mode is the most frequently observed value in our dataset.
>
> > No Mode If all observations in our dataset are observed with the same frequency, there is no mode. If we have the dataset: 1, 1, 2, 2, 3, 3, 4, 4 There is no mode because all observations occur the same number of times.
> >
> > Many Modes If two (or more) numbers share the maximum value, then there is more than one mode. If we have the dataset: 1, 2, 3, 3, 3, 4, 5, 6, 6, 6, 7, 8, 9 There are two modes 3 and 6

Notation	English	Example
X	A random variable	Time spent on website
x_1	First observed value of the random variable X	15 mins
$\sum_{i=1}^{n} x_i$	Sum values beginning at the first observation and ending at the last	5 + 2 + ... + 3
$\frac{1}{n}\sum_{i=1}^{n} x_i$	Sum values beginning at the first observation and ending at the last and divide by the number of observations (the mean)	(5 + 2 + 3)/3
$\bar{x}$	Exactly the same as the above - the mean of our data.	(5 + 2 + 3)/3

Descriptive Statistic II

Measures of Spread

Histogram

THE MOST COMMON VISUAL FOR QUANTITATIVE

Calculating the 5 Number Summary

The five-number summary consist of 5 values:

```
1-Minimum: The smallest number in the dataset.
2-Q1: The value such that 25% of the data fall below.
3-Q2: The value such that 50% of the data fall below.(Median)
4-Q3: The value such that 75% of the data fall below.
5-Maximum: The largest value in the dataset.
```

Range

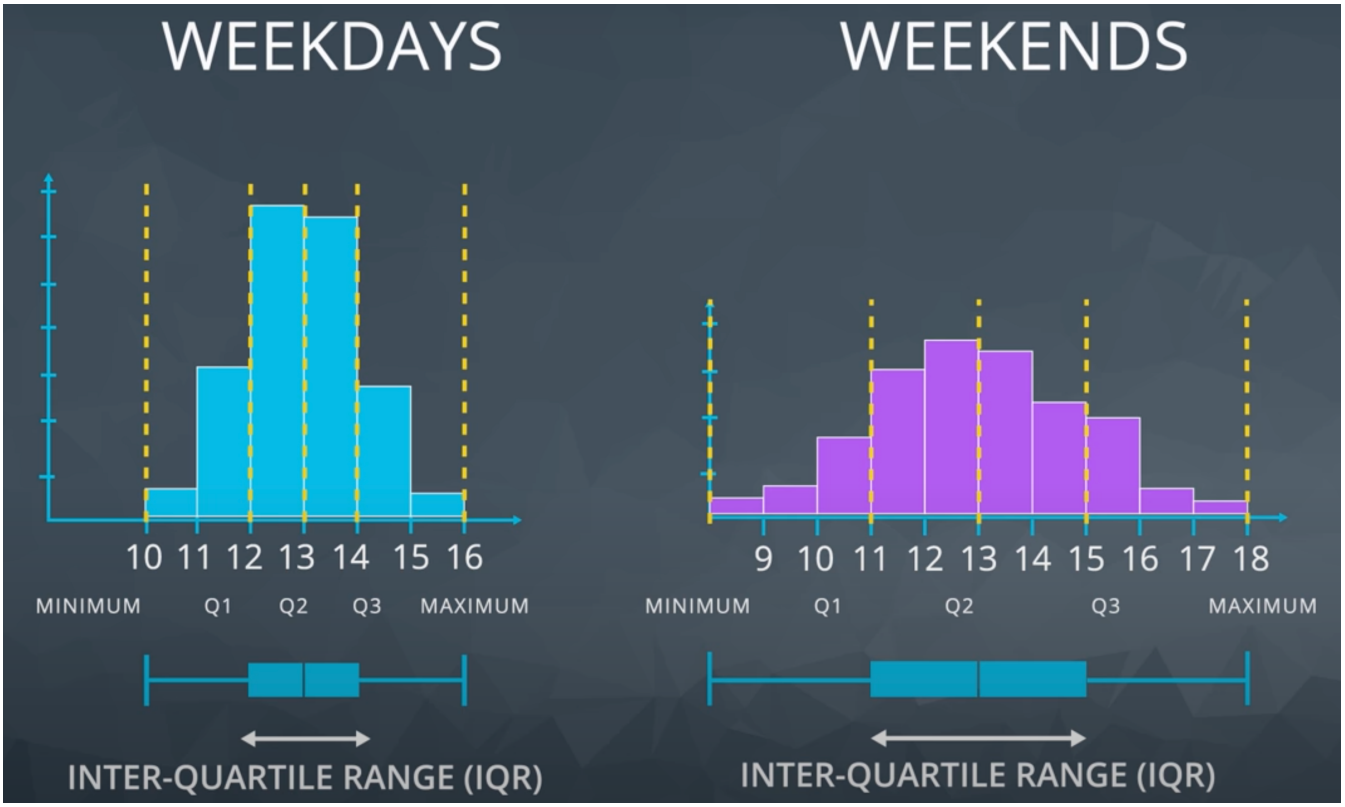
The range is then calculated as the difference between the maximum and the minimum.

IQR

The interquartile range is calculated as the difference between Q3 and Q1.

EXAMPLE:

Find the 5 number summary : 1,2,3,3,5,8,10 Minimum: 1 Q1: 2 Q2: 3 Q3: 8 Maximum: 10 Find the 5 number summary : 1,2,3,3,5,8,10,105 Minimum: 1 Q1: (2+3)/2 = 2.5 Q2: (3+5)/2 = 4 Q3: (8+10)/2 = 9 Maximum: 105 Range: 105 - 1 = 104 IQR: 9 - 2.5 = 6.5 First, you MUST arrange your numbers



Box plots:

are useful for quickly comparing the spread of two data sets across some key metrics, like quartiles, maximum, and minimum.

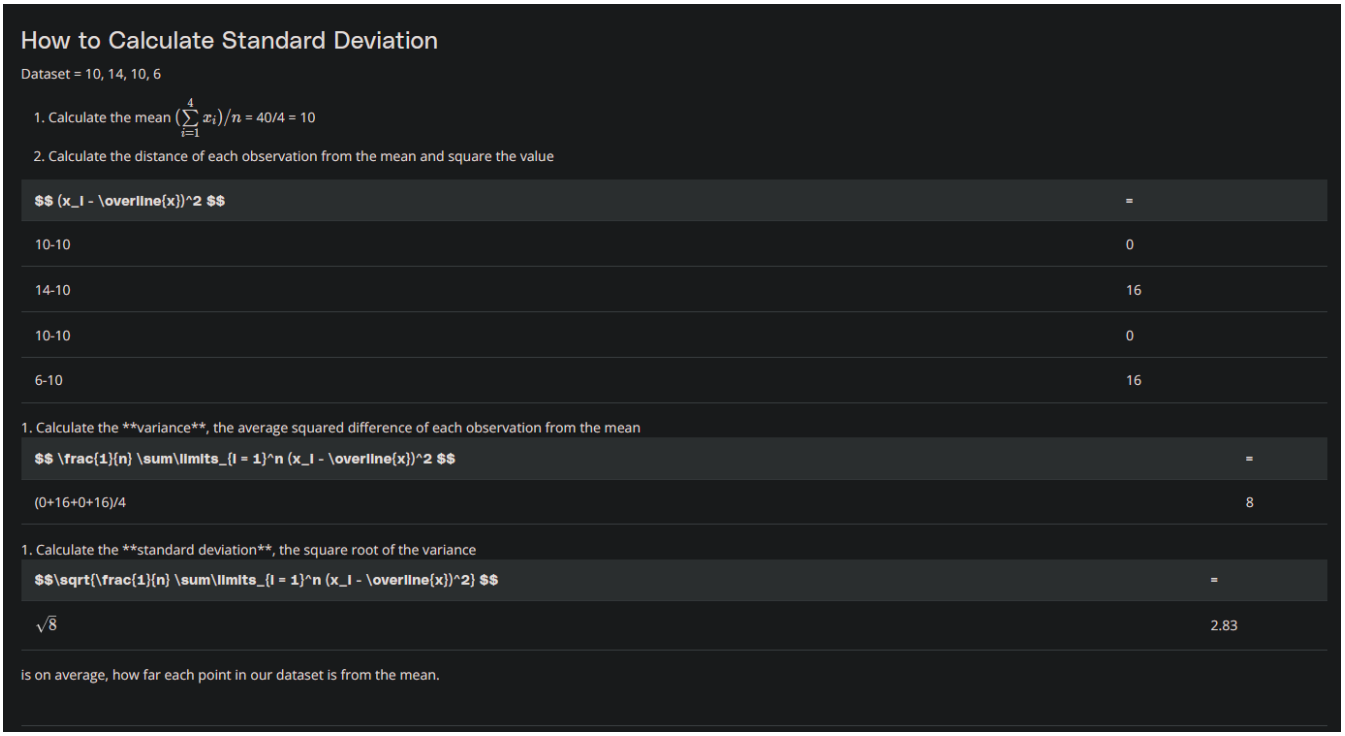
1-The beginning of the line to the left of the box and the end of the line to the right of the box represent the minimum and maximum values in a dataset.

2-The visual distance between these markings is an indication of the range of the values.

3-The box itself represents the IQR. The box begins at the Q1 value, ends at the Q3 value, and Q2, or the median, is represented by a line within the box.

Standard Deviation and Variance

The standard deviation is one of the most common measures for talking about the spread of data. It is defined as the average distance of each observation from the mean.



Standard deviation is a common metric used to compare the spread of two datasets. The benefits of using a single metric instead of the 5 number summary are:

It simplifies the amount of information needed to give a measure of spread

It is useful for inferential statistics

Important Final Points !

1-The variance is used to compare the spread of two different groups. A set of data with higher variance is more spread out than a dataset with lower variance. Be careful though, there might just be an outlier (or outliers) that is increasing the variance when most of the data are actually very close.

2-When comparing the spread between two datasets, the units of each must be the same.

3-When data are related to money or the economy, higher variance (or standard deviation) is associated with higher risk.

4-The standard deviation is used more often in practice than the variance because it shares the units of the original dataset.

Use in the World

The standard deviation is associated with risk in finance, assists in determining the significance of drugs in medical studies, and measures the error of our results for predicting anything from the amount of rainfall we can expect tomorrow to your predicted commute time tomorrow.

Recap

Quantitative Variables

> Measures of Center
>
> Measures of Spread
>
> Shape of the Distribution
>
> Outliers

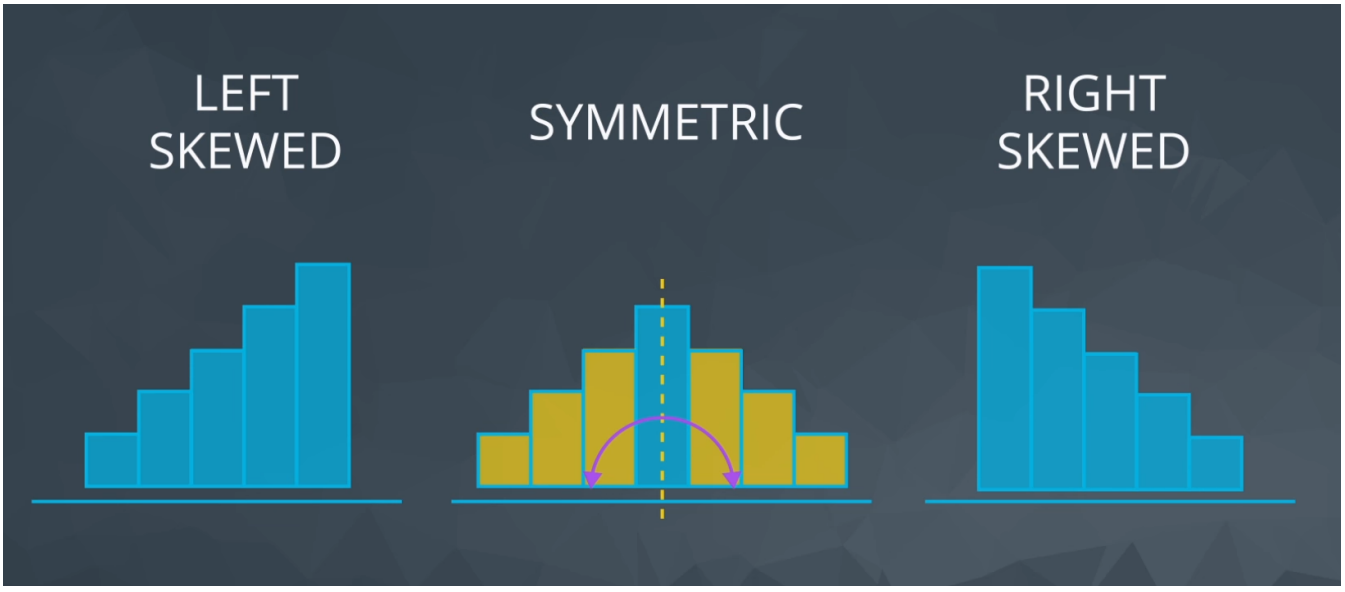
We looked at calculating measures of Center

> Means
>
> Medians
>
> Modes

We also looked at calculating measures of Spread

> Range
>
> Interquartile Range
>
> Standard Deviation
>
> Variance

Shape



Summary

Shape	Mean vs. Median	Real-World Applications
Symmetric (Normal)	Mean equals Median	Height, Weight, Errors, Precipitation
Right-skewed	Mean greater than Median	Amount of drug remaining in a bloodstream, Time between phone calls at a call center, Time until light bulb dies
Left-skewed	Mean less than Median	Grades as a percentage in many universities, Age of death, Asset price changes

Distribution Shape	Types of Data
Bell Shaped	Heights, Weight, Scores
Left Skewed	GPA, Age of Death, Price
Right Skewed	Distribution of Wealth, Athletic Abilities

Outlier

if there an outlier , the BEST meaurment is the median

mean and the standard deviatiom are not good idea

Common Techniques

When outliers are present we should consider the following points.

1. Noting they exist and the impact on summary statistics.
2. If typo - remove or fix
3. Understanding why they exist, and the impact on questions we are trying to answer about our data.
4. Reporting the 5 number summary values is often a better indication than measures like the mean and standard deviation when we have outliers.
5. Be careful in reporting. Know how to ask the right questions.

Outliers Advice

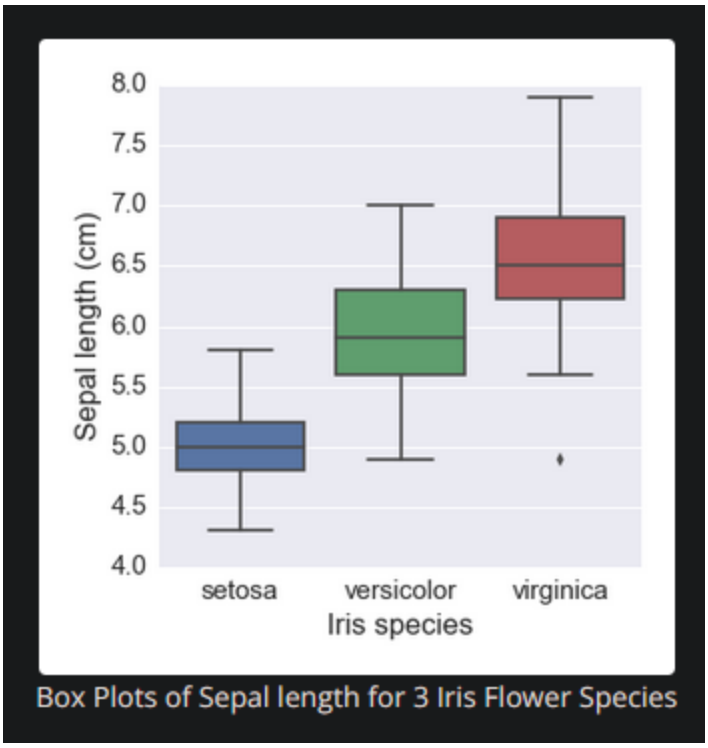
Below are my guidelines for working with any column (random variable) in your dataset.

1. Plot your data to identify if you have outliers.
2. Handle outliers accordingly via the previous methods.
3. If no outliers and your data follow a normal distribution - use the mean and standard deviation to describe your dataset, and report that the data are normally distributed.
4. If you have skewed data or outliers, use the five-number summary to summarize your data and report the outliers.

WE CAN LEARN ALOT WITH THE MEAN AND STANDARD DEVIATION IN NORMALLY DISTRIBUTED DATA

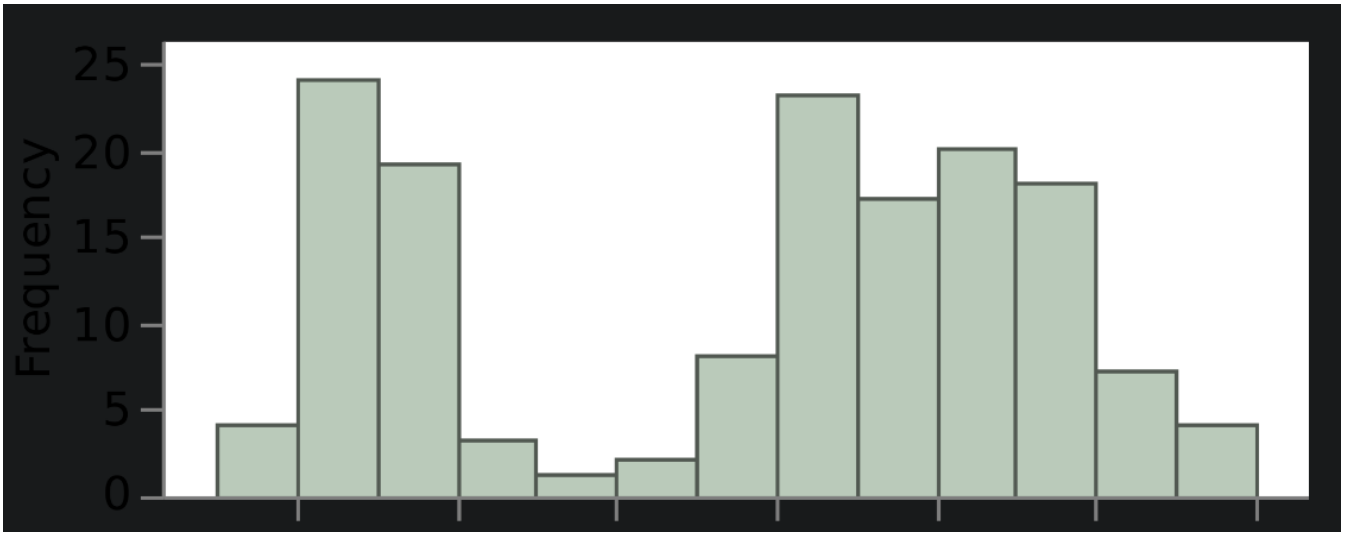
Side note

If you aren't sure if your data are normally distributed, there are plots called (normal quantile plots) and statistical methods like the (Kolmogorov-Smirnov test) that are aimed to help you understand whether or not your data are normally distributed. Implementing this test is beyond the scope of this class, but can be used as a fun fact.

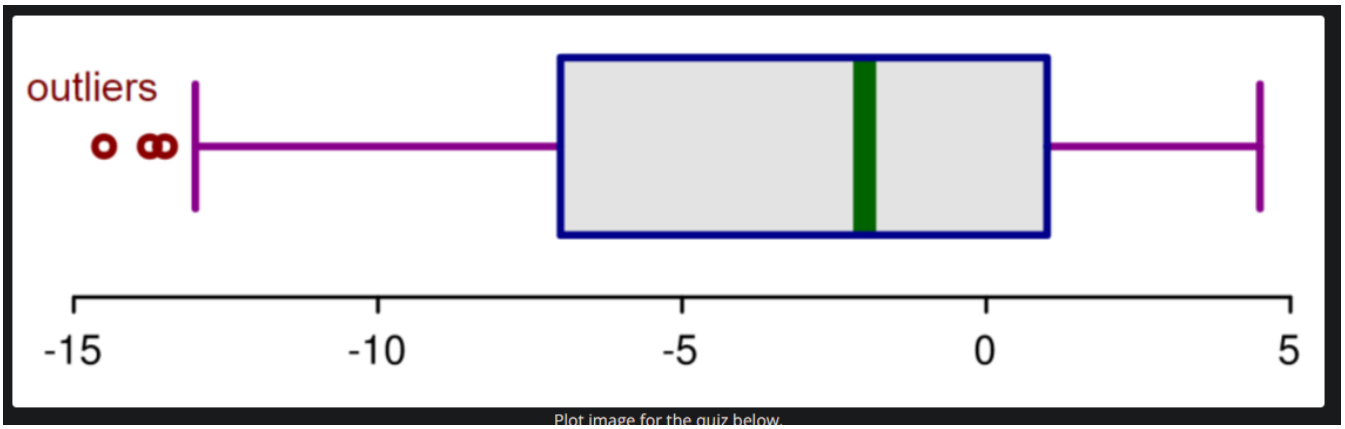


Box Plots of Sepal length for 3 Iris Flower Species

Notice that virginica has an outlier towards the bottom of the plot.



The shape of the above distribution is Bi-modal



The skew is the direction of the longer whisker.So it is left skewed

Descriptive vs. Inferential Statistics

Descriptive Statistics

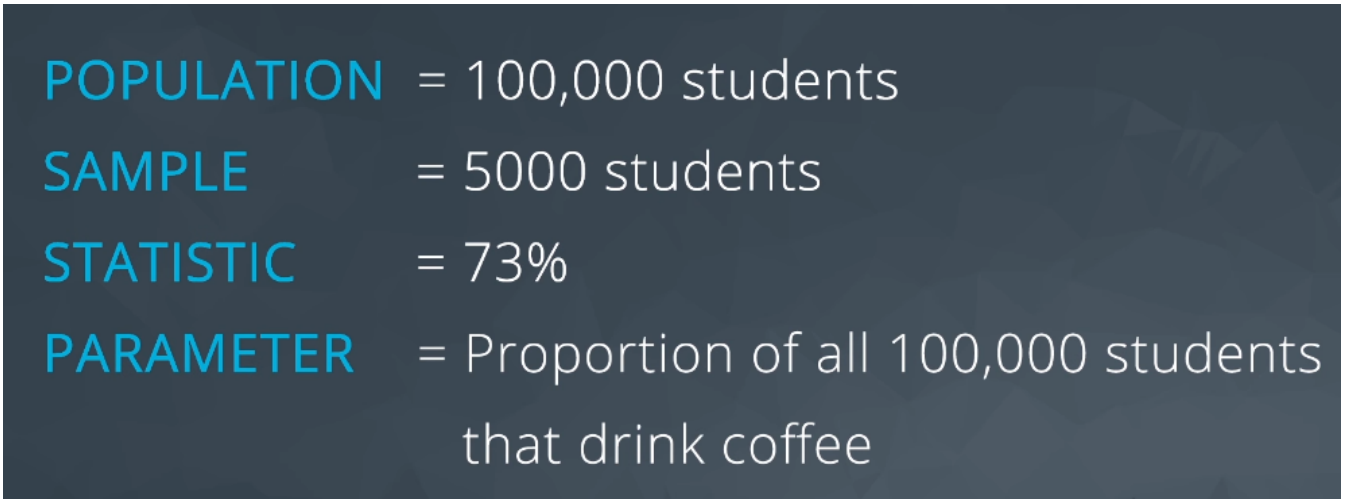
Descriptive statisticsis about describing our collected data.

Inferential Statistics

Inferential Statisticsis about using our collected data to draw conclusions about a larger population.

We looked at specific examples that allowed us to identify the

1. Population - our entire group of interest.
2. Parameter - numeric summary about a population
3. Sample - a subset of the population
4. Statistic numeric summary about a sample



Admissions Case Study

The admissions case study illustrates Simpson's Paradox and how the way you chose to group data can lead to very different results.

how you choose to communicate your data can:

1. Impact people believe to be true
2. Lead to intentional or unintentional false conclusions

Simpson's Paradox

Use admission_data.csv for this exercise.

In [2]:
```
# Load and view first few lines of dataset
import pandas as pd
admits = pd.read_csv('admission_data.csv')
admits.head()
```

Out[2]:

	student_id	gender	major	admitted
0	35377	female	Chemistry	False
1	56105	male	Physics	True
2	31441	female	Chemistry	False
3	51765	male	Physics	True
4	53714	female	Physics	True

Proportion and admission rate for each gender

```
print (len(admits[admits['gender']=='female']))
print (admits.shape[0])
```

```
257
500
```

```
# Proportion of students that are female
(len(admits[admits['gender']=='female']))/admits.shape[0]
```

```
0.514
```

```
# Proportion of students that are male
(len(admits[admits['gender']=='male']))/admits.shape[0]
```

```
0.486
```

```
# Admission rate for females
len(admits[(admits['gender']=='female') & (admits['admitted'])])/(len(admits[admits['gender']=='female']))
```

```
0.28793774319066145
```

```
# Admission rate for males
len(admits[(admits['gender']=='male') & (admits['admitted'])])/(len(admits[admits['gender']=='male']))
```

```
0.48559670781893005
```

Proportion and admission rate for physics majors of each gender

```
# What proportion of female students are majoring in physics?
fem_phys_rate = admits.query("gender == 'female' & major == 'Physics'").count()/ \
    (admits.query("gender == 'female'").count())
print (fem_phys_rate)
```

```
student_id    0.120623
gender        0.120623
major         0.120623
admitted      0.120623
dtype: float64
```

```
# What proportion of male students are majoring in physics?
fem_phys_rate = admits.query("gender == 'male' & major == 'Physics'").count()/ \
    (admits.query("gender == 'male'").count())
print (fem_phys_rate)
```

```
student_id    0.925926
gender        0.925926
major         0.925926
admitted      0.925926
dtype: float64
```

```
# Admission rate for female physics majors
len(admits[(admits["gender"]=='female') & (admits["major"] == 'Physics') & admits["admitted"]]) / len(admits[(admits["gende
r"]=='female') & (admits["major"] == 'Physics')])
```

```
0.7419354838709677
```

```
# Admission rate for male physics majors
len(admits[(admits["gender"]=='male') & (admits["major"] == 'Physics') & admits["admitted"]]) / len(admits[(admits["gende
r"]=='male') & (admits["major"] == 'Physics')])
```

```
0.5155555555555555
```

Proportion and admission rate for chemistry majors of each gender

```
# What proportion of female students are majoring in chemistry?
len(admits[(admits['gender']=='female') & (admits['major'] == 'Chemistry')]) / len(admits[admits['gender']=='female'])
```

```
0.8793774319066148
```

```
# What proportion of male students are majoring in chemistry?
len(admits[(admits['gender']=='male') & (admits['major'] == 'Chemistry')]) / len(admits[admits['gender']=='male'])
```

```
0.07407407407407407
```

```
# Admission rate for female chemistry majors
len(admits[(admits['gender']=='female') & (admits['major'] == 'Chemistry') & admits['admitted']]) / len(admits[(admits['gender']=='female') & (admits['major'] == 'Chemistry')])
```

```
0.22566371681415928
```

```
# Admission rate for male chemistry majors
len(admits[(admits['gender']=='male') & (admits['major'] == 'Chemistry') & admits['admitted']]) / len(admits[(admits['gender']=='male') & (admits['major'] == 'Chemistry')])
```

```
0.1111111111111111
```

Admission rate for each major

```
# Admission rate for physics majors
len(admits[(admits['major'] == 'Physics') & admits['admitted']]) / len(admits[(admits['major'] == 'Physics')])
```

```
0.54296875
```

```
# Admission rate for chemistry majors

len(admits[(admits['major'] == 'Chemistry') & admits['admitted']]) / len(admits[(admits['major'] == 'Chemistry')])
```

```
0.21721311475409835
```

1. it is better to use query
2. The / \ symbol is a line continuation symbol in Python. It is used to split a long line of code into multiple lines for better readability.

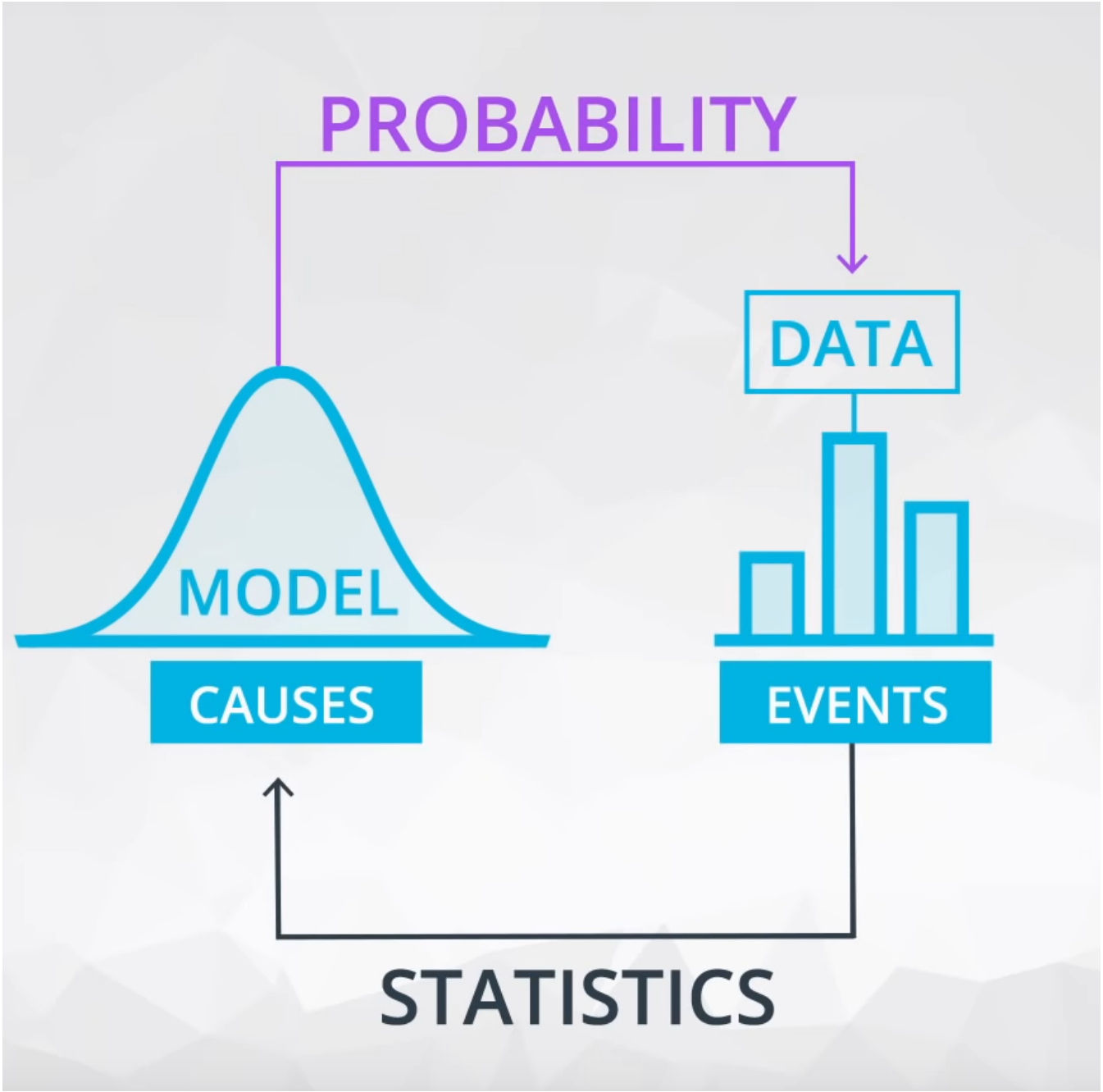
In [ ]:
```
fem_phys_rate = admits.query("gender == 'female' & major == 'Physics'").count() / (admit
# same
fem_phys_rate = admits.query("gender == 'female' & major == 'Physics'").count() / \
                (admits.query("gender == 'female'").count())
```

Introduction to Probability

Probability VS. Statistics

Probability: Make predictions about future events based on models or causes that we assume

Statistics: Analyze the data from past events to infer what the models or causes could be



In this lesson, we will cover and you will be able to:

1. Describe and calculate the probability of one event given another
2. Calculate the opposite of an event
3. Calculate the probability of a composite event

Probability Law

The formula applies to anything that has only two outcomes:

`P(Outcome 1)` + `P(Outcome 2)` will always equal 1.

The probability of `P(A)=1−P(¬A)`.

This is read the probability of A = 1 minus the probability of not A.

Truth Table

A good way to understand this is with a truth table.

Flip 1	Flip 2	Probability of Each Outcome
H	H	0.25
H	T	0.25
T	H	0.25
T	T	0.25

Another way to look at this is

P(H,H) = P(H) * P(H) or the probability of heads times the probability of heads will give you the probability of flipping heads two times in a row.

P(H,H) = 0.5 * 0.5 = 0.25

Loaded Coin Solution

Below is the truth table for this scenario

P(H) = 0.6

Case	Flip 1	Flip 2	Flip 3	Probability
1	H	H	H	0.216
2	H	H	T	0.144
3	H	T	H	0.144
4	H	T	T	0.096
5	T	H	H	0.144
6	T	H	T	0.096
7	T	T	H	0.096
8	T	T	T	0.064

To get each of these you multiply each of the probabilities.

EX:

P(H) = 0.6 P(T) = 0.4

H, H, H would be 0.6 ***** 0.6 ***** 0.6 = 0.216

In this particular case, we only care about the times when one Heads is a possibility which are cases 4, 6, and 7.

Each of those has a probability of 0.096.

If we sum those up we get

0.096 + 0.096 + 0.096 = 0.288

Doubles with Fair Die

Next, if we roll a die twice, what is the probability that we would get doubles or the same number on both rolls.

Solution

P(Double) = 0.1667

Truth Table

Case	Roll 1	Roll 2	Probability
1	1	1	0.0278
2	2	2	0.0278
3	3	3	0.0278
4	4	4	0.0278
5	5	5	0.0278
6	6	6	0.0278

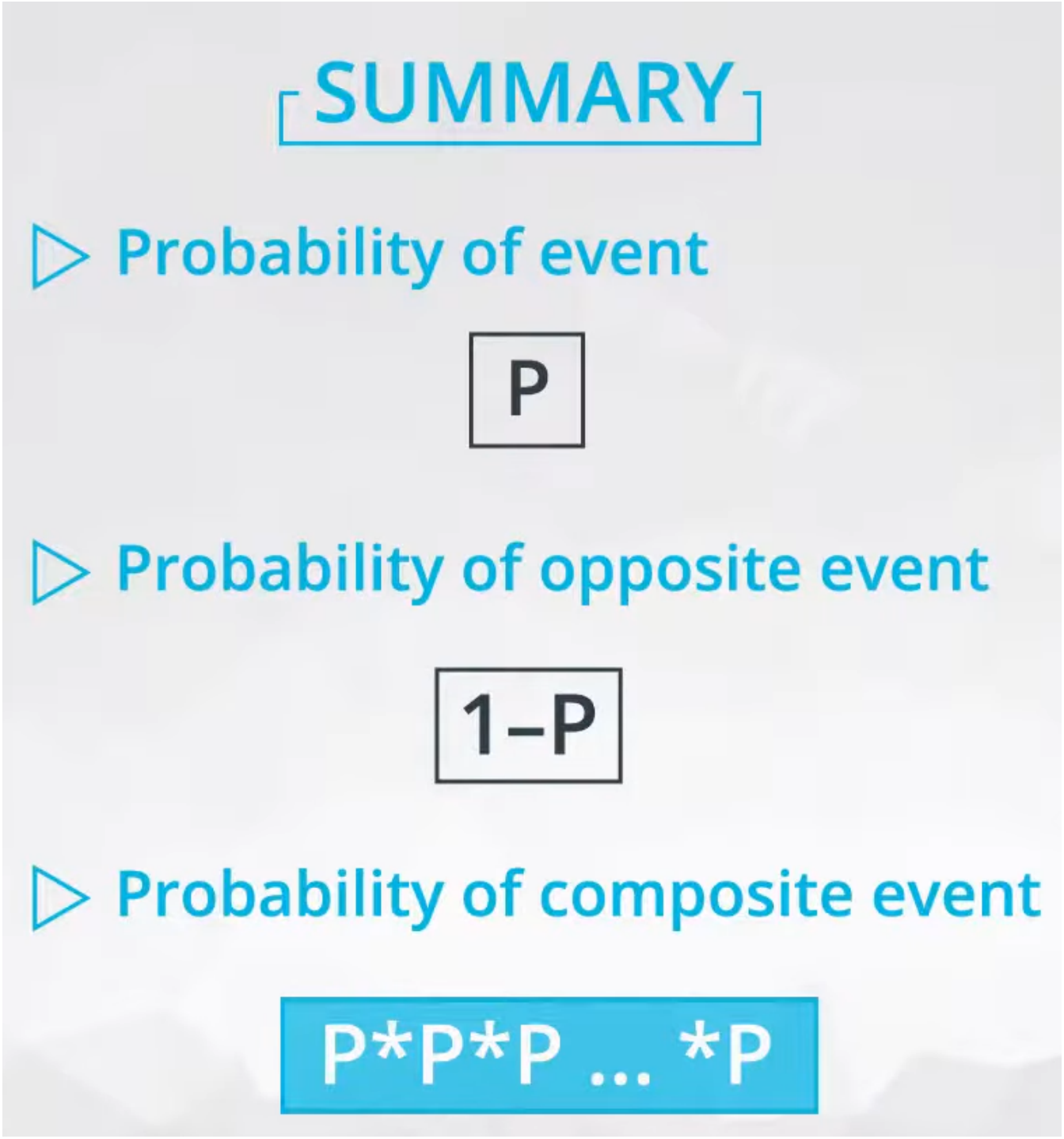
Because we could get a double in each of the six cases, we add them up (or multiple by the number of cases):

0.0278 + 0.0278 + 0.0278 + 0.0278 + 0.0278 + 0.0278 = 0.1667

OR

0.0278 * 6 = 0.1667

The probability of ONE roll is 1/6 which it 0.1667. So, The probability of TWO rolls is 1/6 * 1/6 which it 1/36 = 0.0278



Introduction to Binomials

Binomial Distribution: Determines the probability of a string of independent 'coin flip like events'.

In this lesson, we will cover and you will be able to:

```
Describe and calculate Binomial Distributions
Calculate the probabilities of multiple independent events
```

Truth Table

Case	Coin 1	Coin 2
1	H	H
2	H	T
3	T	H
4	T	T

Quiz Question

In the cases above, given 2 fair coins and one flip how many times could you get Heads exactly once and tails exactly once?

- ○ 1
- ● 2

5 Flips 1 Head

With 5 coin flips, how many outcomes will have exactly 1 Heads and 4 Tails?

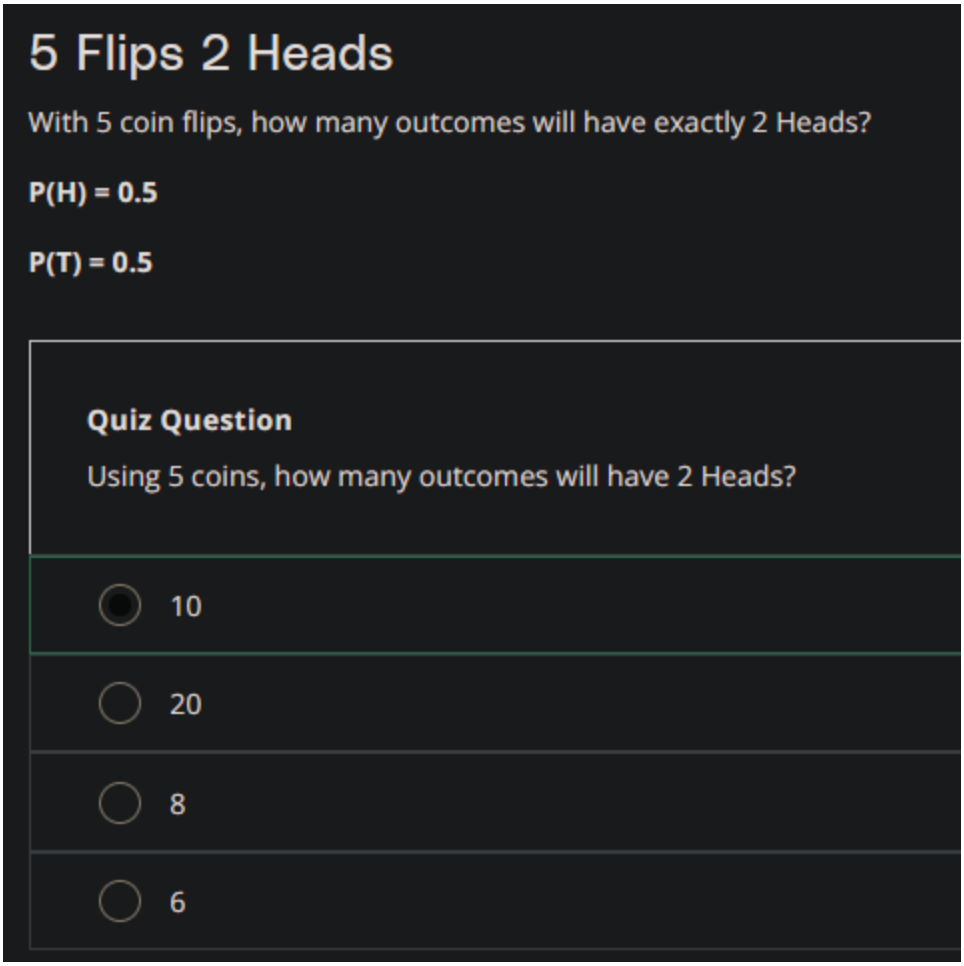
P(H) = 0.5

P(T) = 0.5

Quiz Question

Using 5 fair coins, how many outcomes will have exactly 1 Head and 5 Tails?

- ● 5
- ○ 6
- ○ 3
- ○ 1



The answer is 10. We know from the previous example that the first Head can appear in 5 different positions. This leaves a factor of 4 different ways to position the second Head. To eliminate double counting, the multiple of 5·4 must be divided by 2; thus, 25/2=10.

5 Flips 3 Heads

With 5 coin flips, how many outcomes will have exactly 3 Heads?

P(H) = 0.5

P(T) = 0.5

Quiz Question

Using 5 coins, how many outcomes will have 3 Heads?

- (●) 10
- ( ) 20
- ( ) 8
- ( ) 6

5 Flips 3 Heads

The answer is 10. We know from the previous example that the first Head can appear in 5 different positions. This leaves a factor of 4 different ways to position the second Head and 3 ways to place the third Head.

	# of Positions Available
1st Head	5
2nd Head	4
3rd Head	3

Taking the product of these three numbers, $5 \cdot 4 \cdot 3 = 60$, would result in double counting. To remove duplicates we need to determine how many placement options are remaining if the other Heads are already in place.

	Placement Options	# of Placements Available
1st Head	**H**THHT, **TH**HHT, TTHH**H**	3
2nd Head	HT**H**HT, HTHT**H**	2
3rd Head	HTH**H**T	1

The table above represents the amount of duplication. To remove duplicates, divide the total options 60 by the product of all duplicates, $60/(3 \cdot 2 \cdot 1) = 10$.

10 Flips 5 Heads

The answer is 252.

Step	Calculation	Result
1. Calculate the product of available positions	$10 \cdot 9 \cdot 8 \cdot 7 \cdot 6$ =	30,240
2. Calculate the product of duplication	$5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$ =	120
3. Divide the two results	$\frac{30,240}{120}$	252

Factorial Formula

$$\frac{n!}{k!(n-k)!}$$

where n is = coin flip & k = heads

125 Flips 3 Heads

The answer is 317,750. Evaluating for $\frac{n!}{k!(n-k)!}$

n = 125 coin flips	k = 3 Heads
$\frac{n!}{(n-k)!}$	k!
$\frac{125!}{(125-3)!}$	3!
$125 \cdot 124 \cdot 123$	$3 \cdot 2 \cdot 1$
$1,906,500$	6
$\frac{1,906,500}{6} = 317,750$	

Binomial

`Quiz Question`

What is the probability of having exactly 1 Head in 5 coin flips?

The answer is $\frac{5!}{1!(5-1)!} = 5$. Given the size of the Truth Table, $2^5 = 32$, this means there is a $\frac{5}{32}$, or a 0.15625 chance you will get only 1 Head in 5 flips.

NOTE! : 5 is the number of Heads that would come up. But they are asking for the probability. Divide 5 by the size of your Truth Table to get the correct answer.

BUT , what if the Head is = 0.8 instead of 0.2 ?

3 Flips 1 Head

We have 3 loaded (unfair) coins.

P(H) = 0.8

P(T) = 0.2

Quiz Question

What is the **probability** of having exactly 1 Head in 3 coin flips, using loaded coins above?
Hint: calculate probability using a Truth Table.

Truth Table

Case	Result	Probability of Each	Total Probability
1	HTT	$0.8 \cdot 0.2 \cdot 0.2 = 0.032$	
2	TTH	$0.2 \cdot 0.2 \cdot 0.8 = 0.032$	
3	THT	$0.2 \cdot 0.8 \cdot 0.2 = 0.032$	0.096

5 Flips 4 Heads

We now have 5 loaded (unfair) coins.

P(H) = 0.8

P(T) = 0.2

To **calculate the probability** for the following example:

1. Calculate $\frac{n!}{k!(n-k)!}$
2. Calculate the probability of something occurring by the number above

> **Quiz Question**
>
> What is the **probability** of having exactly 4 Heads in 5 coin flips, using loaded coins above?
> Hint: use the steps above

5 Flips 4 Heads

The answer is 0.4096.

1. $\frac{5!}{4!(5-4)!} = 5$ - This is the number of Truth Table occurrences
2. The probability of each Truth Table occurrence, 4 Heads and 1 Tail, is$0.8 \cdot 0.8 \cdot 0.8 \cdot 0.8 \cdot 0.2 = 0.8^4 \cdot 0.2 = 0.08192$. This probability multiplied by 5 = 0.4096.

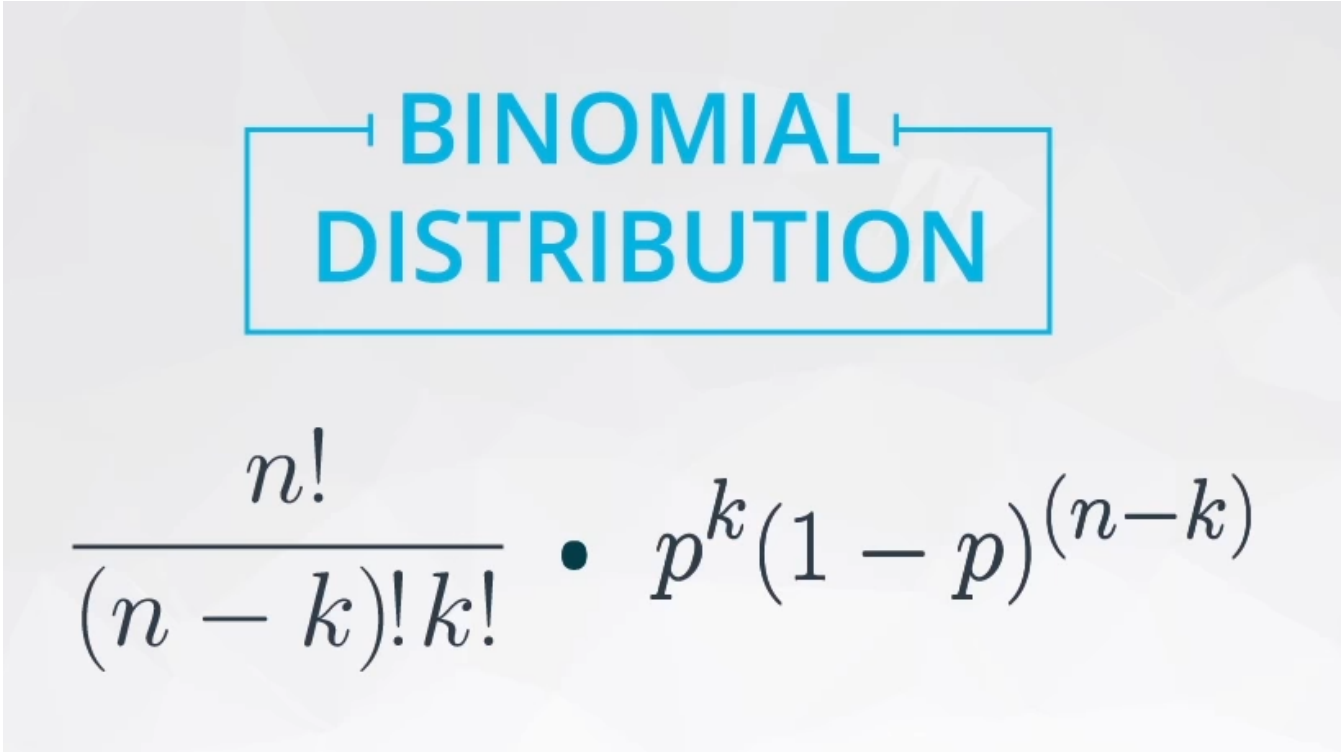
What if 3 head ?

5 Flips 3 Heads

The answer is 0.2048.

1. $\frac{5!}{3!(5-3)!} = 10$ - This is the number of Truth Table occurrences
2. The probability of each Truth Table occurrence, 3 Heads and 2 Tails, is$0.8 \cdot 0.8 \cdot 0.8 \cdot 0.2 \cdot 0.2 = 0.8^3 \cdot 0.2^2 = 0.02048$. This probability multiplied by 10 = 0.2048.

- Number of Ways to get *k* Heads in *n* flips of a coin: $\frac{n!}{k!(n-k)!}$
- Probability of *k* Heads and *(n-k)* Tails: $p^k(1-p)^{(n-k)}$



Introduction to Conditional Probability

Conditional Probability: The probability of the outcome of one event is affected by the outcome of another event.

In this lesson, we will cover and you will be able to:

```
Describe and calculate the Conditional Probabilities
Calculate the probabilities of multiple dependent events
```

Cancer Probability

Understanding the Cancer Example:

- P(C) is the Probability of cancer and is represented as:
  - P_0
- P(Pos|C) is the probability of a positive test for someone who has cancer and is represented as:
  - P_1
- P(Neg|¬C) is the probability of a negative test for someone who does not have Cancer and is represented as:
  - P_2

Case	Probability	Sensitivity	Specificity
P_0	0.1	-	-
P_1	-	0.9	-
P_2	-	-	0.8

What this table represents is that 10% of the general population has cancer. 90% of people **who have cancer**, will **test positive** for cancer. And 80% of people who **don't have cancer**, will **test negative** for cancer.

The question we want to answer is :

- What is the probability of getting a positive test for cancer for *anyone in the general population* given the data above regardless of the fact they have cancer or not?

Quiz Question

What is the probability of someone testing positive for cancer given the information above?

Solution:

The answer was 0.27. We can use P(Pos) to represent the probability of someone testing positive for Cancer.

Using conditional probability theory, we did the following:

$$\mathbf{P(Pos) = P(C) \cdot P(Pos|C) + P(\neg C) + P(Pos|\neg C)}$$

We know that:

$P(\neg C) = 1 - P(C) = 1 - 0.1$

$P(Pos|\neg C) = 1 - P(Neg|\neg C) = 1 - 0.8$

So we can get:

$P(Pos) = 0.1\ 0.9 + (1 - 0.1)\ (1 - 0.8) = 0.27$

Building a Truth Table for Conditional Probability

Table of the Cases from the Video

Case	Probability
P(Cancer)	0.1
P($$\neg$$Cancer)	0.9
P(Positive|Cancer)	0.9
P(Neagative|Cancer)	0.1
P(Positive|$$\neg$$Cancer)	0.2
P(Neagative|$$\neg$$Cancer)	0.8

Aimed with the information from above we can start building out a Truth Table for these cases that looks like this:

Case	Cancer	Test	Probability
1	Y	P	?
2	Y	N	?
3	N	P	?
4	N	N	?

Now we need to figure out the probability of each of the cases.

Quiz Question

What is the probability of Case 1 from the table above where a person has cancer and tests positive for cancer?

Solution:

The answer was 0.09.

To get this we multiplied

0.1 · 0.9 = 0.09

P(Positive) = P(Cancer) · P(Positive|Cancer)

Or the **probability of cancer** 0.1 times the probability of **testing positive and having cancer** 0.9.

Quiz Question

What is the probability of Case 2 from the table above where a person has cancer and tests negative for cancer?

Solution:

The answer was 0.01.

To get this we multiply

0.1 · 0.1 = 0.01

We multiplied **P(Cancer)** 0.1 · **P(Negative|Cancer)** 0.1 to get our answer.

Quiz Question

What is the probability of Case 3 from the table above where a person **does not** have cancer and tests positive for cancer?

Solution:

The answer was 0.18.

To get this we multiply

0.9 · 0.2 = 0.18

We multiplied **P(¬Cancer)** 0.9 · **P(Positive|¬Cancer)** 0.2 to get our answer.

Quiz Question

What is the probability of Case 4 from the table above where a person **does not** have cancer and tests Negative for cancer?

Solution:

The answer was 0.72.

To get this we multiply

0.9 · 0.8 = 0.72

We multiplied **P(¬Cancer) 0.9 · P(Negative|¬Cancer)** 0.8 to get our answer.

Total Probability

Using mathematical notation we can see what our probabilities look like:

$P(C) \sim P(\neg Cancer)$

$P(P|C) \sim P(N|C)$

$P(P|\neg Cancer) \sim P(N|\neg Cancer)$

Total probability can be calculated with the formula:

$P(P) = P(P|C) \cdot P(C) + P(P|\neg Cancer) \cdot P(\neg Cancer)$

Two Coins & Two Flips:

```
`Quiz Question`

   What is the probability of flipping Heads and then Tails or P(H,T) in
the case below
   NOTE: P1(H) = 0.5, P1(T) = 0.1...P2(H) = 0.9, P2(T) = 0.1
```

Case	Pick	Flip	Probability
1	P_1	H	H
2	P_1	H	T
3	P_2	T	H
4	P_2	T	T
1	P_1	H	H
2	P_1	H	T
3	P_2	T	H
4	P_2	T	T

We have case 2 in pick 1 & case 2 in pick 2 Therefore, the chance to choose one of the two picks is 0.5 So

1. The probability of case 2 pick 1 = 0.5(chance to choose one of the two picks) *0.5(P1(H))* 0.5(P1(T)) = 0.125
2. The probability of case 2 pick 2 = 0.5(chance to choose one of the two picks) *0.9(P2(H))* 0.1(P2(T)) = 0.045

Once we know those probabilities, we can sum them to find our answer:

0.125 + 0.045 = 0.17

Summary

Conditional Probability

In the video above, you've learned that the probability of a specific outcome for anyone given a test can be represented as:

$$P(Test) = P(Test|Disease)P(Disease) + P(Test|\neg Disease) \cdot P(\neg Disease)$$

Where Test is a given outcome, can be either positive or negative. P(Disease) is the probability of having the disease and P(¬Disease) is the probability of not having the disease.

In this lesson, you learned about conditional probability. Often events are not independent like with coin flips and dice rolling. Instead, the outcome of one event depends on an earlier event.

For example, the probability of obtaining a positive test result is dependent on whether or not you have a particular condition. If you have a condition, it is more likely that a test result is positive. We can formulate conditional probabilities for any two events in the following way:

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

In other words, the probability of testing positive for a disease given the person has the disease would look like:

$$P(positive|disease) = \frac{P(\text{positive} \cap \text{disease})}{P(disease)}$$

where $|$ represents "given" and $\cap$ represents "and".

Bayes Rule

We will cover and you will be able to:

```
Describe probabilistic inference and Bayes Rule
Normalize to a dataset
Apply probabilistic inference and Bayes Rule in a real-world scenario
```

Cancer Test

Table of the Cases from the Video

Case	Probability	Key Term
P(Cancer)	0.01	
P($$\neg$$Cancer)	0.99	
P(Positive|Cancer)	0.9	**Sensitivity**
P(Negative|$$\neg$$Cancer)	0.9	**Specitivity**

Quiz Question

Using the probability provided above. What is the probability that there is truly cancer if someone has a positive test P(Cancer|Positive)?

- ◯ 8%

This quiz asks the probability that there is truly cancer if someone has a positive test.

Without calculating it mathematically, let's estimate the answer here.

First, we know that only 1% of people have cancer regardless of the test being positive or negative. Imagine someone has already tested positive, the probability of that person having cancer will be higher than 1%. But how much higher?

Essentially, the quiz asks you to solve the probability of P(C|Pos).

From conditional probability theory, we know that

P(C|Pos) = P(C&Pos) / P(Pos) = P(C&Pos) / [P(C&Pos) + P(¬C&Pos)],

which is the ratio of people who have cancer and also test positive to people who test positive whether they have cancer or not. This ratio will be quite small because the majority (99%) of people don't have cancer, which means the number of people who test positive and have cancer (P(C&Pos)) is relatively small compared to the number of people who test positive but don't have cancer (P(¬¬C&Pos)). So the ratio could not be as high as 90%.

Based on the estimation and the 3 choices above, we can say that P(C|Pos) is higher than 1% and below 90%. So only 8% makes sense. But you may ask: how do you know it is 8%? That's a great question and you will see how it is calculated mathematically on the next page.

Prior And Posterior

Bayes Rule

Prior probability x Test Evidence ---> Posterior Probability

posterior is often called joint

Case	Prior Probability	Posterior Probability
P(Cancer)	0.01	
P($$\neg$$Cancer)	0.99	
P(Positive | Cancer)	0.9	
P(Negative | $$\neg$$Cancer)	0.9	
P(Positive | $$\neg$$Cancer)	0.1	
P(Cancer | Positive)	P(Cancer) $$\cdot$$ P(Positive | Cancer)	?
P($$\neg$$Cancer | Positive)	P($$\neg$$Cancer) $$\cdot$$ P(Positive | $$\neg$$Cancer)	?

Quiz Question

What are the posterior probabilities for our sample cancer case, based on the data above?

These are the correct matches.

Case	Probability
P(¬Cancer | Positive)	0.099
P(Cancer | Positive)	0.009

P(¬Cancer|Positive) = P(¬Cancer) x P(Positive|¬Cancer)

P(Positive|Cancer) = P(Cancer) x P(Positive|Cancer)

So P(Cancer|Positive) = 0.01 · 0.90 = 0.009

P(¬Cancer|Positive) = 0.99 · 0.1 = 0.099

Normalizing

Quiz Question

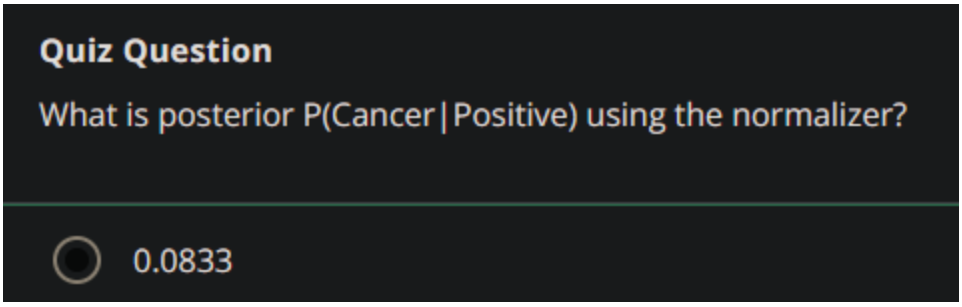
What normalizer should be used for posterior probabilities above?
Hint: We are looking for the sum of the posterior probabilities.

- 0.108

The answer is 0.108.

This means that there is a 0.108 probability of a positive test result.

Normalizer: P(pos) = P(Cancer|Positive) + P(¬Cancer|Positive)



Posterior of normalizer: posterior(joint) / normlaizer

Total Probability = Posterior of normalizer for: P(cancer|pos) + P(¬Cancer|Positive)

0.0833 + 0.9167 = 1

Bayes Rule Diagram

Term	Case
Prior Probability	P(Cancer)
Sensitivity	P(Positive | Cancer)
Specificity	P(Negative | $$\neg$$Cancer)

Let's say we get a positive test and we want to look at both cancer and no cancer hypotheses.

```
Cancer Hypothesis = Prior probability x sensitivity
No Cancer Hypothesis = Prior probability x (1-sensitivity)
Normalizer = Cancer Hypothesis + No Cancer Hypothesis
```

If these numbers (in steps 1 and 2 above) are added together, the result is the normalizer and it will normally not be 1.

The next step in the process is to normalize the hypotheses using the normalizer. Because the normalizer represents the probability of a positive test, it is independent of cancer diagnosis and therefore can be used to normalize both cases.

```
Posterior Probability (Cancer) = Cancer Hypothesis / Normalizer
Posterior Probability (No Cancer) = No Cancer Hypothesis / Normalizer
Adding the posterior probabilities = 1
```

This is the algorithm for Bayes Rule.

Bayes Rule Summary

To summarize Bayes Rule, we have a hidden variable (in this case, having cancer or not) that can not be measured directly, so we use a test.

```
Prior probability – P(C) – Tells us how frequently our variable is true.
```

And the test is characterized by:

```
Sensitivity – P(Positive|Cancer) – Tells us how often the test is
positive when the variable is true.
Specificity – P(Negative|¬¬Cancer) – Tells us how often the test is
negative when the variable is false.
```

Bayes Rule

1. Takes the prior probability and multiplies it by the test measurement, in this case, sensitivity, to create a new variable, P(Cancer, Positive).
2. Takes the prior probability and multiplies it by the opposite of the test measurement, in this case, 1-sensitivity, to create another new variable, P(¬¬Cancer, Positive).
3. These two variables, also known as the joint probabilities of two events, are added to determine the normalizer, P(Positive).
4. The variables in steps 1 and 2 are divided by the normalizer to arrive at the best estimate of the hidden variable, cancer, given our test result, P(Cancer|Positive) and P(¬¬Cancer, Positive). In this case, we were testing for the positive test result.

Robot Sensing 1

Imagine a robot that lives in exactly two places – red place (R) and green place (G). Initially, the robot can not tell where it is so—

```
P(R) = P(G) = 0.5.
```

The robot has unreliable visual sensors—

```
Probability of seeing red when in red is P(see R | at R) = 0.8
Probability of seeing green when in green is P(see G | at G) = 0.8
```

Quiz Question

What are the posterior probabilities of being at a red place given that the robot sees red and not having being in a red place given seeing red?

These are the correct matches.

Case	Probability
P(at Red|see Red)	0.8
P(at Green|see Red)	0.2

Step by Step Walkthrough

The step-by-step breakdown of the solution is pretty quick. Let's recap what's covered in the solution video.

Let's start with what we know: Prior Probabilities

The robot is perfectly ignorant about where it is, so prior probabilities are as follows:

P(at red)=0.5

P(at green)=0.5

Conditional Probabilities

The robot's sensors are not perfect. Just because the robot sees red does not mean the robot is at red.

P(see red|at red)=0.8

P(see green|at green)=0.8

Posterior Probabilities

From these prior and posterior probabilities we are asked to calculate the following posterior probabilities after the robot sees red:

```
P(at red|see red)P(at red|see red)
P(at green|see red)P(at green|see red)
```

and as a reminder, Bayes' rule looks like this:

P(A|B)=(P(B|A)·P(A))/P(B)

or, if we want to use our "versions" of A and B (for posterior #1)...

P(at red|see red)=(P(see red|at red)·P(at red))/P(see red)

Now, we can read two of those terms from what we already know about our prior and conditional probabilities which means we can rewrite this as...

P(at red∣see red)=(0.8·0.5)/P(see red)

But we still have one unknown! What was the probability that we would see red? The answer is 0.5 and there are two ways I can convince myself of that. The first is intuitive and the second is mathematical.

Why is P(see red) 0.5?

Argument 1: Intuitive

Of course, it's 0.5! The robot had a 50% belief that it was in red and a 50% belief that it was in green. Sure, its sensors are unreliable but that unreliability is symmetric and not biased towards mistakenly seeing either color.

So whatever the probability of seeing red is, that will also be the probability of seeing green. Since these two colors are the only possible colors the probability MUST be 50% for each! Argument 2: Mathematical (Law of Total Probability)

There are exactly two situations where the robot would see red.

```
When the robot is in a red square and its sensors work correctly.
When the robot is in a green square and its sensors make a mistake.
```

I just need to add up these two probabilities to get the total probability of seeing red.

P(see red)=P(at red)·P(see red∣at red)+P(at green)·P(see red∣at green)

I can read these quantities from above!

P(see red)=0.5·0.8+0.5·0.2P(see red)=0.5·0.8+0.5·0.2

P(see red)=0.4+0.1P(see red)=0.4+0.1

P(see red)=0.5P(see red)=0.5

Robot Sensing 2

The robot still lives in exactly two places – red place (R) and green place (G), but now:

```
P(R) = 0
P(G) = 1
```

The robot has unreliable visual sensors—

```
Probability of seeing red when in red is P(see R | at R) = 0.8
Probability of seeing green when in green is P(see G | at G) = 0.8
```

Quiz Question

What are the posterior probabilities of being at a red place given that the robot sees red and not having being in a red place given seeing red?

These are the correct matches.

Case	Probability
P(at Red|see Red)	0
P(at Green|see Red)	1

In this example, the prior probabilities are unaffected by the measurements. The joint probabilities of two events and normalizer are:

```
P(Red) x P(see Red|at Red) = (0) x (0.8) = 0
P(Green) x P(see Green|at Red) = (1) x (1-0.8) = 0.2
Normalizer = 0 + 0.2 = 0.2
```

Sebastian At Home

Where in the world is Sebastian? Or more specifically, is he at home. In this last exercise, use your Bayes Rule knowledge to come up with the probability that he is home and it is raining.

These are the facts given—

```
Probability Sebastian is not home – P(gone) = 0.6
Probability Sebastian is home – P(home) = 0.4
Probability of seeing rain when at home – P(rain|home) = 0.01
Probability of seeing rain when not at home – P(rain|gone) = 0.3
```

`Quiz Question`

What is the probability that Sebastian is home, given that he sees rain, P(home| rain)?

$$\frac{P(home)\cdot P(rain|home)}{P(home)\cdot P(rain|home)+P(gone)\cdot(1-P(rain|gone))} = \frac{(0.4)\cdot(0.01)}{(0.4)\cdot(0.01)+(0.6)\cdot(0.3))} = 0.0217$$

Next Up

You will reinforce your understanding of `probability`, `conditional probability`, and `Bayes Rule` .by doing some probability practice in Python.

More specifically, in this lesson, you will create code to calculate the following:

```
Probability
Binomial Distributions
```

Conditional Probability
Bayes Rule

Python Probability Practice

Simulating Coin Flips

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
% matplotlib inline
```

```
UsageError: Line magic function `%` not found.
```

In [4]:
```
# outcome of one coin flip
np.random.randint(2)
```

Out[4]:
```
0
```

In [5]:
```
# outcomes of ten thousand coin flips
np.random.randint(2, size=10000)
```

Out[5]:
```
array([1, 1, 0, ..., 0, 1, 0])
```

In [6]:
```
# mean outcome of ten thousand coin flips
np.random.randint(2, size=10000).mean()
```

Out[6]:
```
0.502
```

In [7]:
```
# outcome of one coin flip
np.random.choice([0, 1])
```

Out[7]:
```
0
```

In [8]:
```
# outcome of ten thousand coin flips
np.random.choice([0, 1], size=10000)
```

Out[8]:
```
array([1, 0, 0, ..., 0, 0, 1])
```

In [9]:
```
# mean outcome of ten thousand coin flips
np.random.choice([0, 1], size=10000).mean()
```

Out[9]:
```
0.494
```

In [10]:
```
# outcomes of ten thousand biased coin flips
np.random.choice([0, 1], size=10000, p=[0.8, 0.2])
# p = [0.8, 0.2] means the probability of the head is 0.8 & 0.2 for the tail
# first value is the head which it 0, and the tail is the second value 1
```

Out[10]:
```
array([0, 0, 0, ..., 1, 0, 1])
```

In [12]:
```
# mean outcome of ten thousand biased coin flips
np.random.choice([10, 1], size=10000, p=[0.8, 0.2]).mean()
```

Out[12]:
```
8.2063
```


coin flips produce exactly two heads

In [13]:
```
# simulate 1 million tests of two fair coin flips
tests = np.random.randint(2, size=(int(1e6), 2))
#                       head&tail  1milion  two flip
# sums of all tests
test_sums = tests.sum(axis=1) #This means that the sum is calculated for each row of the

# proportion of tests that produced exactly two heads
(test_sums == 0).mean()
#You have two flips in each trial. You're checking to see how often you get exactly two
```

Out[13]:
```
0.249995
```

2. Three fair coin flips produce exactly one head

In [14]:
```
# simulate 1 million tests of three fair coin flips
tests = np.random.randint(2, size=(int(1e6), 3))

# sums of all tests
test_sums = tests.sum(axis=1)

# (test_sums == 2).mean()
(test_sums == 2).mean()
```

Out[14]:
```
0.374893
```

3. Three biased coin flips with P(H) = 0.6 produce exactly one head

In [15]:
```
# simulate 1 million tests of three bias coin flips
# hint: use np.random.choice()
tests = np.random.choice([0, 1], size=(int(1e6), 3), p=[0.6, 0.4])

# sums of all tests
test_sums = tests.sum(axis=1)

# proportion of tests that produced exactly one head
(test_sums == 2).mean()
```

Out[15]:
```
0.28773
```

4. A die rolls an even number

In [16]:
```
# simulate 1 million tests of one die roll
tests = np.random.choice(np.arange(1, 7), size=int(1e6))

# proportion of tests that produced an even number
(tests % 2 == 0).mean()
```

Out[16]:
```
0.500029
```

5. Two dice roll a double

In [17]:
```
# simulate the first million die rolls
first = np.random.choice(np.arange(6), size=int(1e6))

# simulate the second million die rolls
second = np.random.choice(np.arange(6), size=int(1e6))
```

```
# proportion of tests where the 1st and 2nd die rolled the same number
(first == second).mean()
```

Out[17]:
```
0.166914
```

Simulating Many Coin Flips

n, p = number of trials, probability of each trial

In [26]:
```
# number of heads from 10 fair coin flips
np.random.binomial(10, 0.5)
```

Out[26]:
```
5
```

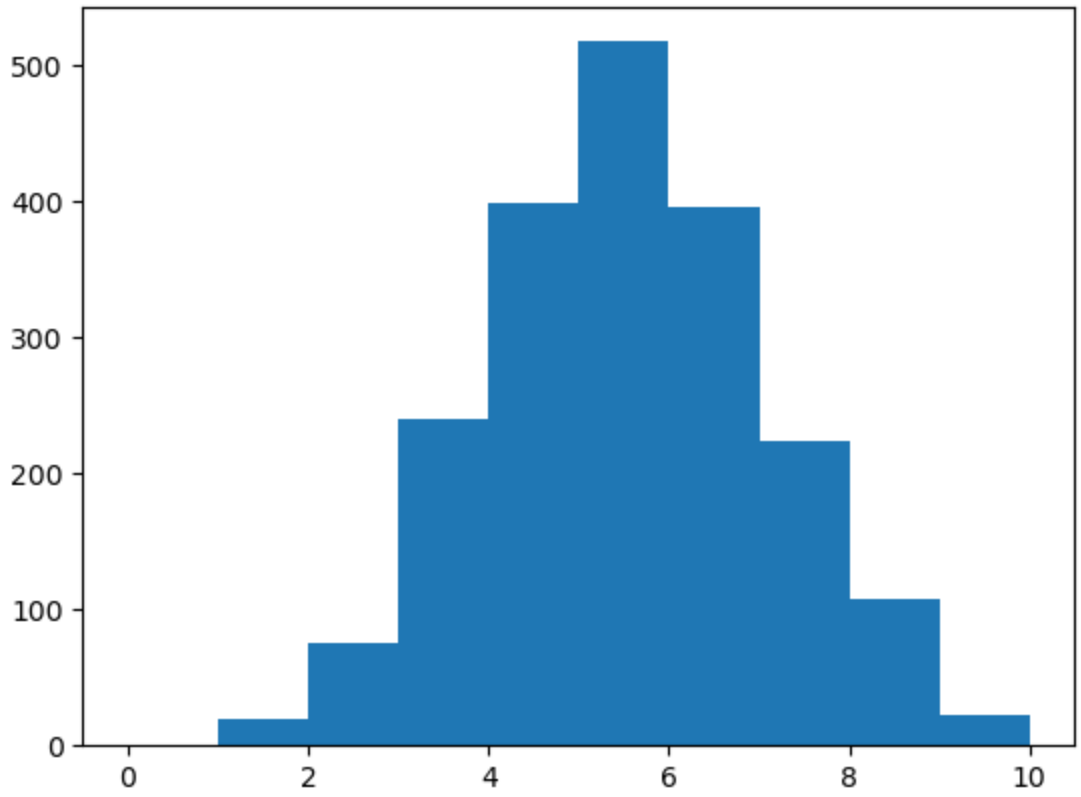
In [21]:
```
np.random.binomial(n=10, p=0.5, size=20).mean()
# it should be close to 5
```

Out[21]:
```
4.35
```

In [27]:
```
# results from 20 tests with 10 coin flips
np.random.binomial(10, 0.5, 20)
#same as the above cell
```

Out[27]:
```
array([5, 5, 5, 5, 6, 6, 6, 3, 4, 6, 3, 8, 6, 2, 4, 4, 7, 6, 5, 2])
```

In [25]:
```
plt.hist(np.random.binomial(n=10, p=0.5, size=2000));
```



1. A fair coin flip produces heads

In [28]:
```
# simulate 1 million tests of one fair coin flip
# remember, the output of these tests are the # successes, or # heads
tests = np.random.binomial(1, 0.5, int(1e6))
```

```
# proportion of tests that produced heads
(tests == 1).mean() #the output of these tests are the # successes, or # heads
```

Out[28]:
```
0.499513
```

2. Five fair coin flips produce exactly one head

In [29]:
```
# simulate 1 million tests of five fair coin flips
tests = np.random.binomial(5, 0.5, int(1e6))

# proportion of tests that produced 1 head
(tests == 1).mean()
```

Out[29]:
```
0.156382
```

3. Ten fair coin flips produce exactly four heads

In [30]:
```
# simulate 1 million tests of ten fair coin flips
tests = np.random.binomial(10, 0.5, int(1e6))

# proportion of tests that produced 4 heads
(tests == 4).mean()
```

Out[30]:
```
0.20528
```

4. Five biased coin flips with P(H) = 0.8 produce exactly five heads

In [31]:
```
# simulate 1 million tests of five biased coin flips
tests = np.random.binomial(5, 0.8, int(1e6))

# proportion of tests that produced 5 heads
(tests == 5).mean()
```

Out[31]:
```
0.327467
```

5. Ten biased coin flips with P(H) = 0.15 produce at least 3 heads

In [33]:
```
# simulate 1 million tests of ten biased coin flips
tests = np.random.binomial(10, 0.15, int(1e6))

# proportion of tests that produced at least 3 heads
(tests >= 3).mean()
```

Out[33]:
```
0.179568
```

Cancer Test Results

In [17]:
```
# load dataset
import pandas as pd
import numpy as np
df = pd.read_csv('cancer_test_data.csv')
df.head()
```

Out[17]:

	patient_id	test_result	has_cancer
0	79452	Negative	False
1	81667	Positive	True
2	76297	Negative	False
3	36593	Negative	False
4	53717	Negative	False

In [18]:
```
# number of patients
df.shape[0]
```

Out[18]: 2914

In [34]:
```
# number of patients with cancer
pCancer = df.query('has_cancer == True').shape[0]
print(pCancer)
```

```
306
```

In [20]:
```
# number of patients without cancer
pNoCancer = df.query('has_cancer == False').shape[0]
print(pNoCancer)
```

```
2608
```

In [21]:
```
# proportion of patients with cancer
propCancer = pCancer / df.shape[0]
print(propCancer)
```

```
0.10501029512697323
```

In [22]:
```
# proportion of patients without cancer
propNoCancer = pNoCancer / df.shape[0]
print(propNoCancer)
```

```
0.8949897048730268
```

In [33]:
```
# proportion of patients with cancer who test positive
# Calculate the total number of patients with cancer
total_cancer = df.query('has_cancer == True').shape[0]

# Calculate the number of patients without cancer who test positive
positive_cancer = df.query('has_cancer == True & test_result == "Positive"').shape[0]

# Calculate the proportion
proportion = positive_cancer / total_cancer
print(proportion)
```

```
0.9052287581699346
```

In [32]:
```
# proportion of patients with cancer who test negative
# Calculate the total number of patients with cancer
total_cancer = df.query('has_cancer == True').shape[0]

# Calculate the number of patients with cancer who test negative
negative_cancer = df.query('has_cancer == True & test_result == "Negative"').shape[0]

# Calculate the proportion
proportion = negative_cancer / total_withototal_cancerut_cancer
print(proportion)
```

```
0.09477124183006536
```

In [31]:
```
# proportion of patients without cancer who test positive
# Calculate the total number of patients without cancer
total_without_cancer = df.query('has_cancer == False').shape[0]

# Calculate the number of patients without cancer who test positive
positive_without_cancer = df.query('has_cancer == False & test_result == "Positive"').shape[0]

# Calculate the proportion
proportion = positive_without_cancer / total_without_cancer
print(proportion)
```

```
0.2036042944785276
```

In [30]:
```
# proportion of patients without cancer who test negative

# Calculate the total number of patients without cancer
total_without_cancer = df.query('has_cancer == False').shape[0]

# Calculate the number of patients without cancer who test negative
negative_without_cancer = df.query('has_cancer == False & test_result == "Negative"').shape[0]

# Calculate the proportion
proportion = negative_without_cancer / total_without_cancer
print(proportion)
```

```
0.7963957055214724
```

Conditional Probability & Bayes Rule Quiz

In [1]:
```
# load dataset
import pandas as pd

df = pd.read_csv('cancer_test_data.csv')
df.head()
```

Out[1]:

	patient_id	test_result	has_cancer
0	79452	Negative	False
1	81667	Positive	True
2	76297	Negative	False
3	36593	Negative	False
4	53717	Negative	False

In [2]:
```
# What proportion of patients who tested positive has cancer?
df.query('test_result == "Positive"')['has_cancer'].mean()
```

Out[2]: 0.34282178217821785

In [3]:
```
# What proportion of patients who tested positive doesn't have cancer?
1 - df.query('test_result == "Positive"')['has_cancer'].mean()
```

Out[3]: 0.65717821782178221

In [4]:
```
# What proportion of patients who tested negative has cancer?
df.query('test_result == "Negative"')['has_cancer'].mean()
```

Out[4]: 0.013770180436847104

In [5]:
```
# What proportion of patients who tested negative doesn't have cancer?
1 - df.query('test_result == "Negative"')['has_cancer'].mean()
```

Out[5]: 0.98622981956315292

Difference between the both quizzes

```
# Proportion of patients who tested positive and have cancer
#Quiz 2
positive_with_cancer = df.query('test_result == "Positive" & has_cancer == True').shape[0]
total_positive = df.query('test_result == "Positive"').shape[0]
proportion_tested_positive_has_cancer = positive_with_cancer / total_positive

# Proportion of patients with cancer who test positive
#Quiz 1
cancer_positive = df.query('has_cancer == True & test_result == "Positive"').shape[0]
total_cancer = df.query('has_cancer == True').shape[0]
proportion_has_cancer_tested_positive = cancer_positive / total_cancer


print(proportion_tested_positive_has_cancer)
print(proportion_has_cancer_tested_positive)
```

```
0.34282178217821785
0.9052287581699346
```

Introduction to Normal Distribution

In this lesson, we will cover:

```
The shape of a normal distribution
Derive the normal distribution function step by step
```

Maximum Probability

Let's say there is a fair coin (P = 0.5), and we flip it 20 times (N = 20).

What is the number of heads (k) that *maximizes* the probability in the binomial distribution?

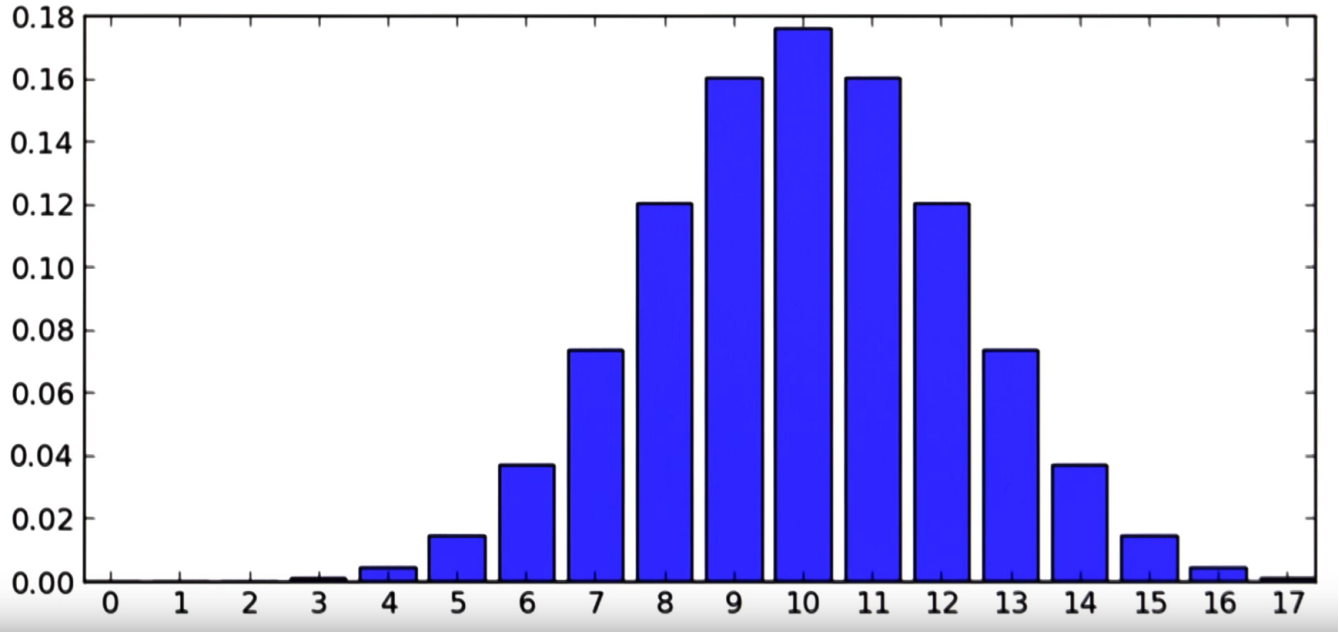
$P(k) = \frac{N!}{(N-k)!k!}p^k(1-p)^{n-k}$

Quiz Question

Using the binomial distribution $P(k) = \frac{N!}{(N-k)!k!}p^k(1-p)^{n-k}$, given that N = 20, p = 0.5. What is the value of k that maximizes

- ○ 1
- ○ 3
- ● 10
- ○ 20

The answer is 10 because once we plot all the possible k values from 1 to 20 and the corresponding probabilities, you can see that k = 10 gives us the highest probability.



The bell-curve distribution not only applies to binomial distributions but also almost any distribution with large samples.

The normal distribution can be written as:

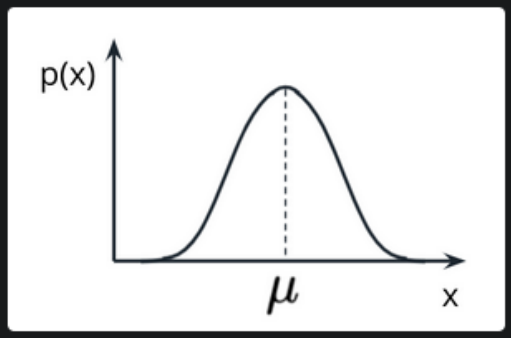
$$N(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} * e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

where μ is the mean and σ^2 is the variance of the normal distribution.

The quadratic term penalizes any deviations from the expectation μ and the exponential squeezes the shape back into the curves.

The shape of the normal distribution is shown below. The y-axis is the probability $p(x)$ and x is the possible mean values of the experiments. There is some information we can extract from the curve:

1. The probability is maximized when $x = \mu$
2. The probability decreases when x is deviating from μ
3. The probability is approaching 0 when x is very far from the mean



Plot of a normal distribution.

Scenario	Distribution
A single coin flip	Single probability
A few coin flips	Binomial distribution
Many coin flips	Normal distribution

The normal distribution is significant in statistics because it can be applied to problems involving other types of distributions if you have sufficiently large samples.

To look at the central limit theorem using the coin flip example again.

1. If you only flip a coin once, you use a single probability to describe the outcome of having Heads.
2. If you flip a coin a few times, you can use the binomial distribution to describe the probability of getting a given number of heads.
3. If you flip a coin 10,000 times, the probability of having a given number of heads is approximately a normal distribution.

Probability to Statistics

This begins a set of lessons that will be more data-oriented in the way that you are applying ideas and less probability-oriented.

Descriptive vs. Inferential Statistics

`Descriptive statisticsis` about describing our collected data.

`Inferential Statisticsis` about using our collected data to draw conclusions about a larger population.

```
Population - our entire group of interest.
Parameter - numeric summary about a population
Sample - a subset of the population
Statistic numeric summary about a sample
```

Introduction to Sampling Distributions

Sampling Distribution: The distribution of a Statistic

`Quiz`

```
Population:

Total cups = 21
Total red cups = 6
Total green cups = 15
```

Sampling Distribution:

```
Red cups = 4
Green cups = 1
```

Term	Value
Population size	21 students
Parameter (percentage of green cup in overall population)	71% of students drink coffee
Sample size	5 students
Statistic (percentage of students who drink coffee in the sampling distribution)	20% of students drink coffee

Sampling Distributions & Python

In [37]:
```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(42) #Even if you run the code multiple times, the output will remain the

students = np.array([1,0,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0])
```

`1.` Find the proportion of students who drink coffee in the above array. Store this value in a variable **p**.

In [38]:
```
students.mean()
```

Out[38]:
```
0.7142857142857143
```

In [45]:
```
p = students.mean()
p
```

Out[45]:
```
0.7142857142857143
```

`2.` Use numpy's **random.choice** to simulate 5 draws from the `students` array. What is proportion of your sample drink coffee?

In [39]:
```
sample1 = np.random.choice(students, 5, replace=True)
sample1.mean()
```

Out[39]:
```
0.6
```

`3.` Repeat the above to obtain 10,000 additional proportions, where each sample was of size 5. Store these in a variable called `sample_props`.

In [40]:
```
sample_props = []
for _ in range(10000):
    sample = np.random.choice(students, 5, replace=True)
    sample_props.append(sample.mean())
```

`4.` What is the mean proportion of all 10,000 of these proportions? This is often called **the mean of the sampling distribution**.

In [41]:
```
sample_props = np.array(sample_props)
sample_props.mean()
```

Out[41]:
```
0.714
```

`5.` What are the variance and standard deviation for the original 21 data values?

In [42]:
```
print('The standard deviation for the original data is {}'.format(students.std()))
print('The variance for the original data is {}'.format(students.var()))
```

```
The standard deviation for the original data is 0.45175395145262565
The variance for the original data is 0.20408163265306126
```

`6.` What are the variance and standard deviation for the 10,000 proportions you created?

In [43]:
```
print('The standard deviation of the sampling distribution of the mean of 5 draws is {}'
ariance for the sampling distribution of the mean of 5 draws is {}'.format(s
```

```
The standard deviation of the sampling distribution of the mean of 5 draws is 0.20436242
31604235
The variance for the sampling distribution of the mean of 5 draws is 0.04176399999999999
6
```

`7.` Compute p(1-p), which of your answers does this most closely match?

In [46]:
```
p*(1-p) # The variance of the original data
```

Out[46]:
```
0.20408163265306123
```

`8.` Compute p(1-p)/n, which of your answers does this most closely match?

In [47]:
```
p*(1-p)/5 # The variance of the sample mean of size 5
```

Out[47]:
```
0.04081632653061225
```

`9.` Notice that your answer to `8.` is commonly called the **variance of the sampling distribution**. If you were to change your first sample to be 20, what would this do for the variance of the sampling distribution? Simulate and calculate the new answers in `6.` and `8.` to check that the consistency you found before still holds.

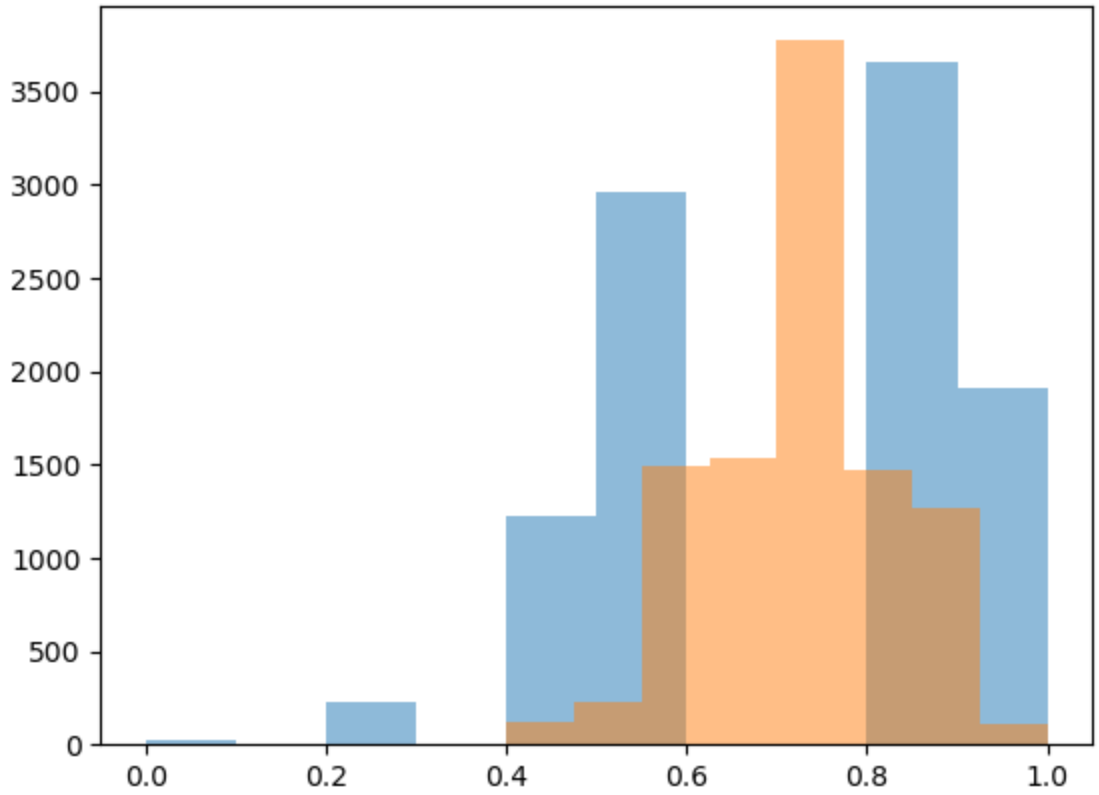
In [48]:
```
##Simulate your 20 draws
sample_props_20 = []
for _ in range(10000):
    sample = np.random.choice(students, 20, replace=True)
    sample_props_20.append(sample.mean())
```

In [49]:
```
##Compare your variance values as computed in 6 and 8,
##but with your sample of 20 values


print(p*(1-p)/20) # The theoretical variance
print(np.array(sample_props_20).var()) # The simulated variance
```

```
0.010204081632653062
0.010300994374999999
```

`10.` Finally, plot a histgram of the 10,000 draws from both the proportions with a sample size of 5 and the proportions with a sample size of 20. Each of these distributions is a sampling distribution. One is for the proportions of sample size 5 and the other a sampling distribution for proportions with sample size 20.

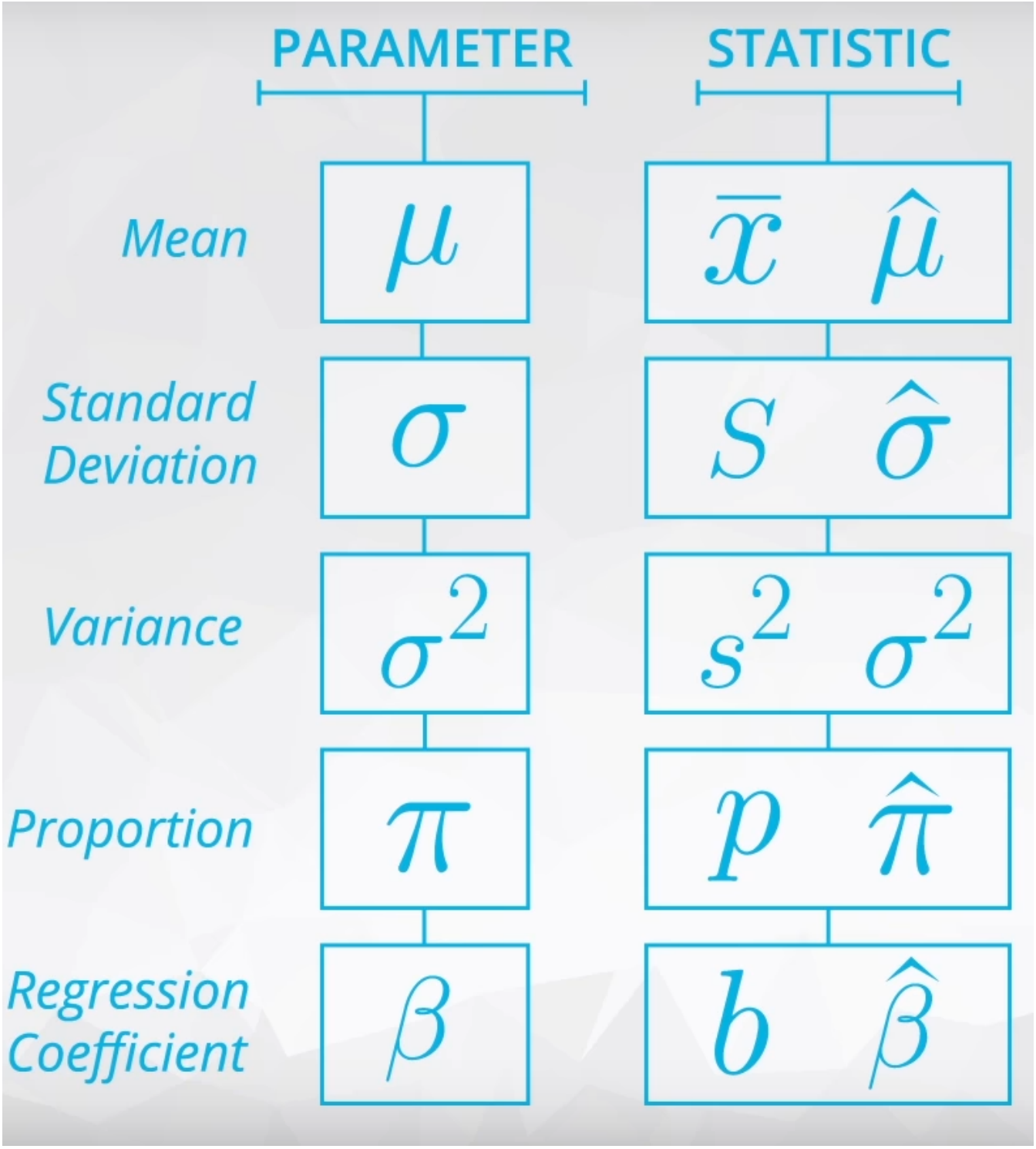
In [50]:
```
plt.hist(sample_props, alpha=.5);
plt.hist(np.array(sample_props_20), alpha=.5);
```



In [ ]:
```
# Notice the 20 is much more normally distributed than the 5
```

Introduction to Notation

Notation: A common Math language used to communicate EX:

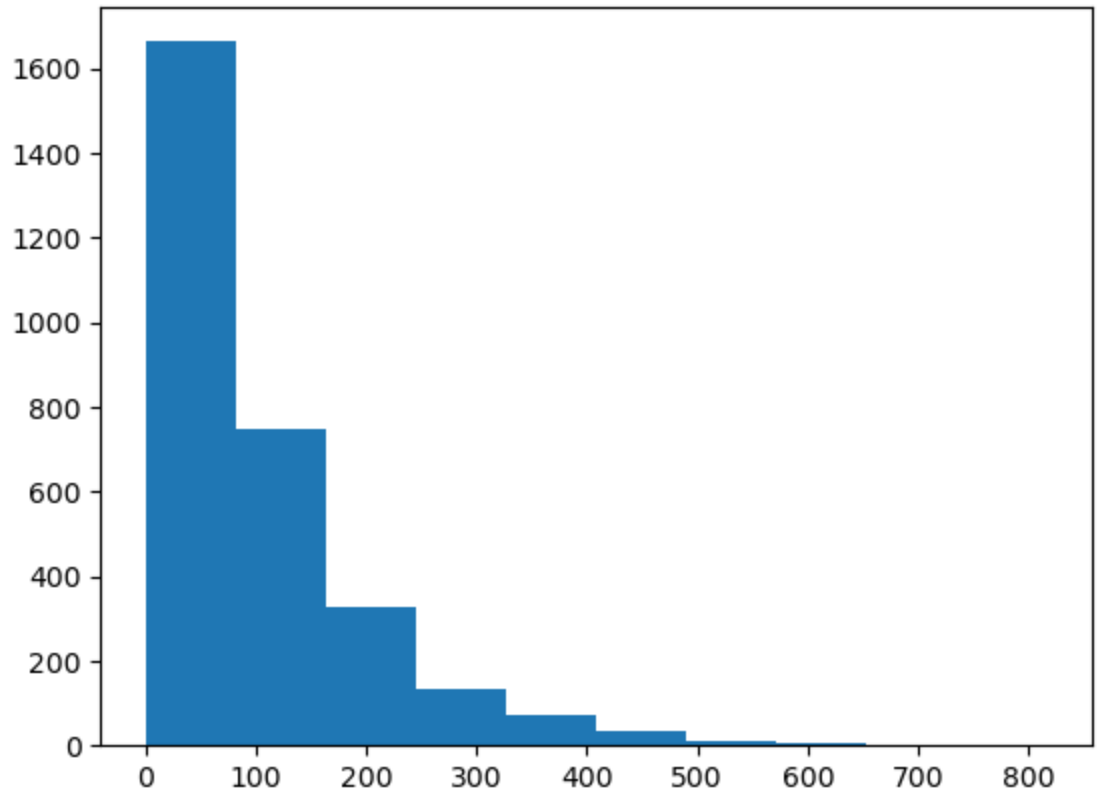


Remember that all parameters pertain to a population, while all statistics pertain to a sample.

Notation & Python

Match the number to the corresponding correct notation.

In [51]:
```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
np.random.seed(42)

pop_data = np.random.gamma(1,100, 3000)
                           data);
```



1. What is the population mean?

In [52]:
```
pop_data.mean()
```

Out[52]:
```
100.35978700795846
```

2. Randomly select 10 draws from the population using **numpy's random.choice**. What is the sample mean for these 10 values?

In [53]:
```
sample1 = np.random.choice(pop_data, 10, replace = True)
sample1.mean()
```

Out[53]:
```
103.47556549464393
```

$\bar{x}$ 103.48

3. What is the sample standard deviation of your 10 draws?

In [54]:
```
print("The sample standard deviation is {}".format(sample1.std()))
```

```
The sample standard deviation is 130.00074035573317
```

n 130

4. What is the population standard deviation?

In [55]:
```
print("The population standard deviation is {}".format(pop_data.std()))
```

```
The population standard deviation is 99.77860187968906
```

5. What is the population variance?

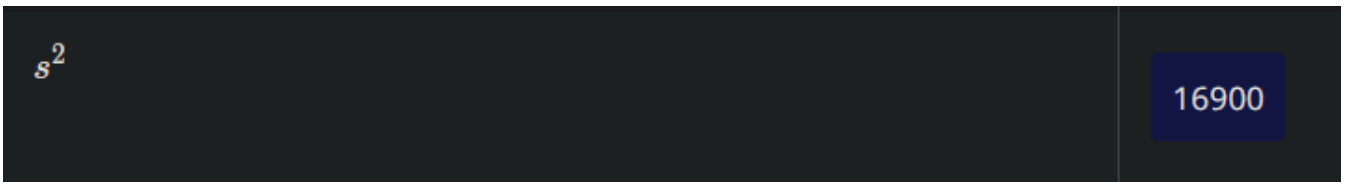
In [56]:
```
print("The population variance is {}".format(pop_data.var()))
```

```
The population variance is 9955.76939306549
```

6. What is the sample variance?

In [57]:
```
print("The sample variance is {}".format(sample1.var()))
```

```
The sample variance is 16900.192493038754
```

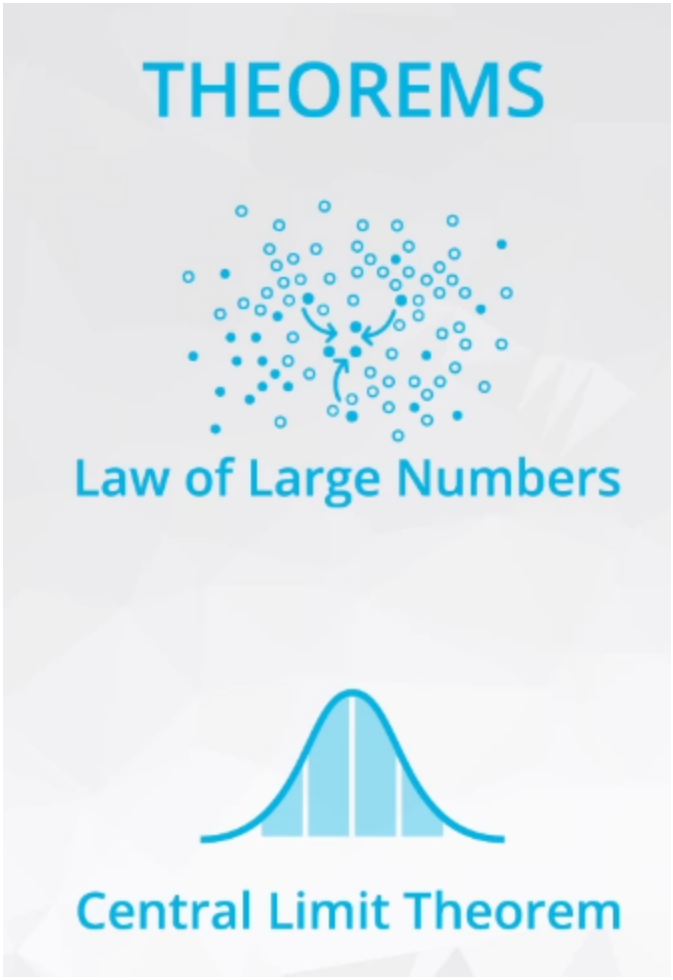


Two important mathematical theorems for working with sampling distributions include:

1. **Law of Large Numbers**: The larger our sample size, the closer our statistic gets to the parameter
2. **Central Limit Theorem**: states that with a large enough sample size the sampling distribution of the mean will be normally distributed.

The **Central Limit Theorem** actually applies to these well-known statistics:

1. Sample means ($\bar{x}$)
2. Sample proportions (p)
3. Difference in sample means ($\bar{x}_1 - \bar{x}_2$)
4. Difference in sample proportions ($p_1 - p_2$)



Which of the following statements were true regarding the values you computed?

- [x] The larger the sample size, the closer the sample mean was to the population mean.
- [ ] The sample mean values were always less than the population mean.
- [ ] The sample mean values were always greater than the population's mean.

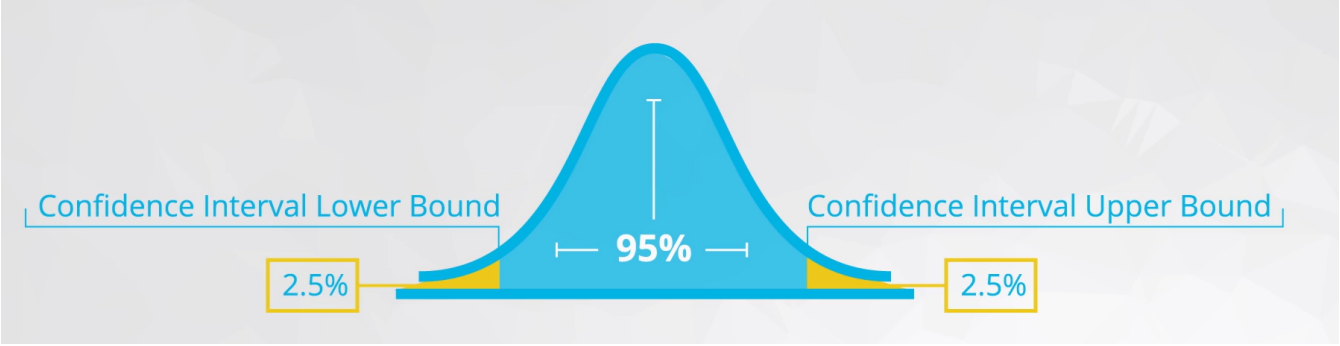
Bootstrapping

Bootstrapping is sampling with replacement

The fact that values could be chosen more than once makes this bootstrap sampling or "bootstrapping".

By bootstrapping and then calculating repeated values of our statistics, we can understand the sampling distribution of our statistics.

From Sampling Distributions to Confidence Intervals



We can use bootstrapping and sampling distributions to build confidence intervals for our parameters of interest.

Quiz: Building Confidence Intervals

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

np.random.seed(42)

coffee_full = pd.read_csv('coffee_dataset.csv')
coffee_red = coffee_full.sample(200) #this is the only data you might actually get in the real world.
```

1. What is the proportion of coffee drinkers in the sample? What is the proportion of individuals that don't drink coffee?

```
coffee_red['drinks_coffee'].mean() # Drink Coffee
```

```
0.59499999999999997
```

```
1 - coffee_red['drinks_coffee'].mean() # Don't Drink Coffee
```

```
0.40500000000000003
```

2. Of the individuals who do not drink coffee, what is the average height?

```
#coffee_red[coffee_red['drinks_coffee'] == False]['height'].mean()
#SAME
coffee_red.query('drinks_coffee == False')['height'].mean()
```

```
66.784922799278775
```

3. Simulate 200 "new" individuals from your original sample of 200. What are the proportion of coffee drinkers in your bootstrap sample? How about individuals that don't drink coffee?

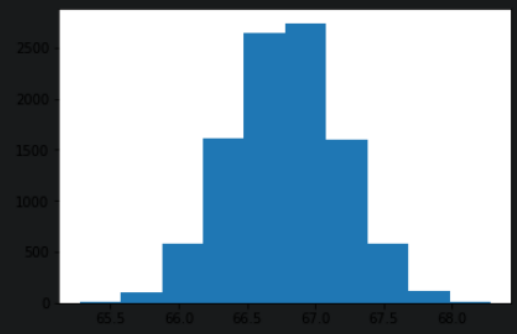
```
bootsamp = coffee_red.sample(200, replace = True)
```

```
bootsamp['drinks_coffee'].mean() # Drink Coffee and 1 minus gives the don't drink
```

```
0.60499999999999998
```

4. Now simulate your bootstrap sample 10,000 times and take the mean height of the non-coffee drinkers in each sample. Plot the distribution, and pull the values necessary for a 95% confidence interval. What do you notice about the sampling distribution of the mean in this example?

```
boot_means = []
for _ in range(10000):
    bootsamp = coffee_red.sample(200, replace = True)
    boot_mean = bootsamp.query('drinks_coffee == False')['height'].mean()
    boot_means.append(boot_mean)

plt.hist(boot_means); # Looks pretty normal
```




```
np.percentile(boot_means, 2.5), np.percentile(boot_means, 97.5)
```

```
(65.992913281575198, 67.584027382815734)
```

Quiz Question

Did the proportion of coffee drinkers in your bootstrap sample exactly match the proportion in the original sample?

- ○ Yes
- ◉ No

What is the lower bound of your 95% confidence interval for the mean height of those who do not drink coffee in your population?	65.99
What is the upper bound of your 95% confidence interval for the mean height of those who do not drink coffee in your population?	67.58
What is the value for the mean height of those who do not drink coffee in your population?	66.44
Did your interval capture the true mean height for the non-coffee drinkers?	Yes
Was the sample mean height for non-coffee drinkers the same as the population mean height for non-coffee drinkers?	No

Difference In Means

WHAT IS THE DIFFERENCE IN THE MEAN HIGHT FOR COFFEE VS. NON-COFFEE DRINKERS?

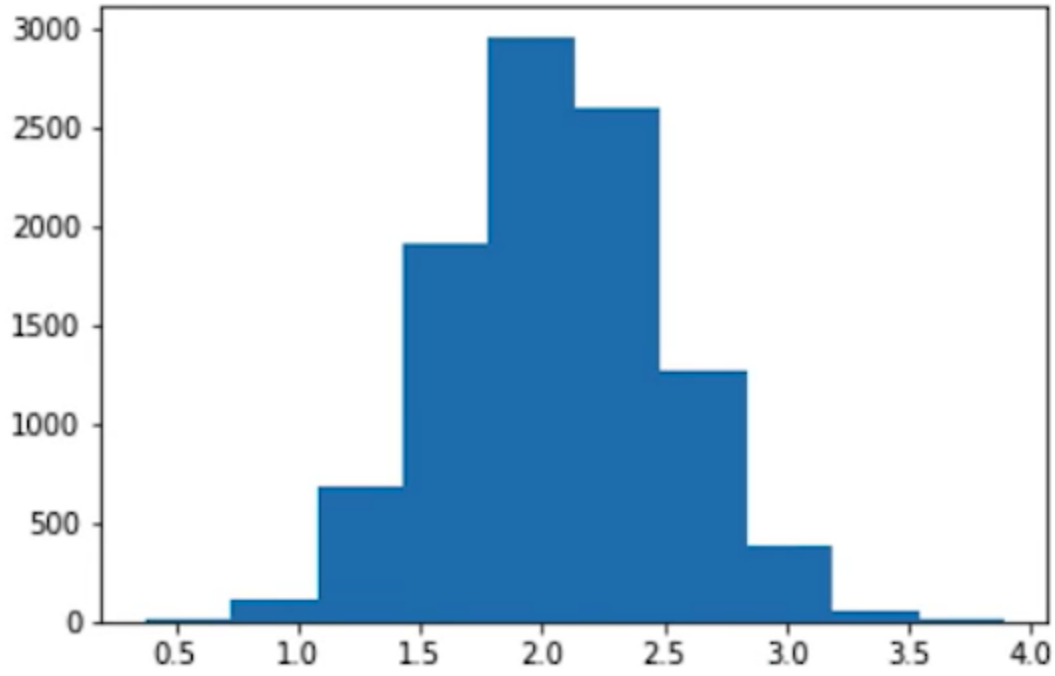
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

coffee_full = pd.read_csv('coffee-dataset.csv')
coffee_red = coffee_full.sample(200)
```

```
bootsample = coffee_red.sample(200, replace=True)
mean_coff = bootsample[bootsample['drinks_coffee'] == True]['height'].mean()
mean_nocoff = bootsample[bootsample['drinks_coffee'] == False]['height'].mean()
mean_coff - mean_nocoff
```

```
1.9604248402959854
```

```
diff = []
for _ in range(10000):
    bootsample = coffee_red.sample(200, replace=True)
    mean_coff = bootsample[bootsample['drinks_coffee'] == True]['height'].mean()
    mean_nocoff = bootsample[bootsample['drinks_coffee'] == False]['height'].mean()
    diff.append(mean_coff - mean_nocoff)
```

```
plt.hist(diff);
```




```
np.percentile(diffs, 2.5), np.percentile(diffs, 97.5)
```

```
(0.5854619923130314, 2.3686622208883668)
```

In this case, you saw that our confidence interval doesn't contain zero. Therefore, this suggest that there is a difference in the population means

SINCE A CONSIDENCE INTERVAL FOR MEAN_COFF - MEAN_NOCOFF IS (0.59, 2.37), WE HAVE EVIDENCE OF THE MEAN HEGHIT FOR COFFEE DRINKERS IS LARGER THAN NON-COFFEE

DRINKERS

Confidence Interval - Difference In Means & Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
np.random.seed(42)

full_data = pd.read_csv('coffee_dataset.csv')
sample_data = full_data.sample(200)
```

1. For 10,000 iterations, bootstrap sample your sample data, compute the difference in the average heights for coffee and non-coffee drinkers. Build a 99% confidence interval using your sampling distribution. Use your interval to start answering the first quiz question below.

```
diffs = []
for _ in range(10000):
    bootsamp = sample_data.sample(200, replace=True)
    coff_mean = bootsamp.query('drinks_coffee == True')['height'].mean()
    nocoff_mean = bootsamp.query('drinks_coffee == False')['height'].mean()
    diff = coff_mean - nocoff_mean
    diffs.append(diff)

np.percentile(diffs, 0.5), np.percentile(diffs, 99.5)


# statistical evidence coffee drinkers are on average taller
```

```
(0.13589915206465278, 2.5634536859777701)
```

```
all_positive = all(diff > 0 for diff in diffs)
all_positive
```

```
False
```

2. For 10,000 iterations, bootstrap sample your sample data, compute the difference in the average heights for those older than 21 and those younger than 21. Build a 99% confidence interval using your sampling distribution. Use your interval to finish answering the first quiz question below.

```
diffs_age = []
for _ in range(10000):
    bootsamp = sample_data.sample(200, replace = True)
    under21_mean = bootsamp[bootsamp['age'] == '<21']['height'].mean()
    over21_mean = bootsamp[bootsamp['age'] != '<21']['height'].mean()
    diff = over21_mean - under21_mean
    diffs_age.append(diff)

np.percentile(diffs_age, 0.5), np.percentile(diffs_age, 99.5)
# statistical evidence that over21 are on average taller
```

```
(3.3846249718386421, 5.1051788925372721)
```

```
all_positive = all(diff > 0 for diff in diffs_age)
all_positive
```

```
True
```

3. For 10,000 iterations bootstrap your sample data, compute the difference in the average height for coffee drinkers and the average height non-coffee drinkers for individuals under 21 years old. Using your sampling distribution, build a 95% confidence interval. Use your interval to start answering question 2 below.

```
diffs_coff_under21 = []
for _ in range(10000):
    bootsamp = sample_data.sample(200, replace = True)
    under21_coff_mean = bootsamp.query("age == '<21' and drinks_coffee == True")['height'].mean()
    under21_nocoff_mean = bootsamp.query("age == '<21' and drinks_coffee == False")['height'].mean()
    diffs_coff_under21.append(under21_nocoff_mean - under21_coff_mean)

np.percentile(diffs_coff_under21, 2.5), np.percentile(diffs_coff_under21, 97.5)
# For the under21 group, we have evidence that the non-coffee drinkers are on average taller
```

```
(1.0593651244624331, 2.5931557940679251)
```

4. For 10,000 iterations bootstrap your sample data, compute the difference in the average height for coffee drinkers and the average height non-coffee drinkers for individuals under 21 years old. Using your sampling distribution, build a 95% confidence interval. Use your interval to finish answering the second quiz question below. As well as the following questions.

```
diffs_coff_over21 = []
for _ in range(10000):
    bootsamp = sample_data.sample(200, replace = True)
    over21_coff_mean = bootsamp.query("age != '<21' and drinks_coffee == True")['height'].mean()
    over21_nocoff_mean = bootsamp.query("age != '<21' and drinks_coffee == False")['height'].mean()
    diffs_coff_over21.append(over21_nocoff_mean - over21_coff_mean)

np.percentile(diffs_coff_over21, 2.5), np.percentile(diffs_coff_over21, 97.5)
# For the over21 group, we have evidence that on average the non-coffee drinkers are taller
```

```
2]: (1.827895397088422, 4.4026329654774772)
```

Within the under 21 and over 21 groups, we saw that on average non-coffee drinkers were taller. But, when combined, we saw that on average coffee drinkers were on average taller. This is again **Simpson's paradox**, and essentially there are more adults in the dataset who were coffee drinkers. So these individuals made it seem like coffee drinkers were on average taller - which is a misleading result.

A larger idea for this is the idea of confounding variables altogether. You will learn even more about these in the regression section of the course.

Statement	True or False
Based on the confidence interval in the first question, you have evidence that coffee drinkers are on average taller than non-coffee drinkers.	True
In every bootstrapped instance in the first question, the difference in your averages suggested that coffee drinkers are on average taller than non-coffee drinkers.	False
Based on the confidence interval in the second question, you have evidence that those older than 21 are on average taller than those younger than 21.	True
In every bootstrapped instance in the second question, the difference in your averages suggested that those older than 21 are on average taller than those younger than 21.	True

SINCE THE np.perecentile NOT EQUAL ZERO SO THE CONFIDENCE INTERVAL IS TRUE ... IF THE BOOTSTRAPPED IS TRUE. THEREFORE, all_positive is True; otherwise, it is false.

Quiz Question

in the first intervals, you had evidence that the average height of coffee drinkers was taller, but in the final intervals, you had evidence that coffee drinkers in each group were actually shorter. What was this an example of:

- ○ The Empirical Rule
- ○ Occam's Razor
- ○ Mathematical Error
- ◉ Simpson's Paradox ✓

Confidence Interval Applications

A/B testing is one of the most important technique to compare between two groups

Statistical vs. Practical Significance

Using confidence intervals and hypothesis testing, you are able to provide **statistical significance** in making decisions.

Practical significance takes into consideration other factors of your situation that might not be considered directly in the results of your hypothesis test or confidence interval. Constraints like **space**, **time**, or **money** are important in business decisions.

Quiz Question

If a weight loss drug helped individuals lose on average 0.5 lbs over the course of 5 years, which of the following are true statements about this result?

- [ ] This would definitely **not** be statistically significant.
- [x] This would definitely **not** be practically significant. ✓
- [ ] This would be more likely to be statistically significant if we had smaller sample sizes.
- [x] This would be more likely to be statistically significant if we had larger sample sizes. ✓

Actually smaller sample sizes make it harder to prove statistical differences.

REGERDING OF OUR BEFORE EXAMPLES , BOOTSTRAPPING IS THE BEST WAY FOR CREATING THE CONFIDENCE INTERVAL

```
diff = []

for _ in range(10000):
    bootsample = coffee_red.sample(200, replace=True)
    mean_coff = bootsample[bootsample['drinks_coffee'] == True]['height'].mean()
    mean_nocoff = bootsample[bootsample['drinks_coffee'] == False]['height'].mean()
    diff.append(mean_coff - mean_nocoff)

np.percentile(diff, 2.5), np.percentile(diff, 97.5)
```

```
(0.39656867909086274, 2.2432588681124224)
```

```
import statsmodels.stats.api as sms

X1 = coffee_red[coffee_red['drinks_coffee'] == True]['height']
X2 = coffee_red[coffee_red['drinks_coffee'] == False]['height']

cm = sms.CompareMeans(sms.DescrStatsW(X1), sms.DescrStatsW(X2))
cm.tconfint_diff(usevar='unequal')
```

```
/Users/clamps/anaconda/envs/RoboND/lib/python3.5/site-packages/statsmodels/compat/pand
ndas.core.datetools module is deprecated and will be removed in a future version. Plea
e instead.
  from pandas.core import datetools
```

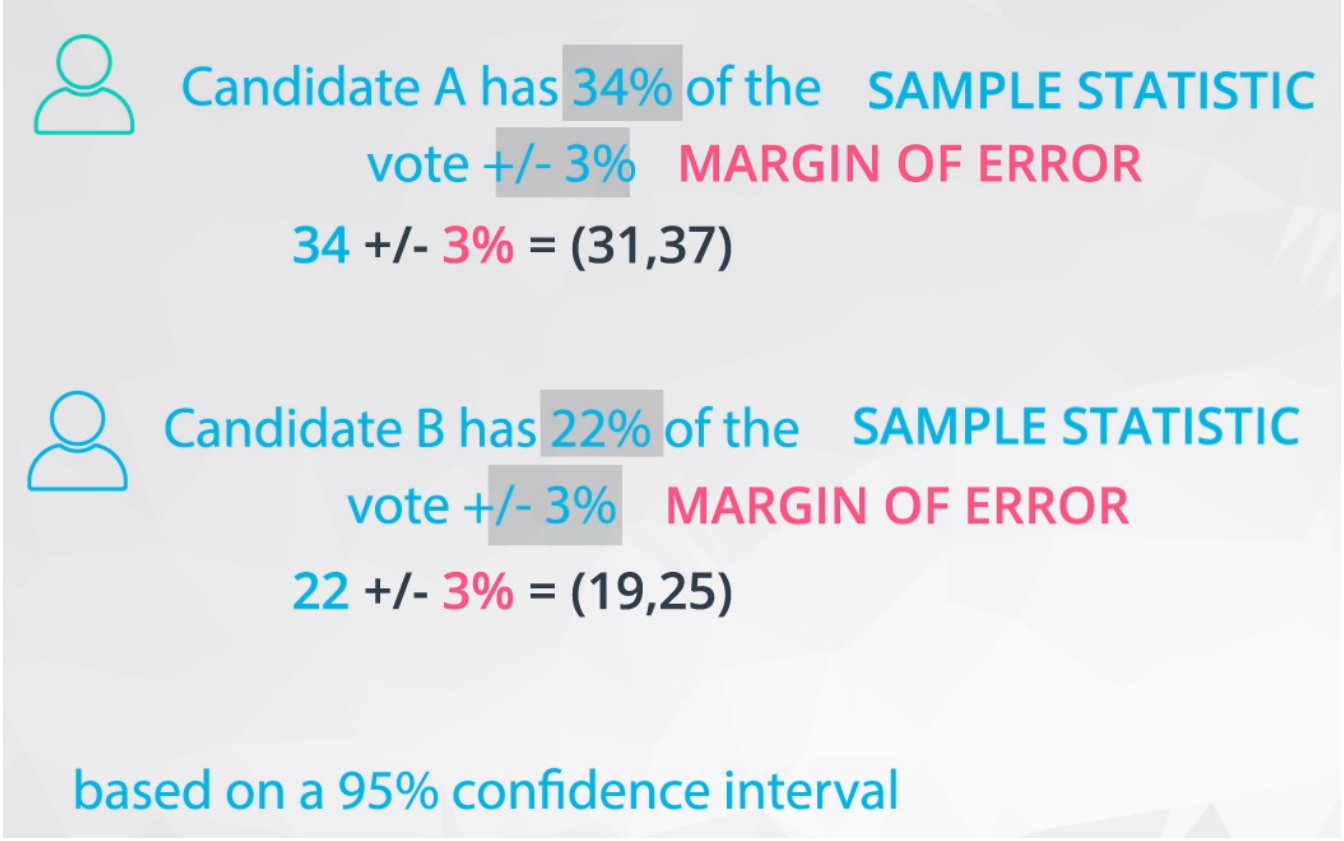
```
(0.39600106159185644, 2.2734131570228908)
```

There are many different Confidence Interval and Hypothesis Tests

- T-Test
- Two-Sample T-Test
- Paired T-Test
- Z-Test
- Chi-Squared Test
- F-Test

Bootstrapping can be used in place of any of these techniques.

Other Language Associated with Confidence Intervals



Quiz Question

Imagine we build a confidence interval for a population mean to obtain a confidence interval with an upper bound of 20 and a confidence interval width of 8. Use this information to provide the value to each of the corresponding additional values.

These are the correct matches.

Description	Value
Upper Bound of the Confidence Interval	20
Lower Bound of the Confidence Interval	12
The Margin of Error	4
The Sample Mean	16

Quiz Question

Which of the following statements are true?

- [x] If you increase your sample size, holding all other items constant, your confidence interval will narrow.
- [ ] If you increase your sample size, holding all other items constant, your confidence interval will widen.
- [ ] If you increase your confidence level, holding all other items constant, your confidence interval will narrow.
- [x] If you increase your confidence level, holding all other items constant, your confidence interval will widen.

Increasing your sample size will narrow your confidence interval, and increasing your confidence level will widen your interval.

Confidence Intervals (& Hypothesis Testing) vs. Machine Learning

Confidence intervals take an aggregate approach towards the conclusions made based on data, as these tests are aimed at understanding population parameters (which are aggregate population values).

Alternatively, machine learning techniques take an individual approach towards making conclusions, as they attempt to predict an outcome for each specific data point.

In the final lessons of this class, you will learn about two of the most fundamental machine learning approaches used in practice: linear and logistic regression.

What's Next

The topics of confidence intervals and hypothesis testing essentially do the same thing, but depending on who you talk to or what source you are reading from, it is important to understand both.

Important

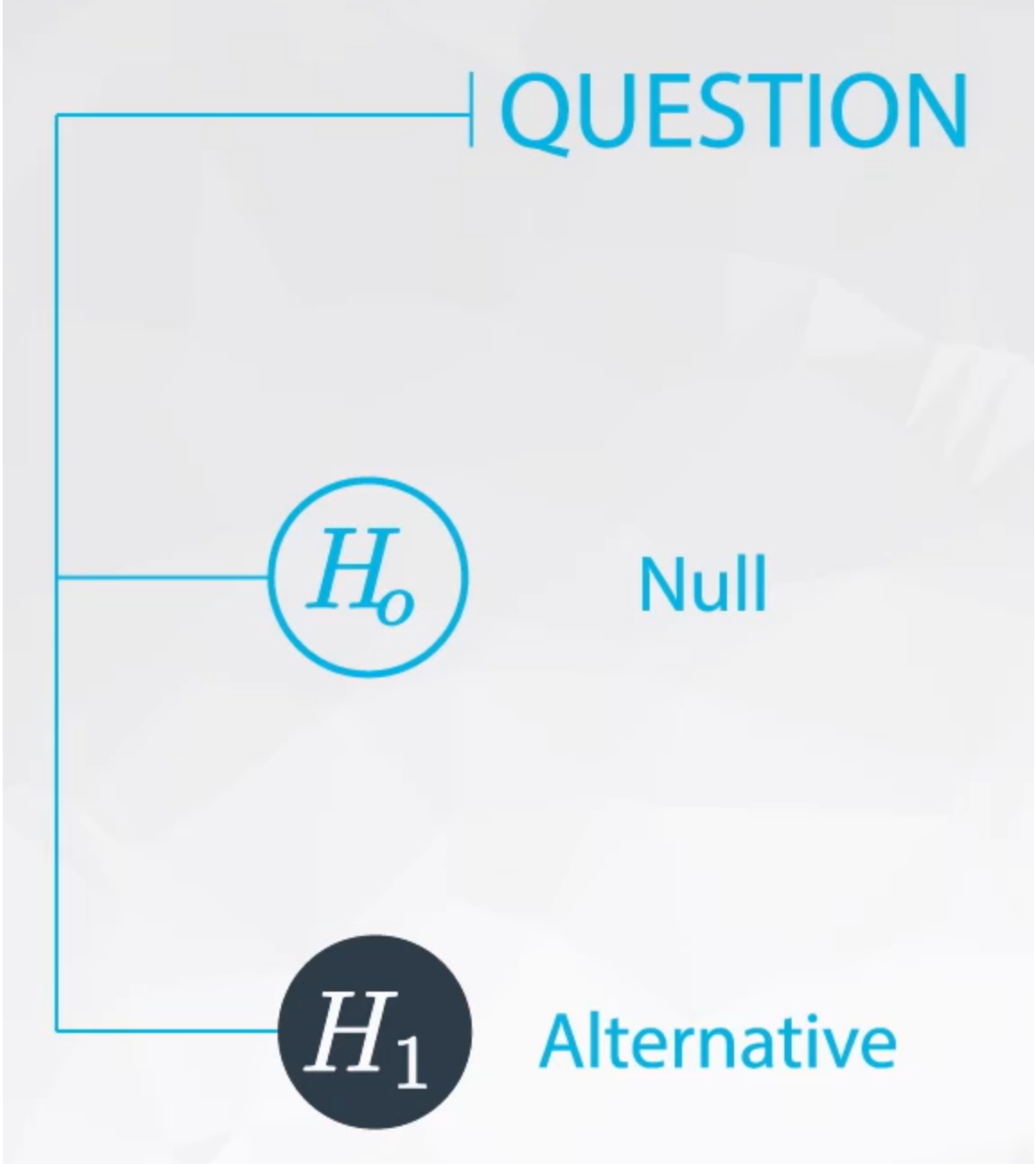
If you have not been introduced to this topic before, hypothesis testing can be a challenging subject. In fact, this is often reported as the most difficult lesson.

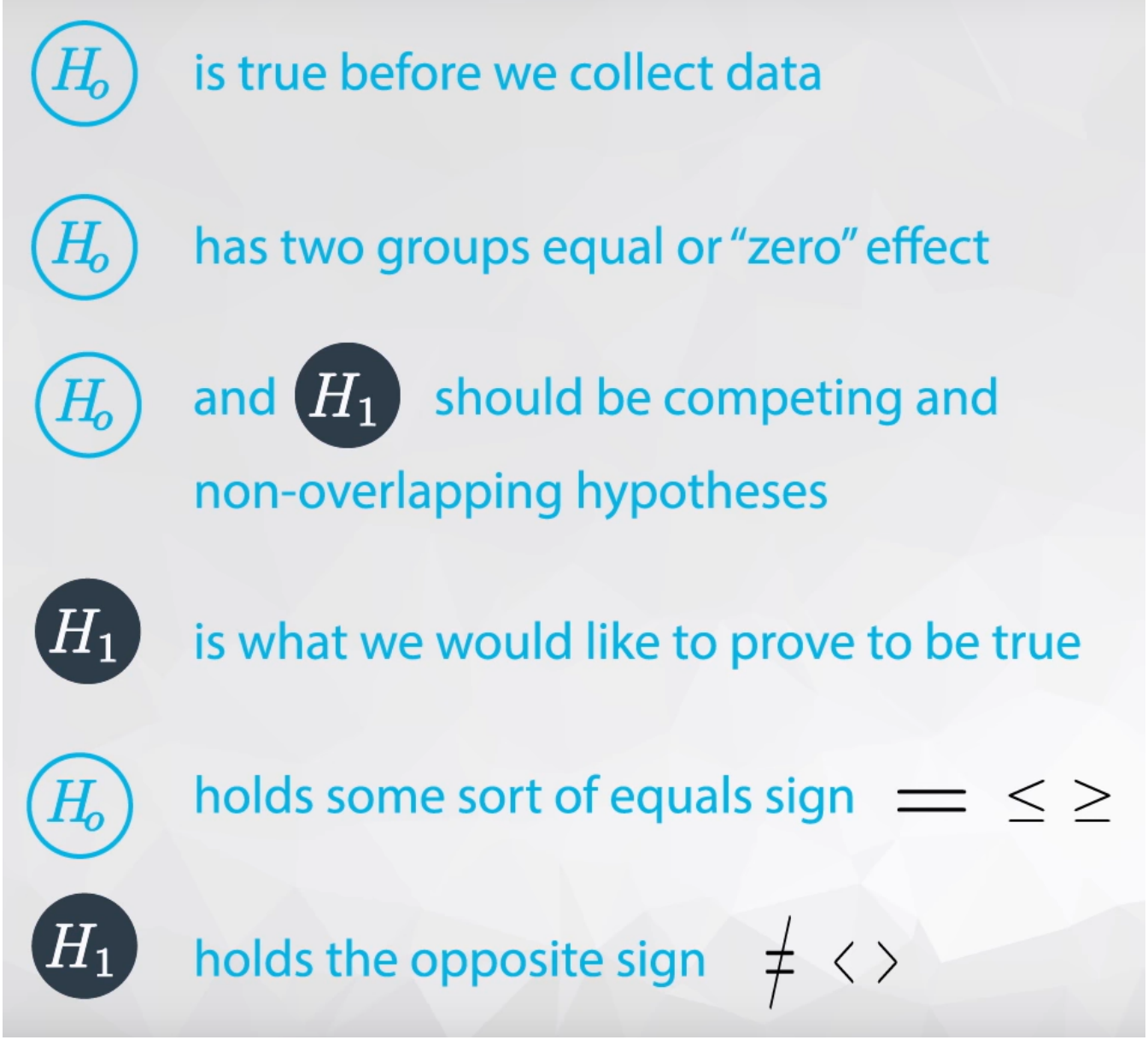
Hypothesis Testing

To answer questions, we need to break down a question into a hypothesis. Then we collect data to justify which hypothesis is likely to be true.

Hypothesis testing and confidence intervals allow for the use of sample data to draw conclusions about an entire population.

Setting Up Hypothesis Tests





EXAMPLE: "Innocent until proven guilty" suggests the following hypotheses are true:

H0: Innocent

H1: Guilty

We can relate this to the idea that "innocent" is true before we collect any data. Then the alternative must be a competing, non-overlapping hypothesis. Hence, the alternative hypothesis is that an individual is guilty.

Quiz Question

Which hypothesis is the hypothesis we believe before collecting any data?

- (●) The null hypothesis.
- ( ) The alternative hypothesis.
- ( ) We could believe either to be true.
- ( ) We don't believe **either** hypothesis is true until after we collect data.

Quiz Question

Select **all** of the below statements that are true regarding the null and alternative hypotheses.

- [x] We assume the null hypothesis is true before we collect any data.
- [x] The null and alternative hypotheses should be competing, non-overlapping hypotheses.
- [x] I never lose at chess is a possible null hypothesis.
- [x] Null hypotheses usually contain an equals sign when they pertain to mathematical ideas.

The alternative would be I sometimes lose at chess and would make this null hypothesis viable.

Quiz Question

Consider the following statement:

"Dyson air purifiers are the best."

Which set of hypotheses could accurately test this statement?

- ( ) Null: Dyson air purifiers **are not** the best. Alternative: Dyson air purifiers **are not** the best.
- ( ) Null: Dyson air purifiers **are** the best. Alternative: Honeywell air purifiers **are not** as good as Dyson air purifiers.
- (●) Null: Dyson air purifiers **are not** the best. Alternative: Dyson air purifiers **are** the best.
- ( ) It's impossible to set up null and alternative hypotheses based on the above statement.

This statement suggests that the individual believes that Dyson air purifiers are the best. Then the competing, non-overlapping alternative hypothesis is that the Dyson air purifier is not the best. Even if we should better define metrics aimed at "best", this does constitute a null and alternative based on this statement.

Statement	Hypothesis
The average height of all people in the world is different than 68 inches tall.	Alternative
The average height of all people in the world is less than 68 inches tall.	Alternative
The average height of all people in the world is less than or equal to 68 inches	Null
The impact of Drug A is on average the same as Drug B.	Null
Drug A will on average have a larger impact than Drug B.	Alternative

Quiz Question

Consider the following statement:

"The average height of a human is smaller than or equal to 70 inches."

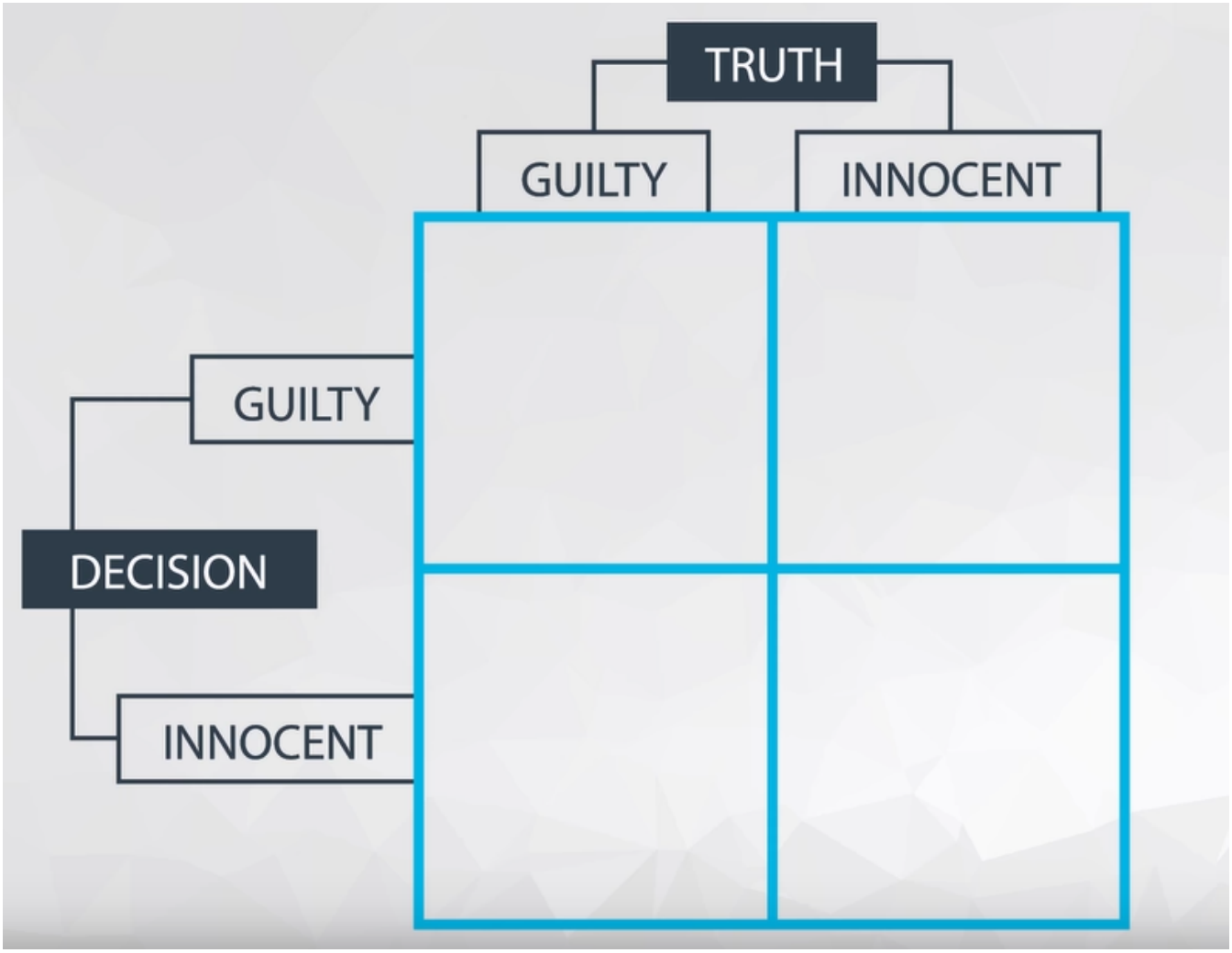
Which set of hypotheses below is correctly set up based on this assertion?

- ○ Null: The average height of a human is greater than 70 inches. Alternative: The average height of a human is smaller than or equal to 70 inches.
- ○ Null: The average height of a human is greater than 70 inches. Alternative: The average height of a human is not greater than 70 inches.
- ◉ Null: The average height of a human is not greater than 70 inches. Alternative: The average height of a human is greater than 70 inches.
- ○ It's impossible to set up null and alternative hypotheses based on the above statements.

The null is usually set up with an equal (you can image less than or equal to 70 here). The one-sided part of the hypothesis is generally placed in the alternative hypothesis.

Types of Errors

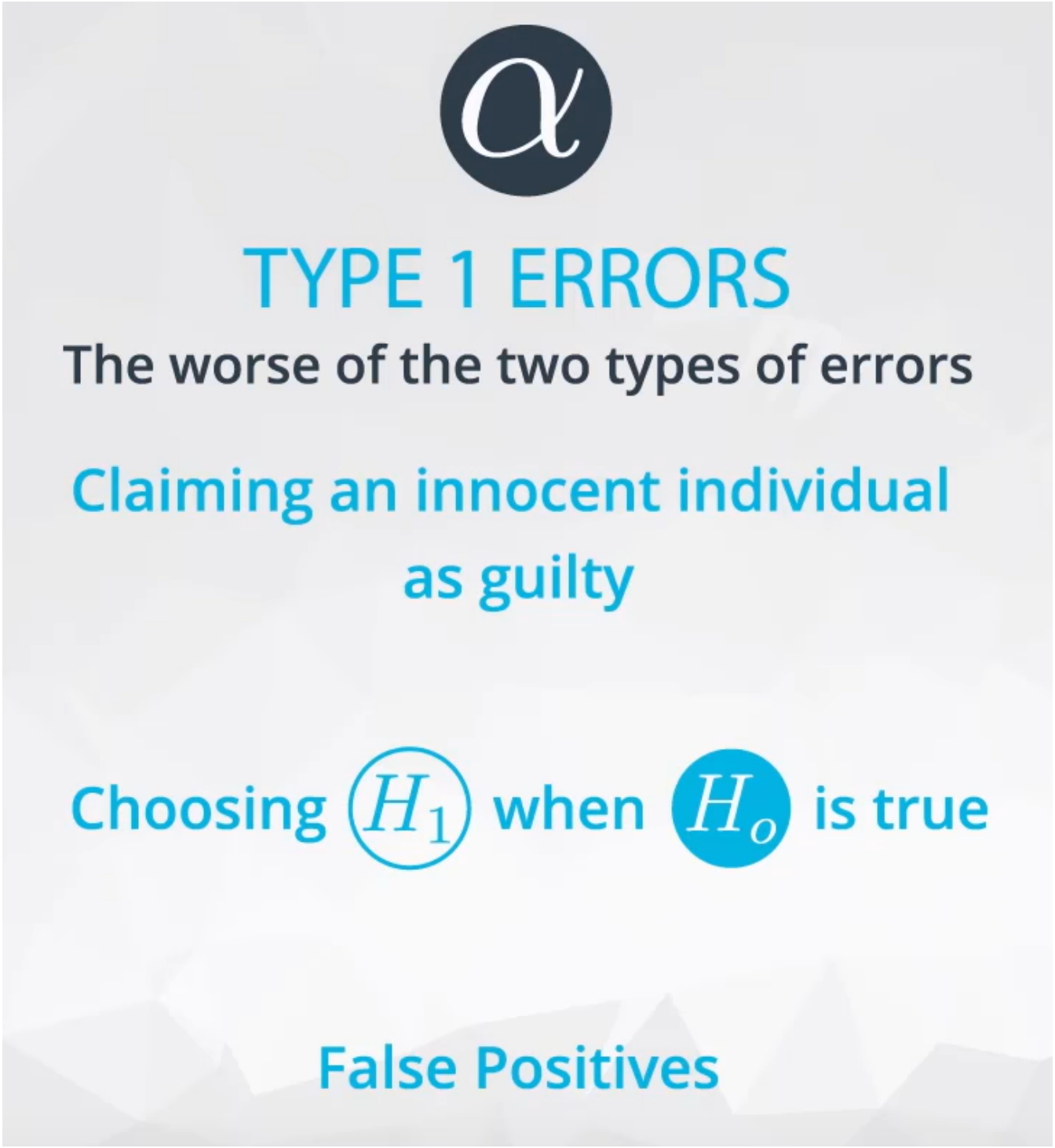
To better understand the null and alternative hypothesis matter, let's look at this real-world judicial example:



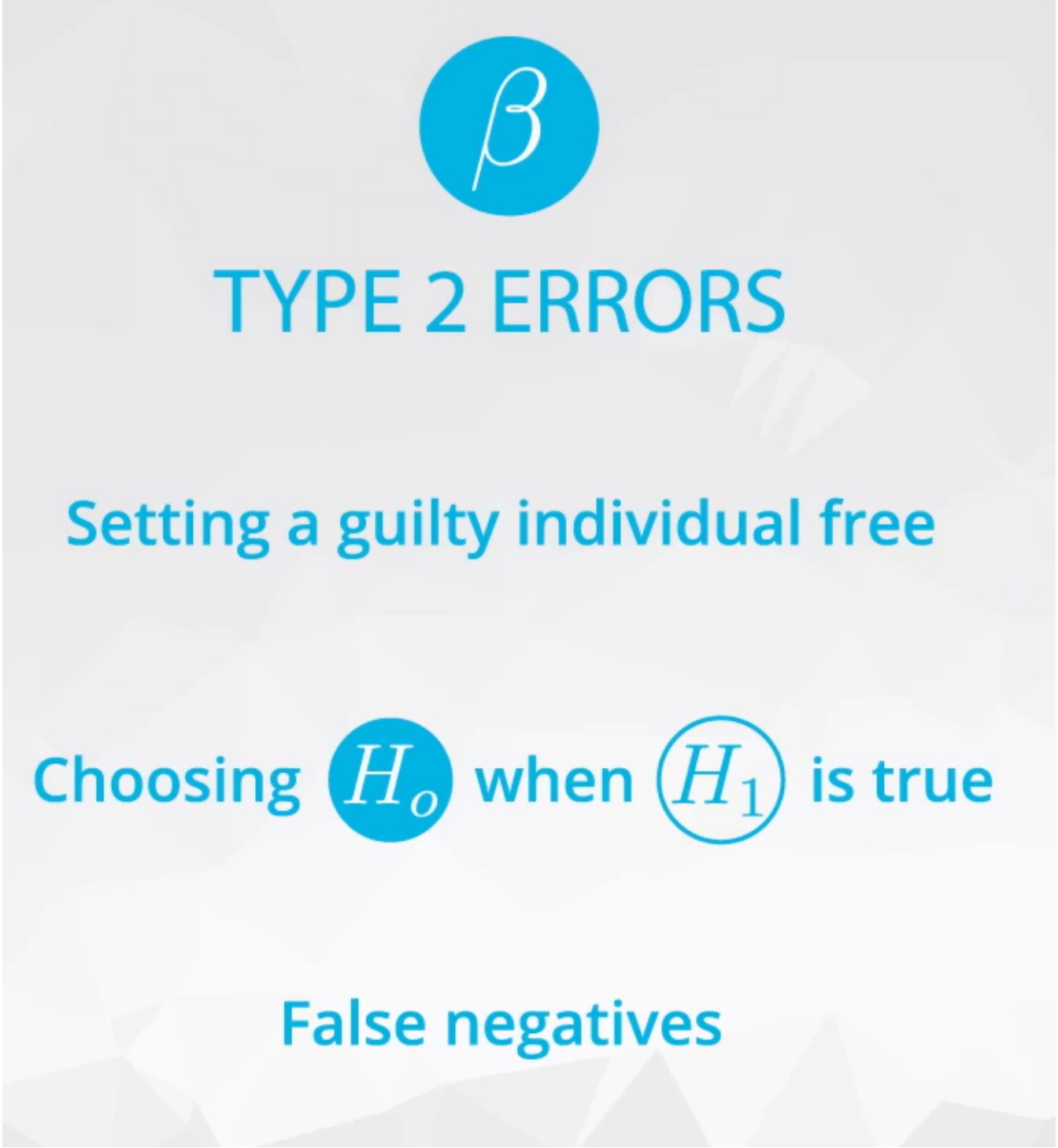
	Truly Guilty	Truly Innocent
Guilty Decision	Truly Guilty/Guilty Decision	Truly Innocent/Guilty Decision
Innocent Decision	Truly Guilty/Innocent Decision	Truly Innocent/Innocent Decision

In the first error, a truly guilty person goes free, and the person is not punished for something they did do.

A truly innocent person is found guilty and punished for something they did not do in the second error.



The greek symbol alpha is most frequently used to represent the type I error rate. Commonly these rates are 1-5%.



Beta is frequently used to identify a type II error rate. Frequently people will talk about the "power" of a statistical test as 1 - beta (or 1 minus the type two error rate). This is the ability of an individual to correctly choose the alternative hypothesis. There are a lot of ways to look at these errors, so it can be easy to get overwhelmed.

Hypothesis tests and Confidence Interval tell us about parameters, NOT statistics

Common Types of Hypothesis Tests

1. Testing a population mean (**One sample t-test**).
2. Testing the difference in means (**Two-sample t-test**)
3. Testing the difference before and after some treatment on the same individual (**Paired t-test**)
4. Testing a population proportion (**One sample z-test**)
5. difference between population proportions (**Two sample z-test**)

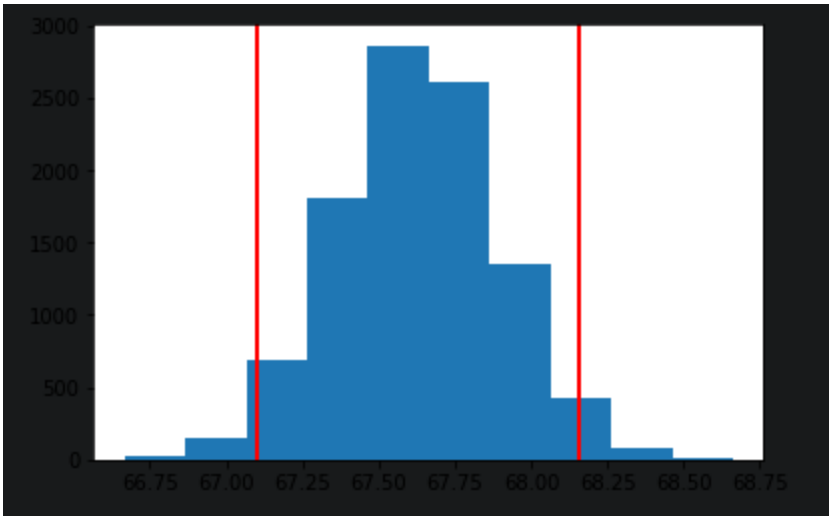
Using a Confidence Interval to Make a Decision

$$H_0 : \mu \leq 70$$
$$H_1 : \mu > 70$$

```
#Import libraries, set the seed, and read in the data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
np.random.seed(42)

df = pd.read_csv('coffee_dataset.csv')
```

```
#create a sample from our data
sample_df = df.sample(150)

#create our bootstrap sample
bootsample = sample_df.sample(150, replace=True)


means = []
for _ in range(10000):
    bootsample = sample_df.sample(150,replace=True)
    means.append(bootsample.height.mean())
```

```
low, high = np.percentile(means, 2.5), np.percentile(means,97.5)

plt.hist(means);
plt.axvline(x=low, color='r', linewidth=2);
plt.axvline(x=high, color='r', linewidth=2);
```



In the above case, our interval was entirely below 70, which would suggest the null (the population mean is less than 70) is actually true.

Using Standard Deviation of the Sampling Distribution to Make the Decision

$$H_0 : \mu \leq 70$$
$$H_1 : \mu > 70$$

```
#Import libraries, set the seed, and read in the data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
np.random.seed(42)

df = pd.read_csv('coffee_dataset.csv')
```

```
#create a sample from our data
sample_df = df.sample(150)
```

```
#create our bootstrap sample
bootsample = sample_df.sample(150, replace=True)
```

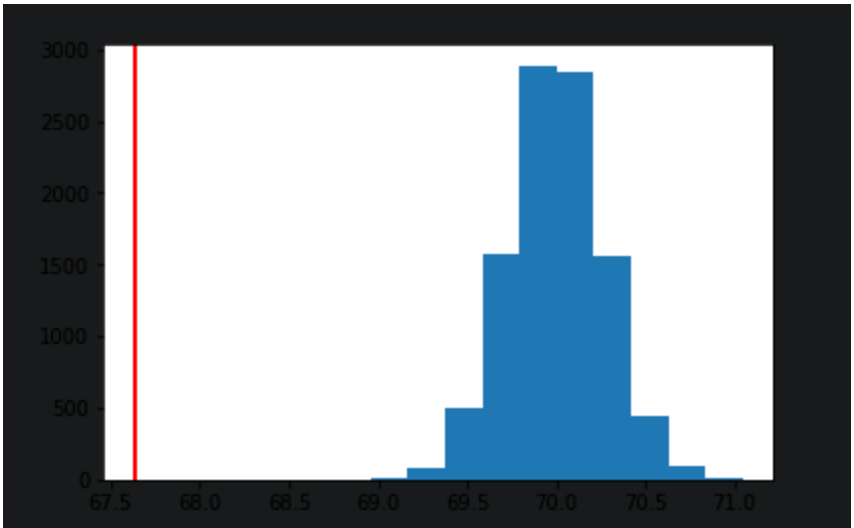
```
#Bootstrap from the sample
means = []
for _ in range(10000):
    bootsample = sample_df.sample(150,replace=True)
    means.append(bootsample.height.mean())
```

```
#Get the standard deviation
np.std(means)
```

```
0.26582463905558962
```

```
null_vals = np.random.normal(70, np.std(means),10000)
```

```
plt.hist(null_vals);
plt.axvline(sample_df.height.mean(),color='r', linewidth=2)
```




```
sample_df.height.mean()
```

```
67.632976882280587
```

In this case with our sample means so

far out in the tail, it is far enough that we donnot think it probably came from this null hypothesized value. However, since our null is that the population mean is less than or equal to 70, we do have evidence to support this claim with our sample mean of approximately 67. This would suggest not rejecting our one-sided null alternative.

If we had a null where we asked if the population mean was equal to 70, then we would reject this null in favor of an alternative that suggested the population mean was actually different from 70

Quiz: Simulating from the Null

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
np.random.seed(42)

full_data = pd.read_csv('coffee_dataset.csv')
sample_data = full_data.sample(200)
```

1. If you were interested in studying whether the average height for coffee drinkers is the same as for non-coffee drinkers, what would the null and alternative hypotheses be? Write them in the cell below, and use your answer to answer the first quiz question below.

Since there is no directional component associated with this statement, a not equal to seems most reasonable.

$$H_0 : \mu_{coff} - \mu_{no} = 0$$

$$H_1 : \mu_{coff} - \mu_{no} \neq 0$$

μ_{coff} and μ_{no} are the population mean values for coffee drinkers and non-coffee drinkers, respectivley.

2. If you were interested in studying whether the average height for coffee drinkers is less than non-coffee drinkers, what would the null and alternative be? Place them in the cell below, and use your answer to answer the second quiz question below.

In this case, there is a question associated with a direction - that is the average height for coffee drinkers is less than non-coffee drinkers. Below is one of the ways you could write the null and alternative. Since the mean for coffee drinkers is listed first here, the alternative would suggest that this is negative.

$$H_0 : \mu_{coff} - \mu_{no} \geq 0$$

$$H_1 : \mu_{coff} - \mu_{no} < 0$$

μ_{coff} and μ_{no} are the population mean values for coffee drinkers and non-coffee drinkers, respectivley.

3. For 10,000 iterations: bootstrap the sample data, calculate the mean height for coffee drinkers and non-coffee drinkers, and calculate the difference in means for each sample. You will want to have three arrays at the end of the iterations - one for each mean and one for the difference in means. Use the results of your sampling distribution, to answer the third quiz question below.

```
nocoff_means, coff_means, diffs = [], [], []

for _ in range(10000):
    bootsamp = sample_data.sample(200, replace = True)
    coff_mean = bootsamp[bootsamp['drinks_coffee'] == True]['height'].mean()
    nocoff_mean = bootsamp[bootsamp['drinks_coffee'] == False]['height'].mean()
    # append the info
    coff_means.append(coff_mean)
    nocoff_means.append(nocoff_mean)
    diffs.append(coff_mean - nocoff_mean)
```

```
np.std(nocoff_means) # the standard deviation of the sampling distribution for nocoff
```

```
0.40512631277475264
```

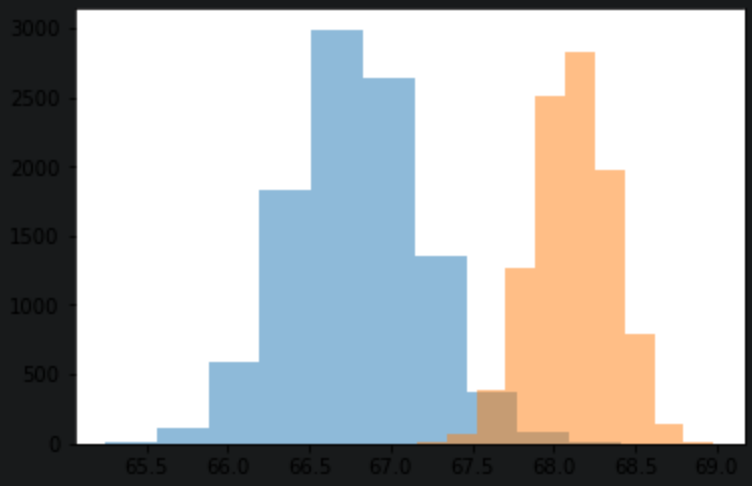
```
np.std(coff_means) # the standard deviation of the sampling distribution for coff
```

```
0.24073763373473001
```

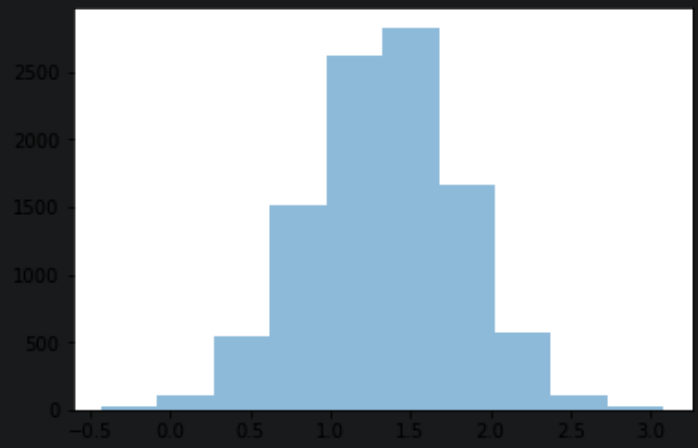
```
np.std(diffs) # the standard deviation for the sampling distribution for difference in means
```

```
0.46980910743871468
```

```
plt.hist(nocoff_means, alpha = 0.5);
plt.hist(coff_means, alpha = 0.5); # They look pretty normal to me!
```




```
plt.hist(diffs, alpha = 0.5); # again normal - this is by the central limit theorem
```

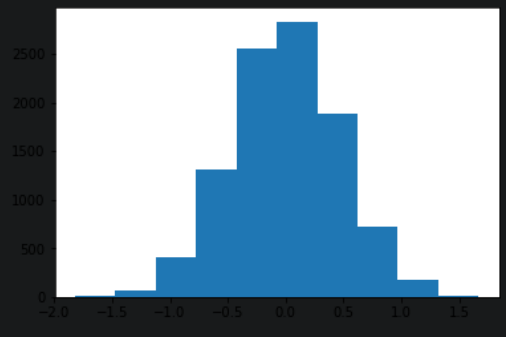


4. Now, use your sampling distribution for the difference in means and the docs to simulate what you would expect if your sampling distribution were centered on zero. Also, calculate the observed sample mean difference in `sample_data`. Use your solutions to answer the last questions in the quiz below.

** We would expect the sampling distribution to be normal by the Central Limit Theorem, and we know the standard deviation of the sampling distribution of the difference in means from the previous question, so we can use this to simulate draws from the sampling distribution under the null hypothesis. If there is truly no difference, then the difference between the means should be zero.**

```
null_vals = np.random.normal(0, np.std(diffs), 10000) # Here are 10000 draws from the sampling distribution under the nul
```

```
plt.hist(null_vals); #Here is the sampling distribution of the difference under the null
```



What is the shape of each sampling distribution?	Normal
What is the reason for the shape of each of these distributions?	The Central Limit Theorem

Quiz Question

Use your solutions about the sampling distribution, and what difference in means would be likely from the null hypothesis to mark all of the below that are true.

- [x] We can expect the standard deviation of the sampling distribution for differences from the null to be essentially the same as what we observed from the data.
- [x] If the null hypothesis is true, we would expect the difference in means of coffee drinkers and non-drinkers to be zero.

P-value (مقدار الصدفة)

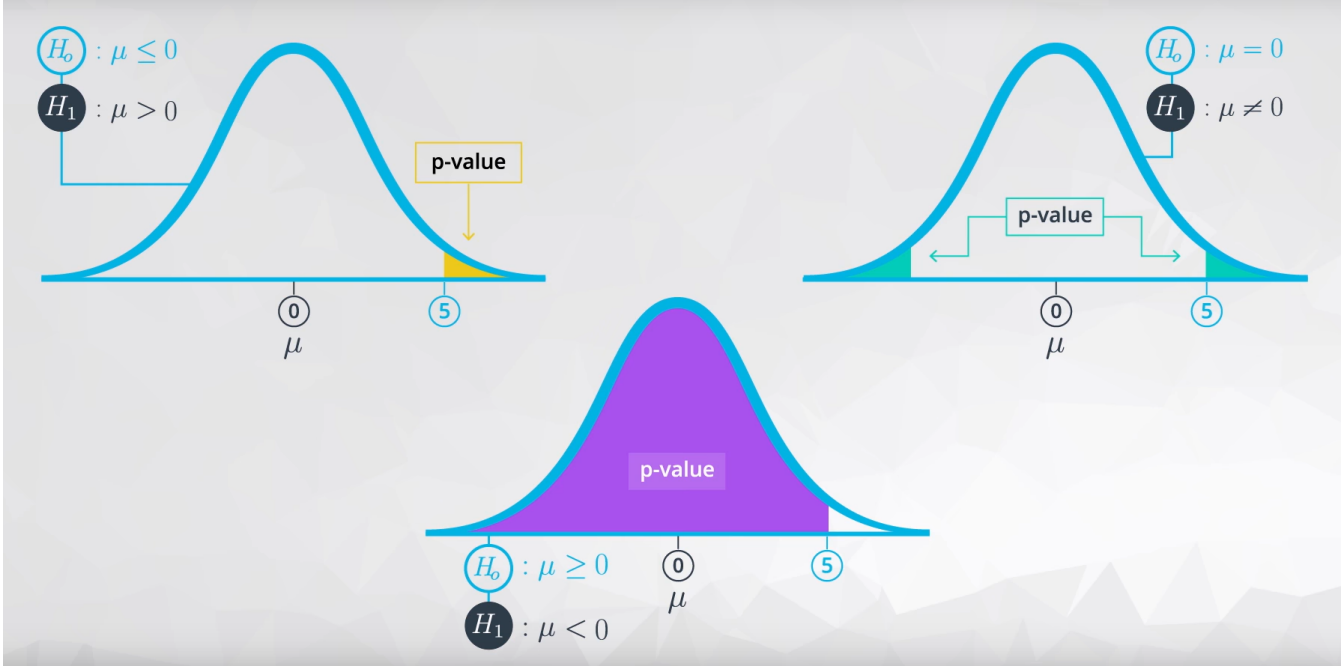
The definition of a p-value is the probability of observing your statistic (or one more extreme in favor of the alternative) if the null hypothesis is true.

The more extreme in favor of the alternative portion of this statement determines the shading associated with your p-value.

Understanding **p-values** involves

Sampling Distributions

Conditional Probability



If P-value $\leq \alpha$, then **reject** Ho.

If P-value $> \alpha$, then **fail to reject** Ho.

REMEMBER: fail to reject means the null hypothesis is correct

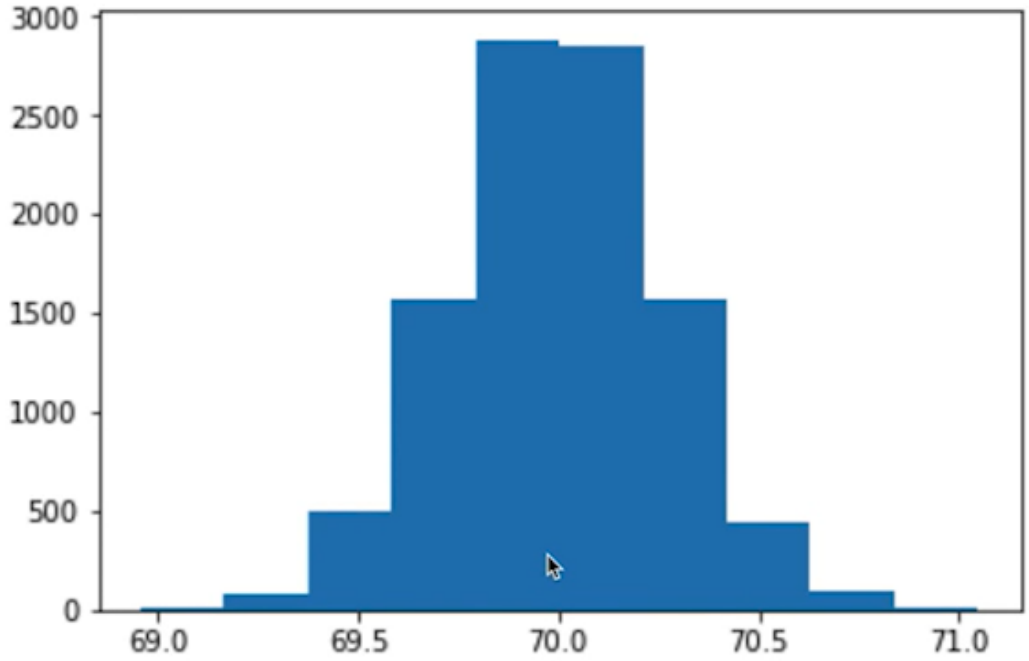
Calculating the p-value

In simulate draws from null hypothesis we have seen that, if our statistic was in the bulk of the distribution, this suggested that the statistic was likely from that null. However, if the statistic was farther out from the

bulk of the distribution, this suggested that the null wasn't likely to have generated our statistic.

```
null_vals = np.random.normal(70, np.std(means), 10000)
```

```
plt.hist(null_vals);
```




```
sample_df.height.mean()
```

```
67.632976882280587
```

REMEMBER: Large P-value suggests that we should not move away from the null hypothesis

IF our null & alternative hypothesis are:

`H0: u <= 70, H1: u > 70`

In[ ]: (null_vals > sample_mean).mean()

Out[ ]: 1.0

`Since this is one, our p-value is large and therefore we would not move away from the null hypothesis. That is mean we fail to reject the null hypothesis`

IF our null & alternative hypothesis are:

`H0: u >= 70, H1: u < 70`

In[ ]: (null_vals < sample_maen).mean()

Out[ ]: 0.0

`Since our output is 0, this suggest we reject the null hypothesis`

ternative hypothesis are:

`H0: u = 70, H1: u != 70`

In[ ]: null_mean = 70 (null_vals < sample_maen).mean() + (null_vals_mean + (null_mean - sample_mean)).mean()

Out[ ]: 0.0

`Since our output is 0, this suggest we reject the null hypothesis`

In [2]:
```
#imort the libraries, set the seed, and read in the data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
np.random.seed(42)

df= pd.read_csv('coffee_dataset.csv')
```

In [3]:
```
#create a simple from our data
sample_df = df.sample(150)
```

In [4]:
```
#create our bootstrap sample
bootsample = sample_df.sample(150, replace = True)
```

In [5]:
```
#Bootstrap from the sample
means = []
for _ in range (10000):
    bootsample = sample_df.sample(150, replace = True)
    means.append(bootsample.height.mean())
```

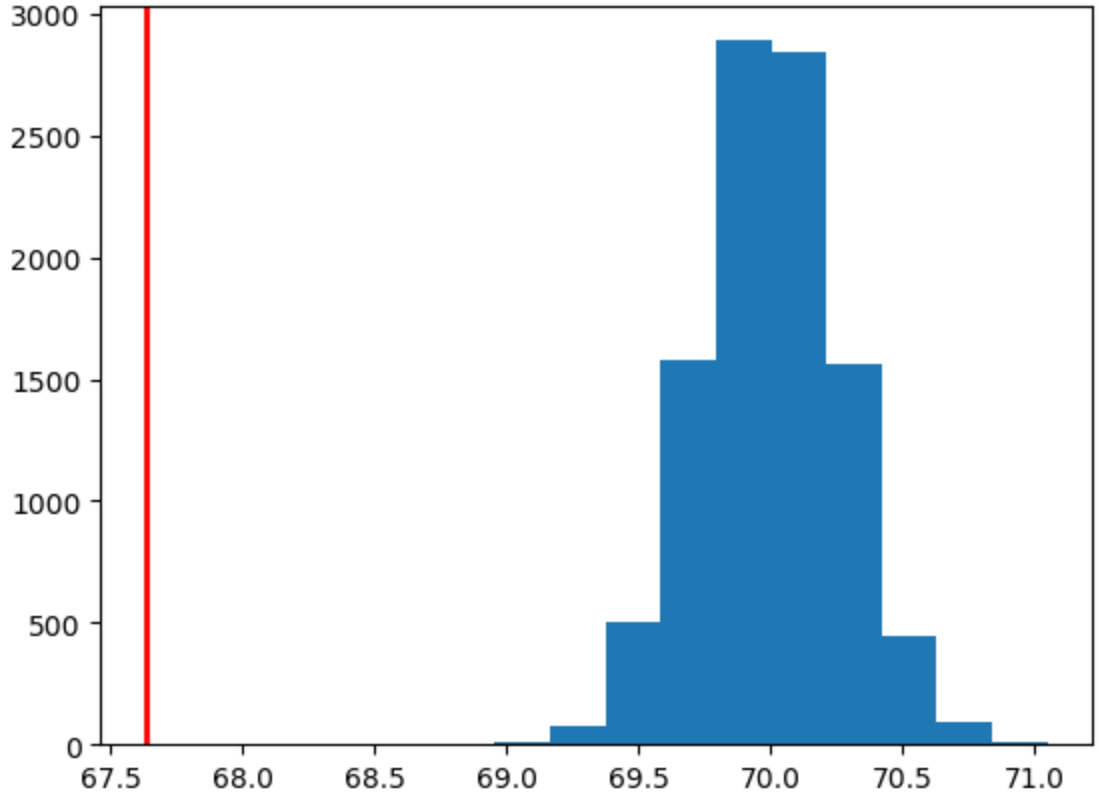
In [6]:
```
#Get the standard deviation
np.std(means)
```

Out[6]:
```
0.2658246390555901
```

In [7]:
```
null_vals = np.random.normal(70, np.std(means), 10000)
```

In [9]:
```
#Plot the null values
plt.hist(null_vals);
plt.axvline(sample_df.height.mean(), color = 'r', linewidth=2)
```

Out[9]:
```
<matplotlib.lines.Line2D at 0x1ec60e92df0>
```



In [10]:
```
#Find the sample mean
sample_mean = sample_df.height.mean()
```

In [12]:
```
#calculate the P-value
(null_vals > sample_mean).mean()
```

Out[12]:
```
1.0
```

Quiz Question

Using the same example from the first quiz, match each of the below alternative hypothesis with the way the distribution would be shaded. **The statistic we compute is the sample mean after the program minus the sample mean before the program.**

These are the correct matches.

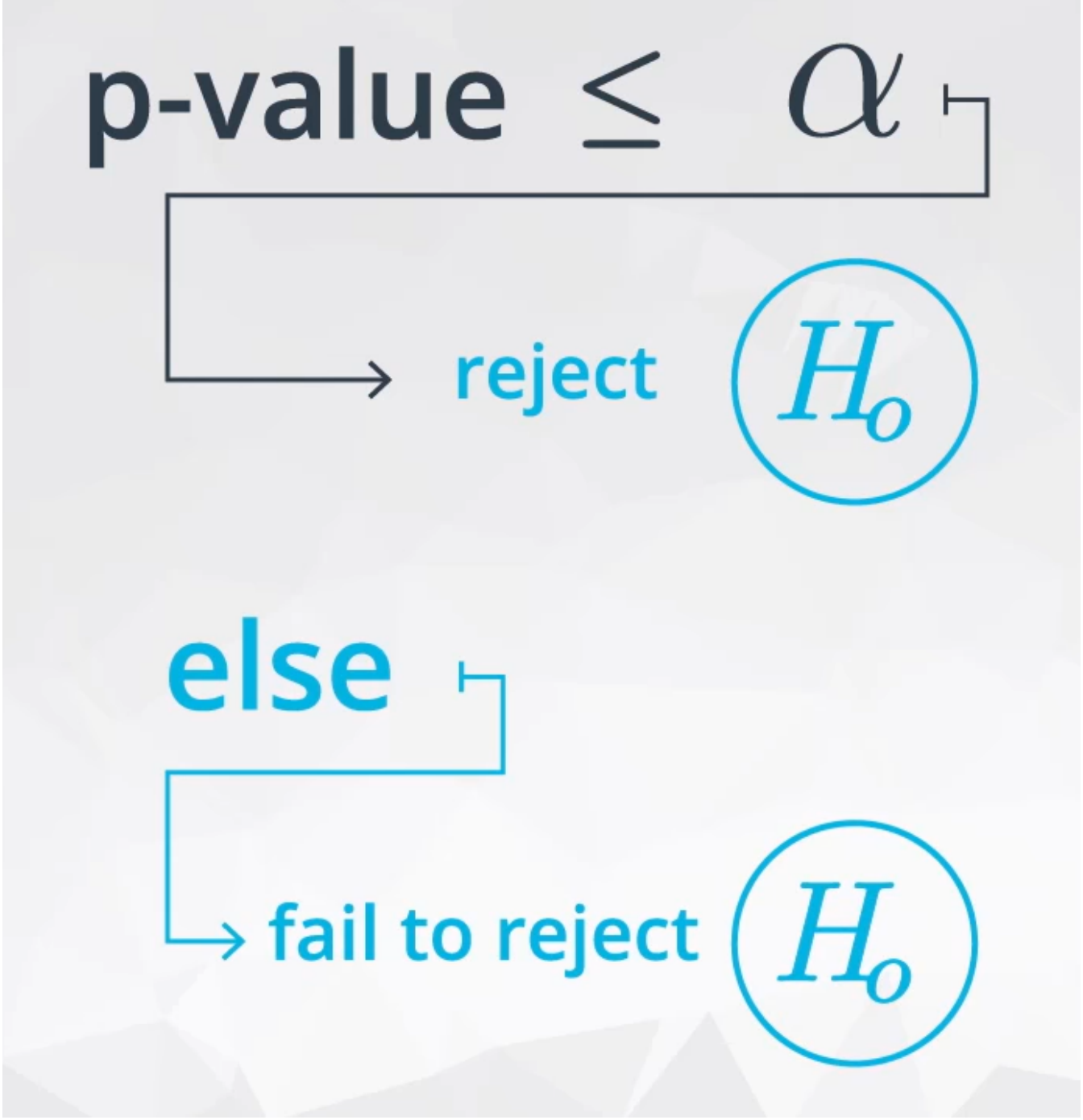
Alternative Hypothesis	Shading
There is no difference in the average morale.	Shade greater than the positive of our observed statistic and below the negative of our observed statistic to obtain the p-value.
The average morale after the program is higher than before the program.	Shade greater than the observed statistic to obtain the p-value.
The average morale after the program is lower than before the program.	Shade less than the observed statistic to obtain the p-value.

Connecting Errors and P-Values

The p-value is the probability of getting our statistic or a more extreme value if the null is true.

Therefore, small p-values suggest our null is not true. Rather, our statistic is likely to have come from a different distribution than the null.

When the p-value is large, we have evidence that our statistic was likely to come from the null hypothesis. Therefore, we do not have evidence to reject the null.



Quiz Question

For each p-value, alpha pair below: state whether you would reject the null or fail to reject the null.

These are the correct matches.

p-value/alpha	Conclusion
p-value = 0.03, alpha = 0.05	Reject the null
p-value = 0.20, alpha = 0.01	Fail to reject the null
p-value = 0.10, alpha = 0.05	Fail to reject the null

Conclusions in Hypothesis Testing

The word accept is one that is avoided when making statements regarding the null and alternative. You are not stating that one of the hypotheses is true. Rather, you are making a decision based on the likelihood of your data coming from the null hypothesis concerning your type I error threshold.

Therefore, the wording used in conclusions of hypothesis testing includes: We reject the null hypothesis, or We fail to reject the null hypothesis. This lends itself to the idea that you start with the null hypothesis true by default, and "choosing" the null at the end of the test would have been the choice even if no data were collected.

Quiz: Drawing Conclusions

```
import numpy as np
import pandas as pd

jud_data = pd.read_csv('judicial_dataset_predictions.csv')
par_data = pd.read_csv('parachute_dataset.csv')
```

```
jud_data.head()
```

	defendant_id	actual	predicted
0	22574	innocent	innocent
1	35637	innocent	innocent
2	39919	innocent	innocent
3	29610	guilty	guilty
4	38273	innocent	innocent

```
par_data.head()
```

	parachute_id	actual	predicted
0	3956	opens	opens
1	2147	opens	opens
2	2024	opens	opens
3	8325	opens	opens
4	6598	opens	opens

1. Above, you can see the actual and predicted columns for each of the datasets. Using the **jud_data**, find the proportion of errors for the dataset, and furthermore, the percentage of errors of each type. Use the results to answer the questions in quiz 1 below.

```
jud_data[jud_data['actual'] != jud_data['predicted']].shape[0]/jud_data.shape[0] # Number of errors
```

```
]: 0.042152958945489497
```

```
jud_data.query("actual == 'innocent' and predicted == 'guilty'").count()[0]/jud_data.shape[0] # Type 1 errors
```

```
]: 0.001510366607167376
```

```
jud_data.query("actual == 'guilty' and predicted == 'innocent'").count()[0]/jud_data.shape[0] # Type 2 errors
```

```
]: 0.040642592338322119
```

```
# If everyone was predicted to be guilty, then every actual innocent
# person would be a type I error.

# Type I = pred guilty, but actual = innocent
jud_data[jud_data['actual'] == 'innocent'].shape[0]/jud_data.shape[0]
```

```
]: 0.45159961554304545
```

```
#If everyone has prediction of guilty, then no one is predicted inncoent
#Therefore, there would be no type 2 errors in this case

# Type II errs = pred innocent, but actual = guilty
0
```

```
]: 0
```

2. Above, you can see the actual and predicted columns for each of the datasets. Using the **par_data**, find the proportion of errors for the dataset, and furthermore, the percentage of errors of each type. Use the results to answer the questions in quiz 2 below.

```
par_data[par_data['actual'] != par_data['predicted']].shape[0]/par_data.shape[0] # Number of errors
```

```
1]: 0.039972551037913875
```

```
par_data.query("actual == 'fails' and predicted == 'opens'").count()[0]/par_data.shape[0] # Type 1 errors
```

```
2]: 0.00017155601303825698
```

```
par_data.query("actual == 'opens' and predicted == 'fails'").count()[0]/par_data.shape[0] # Type 2 errors
```

```
3]: 0.039800995024875621
```

```
# If every parachute is predicted to fail, what is the proportion
# of type I errors made?

# Type I = pred open, but actual = fail
# In the above situation since we have none predicted to open,
# we have no type I errors

0
```

```
4]: 0
```

```
# If every parachute is predicted to fail, what is
# the proportion of Type II Errors made?

# This would just be the total of actual opens in the dataset,
# as we would label these all as fails, but actually they open

# Type II = pred fail, but actual = open
par_data[par_data['actual'] == 'opens'].shape[0]/par_data.shape[0]
```

```
5]: 0.9917653113741637
```

Quiz Question

Quiz 1: Use your results from the judicial dataset to match each description to the correct corresponding value.

These are the correct matches.

Description	Value
Total percentage of errors	4.2%
Percentage of Type I Errors	0.15%
Percentage of Type II Errors	4%
If everyone was predicted to be guilty, the percentage of Type I Errors made.	45%
If everyone was predicted to be guilty, the proportion of Type II Errors made.	0

Quiz Question

Quiz 2: Use your results from the parachute dataset to match each description to the correct corresponding value.

These are the correct matches.

Description	Value
Total percentage of errors	0.03997
Percentage of Type I Errors	0.00017
Percentage of Type II Errors	0.03980
If every parachute was predicted to not open, the proportion of Type I Errors made.	0
If every parachute was predicted to not open, the proportion of Type II Errors made.	0.99177

Impact of Large Sample Size

Hypothesis Testing vs. Machine Learning

With large sample sizes, hypothesis testing leads to even the smallest of findings as statistically significant. However, these findings might not be practically significant at all.

For example, imagine you find that statistically more people prefer beverage 1 to beverage 2 in a study of more than one million people. Based on this, you decide to open a shop to sell beverage 1. You then find out that beverage 1 is only more popular than beverage 2 by 0.0002% (but a statistically significant amount with your large sample size). Practically, maybe you should have opened a store that sold both.

Hypothesis testing takes an aggregate approach towards the conclusions made based on data, as these tests aim to understand population parameters (aggregate population values).

Alternatively, machine learning techniques take an individual approach towards making conclusions, as they attempt to predict an outcome for each specific data point.

What If We Test More Than Once?

When performing more than one hypothesis test, your type I error compounds. To correct this, a common technique is called the **Bonferroni correction**. This correction is very conservative but says that your new type I error rate should be the error rate you actually want to be divided by the number of tests you perform.

Therefore, if you would like to hold a type I error rate of 1% for each of the 20 hypothesis tests, the Bonferroni corrected rate would be 0.01/20 = 0.0005. This would be the new rate you should use to compare the p-value for each of the 20 tests to make your decision.

Other Techniques

Additional techniques to protect against compounding type I errors include:

1. Tukey correction
2. Q-values

You will learn that in ML

How Do CIs and HTs Compare?

CIs: Confidence Intervals

HTs: Hypothesis Tests

A two-sided hypothesis test (that is, a test involving a ≠ in the alternative) is the same in terms of the conclusions made as a confidence interval as long as:

$1-CI=\alpha$

For example, a 95% confidence interval will draw the same conclusions as a hypothesis test with a type I error rate of 0.05 in terms of which hypothesis to choose because:

$1-0.95=0.051$

assuming that the alternative hypothesis is a two-sided test.

Quiz : The Impact of Large Sample Sizes

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
np.random.seed(42)

full_data = pd.read_csv('coffee_dataset.csv')
```

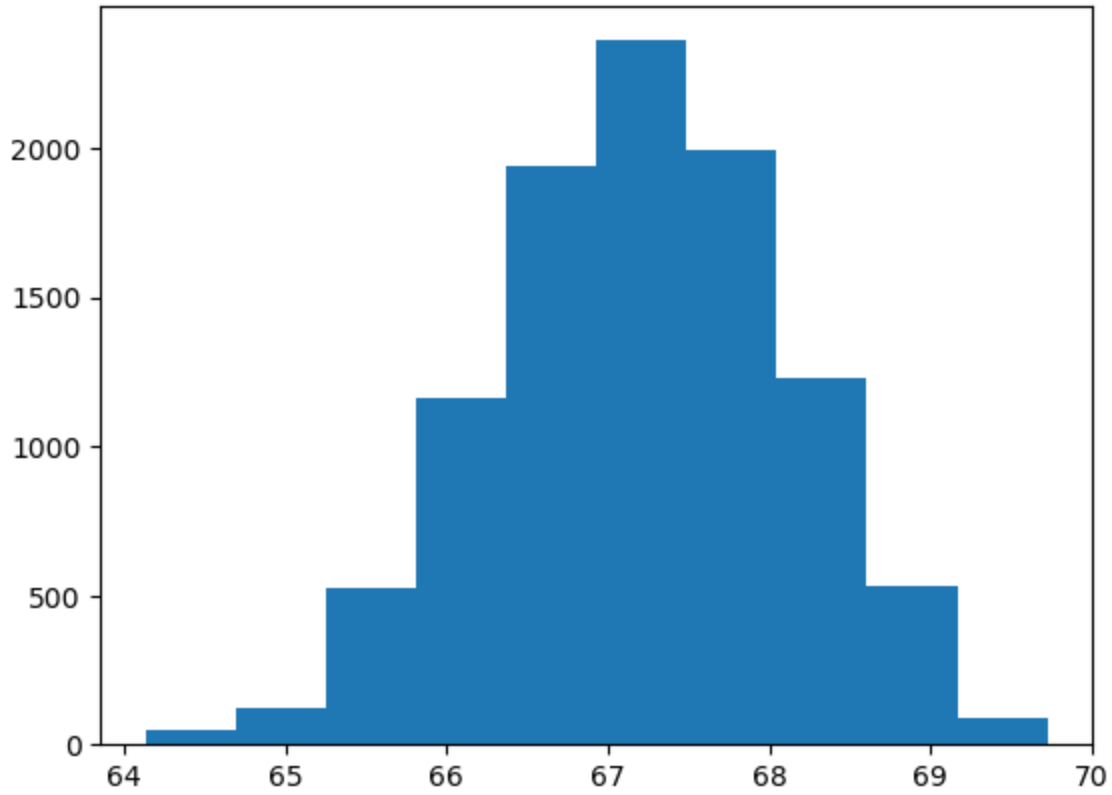
`1.` In this case, imagine we are interested in testing if the mean height of all individuals in `full_data` is equal to 67.60 inches or **different** than 67.60. First, use **quiz 1** below to state the null and alternative hypotheses for these cases in the cell below.

`2.` Find:

- What is the population mean height?
- What is the standard deviation of the population heights? Create a sample set of data using the code below.
- What is the sample mean height? Simulate the sampling distribution for the mean of five values to see the shape and plot a histogram.
- What is the standard deviation of the sampling distribution of the mean of five draws? Use **quiz 2** below to assure your answers are correct.

In [14]:
```
sample1 = full_data.sample(5, random_state=1)

pop_height_mean = np.round(full_data['height'].mean(), 4)
pop_height_std = np.round(full_data['height'].std(), 4)
sample1_height_mean = np.round(sample1['height'].mean(), 4)
sample1_height_std = np.round(sample1['height'].std(), 4)

print(f"Population mean height: {pop_height_mean}")
print(f"Standard deviation of population heights: {pop_height_std}")
print(f"Sample 1 mean height: {sample1_height_mean}")
print(f"Standard deviation of Sample 1 heights: {sample1_height_std}")

# plot histogram of sample heights
sampling_dist_mean5 = []

for _ in range(10000):
    sample_of_5 = sample1.sample(5, replace = True)
    sample_mean = sample_of_5.height.mean()
    sampling_dist_mean5.append(sample_mean)

std_sampling_dist = np.round(np.std(sampling_dist_mean5),4)
print(f"Standard deviation of the sampling distribution: {std_sampling_dist}")
plt.hist(sampling_dist_mean5);
```

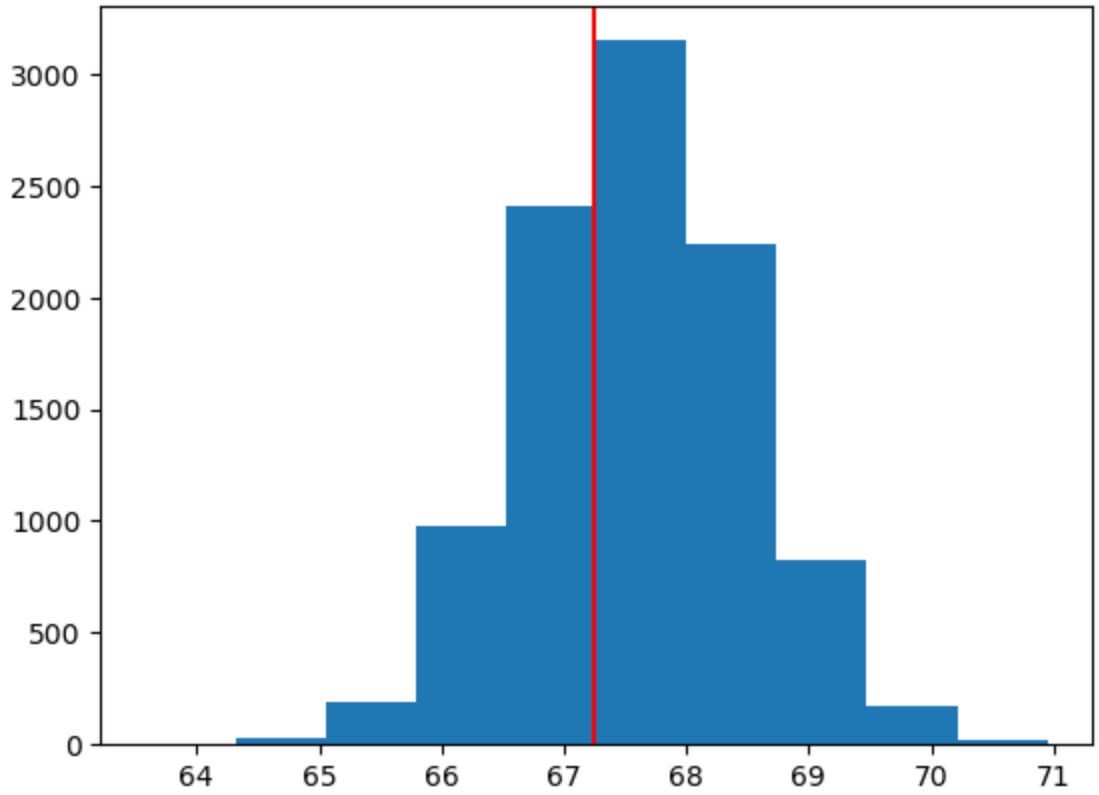
```
Population mean height: 67.5975
Standard deviation of population heights: 3.1194
Sample 1 mean height: 67.2522
Standard deviation of Sample 1 heights: 2.2575
Standard deviation of the sampling distribution: 0.8992
```



3. Using the null and alternative hypotheses set up in question 1 and the results of your sampling distribution in question 2, simulate the mean values you would expect from the null hypothesis. Use these simulated values to determine a p-value to make a decision about your null and alternative hypotheses. Check your solution using **quiz 3** and **quiz 4** below.

Hint: Use the numpy documentation here to assist with your solution.

In [15]:
```
null_mean = 67.5975
null_vals = np.random.normal(null_mean, std_sampling_dist, 10000)

plt.hist(null_vals);
plt.axvline(x=sample1.height.mean(), color = 'red'); # where our sample mean falls on nu
```



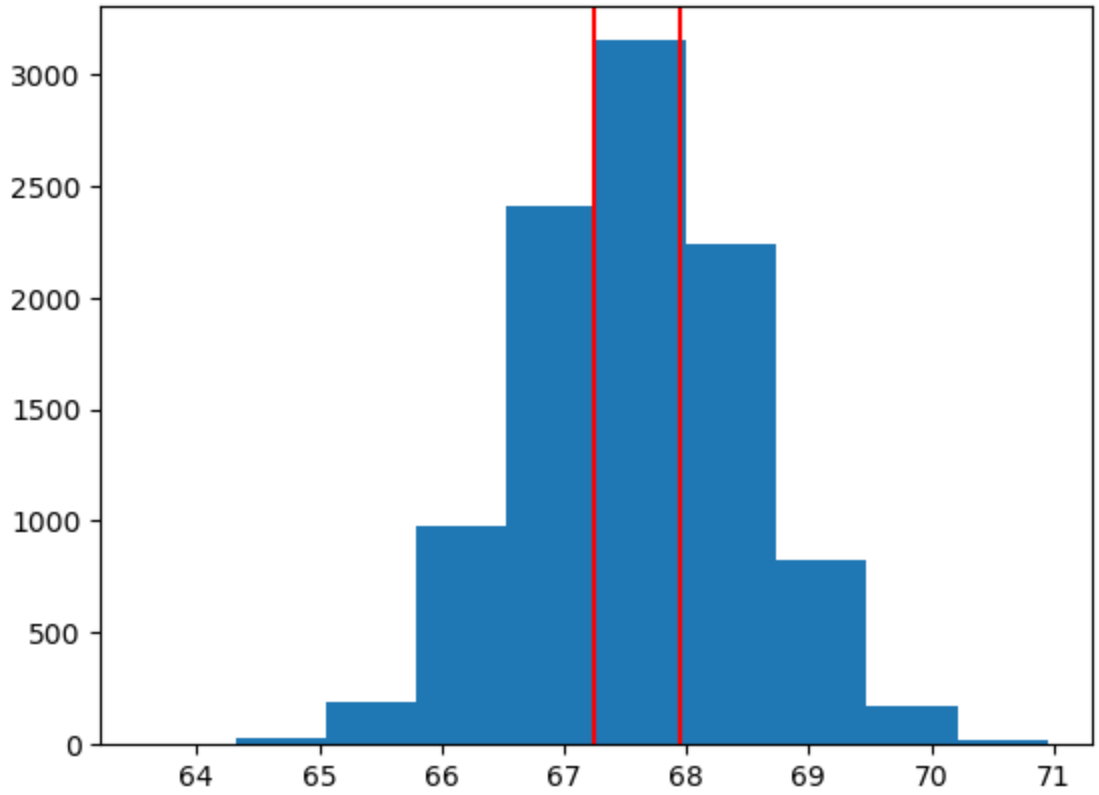
In [16]:
```
# for a two sided hypothesis, we want to look at anything
# more extreme from the null in both directions
obs_mean = sample1.height.mean()

# probability of a statistic higher than observed
prob_more_extreme_high = (null_vals > obs_mean).mean()

# probability a statistic is more extreme lower
prob_more_extreme_low = (null_mean - (obs_mean - null_mean) > null_vals).mean()

pval = prob_more_extreme_low + prob_more_extreme_high
pval = np.round(pval, 4)
print(pval)
```

```
1.2952
```

The above shows a second possible method for obtaining the p-value. These are pretty different, stability of these values with such a small sample size is an issue. We are essentially shading outside the lines below.

In [17]:
```
upper_bound = obs_mean
lower_bound = null_mean - (obs_mean - null_mean)
print(upper_bound, lower_bound)

plt.hist(null_vals);
plt.axvline(x=lower_bound, color = 'red'); # where our sample mean falls on null dist
plt.axvline(x=upper_bound, color = 'red'); # where our sample mean falls on null dist
```

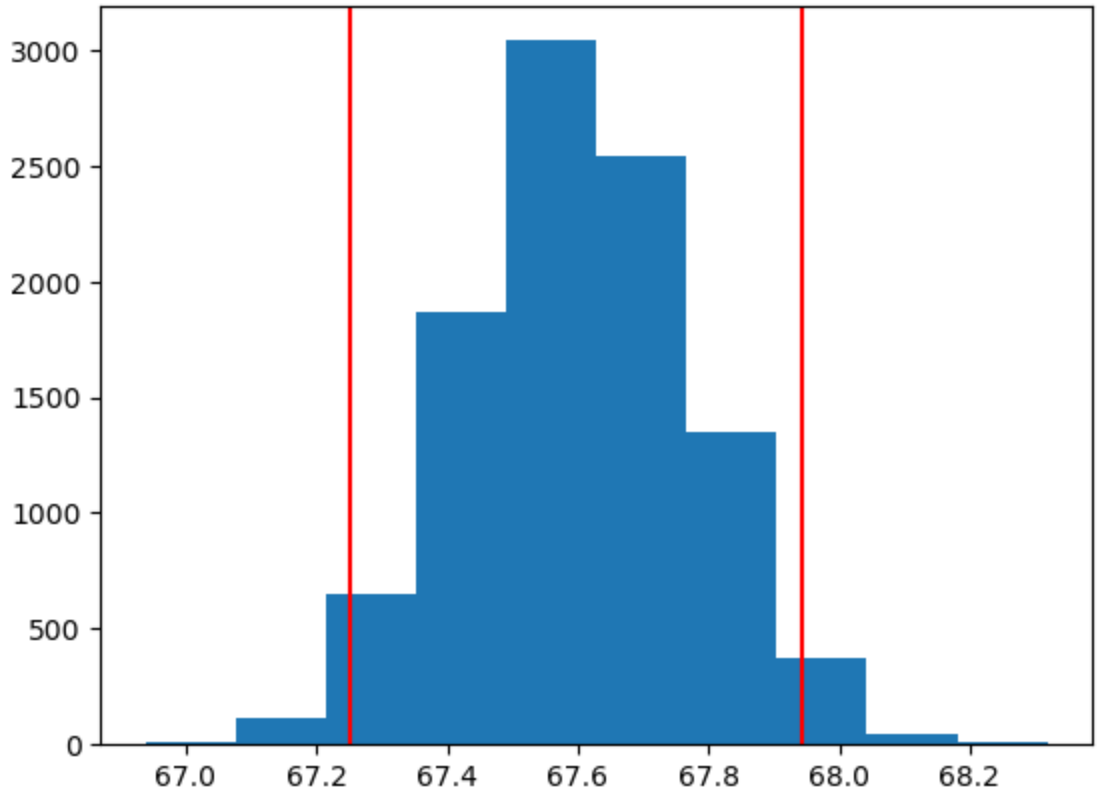
```
67.25218816122813 67.94281183877186
```



4. Now imagine if you received the same sample mean as you calculated from the sample in question 1 above, but that you actually retrieved it from a sample of 300. What would the new standard deviation be for your sampling distribution for the mean of 300 values? Additionally, what would your new p-value be for choosing between the null and alternative hypotheses you set up? Simulate the sampling distribution for the mean of five values to see the shape and plot a histogram. Use your solutions here to answer the second to last quiz question below.

Hint: If you get stuck, notice you can use the solution from quiz regarding finding the p-value earlier to assist with obtaining this answer with just a few small changes.

In [18]:
```
sample2 = full_data.sample(300)
# obs_mean = sample2.height.mean()

sampling_dist_mean300 = []

for _ in range(10000):
    sample_of_300 = sample2.sample(300, replace = True)
    sample_mean = sample_of_300.height.mean()
    sampling_dist_mean300.append(sample_mean)

std_sampling_dist300 = np.std(sampling_dist_mean300)
null_vals = np.random.normal(null_mean, std_sampling_dist300, 10000)

upper_bound = obs_mean
lower_bound = null_mean - (obs_mean - null_mean)

plt.hist(null_vals);
plt.axvline(x=lower_bound, color = 'red'); # where our sample mean falls on null dist
plt.axvline(x=upper_bound, color = 'red'); # where our sample mean falls on null dist
```



5. Reflect on what happened by answering the final quiz in this concept.

Quiz Question

Use the first statement in the notebook above to set up the null and alternative hypotheses.

These are the correct matches.

Hypotheses	Statement
Null	The mean height of everyone in the population is equal to 67.60 inches.
Alternative	The mean height of everyone in the population is different from 67.60 inches.

Quiz Question

Based on the p-value retrieved using the sample of size 5 and any reasonable type I error rate (say 5%),

- ○ We have statistically significant evidence to suggest the population mean is not equal to 67.60 inches. (Reject the null)
- ◉ We do not have statistically significant evidence to suggest the population mean is different from 67.60 inches. (Fail to reject the null) ✓
- ○ We do not have enough data to draw any sort of conclusion.

Quiz Question

In this concept the big takeaway is:

- ○ Statistics are confusing.
- ◉ Even the smallest of differences between a sample mean and a hypothesized population mean are significant when we have large sample sizes. ✓
- ○ With large sample sizes, we can do anything we want!
- ○ Sampling distributions are the distribution of a sample statistic, and the distribution of a sample mean has a standard deviation of the population divided by the square root of the sample size.

Quiz: Multiple Tests

In [19]:
```
import numpy as np
import pandas as pd

df = pd.read_csv('judicial_dataset_pvalues.csv')
df.head()
```

Out[19]:

	defendant_id	actual	pvalue
0	22574	innocent	0.294126
1	35637	innocent	0.417981
2	39919	innocent	0.177542
3	29610	guilty	0.015023
4	38273	innocent	0.075371

`1.` Remember back to the null and alternative hypotheses for this example. Use that information to determine the answer for **Quiz 1** and **Quiz 2** below.

A p-value is the probability of observing your data or more extreme data, if the null is true. Type I errors are when you choose the alternative when the null is true, and vice-versa for Type II. Therefore, deciding an individual is guilty when they are actually innocent is a Type I error. The alpha level is a threshold for the percent of the time you are willing to commit a Type I error.

`2.` If we consider each individual as a single hypothesis test, find the conservative Bonferroni corrected p-value we should use to maintain a 5% type I error rate.

In [20]:
```
bonf_alpha = 0.05/df.shape[0]
bonf_alpha
```

Out[20]:
```
6.86530275985171e-06
```

`3.` What is the proportion of type I errors made if the correction isn't used? How about if it is used?

Use your answers to find the solution to **Quiz 3** below.

In order to find the number of type I errors made without the correction - we need to find all those that are actually innocent with p-values less than 0.05.

In [21]:
```
df.query("actual == 'innocent' and pvalue < 0.05").count()[0]/df.shape[0] # If not used
```

Out[21]:
```
0.001510366607167376
```

In [22]:
```
df.query("actual == 'innocent' and pvalue < @bonf_alpha").count()[0]/df.shape[0] # If us
```

Out[22]:
```
0.0
```

`4.` Think about how hypothesis tests can be used, and why this example wouldn't exactly work in terms of being able to use hypothesis testing in this way. Check your answer with **Quiz 4** below.

This is looking at individuals, and that is more of the aim for machine learning techniques. Hypothesis testing and confidence intervals are for population parameters. Therefore, they are not

meant to tell us about individual cases, and we wouldn't obtain p-values for individuals in this way. We could get probabilities, but that isn't the same as the probabilities associated with the relationship to sampling distributions as you have seen in these lessons.

Quiz Question

Quiz 1: What does a p-value signify in this example?

- ○ The probability of an individual being guilty.
- ○ The probability of an individual being innocent.
- ○ The probability of us deciding an individual is guilty when they are actually innocent.
- ◉ The probability of us observing the facts about an individual's case that are in favor of them being guilty, assuming they are truly innocent. ✓

Quiz Question

Quiz 3:

✓ These are the correct matches.

Description	Value
The new Type I Error rate after the Bonferroni correction.	0.0000069
The proportion of p-values less than the Bonferroni corrected Type I Error rate.	0
The proportion of Type I Errors committed if the Bonferroni correction is used.	0
The proportion of Type I Errors committed if the Bonferroni correction isn't used.	0.002

Quiz Question

Quiz 4: Why is this not how multiple comparison tests work in practice?

- ○ The updated Bonferroni rate is clearly smaller than any rate we would see in the real world.
- ○ The alpha rate starting at 0.05 is too low to begin with.
- ○ We cannot associate a p-value with the real world of judicial cases.
- ◉ Hypothesis testing is used on parameters. This is looking at individuals in a population, not aggregates. ✓

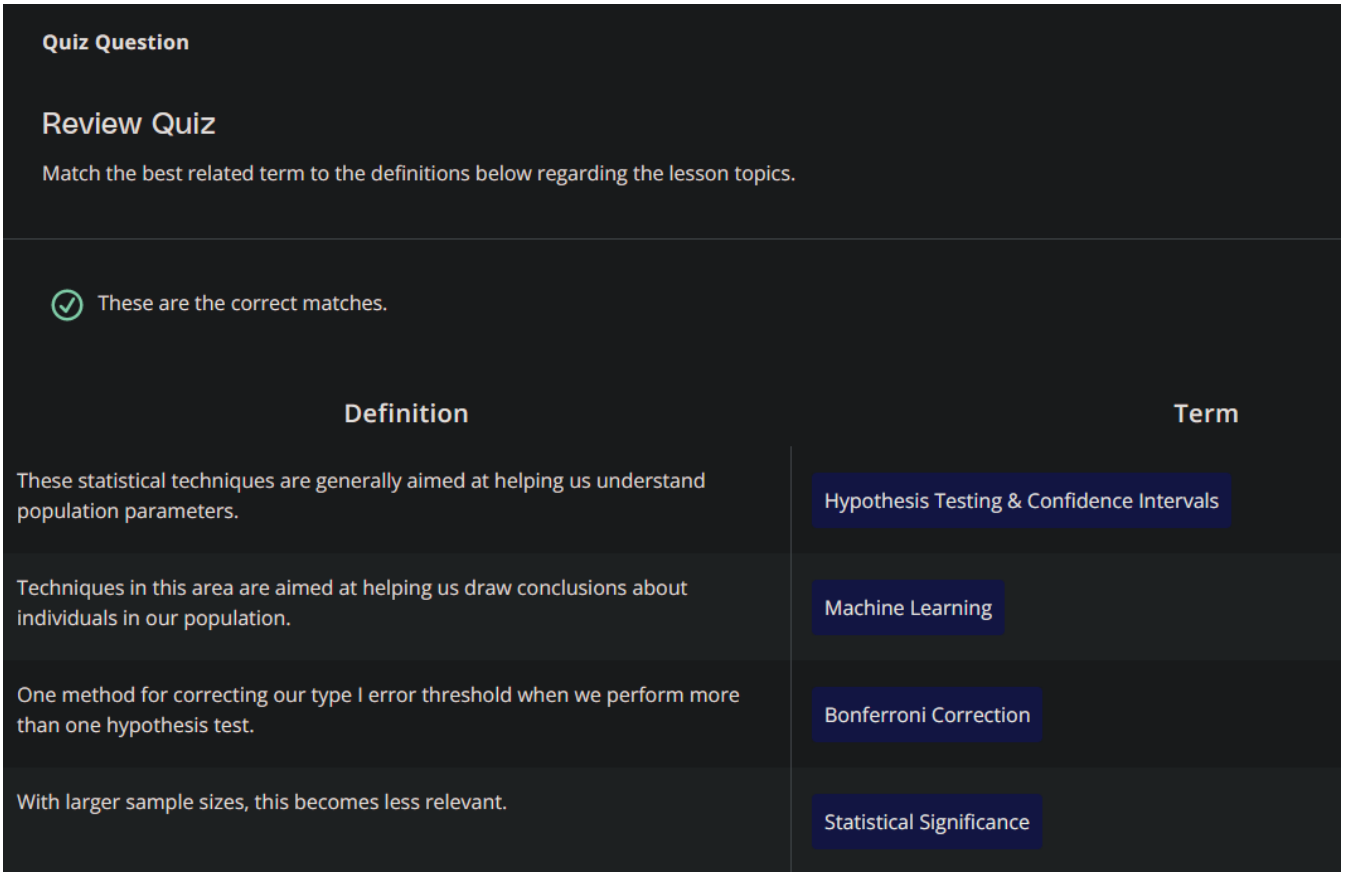
Hypothesis testing is used on aggregates - population parameters. Performing hypothesis tests for any one individual is one of the things noted in an earlier video. A better use case here would be logistic regression, which you will see in a later lesson.

Hypothesis Testing Conclusion

That was a ton. You learned:

- How to set up hypothesis tests. You learned the null hypothesis is what we assume to be true before we collect any data, and the alternative is usually what we want to try and prove to be true.
- You learned about Type I and Type II errors. You learned that Type I errors are the worst type of errors, and these are associated with choosing the alternative when the null hypothesis is actually true.
- You learned that p-values are the probability of observing your data or something more extreme in favor of the alternative given the null hypothesis is true. You learned that using a confidence interval from the bootstrapping samples, you can essentially make the same decisions as hypothesis testing (without all the confusion of p-values).
- You learned how to make decisions based on p-values. That is, if the p-value is less than your Type I error threshold, then you have evidence to reject the null and choose the alternative. Otherwise, you fail to reject the null hypothesis.
- You learned that when sample sizes are enormous, everything appears statistically significant (that is, you end up rejecting essentially every null), but these results may not be practically significant.
- You learned that when performing multiple hypothesis tests, your errors will compound. Therefore, using some corrections to maintain your true Type I error rate is important. A simple but very conservative approach is to use what is known as a Bonferroni correction, which says you should divide your $\alpha\alpha$ level (or Type I error threshold) by the number of tests performed.

This lesson is often the most challenging for students throughout the entire course. To really have the ideas here stick, it can help to put them down in your own words.



Case Study: A/B Tests

After completing this case study, you'll be able to:

- Apply confidence intervals and hypothesis testing to real-world scenarios
- Analyze results from A/B testing

A/B tests test changes on a web page by running an experiment where a control group sees the old version while the experiment group sees the new version. A metric is then chosen to measure the level of engagement from users in each group. These results are then used to judge whether one version is more effective than the other. A/B testing is very much like hypothesis testing with the following hypotheses:

- Null Hypothesis: The new version is no better, or even worse, than the old version
- Alternative Hypothesis: The new version is better than the old version

If we fail to reject the null hypothesis, the results would suggest keeping the old version. If we reject the null hypothesis, the results would suggest launching the change. These tests can be used for a wide variety of changes, from large feature additions to small adjustments in color, to see what change maximizes your metric the most.

A/B testing also has its drawbacks. It can help you compare two options, but it can't tell you about an option you haven’t considered. It can also produce bias results when tested on existing users due to factors like change aversion and novelty effect.

- Change Aversion: Existing users may give an unfair advantage to the old version simply because they are unhappy with the change, even if it’s ultimately for the better.
- Novelty Effect: Existing users may give an unfair advantage to the new version because they’re excited or drawn to the change, even if it isn’t any better in the long run. You'll learn more about factors like these later.

Business Example

In this case study, you’ll analyze the A/B test results for Audacity. Here's the customer funnel for typical new users on their site:

View home page > Explore courses > View course overview page > Enroll in course > Complete course

Audacity loses users as they go down the stages of this funnel, with only a few making it to the end. To increase student engagement, Audacity is performing A/B tests to try out changes that will hopefully increase conversion rates from one stage to the next.

We’ll analyze test results for two changes they have in mind and then make a recommendation on whether they should launch each change.

Experiment I

The first change Audacity wants to try is on their homepage. They hope that this new, more engaging design will increase the number of users who explore their courses and move on to the second stage of the funnel.

The metric we will use is the click-through rate for the Explore Courses button on the home page. Click-through rate (CTR) is often defined as the number of clicks divided by the number of views. Since Audacity uses cookies, we can identify unique users and make sure we don't count the same one multiple times. For this experiment, we'll define our click-through rate as:

CTR: # clicks by unique users / # views by unique users

Now that we have our metric, let's set up our null and alternative hypotheses:

H0:CTRnew≤CTRold

H1:CTRnew>CTRold

Our alternative hypothesis is what we want to prove to be true. In this case, the new homepage design has a higher click-through rate than the old homepage design. And the null hypothesis is what we assume to be true before analyzing data, which is that the new homepage design has a click-through rate that is less than or equal to that of the old homepage design. As you've seen before, we can rearrange our hypotheses to look like this:

H0:CTRnew−CTRold≤0 H1:CTRnew−CTRold>0

In [34]:
```
import numpy as np, pandas as pd, matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('homepage_actions.csv')
df.head()
```

Out[34]:

	timestamp	id	group	action
0	2016-09-24 17:42:27.839496	804196	experiment	view
1	2016-09-24 19:19:03.542569	434745	experiment	view
2	2016-09-24 19:36:00.944135	507599	experiment	view
3	2016-09-24 19:59:02.646620	671993	control	view
4	2016-09-24 20:26:14.466886	536734	experiment	view

1. Match the following characteristics of this dataset:

- total number of actions
- number of unique users
- sizes of the control and experiment groups (i.e., the number of unique users in each group)

In [24]:
```
# total number of actions
df.shape[0]
```

Out[24]:
```
8188
```

In [25]:
```
# number of unique users
df.nunique()
```

Out[25]:
```
timestamp    8188
id           6328
group           2
action          2
dtype: int64
```

In [31]:
```
# size of control group and experiment group (The uniques)
control_size = df.query("group == 'control'").nunique()
experiment_size = df.query("group == 'experiment'").nunique()
print("Cotrol Size is: {}".format(control_size))
print("Experiment Size is: {}".format(experiment_size))

#you can use:
df.groupby('group').nunique()
```

```
Cotrol Size is: timestamp    4264
id           3332
group           1
action          2
dtype: int64
Experiment Size is: timestamp    3924
id           2996
group           1
action          2
dtype: int64
```

Out[31]:

group	timestamp	id	action
control	4264	3332	2
experiment	3924	2996	2

2. How long was the experiment run for?

Hint: the records in this dataset are ordered by timestamp in increasing order

In [27]:
```
# duration of this experiment
df['timestamp'] = pd.to_datetime(df['timestamp'])
min_time = df['timestamp'].min()
max_time = df['timestamp'].max()
experiment_duration = max_time - min_time
experiment_duration
```

Out[27]:
```
Timedelta('115 days 16:41:40.789831')
```

3. What action types are recorded in this dataset?

(i.e., What are the unique values in the action column?)

In [32]:
```
# action types in this experiment
df.action.value_counts()
```

Out[32]:
```
view     6328
click    1860
Name: action, dtype: int64
```

4. Does the experiment page drive higher traffic than the control page?

In [35]:
```
control_df = df.query('group == "control"')
```

In [36]:
```
control_ctr = control_df.query('action == "click"').id.nunique() / control_df.query('act
```

In [37]:
```
control_ctr
```

Out[37]:
```
0.2797118847539016
```

In [38]:
```
experiment_df = df.query('group == "experiment"')
```

In [39]:
```
experiment_ctr = experiment_df.query('action == "click"').id.nunique() / experiment_df.q
```

In [40]:
```
experiment_ctr
```

Out[40]:
```
0.3097463284379172
```

computed the observed difference between the metric, click-through rate, for the control and experiment groups.

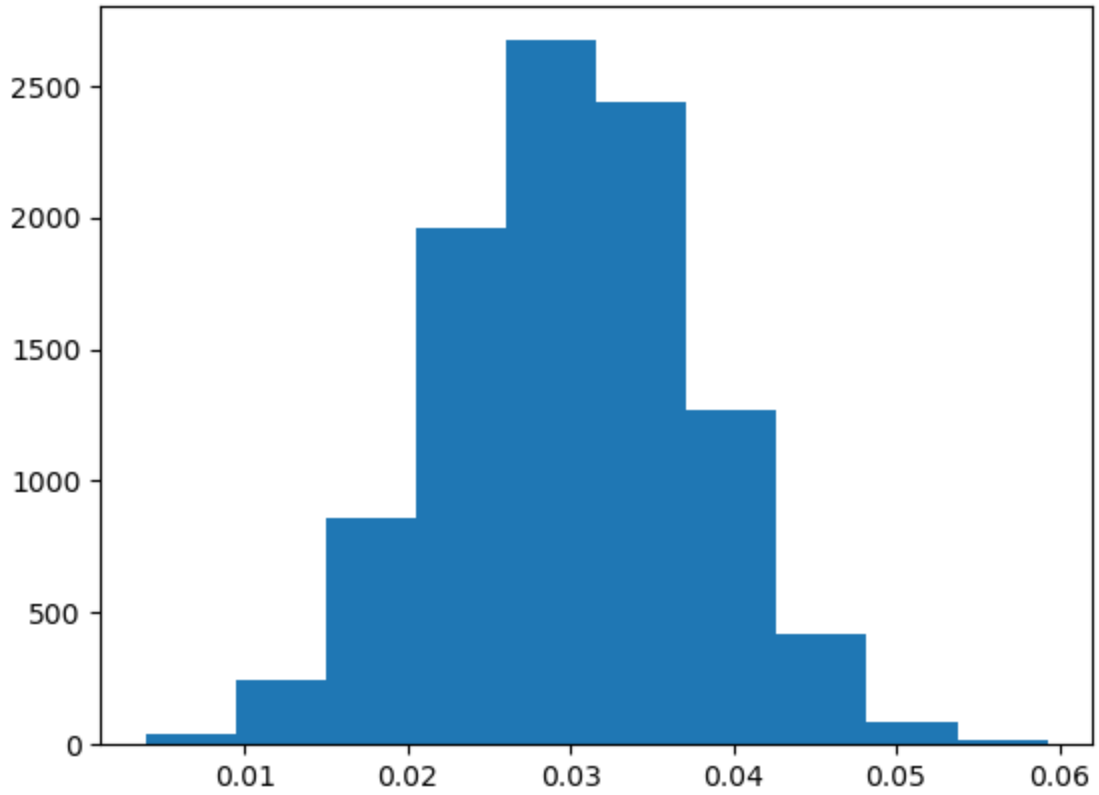
In [41]:
```
obs_diff = experiment_ctr - control_ctr
obs_diff
```

Out[41]:
```
0.030034443684015644
```

simulated the sampling distribution for the difference in proportions (or difference in click-through rates).

In [42]:
```
#Bootstrap
diffs = []
for _ in range(10000):
    b_samp = df.sample(df.shape[0], replace = True)
    control_df = b_samp.query('group == "control"')
    experiment_df = b_samp.query('group == "experiment"')
    control_ctr = control_df.query('action == "click"').id.nunique() / control_df.query(
    expermiment_ctr = experiment_df.query('action == "click"').id.nunique() / experiment
    diffs.append(experiment_ctr-control_ctr)
```

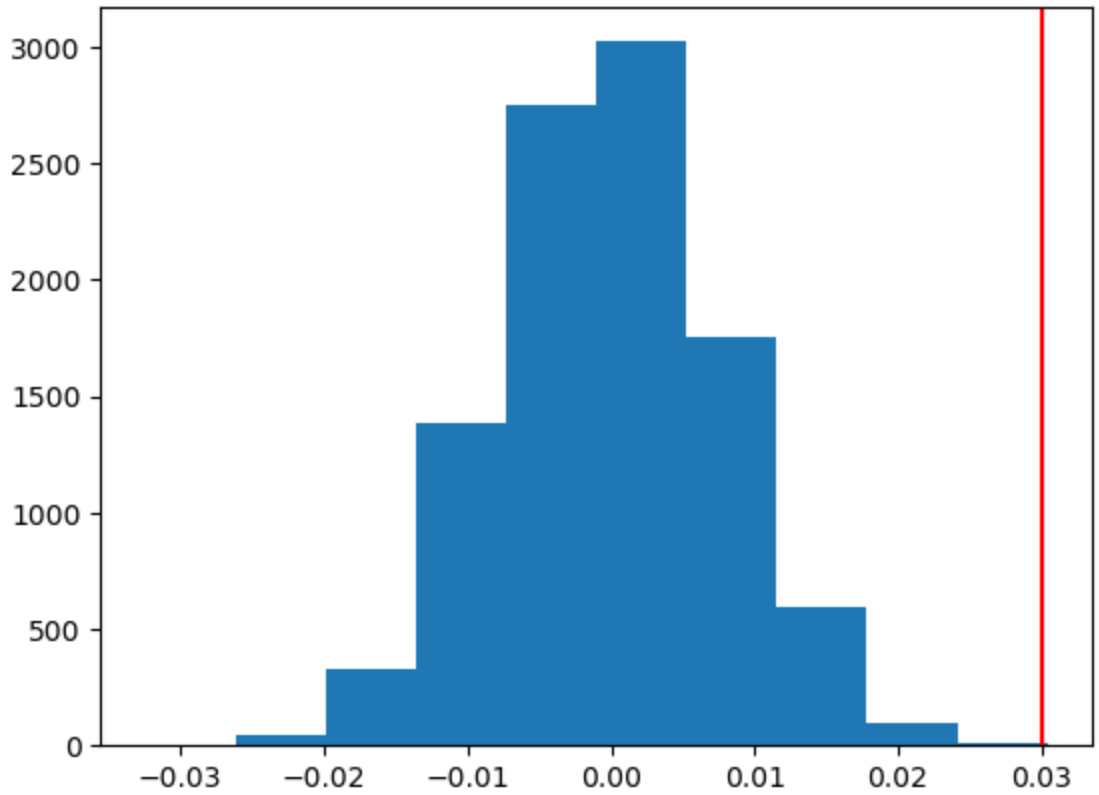
In [43]:
```
#histogram of the bootstrap sample for alternative
plt.hist(diffs);
```



used this sampling distribution to simulate the distribution under the null hypothesis by creating a random normal distribution centered at 0 with the same spread and size.

In [44]:
```
#histogram for the null
diffs = np.array(diffs)
null_vals = np.random.normal(0,diffs.std(),diffs.size)
```

In [45]:
```
plt.hist(null_vals);
plt.axvline(x=obs_diff, color='red');
```



computed the p-value by finding the proportion of values in the null distribution greater than our observed difference.

In [46]:
```
(null_vals > obs_diff).mean()
```

Out[46]:
```
0.0001
```

With a P-value of less than 0.01, it seems unlikely that our statistic is from this null, this means we reject the null hypothesis

Let's recap the steps we took to analyze the results of this A/B test.

1. We computed the observed difference between the metric, click-through rate, for the control and experiment groups.
2. We simulated the sampling distribution for the difference in proportions (or difference in click-through rates).
3. We used this sampling distribution to simulate the distribution under the null hypothesis by creating a random normal distribution centered at 0 with the same spread and size.
4. We computed the p-value by finding the proportion of values in the null distribution greater than our observed difference.
5. We used this p-value to determine the statistical significance of our observed difference.

- Based on this conclusion, we recommend to implement the new experiment homepage

Experiment II

The second change, Audacity is A/B testing is a more career-focused description on a course overview page. They hope that this change may encourage more users to enroll and complete this course. In this experiment, we're going to analyze the following metrics:

1. **Enrollment Rate**: Click-through rate for the Enroll button on the course overview page
2. **Average Reading Duration**: Average number of seconds spent on the course overview page
3. **Average Classroom Time**: Average number of days spent in the classroom for students enrolled in the course
4. **Completion Rate**: Course completion rate for students enrolled in the course

First, let's determine if the difference observed for each metric is statistically significant individually.

Metric - Enrollment Rate

In [48]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

np.random.seed(42)
```

In [49]:
```
df = pd.read_csv('course_page_actions.csv')
df.head()
```

Out[49]:

	timestamp	id	group	action	duration
0	2016-09-24 17:14:52.012145	261869	experiment	view	130.545004
1	2016-09-24 18:45:09.645857	226546	experiment	view	159.862440
2	2016-09-24 19:16:21.002533	286353	experiment	view	79.349315
3	2016-09-24 19:43:06.927785	842279	experiment	view	55.536126
4	2016-09-24 21:08:22.790333	781883	experiment	view	204.322437

In [50]:
```
# Get dataframe with all records from control group
control_df = df.query('group == "control"')

# Compute click through rate for control group
control_ctr = control_df.query('action == "enroll"').id.nunique() / control_df.query('ac

# Display click through rate
control_ctr
```

Out[50]:
```
0.2364438839848676
```

In [51]:
```
# Get dataframe with all records from experiment group
experiment_df = df.query('group == "experiment"')

# Compute click through rate for experiment group
experiment_ctr = experiment_df.query('action == "enroll"').id.nunique() / experiment_df.

# Display click through rate
experiment_ctr
```

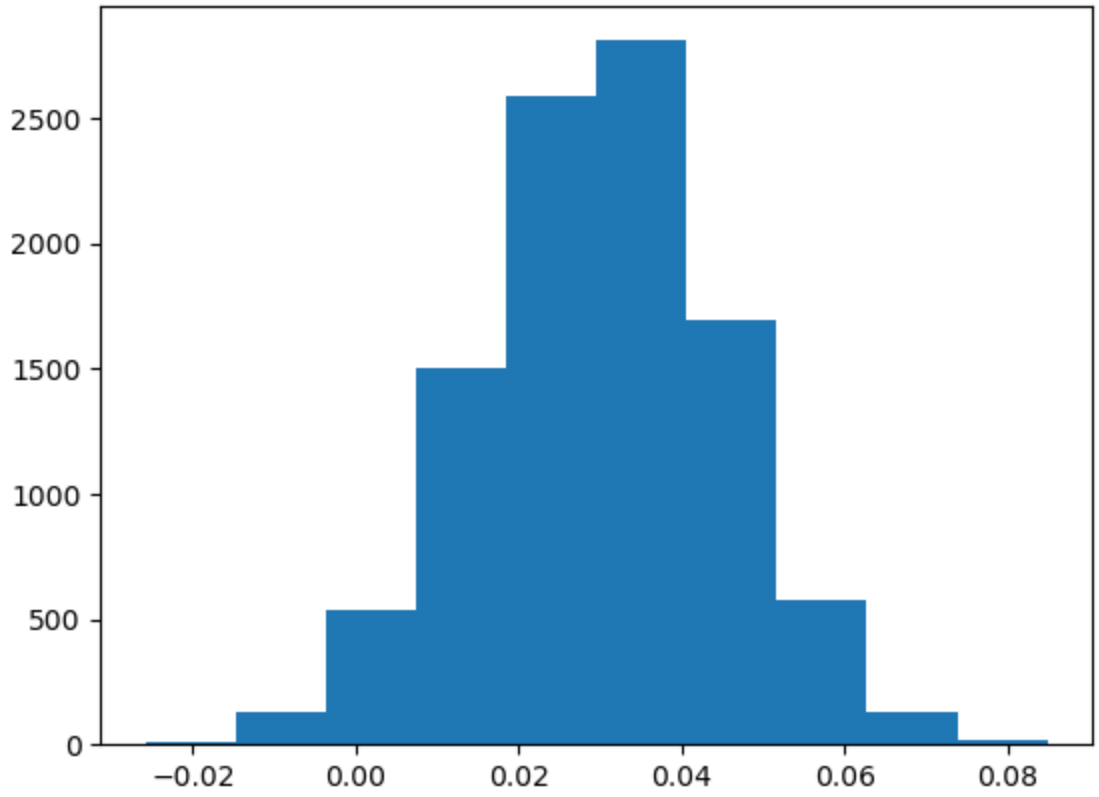
Out[51]:
```
0.2668693009118541
```

In [52]:
```
# Compute the observed difference in click through rates
obs_diff = experiment_ctr - control_ctr

# Display observed difference
obs_diff
```

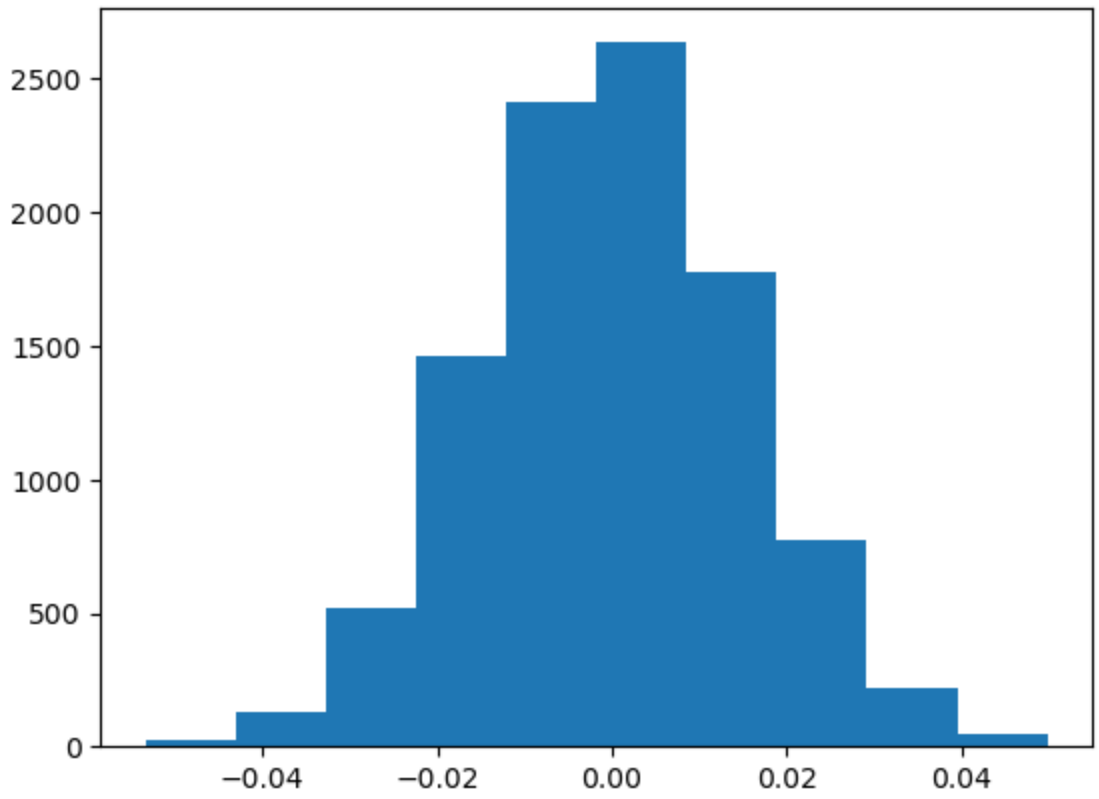
Out[52]:
```
0.030425416926986526
```

In [53]:
```
# Create a sampling distribution of the difference in proportions
# with bootstrapping
diffs = []
size = df.shape[0]
for _ in range(10000):
    b_samp = df.sample(size, replace=True)
    control_df = b_samp.query('group == "control"')
    experiment_df = b_samp.query('group == "experiment"')
    control_ctr = control_df.query('action == "enroll"').id.nunique() / control_df.query
    experiment_ctr = experiment_df.query('action == "enroll"').id.nunique() / experiment
    diffs.append(experiment_ctr - control_ctr)
```

In [58]:
```
# Convert to numpy array
diffs = np.array(diffs)

# Plot sampling distribution
plt.hist(diffs);
```

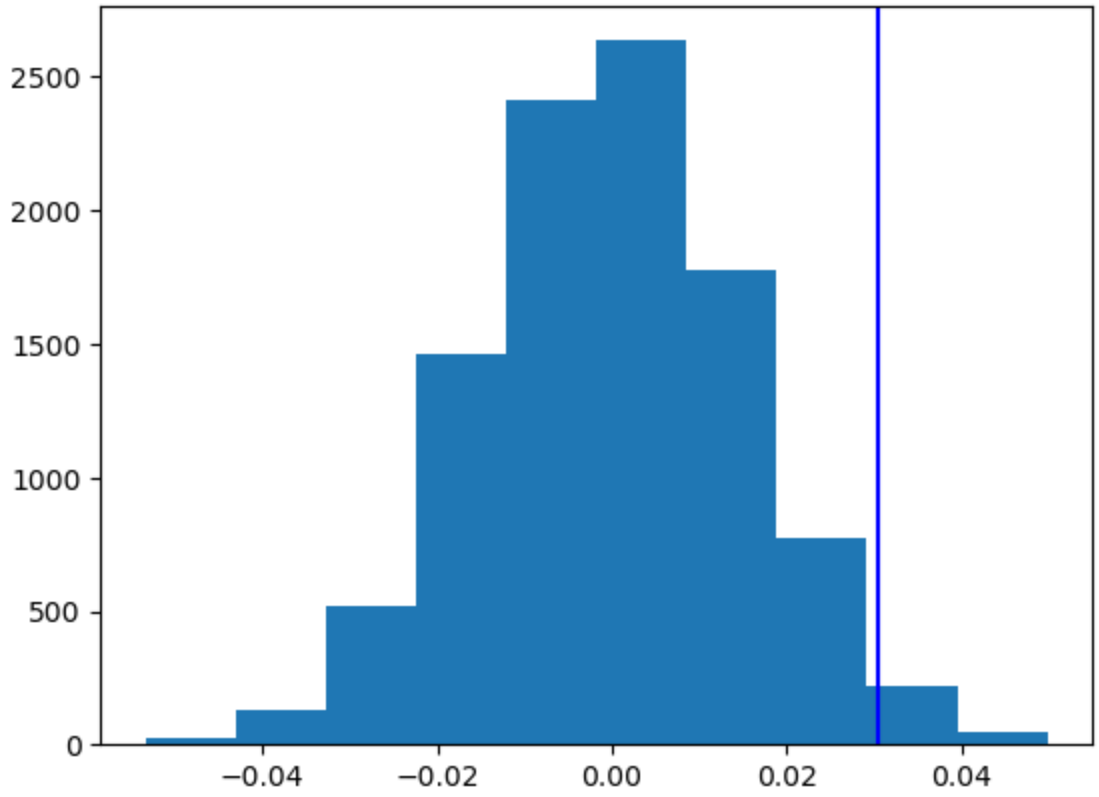


In [62]:
```
# Simulate distribution under the null hypothesis
null_vals = np.random.normal(0,diffs.std(),diffs.size)

# Plot the null distribution
plt.hist(null_vals);
```



In [64]:
```
# Plot observed statistic with the null distibution
plt.hist(null_vals);
plt.axvline(obs_diff, c='blue')
```

Out[64]:
```
<matplotlib.lines.Line2D at 0x1ec66175340>
```



In [61]:
```
# Compute p-value
(null_vals > obs_diff).mean()
```

Out[61]:
```
0.0188
```

Metric - Average Reading Duration

In [72]:
```
import numpy as np, pandas as pd, matplotlib.pyplot as plt
%matplotlib inline
```

In [73]:
```
df = pd.read_csv("course_page_actions.csv")
df.head()
```

Out[73]:

	timestamp	id	group	action	duration
0	2016-09-24 17:14:52.012145	261869	experiment	view	130.545004
1	2016-09-24 18:45:09.645857	226546	experiment	view	159.862440
2	2016-09-24 19:16:21.002533	286353	experiment	view	79.349315
3	2016-09-24 19:43:06.927785	842279	experiment	view	55.536126
4	2016-09-24 21:08:22.790333	781883	experiment	view	204.322437

In [74]:
```
#get only views
views = df.query('action == "view"')
```

In [75]:
```
#Group by id and group with the duration mean.
reading_times = views.groupby(['id', 'group'])['duration'].mean()
```

In [77]:
```
#reset index to keep this as a dataframe
reading_times = reading_times.reset_index()
#A new integer-based index is assigned to the DataFrame, starting from 0 and incrementin
```

In [78]:
```
reading_times.head()
```

Out[78]:

	id	group	duration
0	183260	control	107.331484
1	183615	experiment	24.627594
2	184277	experiment	193.212489
3	184360	experiment	226.586283
4	184589	experiment	12.052097

In [79]:
```
#get means for control/experiment
control_mean = df.query('group == "control"')['duration'].mean()
experiment_mean = df.query('group == "experiment"')['duration'].mean()
control_mean , experiment_mean
```

Out[79]:
```
(115.40710650582038, 130.93220512539477)
```

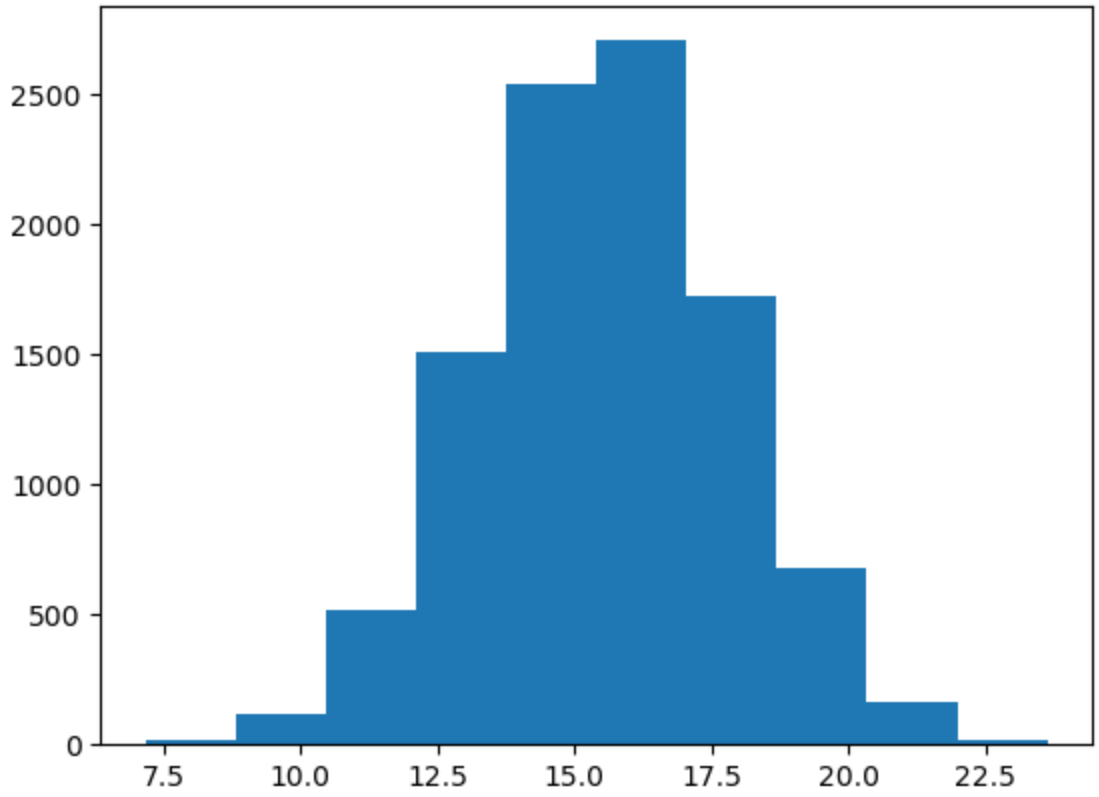
In [80]:
```
obs_diff = experiment_mean - control_mean
obs_diff
```

Out[80]:
```
15.525098619574393
```

In [81]:
```
#bootstrap sample
diffs = []
for _ in range(10000):
    b_samp = df.sample(df.shape[0], replace=True)
    control_mean = b_samp.query('group == "control"')['duration'].mean()
    experiment_mean = b_samp.query('group == "experiment"')['duration'].mean()
    diffs.append(experiment_mean - control_mean)
```

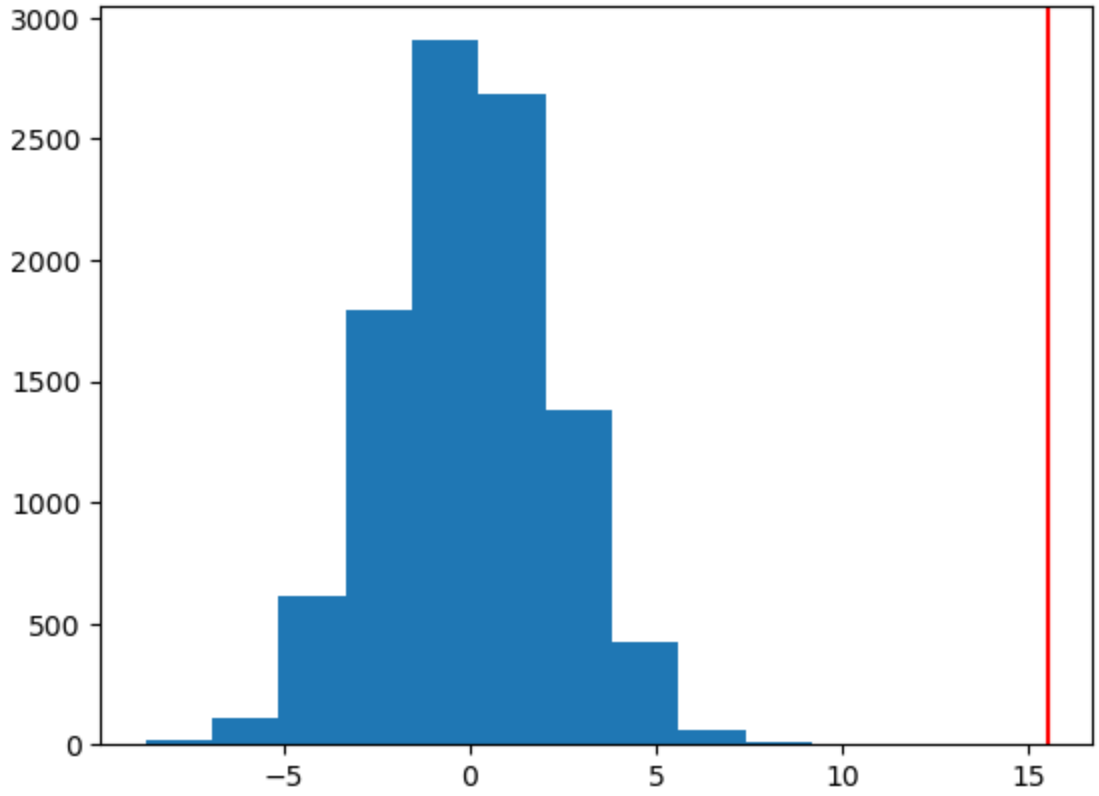
In [82]:
```
diffs = np.array(diffs)
```

In [83]:
```
plt.hist(diffs);
```



In [89]:
```
null_vals = np.random.normal(0.,diffs.std(), diffs.size)
plt.hist(null_vals);
plt.axvline(x=obs_diff, c='r')
```

Out[89]:
```
<matplotlib.lines.Line2D at 0x1ec660bf640>
```



Again, let's recap the steps we took to analyze the results of this A/B test.

1. We computed the observed difference between the metric, average reading duration for the control and experiment groups.
2. ed the sampling distribution for the difference in means (or average reading durations).

3. We used this sampling distribution to simulate the distribution under the null hypothesis by creating a random normal distribution centered at 0 with the same spread and size.
4. We computed the p-value by finding the proportion of values in the null distribution greater than our observed difference.
5. We used this p-value to determine the statistical significance of our observed difference.

Metric - Average Classroom Time

In [104…]
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

np.random.seed(42)
```

In [105…]
```
df = pd.read_csv('classroom_actions.csv')
df.head()
```

Out[105]:

	timestamp	id	group	total_days	completed
0	2015-08-10 17:06:01.032740	610019	experiment	97	True
1	2015-08-10 17:15:28.950975	690224	control	75	False
2	2015-08-10 17:34:40.920384	564994	experiment	128	True
3	2015-08-10 17:50:39.847374	849588	experiment	66	False
4	2015-08-10 19:10:40.650599	849826	experiment	34	False

In [106…]
```
# The total_days represents the total amount of time
# each student has spent in classroom.
# get the average classroom time for control group
control_mean = df.query('group == "control"')['total_days'].mean()

# get the average classroom time for experiment group
experiment_mean = df.query('group == "experiment"')['total_days'].mean()

# display average classroom time for each group
control_mean, experiment_mean
```

Out[106]:
```
(73.36899038461539, 74.6715935334873)
```

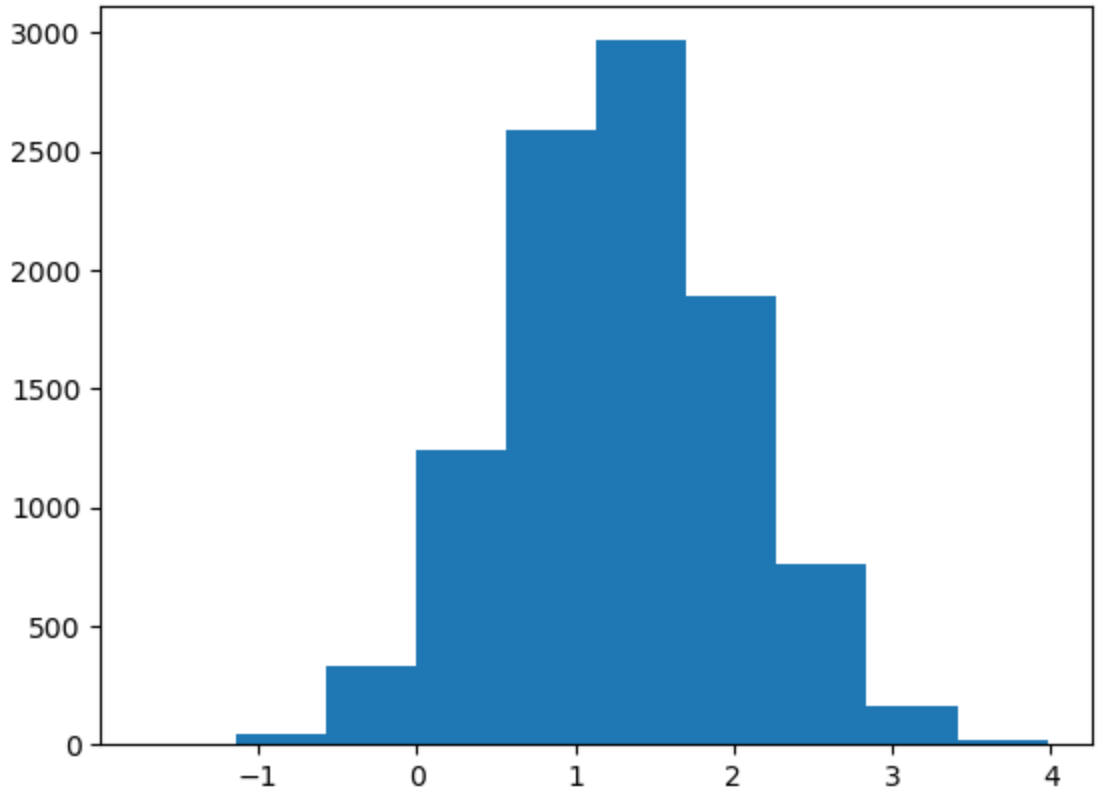
In [107…]
```
# compute observed difference in classroom time
obs_diff = experiment_mean - control_mean

# display observed difference
obs_diff
```

Out[107]:
```
1.3026031488719099
```

In [109…]
```
# create sampling distribution of difference in average classroom times
# with boostrapping
diffs = []
for _ in range(10000):
    df_sample = df.sample(df.shape[0], replace = True)
    control_means = df_sample.query('group == "control"')['total_days'].mean()
    experiment_means = df_sample.query(' group == "experiment"')['total_days'].mean()
    diffs.append(experiment_means - control_means)
```

In [110…
```
# convert to numpy array
diffs = np.array(diffs)
```

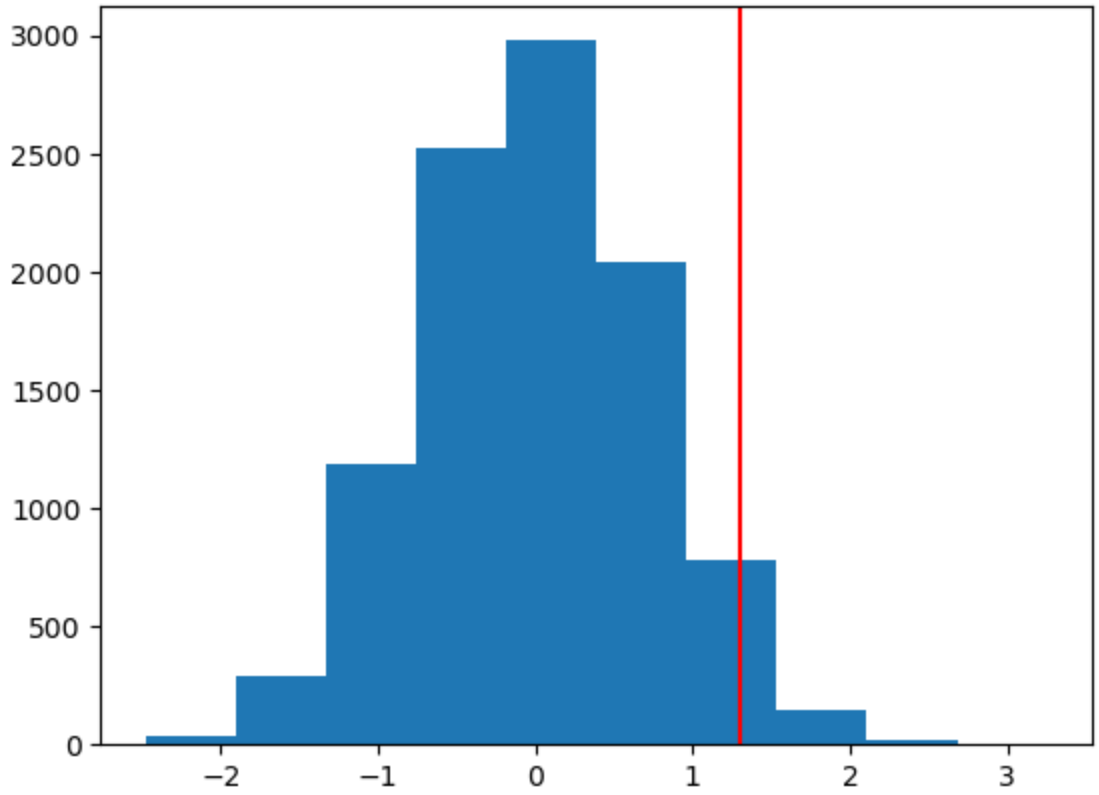
In [111…
```
# plot sampling distribution
plt.hist(diffs);
```



In [112…
```
# simulate distribution under the null hypothesis
null_vals =np.random.normal(0, diffs.std(),diffs.size)
```

In [113…
```
# plot null distribution
plt.hist(null_vals);

# plot line for observed statistic
plt.axvline(x=obs_diff, c='r')
```

Out[113]:
```
<matplotlib.lines.Line2D at 0x1ec663418b0>
```



In [114…
```
# compute p value
(null_vals > obs_diff).mean()
```

Out[114]:
```
0.0389
```

Quiz Question

Do you have evidence, with a type I error rate of 0.05, that users spend more time in the classroom after seeing the experimental description in the course overview page?

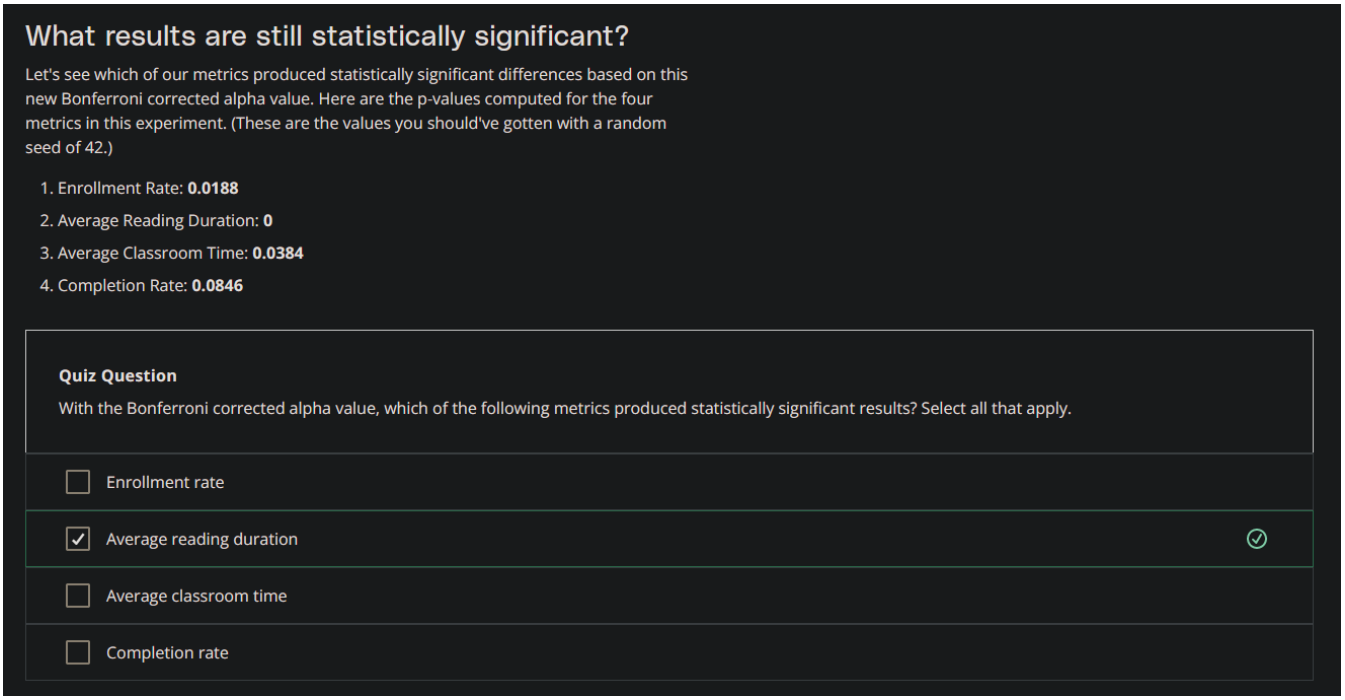
- ◯ Yes ✓
- ◯ No

Analyzing Multiple Metrics

The more metrics you evaluate, the more likely you are to observe significant differences just by chance - similar to what you saw in previous lessons with multiple tests. Luckily, this multiple comparisons problem can be handled in several ways.

If you remember from the previous lesson, the **Bonferroni Correction** is one way we could handle experiments with multiple tests or metrics in this case. To compute the new Bonferroni correct alpha value,

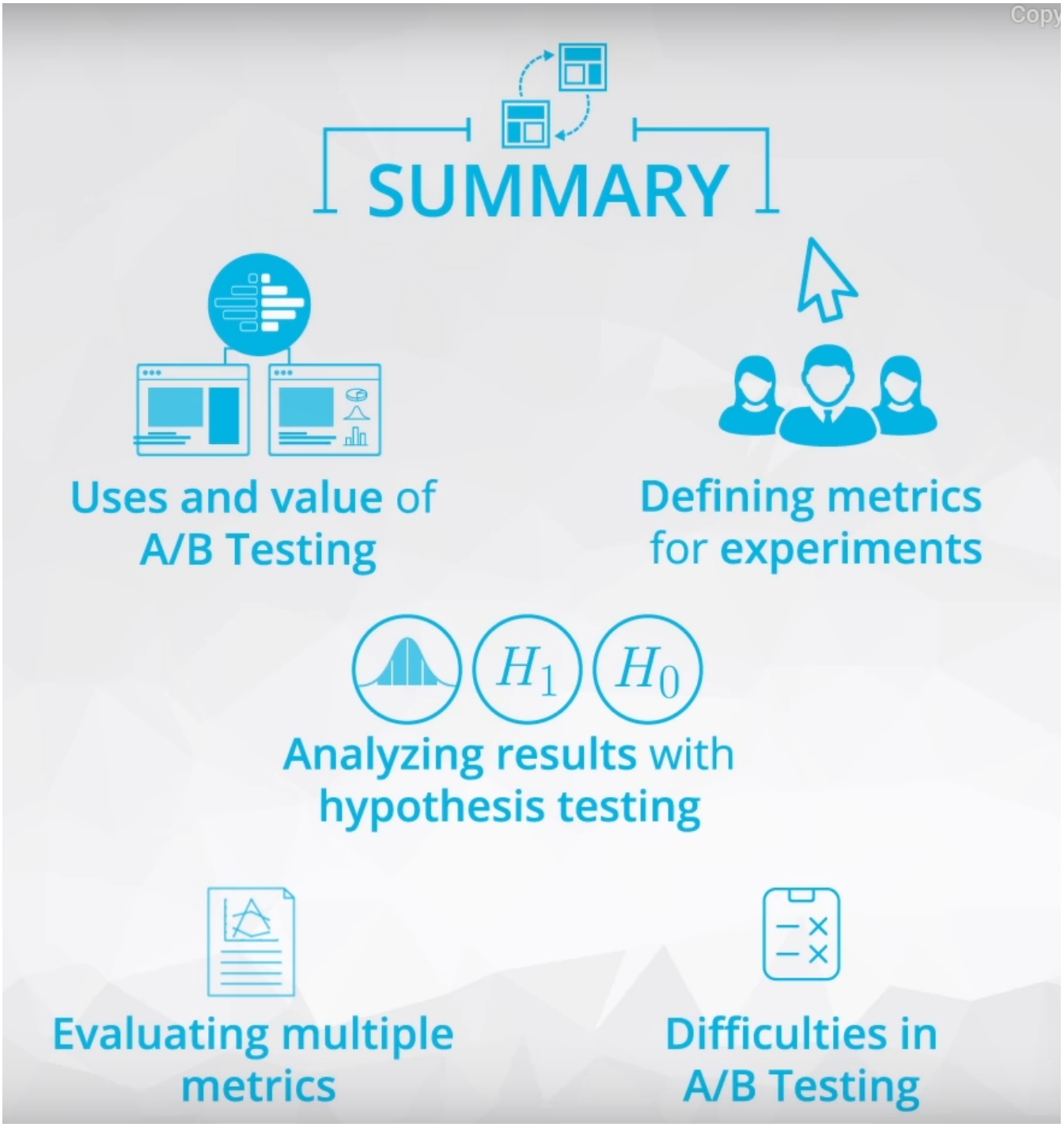
we need to divide the original alpha value by the number of tests.



e.g.Difficulties in A/B Testing

There are many factors to consider when designing an A/B test and drawing conclusions based on its results. To conclude, here are some common ones to consider.

- Novelty effect and change aversion when existing users first experience a change
- Sufficient traffic and conversions to have significant and repeatable results
- The best metric choice for making the ultimate decision (eg. measuring revenue vs. clicks)
- Long enough run time for the experiment to account for changes in behavior based on time of day/week or seasonal events.
- The practical significance of a conversion rate (the cost of launching a new feature vs. the gain from the increase in conversion)
- Consistency among test subjects in the control and experiment group (imbalance in the population represented in each group can lead to situations like Simpson's Paradox)



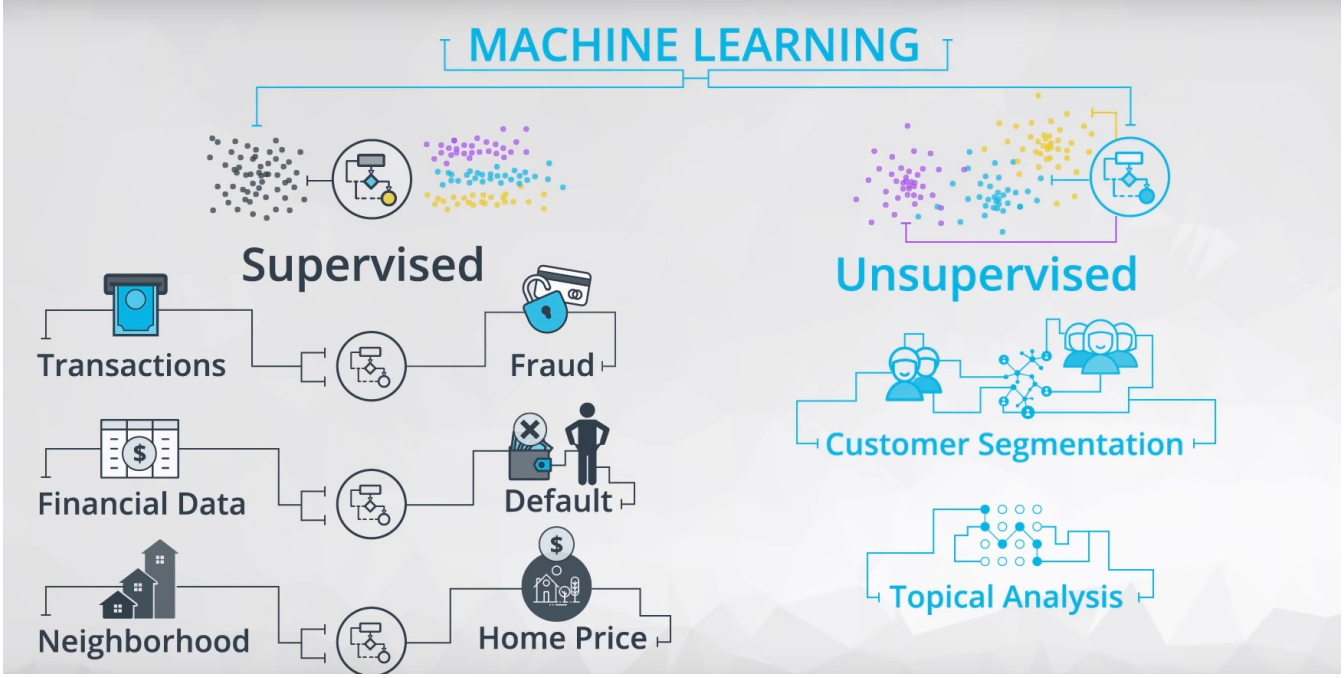
Regression

In this lesson, you will:

1. Identify Regression Applications
2. Learn How Regression Works
3. Apply Regression to Problems Using Python

Introduction to Machine Learning

Machine Learning is frequently split into supervised and unsupervised learning. Regression, is an example of supervised machine learning.



In supervised machine learning, you are interested in predicting a label for your data. Commonly, you might want to predict fraud, customers that will buy a product, or home values in an area.

In unsupervised machine learning, you are interested in clustering data together that isn't already labeled. However, we will not be going into the details of these algorithms in this course.

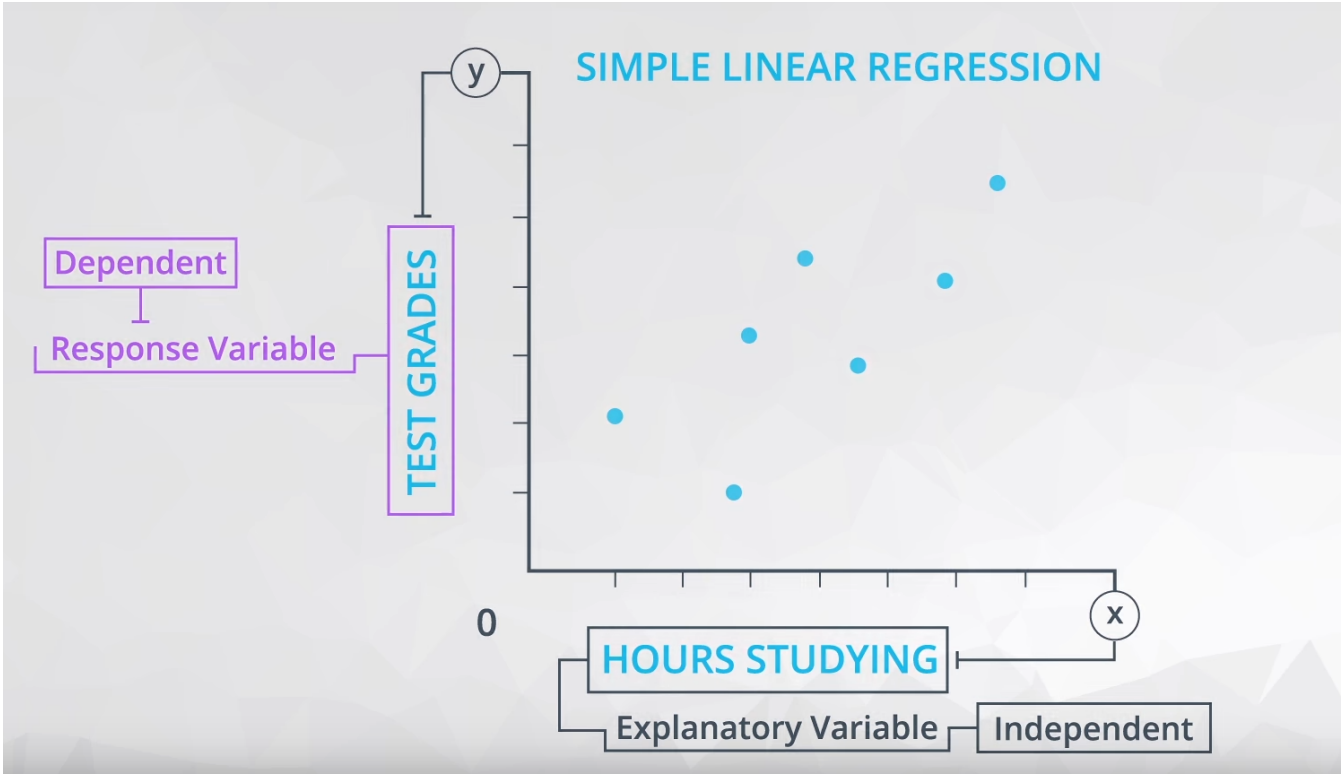
Description	Term
A machine learning technique where we are attempting to predict a label based on inputs.	Supervised Learning
A machine learning technique where we are attempting to group together unlabeled data based on similar characteristics.	Unsupervised Learning
A key supervised learning technique you will be learning about in this lesson.	Regression (Multiple Linear and Logistic)

Introduction to Linear Regression

In simple linear regression, we compare **two quantitative** variables to one another.

The response variable is what you want to predict, while the explanatory variable is the variable you use to predict the response. A common way to visualize the relationship between two variables in linear regression

is using a scatterplot.



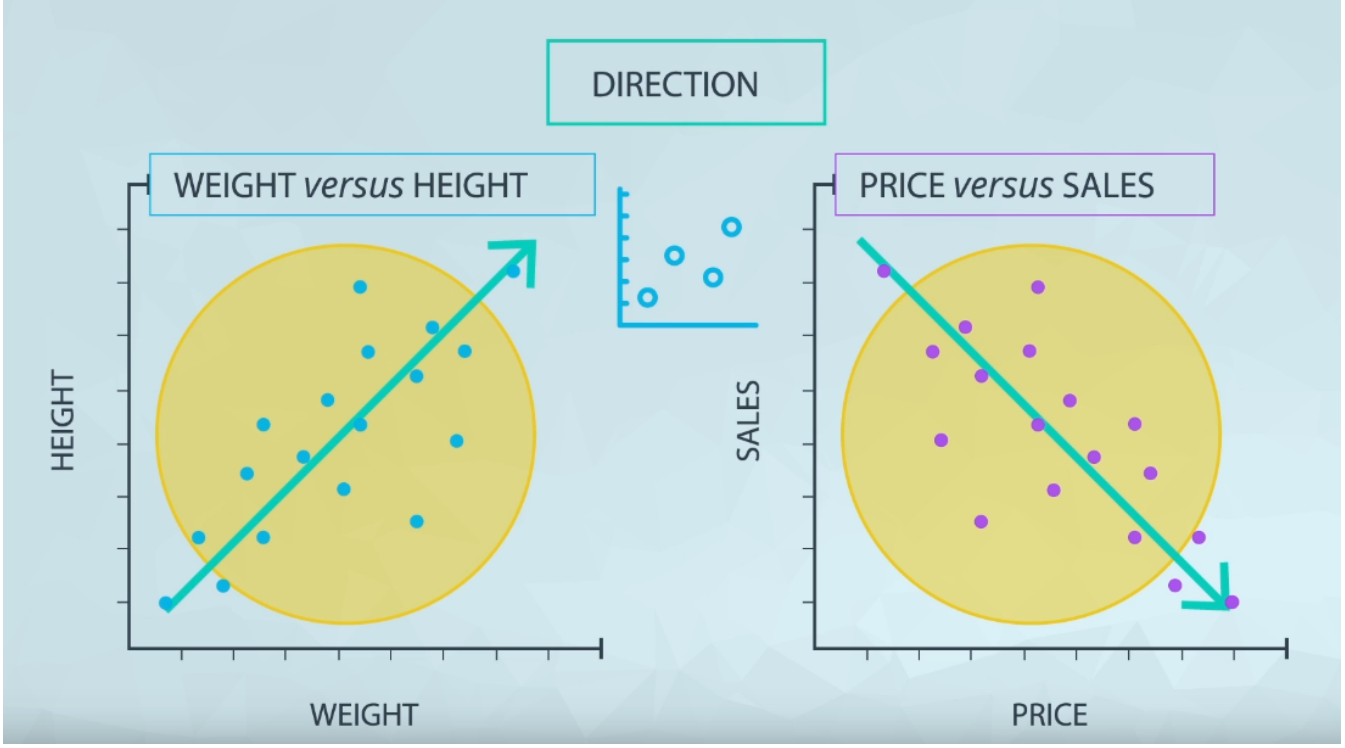
Scatter Plots

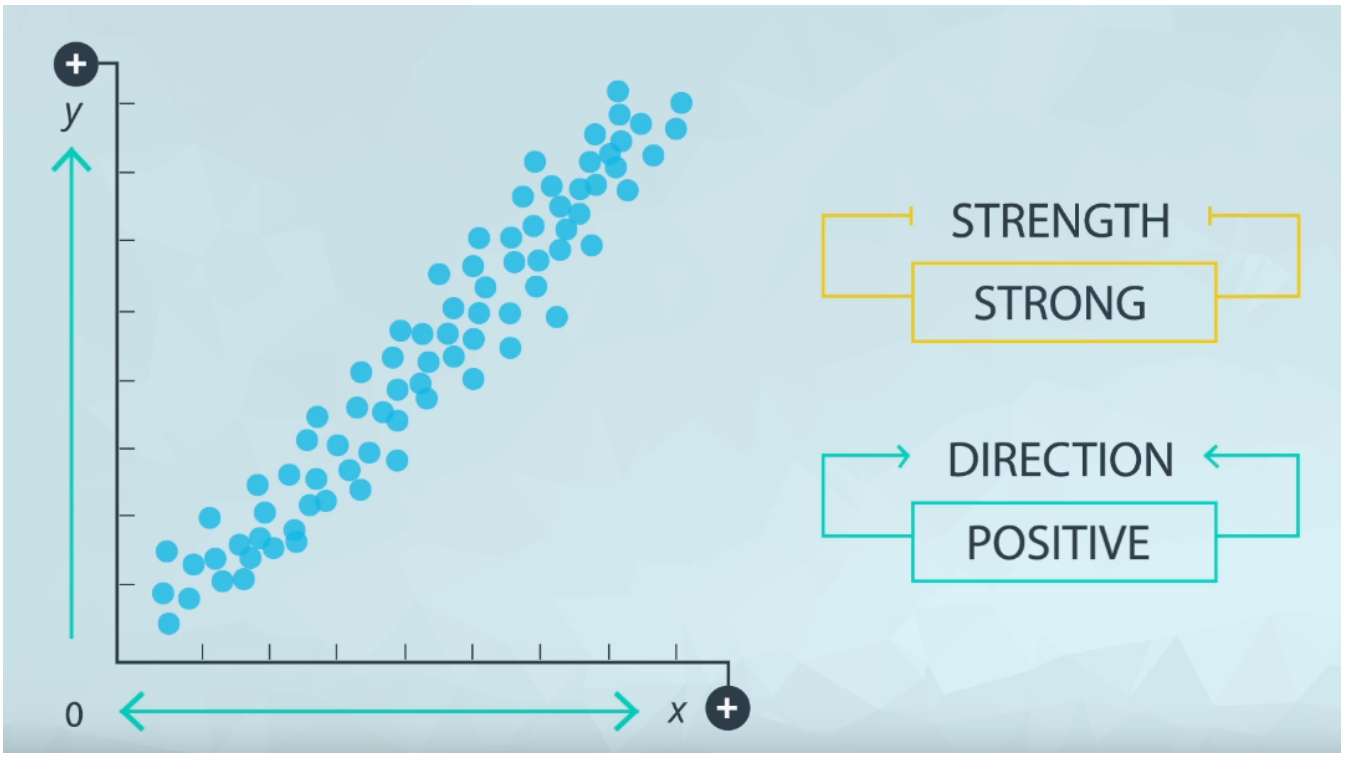
Scatter plots are a common visual for comparing two quantitative variables. A common summary statistic that relates to a scatter plot is the correlation coefficient commonly denoted by r.

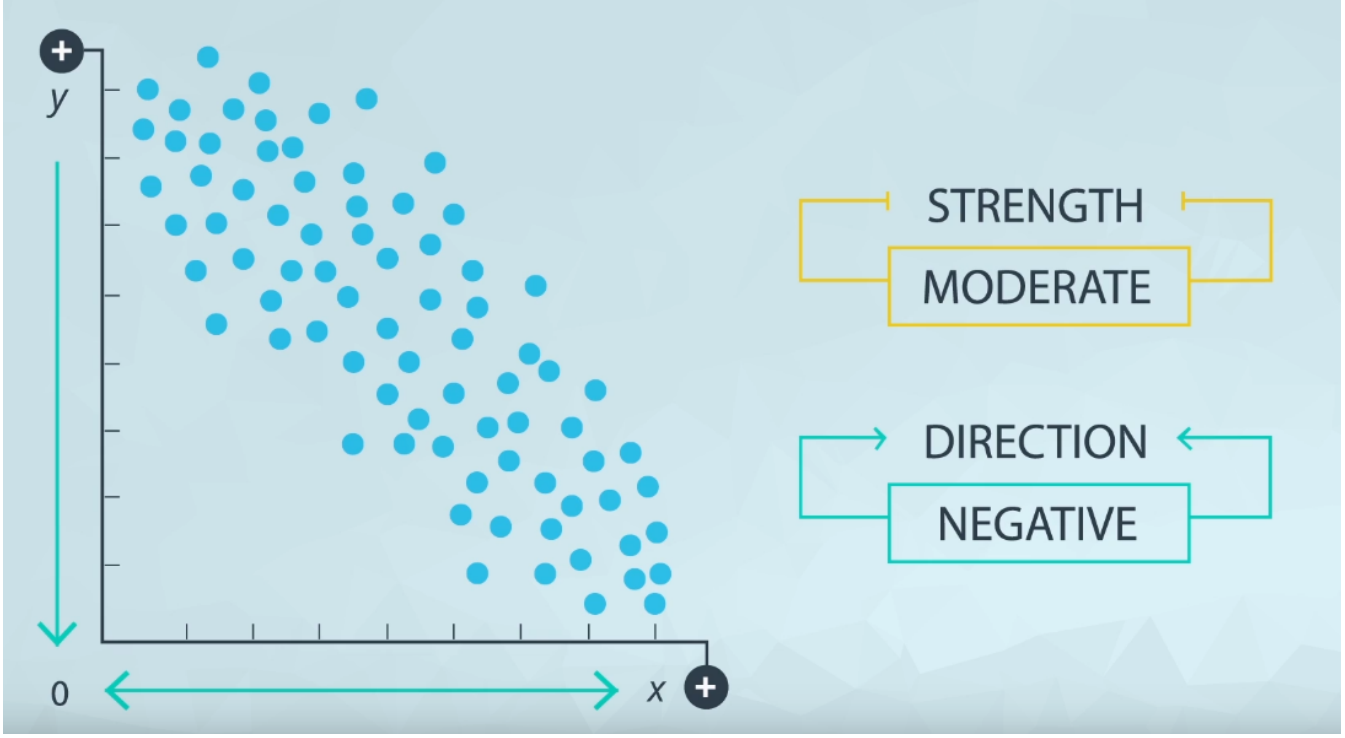
Though there are a few different ways to measure correlation between two variables, the most common way is with Pearson's correlation coefficient. Pearson's correlation coefficient provides the:

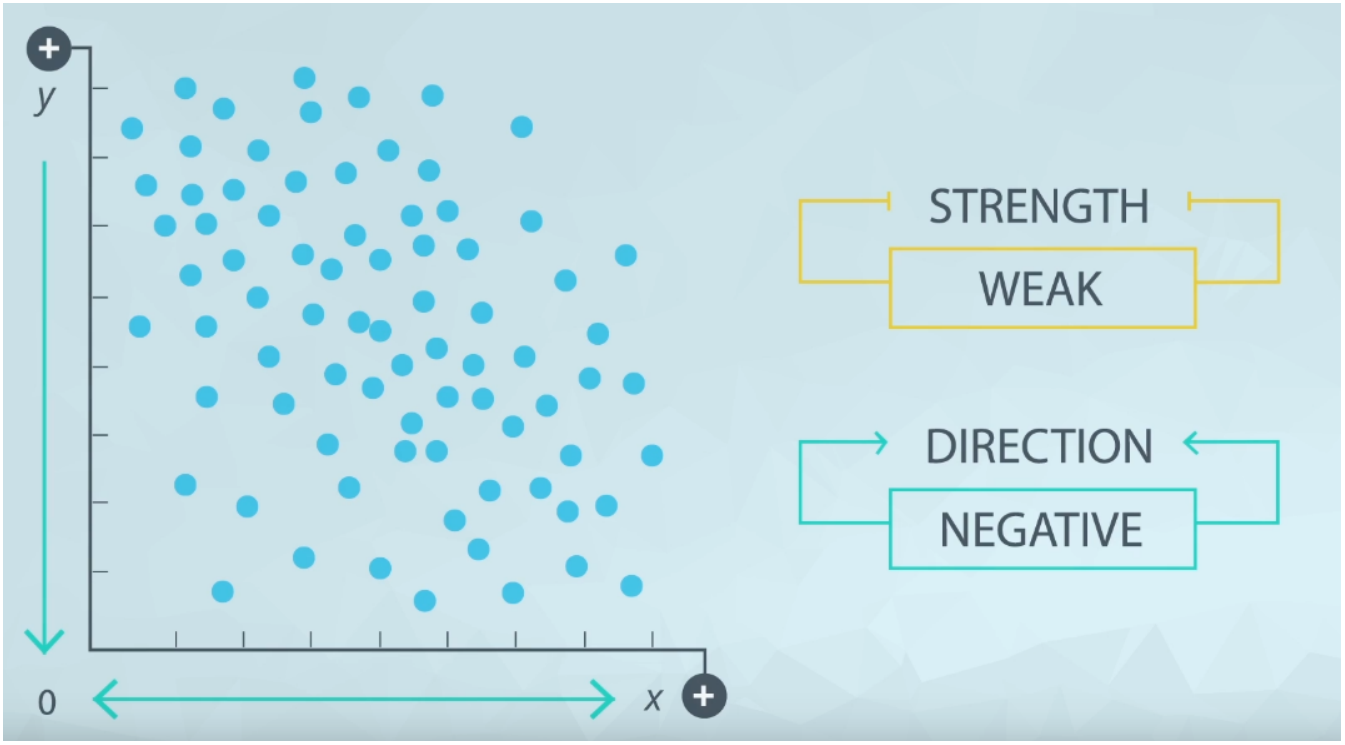
1. Strength
2. Direction

of a linear relationship. Spearman's Correlation Coefficient does not measure linear relationships specifically, and it might be more appropriate for certain cases of associating two variables.



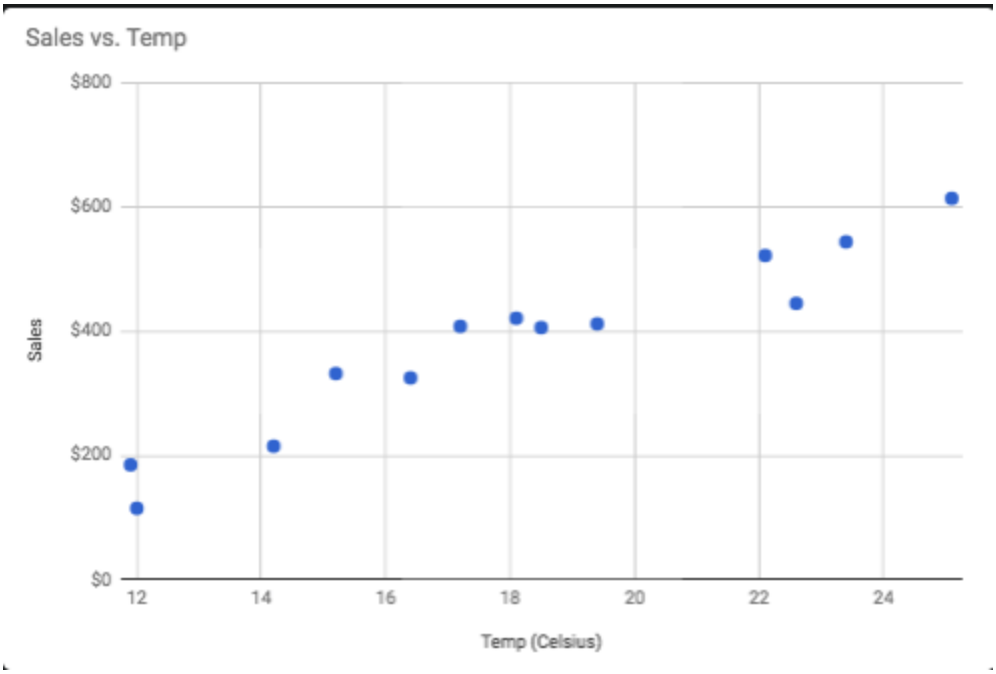






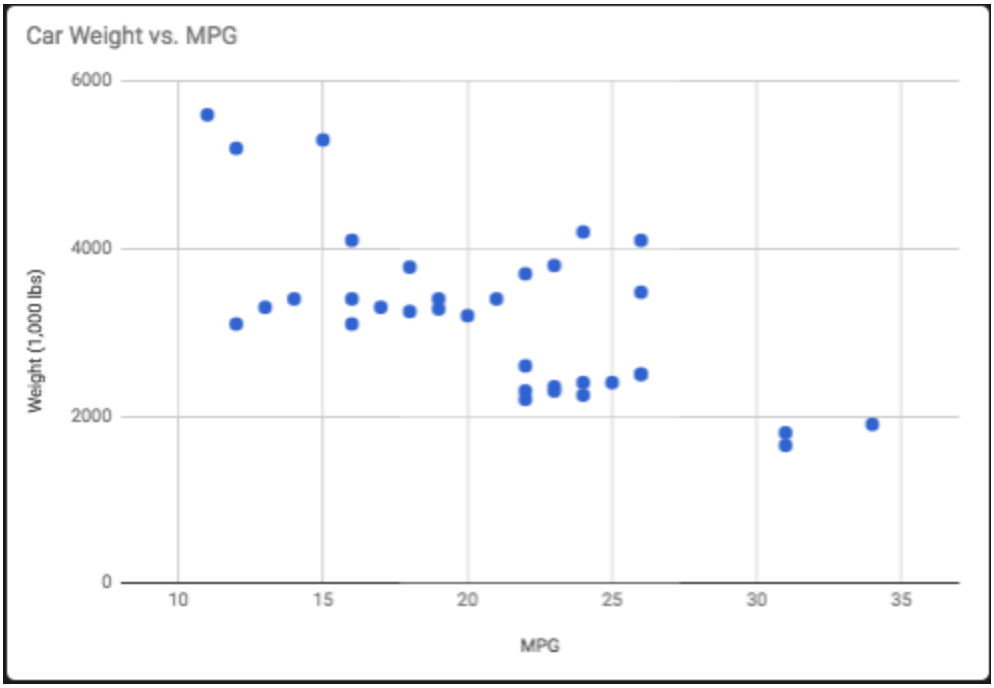
Quizzes On Scatter Plots

Check Your Scatterplot Skills



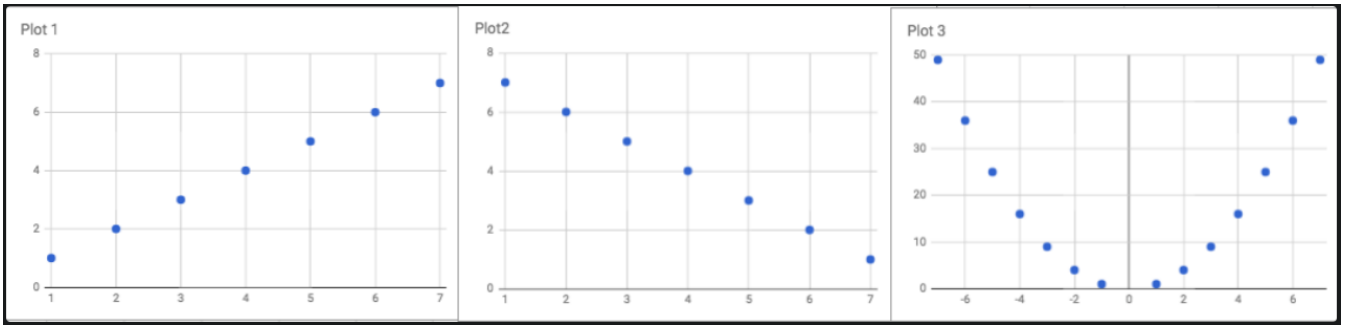
Term	Description
Strength	Strong
Direction	Positive (Direct)
Correlation Coefficient	Close to 1

Check Your Scatterplot Skills (Part II)



Term	Description
Strength	Moderate
Direction	Negative (Indirect)
Correlation Coefficient	-0.67

Check Your Scatterplot Skills (Part III)



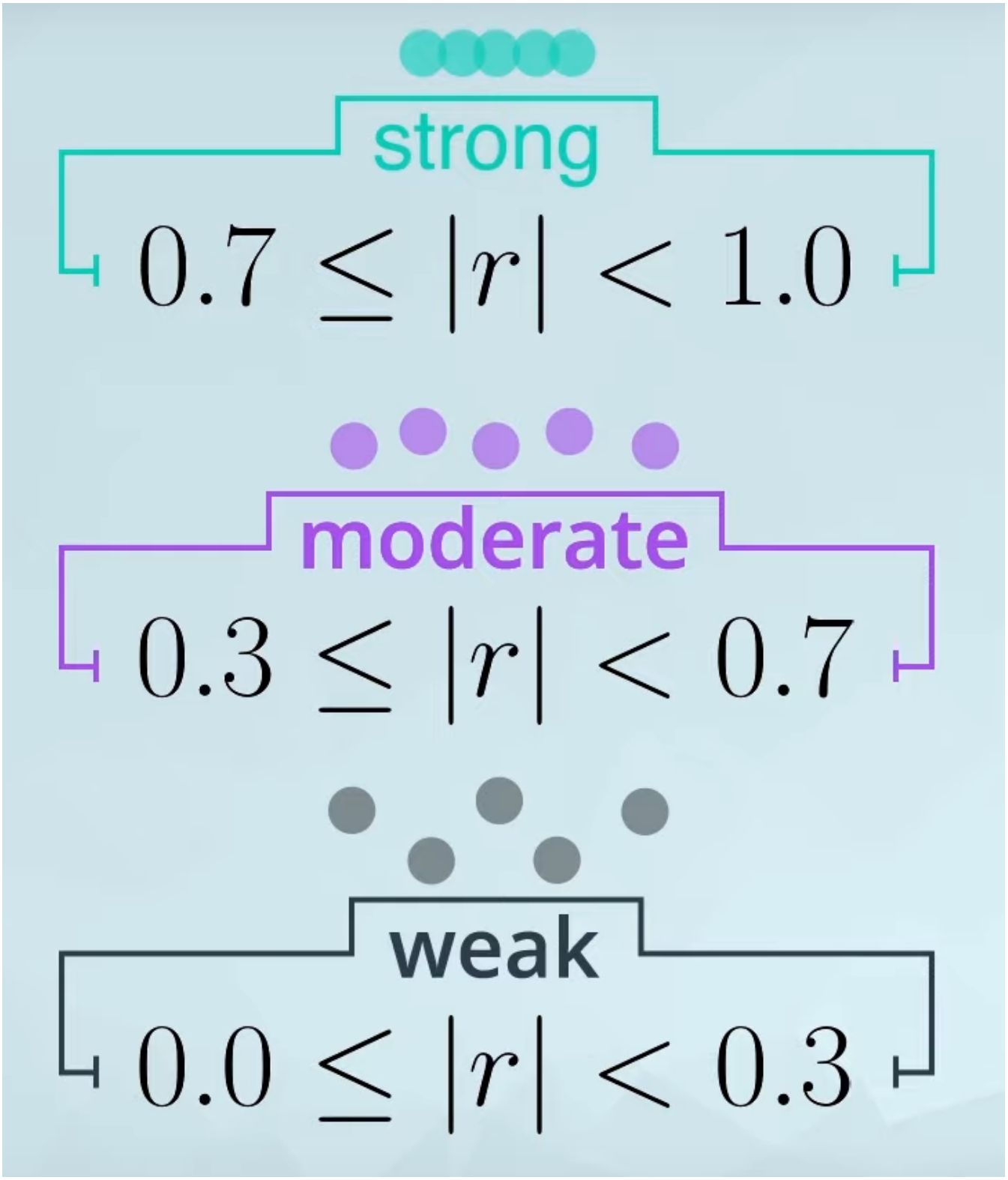
Scatter plot	Correlation Coefficient (r) Value
Plot 1	1.0
Plot 2	-1.0
Plot 3	0.0

Correlation Coefficients

Correlation coefficients provide a measure of the strength and direction of a linear relationship.

We can tell the direction based on whether the correlation is positive or negative.

A rule of thumb for judging the strength:



Quiz Question

Which of the below are true statements regarding the correlation coefficient?

- [ ] The correlation coefficient can be any value.
- [ ] A correlation coefficient of 0 means there is no relationship between two variables.
- [ ] The correlation coefficient can be used to compare more than two variables to one another.
- [x] The correlation coefficient must be between 1 and -1 (inclusively). ✓
- [ ] A smaller correlation coefficient means a weaker relationship between two variables.
- [x] The correlation coefficient (specifically Pearson's correlation) is for measuring linear relationships. ✓

Does smaller always mean weaker? A correlation of -1 is smaller than a correlation of 0. Does this mean the relationship from variables with a correlation of -1 is weaker?

What Defines A Line?

A line is commonly identified by an intercept and a slope.

The intercept is defined as the predicted value of the response when the x-variable is zero.

The slope is defined as the predicted change in the response for every one unit increase in the x-variable.

We notate the line in linear regression in the following way:

$\hat{y}=b_0+b_1x_1$

where

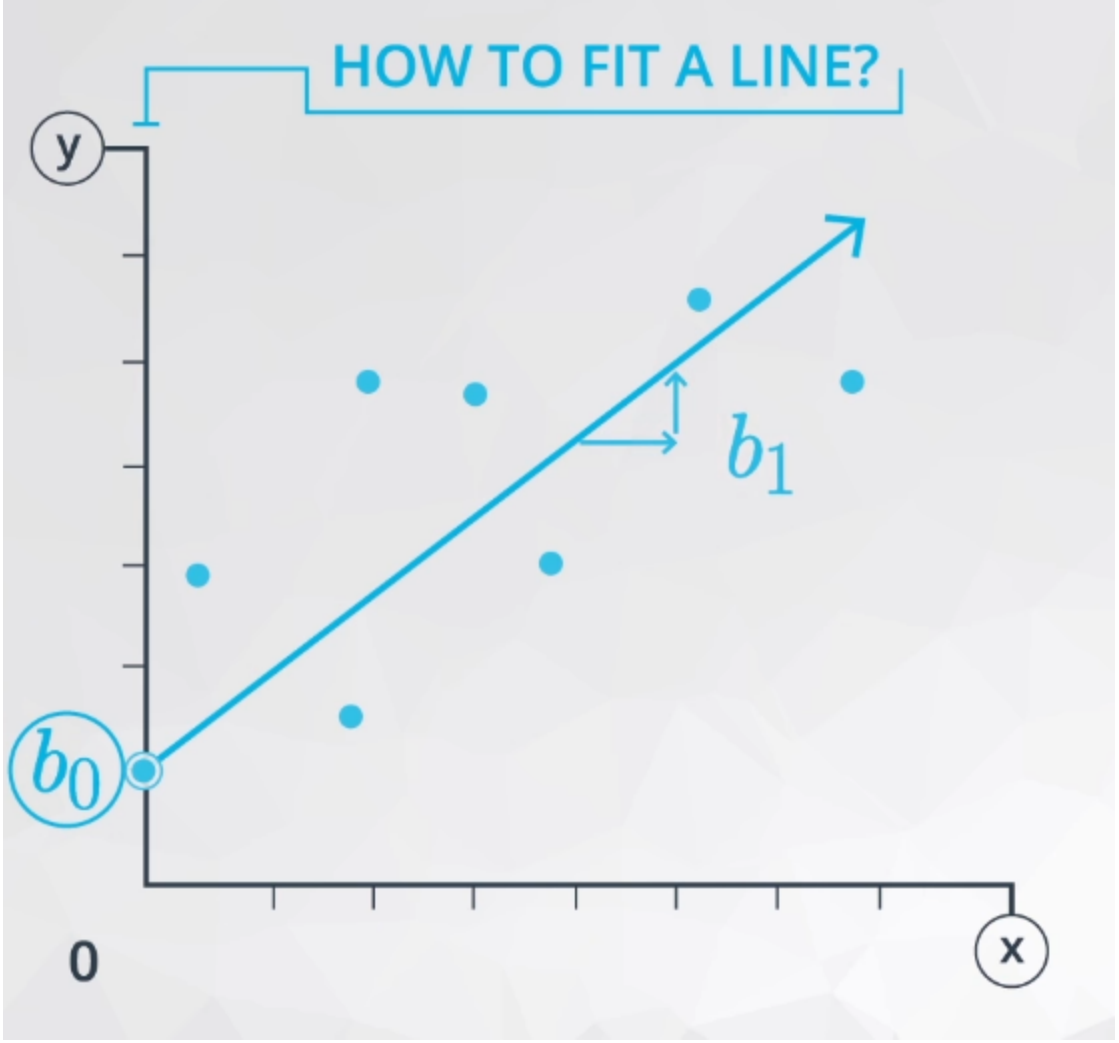
$\hat{y}$ is the predicted value of the response from the line.

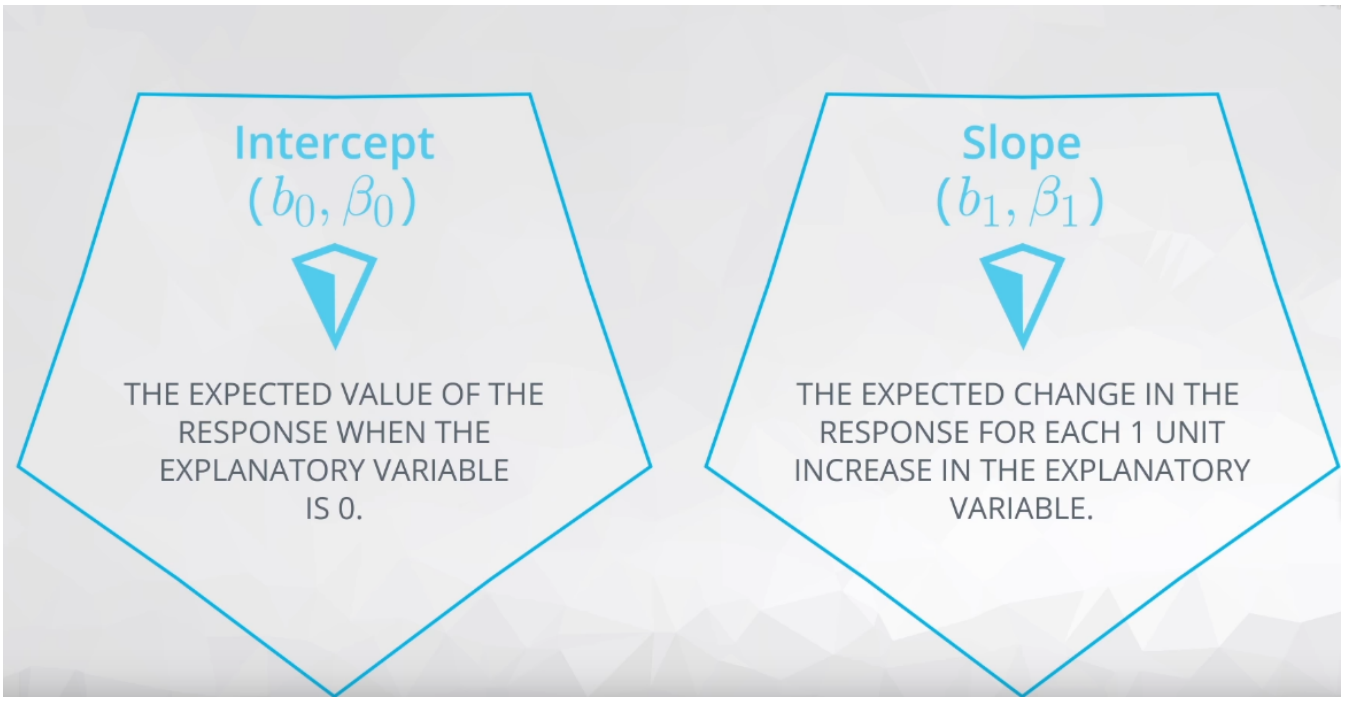
b_0 is the intercept.

b_1 is the slope.

x_1 is the explanatory variable.

Note that in the equation above, there is a hat in the $\hat{y}$ term. This hat indicates that the value is the predicted value from the fitted line and it is not the real value. We use y (without the hat) to denote the actual response value for a data point in our dataset.





Notation for Quizzes

A. $\mathbf{y}$	E. β_0
B. $\hat{\mathbf{y}}$	F. β_1
C. $\mathbf{b_0}$	G. $\mathbf{b_1}$
D. $\mathbf{n}$	

For the below quiz, let the following letters denote the corresponding notation:

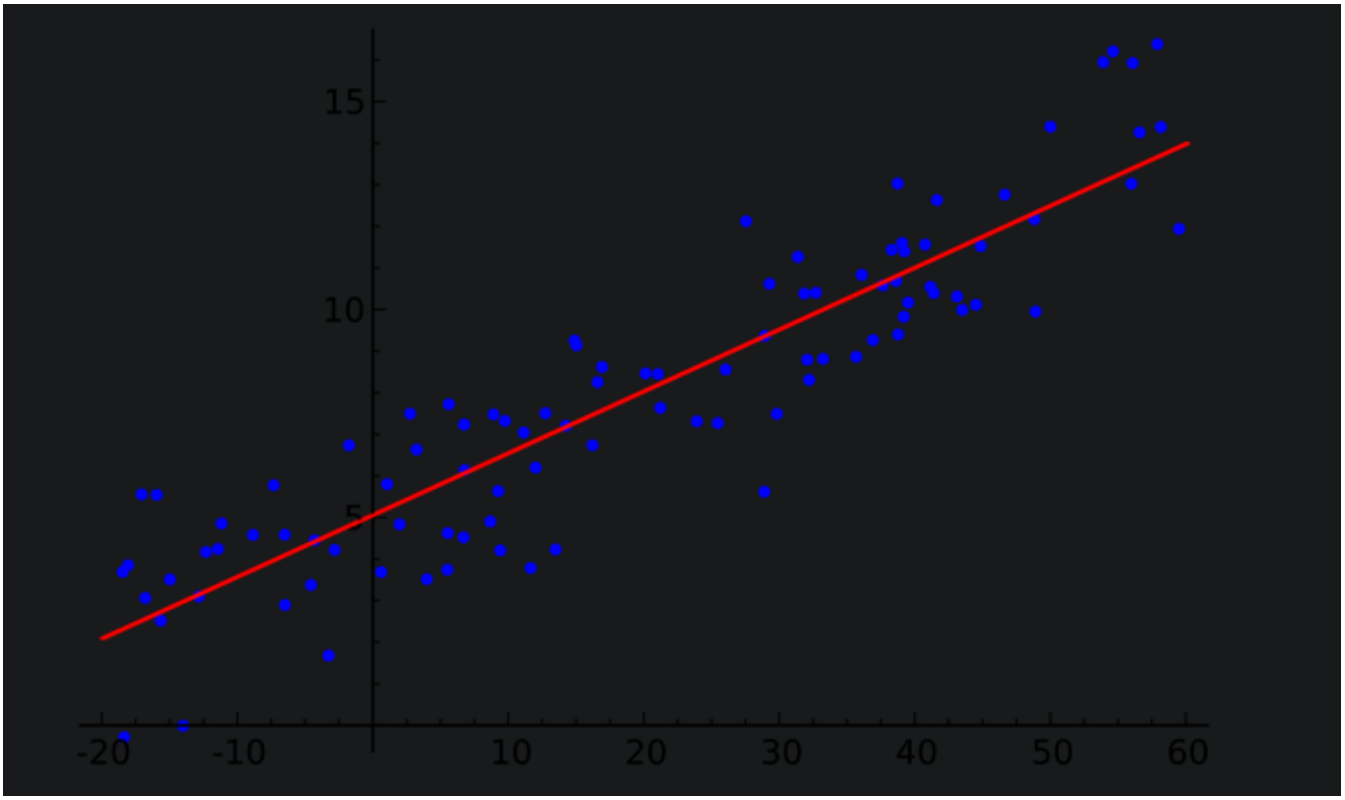
Description	Notation Letter
A predicted value of the response.	B
An actual value of the response.	A
The predicted value of the response when the explanatory variable is zero.	C
The number of rows in the dataset.	D

Description	Notation Letter
The predicted change in the response for every one unit increase in the explanatory variable.	G
The actual average change in the response for the population with every one unit increase in the explanatory variable.	F
The actual average response value for the population when the explanatory variable is zero.	E
A predicted response value.	B

Quiz - Line Basics

For many months, I keep track of the number of hours I work compared to a standard 40 hour week, where -20 represents 20 total hours of work for the week, and 60 is +60 on top of the standard for (or 100 hours for the week). The hours are represented on the x-axis. On the y-axis, is some measure of how happy my boss is with me for the same week. The higher this value, the happier my boss is with me.

Use this information and the plot to answer the following questions.



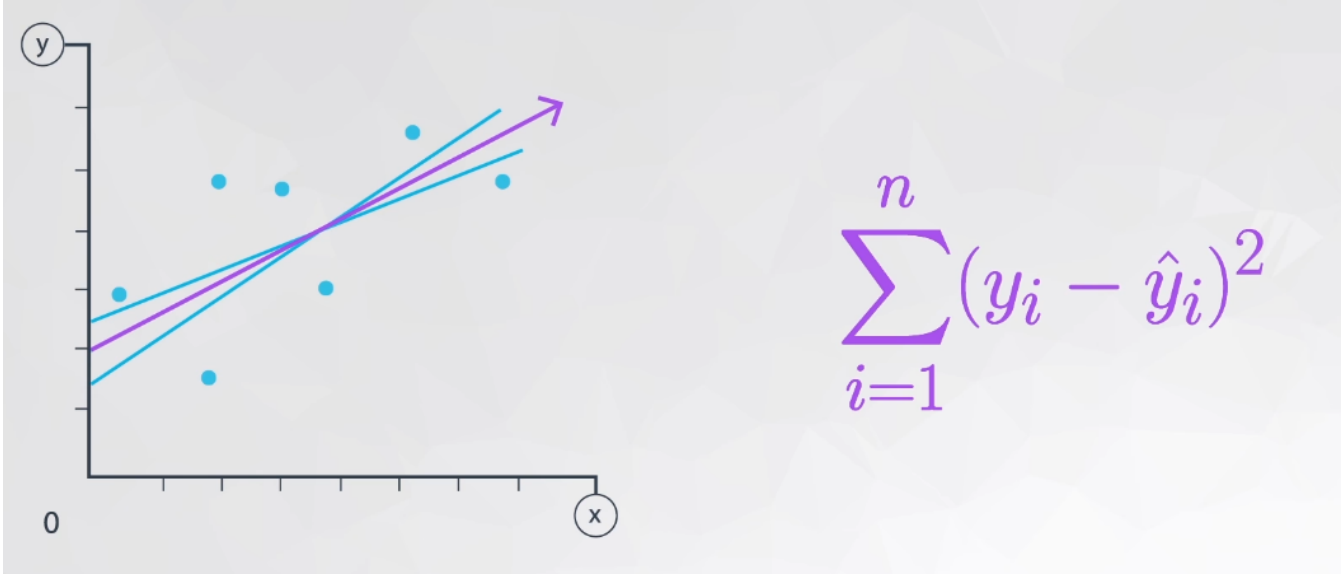
Using the scenario described and the above plot, mark all of the below that are correct.

- [x] If you work 40 hours per week, we expect your boss to be at a happiness level of 5.
- [x] As you work more hours, we expect the happiness of your boss to increase.
- [ ] Working more causes your boss to become more happy.
- [x] For each additional hour worked, we can predict an approximately 0.125 increase in the happiness of your boss.
- [ ] For each additional hour worked, we can predict approximately a 5 unit increase in the happiness of your boss.

Notice zero is actually 40 hours a week here. 40 on the x-axis is 40 more hours than 40 hours a week.

Fitting A Regression Line

The main algorithm used to find the best fit line is called the least-squares algorithm, which finds the line that minimizes: $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.



There are other ways we might choose a "best" line, but this algorithm tends to do a good job in many scenarios.

Fitting A Regression Line in Python

In [115…
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
```

In [ ]:
```
df = pd.read_csv('')
df.head()
```

	Unnamed: 0	price	area
0	0	598291	1188
1	1	1744259	3512
2	2	571669	1134
3	3	493675	1940
4	4	1101539	2208

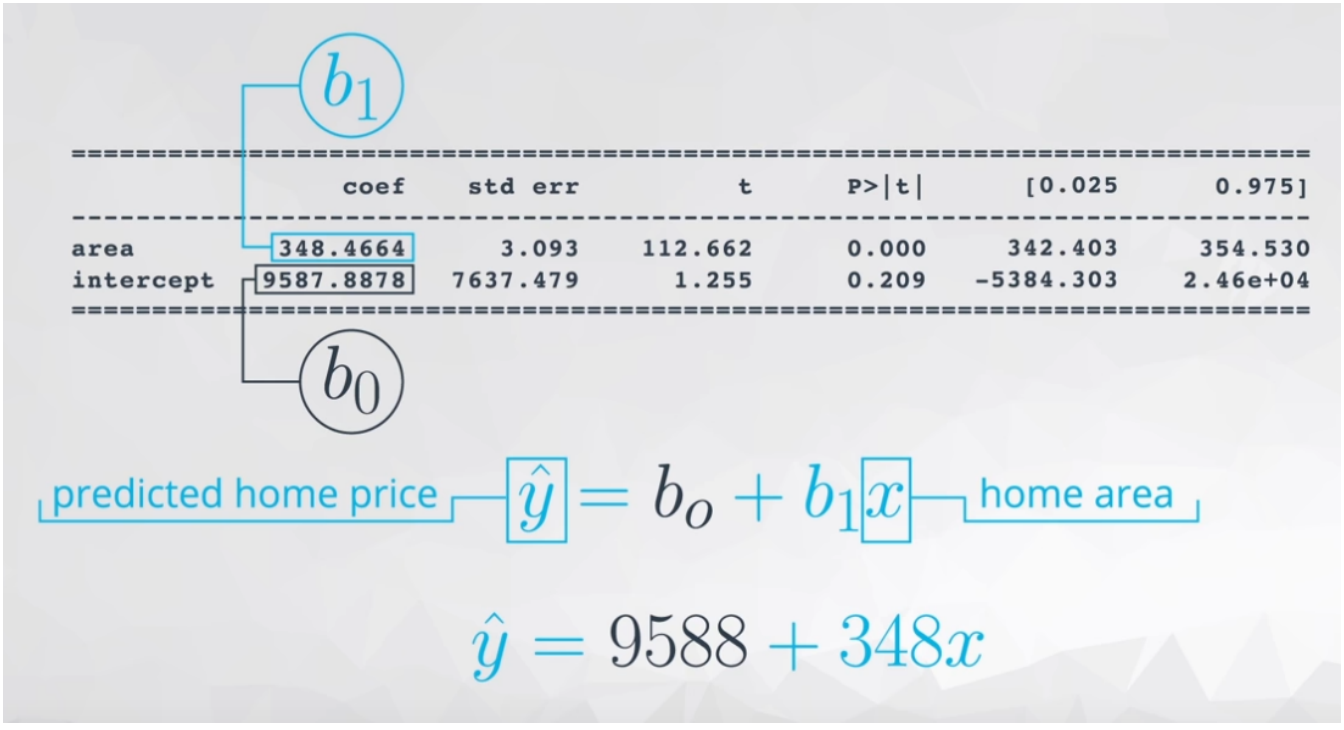
In [ ]:
```
# make sure to add a column for our intercept
df['intercept'] = 1

# upload ordinary least squares(OLS) for x & y variable
# first we privide Y then a list of the X variable
lm = sm.OLS(df['price'], df[['intercept','area']])

# fitting the model with .fit
results = lm.fit()

# look at the summary
results.summary()
```

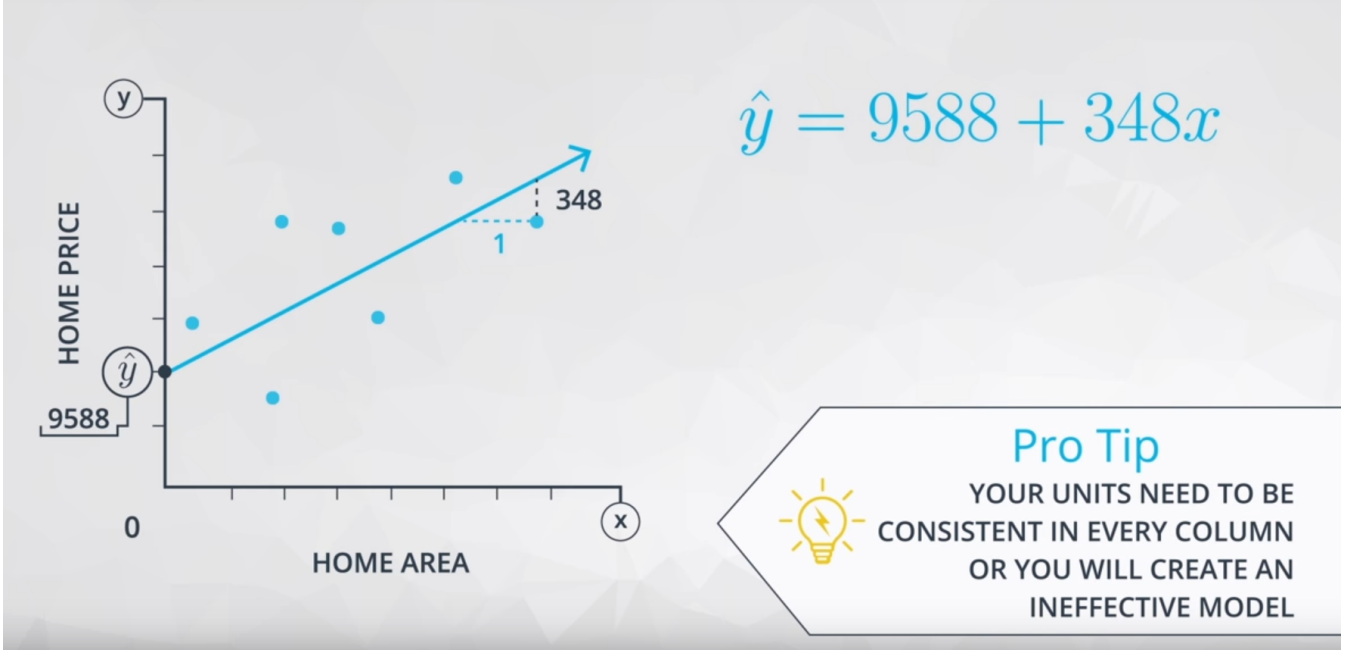
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.678
Model:                            OLS   Adj. R-squared:                  0.678
Method:                 Least Squares   F-statistic:                 1.269e+04
Date:                Thu, 02 Nov 2017   Prob (F-statistic):               0.00
Time:                        13:03:24   Log-Likelihood:                -84517.
No. Observations:                6028   AIC:                         1.690e+05
Df Residuals:                    6026   BIC:                         1.681e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
area         348.4664      3.093    112.662      0.000     342.403     354.530
intercept   9587.8878   7637.479      1.255      0.209   -5384.303    2.46e+04
==============================================================================
Omnibus:                      368.609   Durbin-Watson:                   2.007
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              349.279
Skew:                           0.534   Prob(JB):                     1.43e-76
Kurtosis:                       2.499   Cond. No.                     4.93e +03
==============================================================================
```

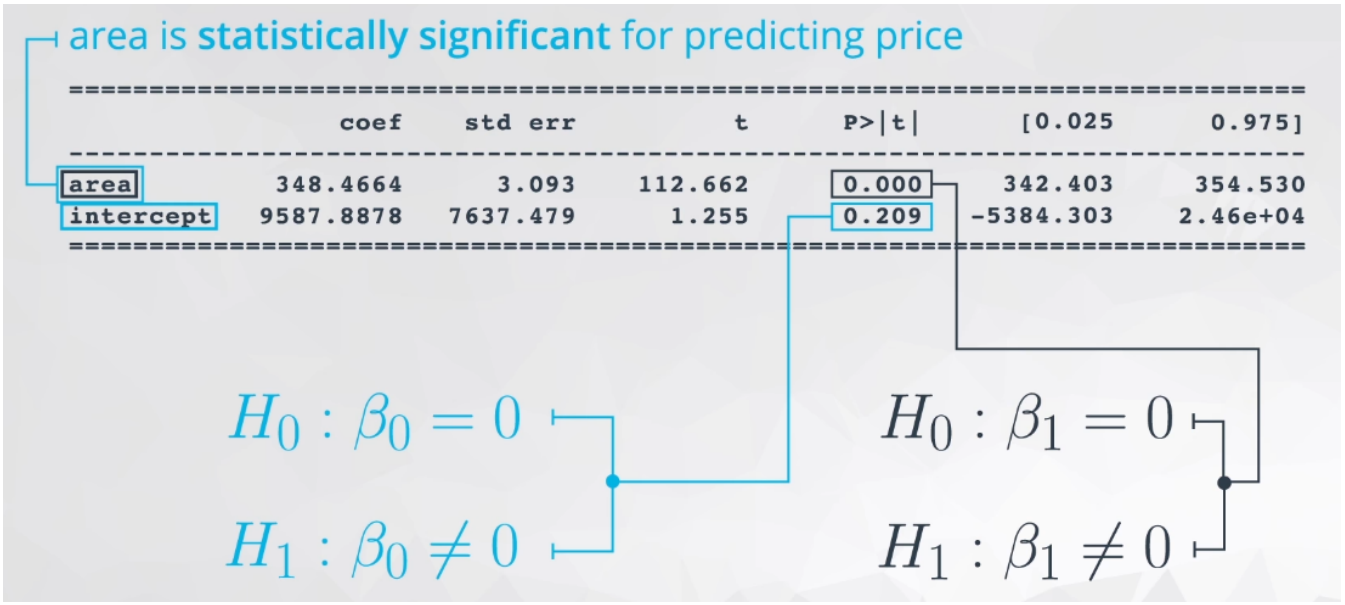
How to Interpret the Results?

it's important that you can interpret the results that you obtain from the line you fit, you might have noticed that the summary of the linear model this part

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
area         348.4664      3.093    112.662      0.000     342.403     354.530
intercept   9587.8878   7637.479      1.255      0.209   -5384.303    2.46e+04
==============================================================================
```







We can perform hypothesis tests for the coefficients in our linear models using Python (and other software). These tests help us determine if there is a statistically significant linear relationship between a particular variable and the response. The hypothesis test for the intercept isn't useful in most cases.

However, the hypothesis test for each x-variable is a test of if that population slope is equal to zero vs. an alternative where the parameter differs from zero. Therefore, if the slope is different than zero (the alternative is true), we have evidence that the x-variable attached to that coefficient has a statistically significant linear relationship with the response. This in turn suggests that the x-variable should help us in predicting the response (or at least be better than not having it in the model).

Does the Line Fit the Data Well?

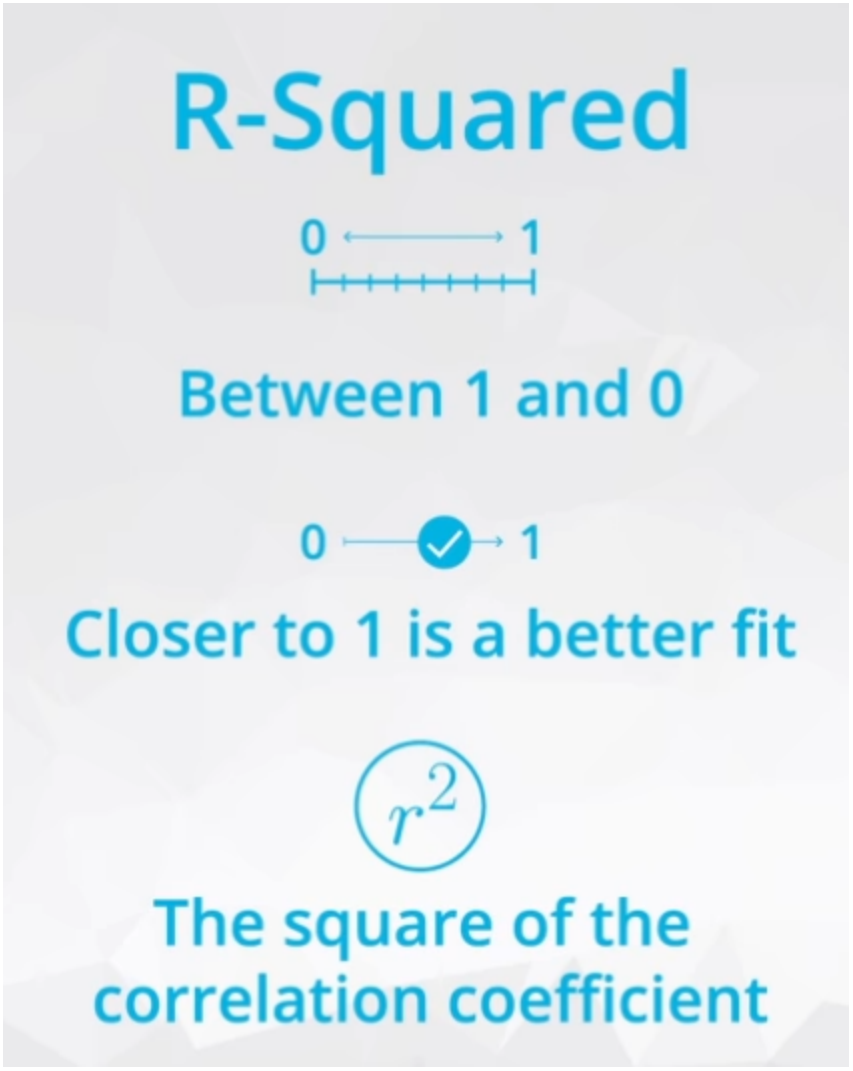
One of the most common techniques for understanding the relationship between two variables in regression, is the colleration coefficient. This part of output provide a bunch of additional measures,

OLS Regression Results

Dep. Variable:	price	R-squared:	0.678
Model:	OLS	Adj. R-squared:	0.678
Method:	Least Squares	F-statistic:	1.269e+04
Date:	Thu, 02 Nov 2017	Prob (F-statistic):	0.00
Time:	13:03:24	Log-Likelihood:	-84517.
No. Observations:	6028	AIC:	1.690e+05
Df Residuals:	6026	BIC:	1.681e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
area	348.4664	3.093	112.662	0.000	342.403	354.530
intercept	9587.8878	7637.479	1.255	0.209	-5384.303	2.46e+04

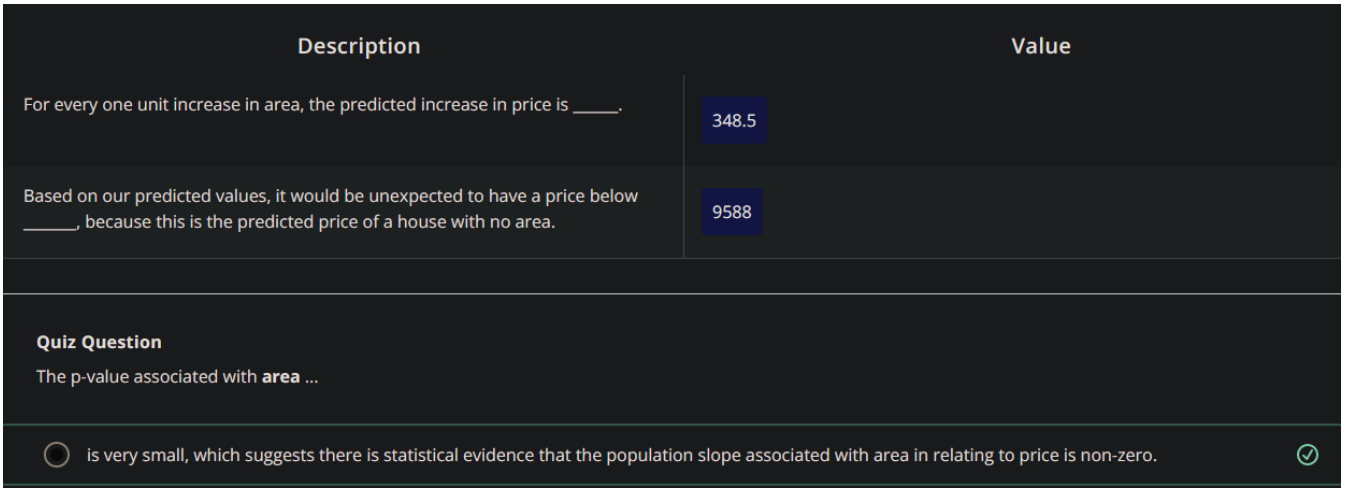
Omnibus:	368.609	Durbin-Watson:	2.007
Prob(Omnibus):	0.000	Jarque-Bera (JB):	349.279
Skew:	0.534	Prob(JB):	1.43e-76
Kurtosis:	2.499	Cond. No.	4.93e +03



The R-squared value is the square of the correlation coefficient.

A common definition for the R-squared variable is that it is the amount of variability in the response variable that can be explained by the x-variable in our model. In general, the closer this value is to 1, the better our model fits the data.

Many feel that R-squared isn't a great measure (which is possibly true), but I would argue that using cross-validation can assist us with validating any measure that helps us understand the fit of a model to our data.



Quiz: Regression - Your Turn - Part I

Regression Carats vs. Price

In this notebook, you will perform a similar analysis to the one you did in the previous notebook, but using a dataset holding the weight of a diamond in carats, and the price of the corresponding diamond in dollars.

To get started, let's read in the necessary libraries and the dataset.

In [117…
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('./carats.csv', header= None)
df.columns = ['carats', 'price']
df.head()
```

Out[117]:

	carats	price
0	0.17	355
1	0.16	328
2	0.17	350
3	0.18	325
4	0.25	642

`1.` Similar to the last notebook, fit a simple linear regression model to predict price based on the weight of a diamond. Use your results to answer the first question below. Don't forget to add an intercept.

In [120…
```
df['intercept'] = 1

lm = sm.OLS(df['carats'], df[['intercept', 'price']])
results = lm.fit()
results.summary()
```

Out[120]:

OLS Regression Results

Dep. Variable:	carats	**R-squared:**	0.978
Model:	OLS	**Adj. R-squared:**	0.978
Method:	Least Squares	**F-statistic:**	2070.
Date:	Mon, 03 Jul 2023	**Prob (F-statistic):**	6.75e-40
Time:	15:50:09	**Log-Likelihood:**	161.97
No. Observations:	48	**AIC:**	-319.9
Df Residuals:	46	**BIC:**	-316.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	0.0727	0.003	23.171	0.000	0.066	0.079
price	0.0003	5.78e-06	45.497	0.000	0.000	0.000

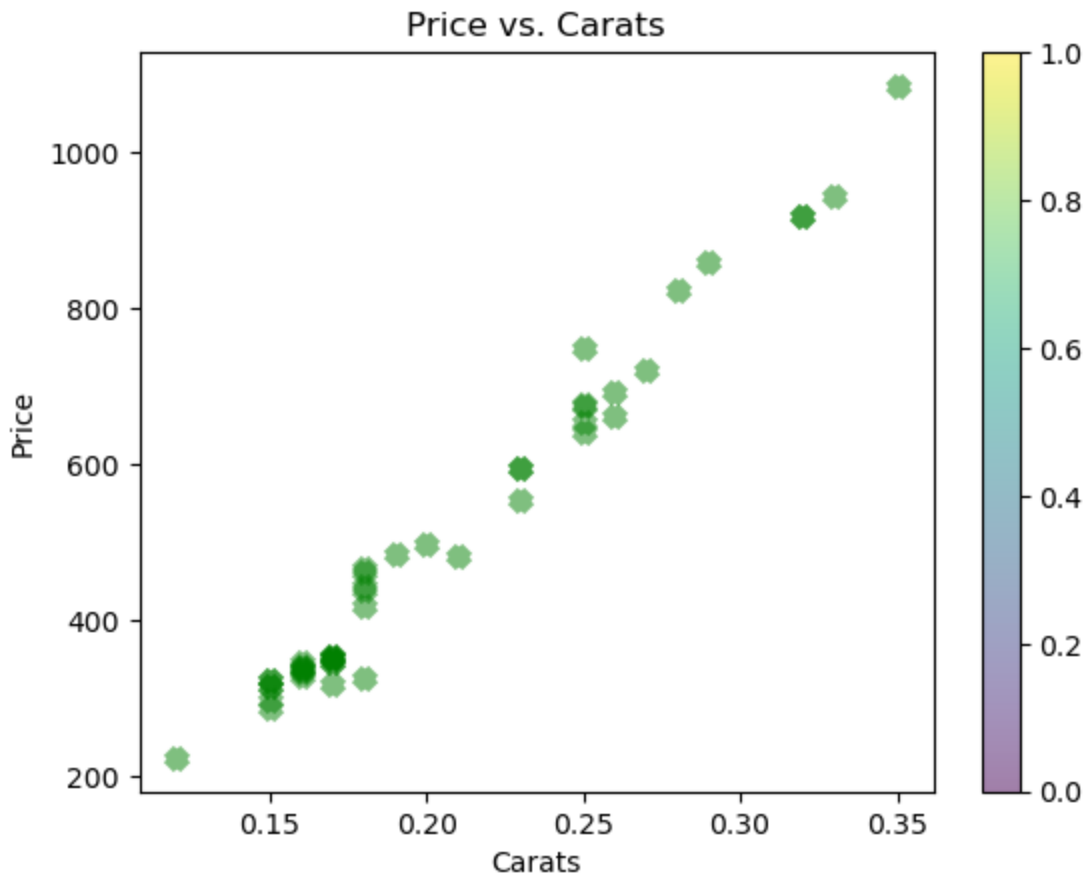
Omnibus:	0.138	**Durbin-Watson:**	2.047
Prob(Omnibus):	0.933	**Jarque-Bera (JB):**	0.002
Skew:	-0.001	**Prob(JB):**	0.999
Kurtosis:	2.966	**Cond. No.**	1.39e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.39e+03. This might indicate that there are strong multicollinearity or other numerical problems.

`2.` Use scatter to create a scatterplot of the relationship between price and weight. Then use the scatterplot and the output from your regression model to answer the second quiz question below.

In [133…

```
plt.scatter(df['carats'], df['price'], c = 'g', alpha = 0.5, marker='x', linewidth=5);
cbar = plt.colorbar()

plt.xlabel('Carats');
plt.ylabel('Price');
plt.title('Price vs. Carats');
```



Description	Value
For every 0.01 carat increase in the carat size, we can expect the price to increase by _____ dollars.	37.21
____% of the variability in price can be explained by the diamonds size.	97.8
There are _____ diamonds in the dataset.	48

Quiz Question

Using the R-squared value and the relationship in the scatterplot, what is the value of the correlation coefficient?

- ○ 0.978
- ○ -0.978
- ● 0.99 ✓
- ○ -0.99

The correlation coefficient is the square root of the Rsquared value.

If you were to see a scatterplot with a negative relationship, you would need to take the negative of the square root of the Rsquared value.

Quiz: Your Turn - Part II

In [3]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.datasets import load_boston
# This imports the load_boston function
# from the sklearn.datasets module.
# It allows us to load the Boston Housing dataset,
# which is a popular dataset used for regression analysis.
import matplotlib.pyplot as plt
%matplotlib inline

boston_data = load_boston()
df = pd.DataFrame()
df['MedianHomePrice'] = boston_data.target
df2 = pd.DataFrame(boston_data.data)
df['CrimePerCapita'] = df2.iloc[:,0];
df.head()
```

```
C:\Users\Tsgts\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarnin
g: Function load_boston is deprecated; `load_boston` is deprecated in 1.0 and will be re
moved in 1.2.

    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np


        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.

  warnings.warn(msg, category=FutureWarning)
```

Out[3]:

	MedianHomePrice	CrimePerCapita
0	24.0	0.00632
1	21.6	0.02731
2	34.7	0.02729
3	33.4	0.03237
4	36.2	0.06905

The Boston housing data is a built in dataset in the sklearn library of python. You will be using two of the variables from this dataset, which are stored in **df**. The median home price in thousands of dollars and the crime per capita in the area of the home are shown above.

`1.` Use this dataframe to fit a linear model to predict the home price based on the crime rate. Use your output to answer the first quiz below. Don't forget an intercept.

In [4]:
```
df['intercept'] = 1

lm = sm.OLS(df['MedianHomePrice'], df[['intercept', 'CrimePerCapita']])
results = lm.fit()
results.summary()
```

Out[4]:

OLS Regression Results

Dep. Variable:	MedianHomePrice	**R-squared:**	0.151
Model:	OLS	**Adj. R-squared:**	0.149
Method:	Least Squares	**F-statistic:**	89.49
Date:	Mon, 03 Jul 2023	**Prob (F-statistic):**	1.17e-19
Time:	16:58:10	**Log-Likelihood:**	-1798.9
No. Observations:	506	**AIC:**	3602.
Df Residuals:	504	**BIC:**	3610.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	24.0331	0.409	58.740	0.000	23.229	24.837
CrimePerCapita	-0.4152	0.044	-9.460	0.000	-0.501	-0.329

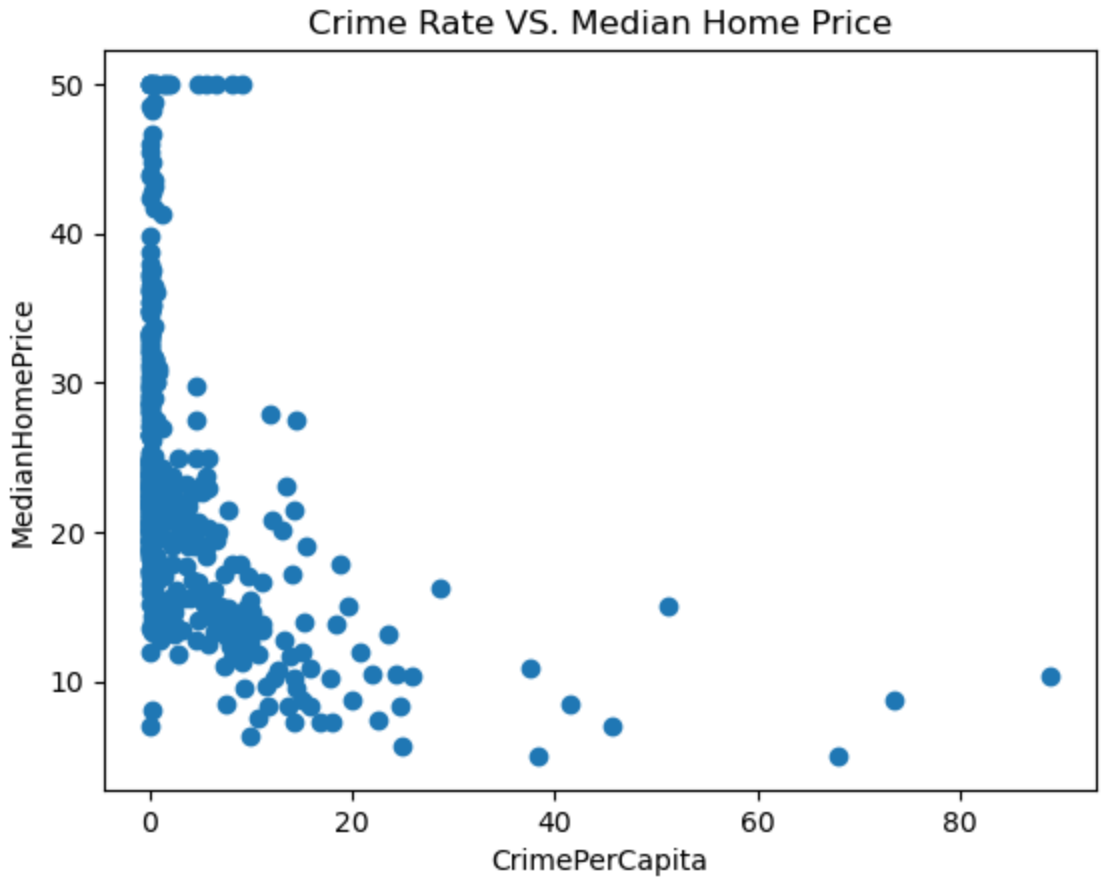
Omnibus:	139.832	**Durbin-Watson:**	0.713
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	295.404
Skew:	1.490	**Prob(JB):**	7.14e-65
Kurtosis:	5.264	**Cond. No.**	10.1

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

`2.` Plot the relationship between the crime rate and median home price below. Use your plot and the results from the first question as necessary to answer the remaining quiz questions below.

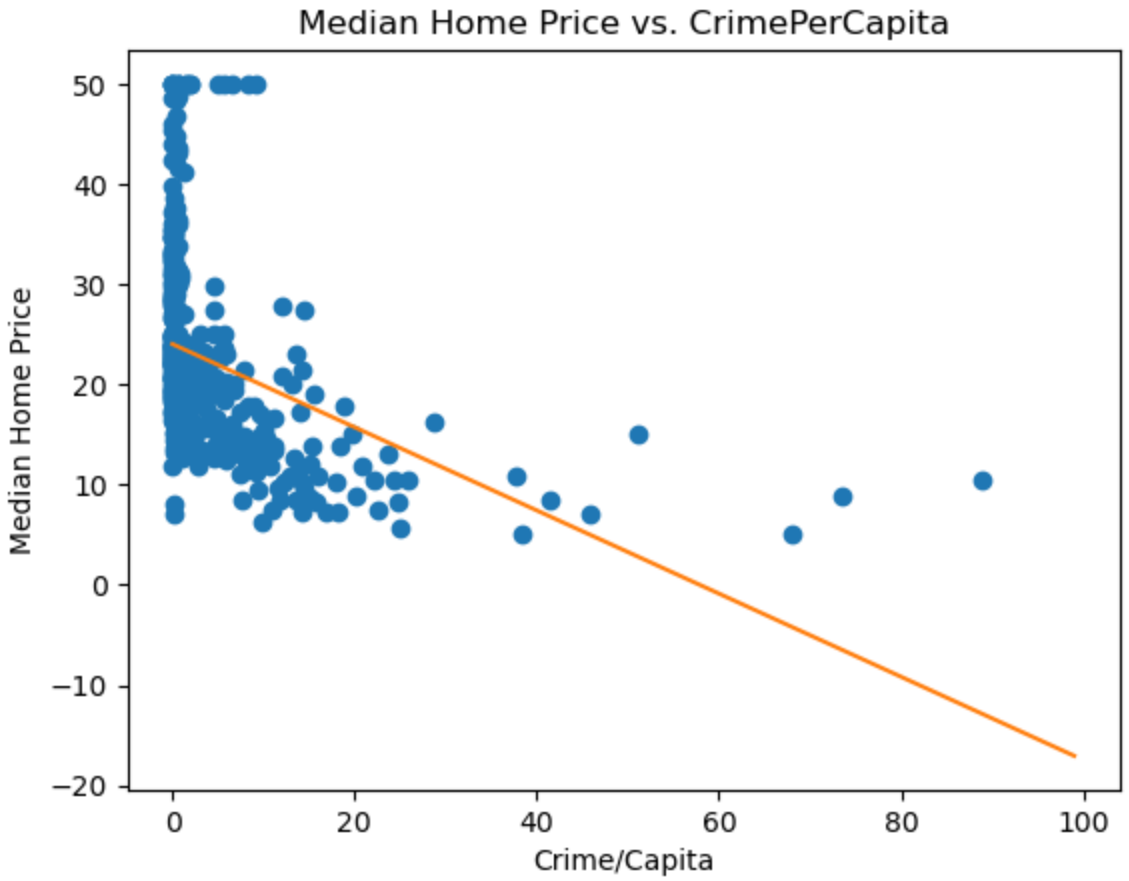
In [5]:
```
plt.scatter(df['CrimePerCapita'], df['MedianHomePrice'])

plt.xlabel('CrimePerCapita')
plt.ylabel('MedianHomePrice')
plt.title('Crime Rate VS. Median Home Price')
```

Out[5]:
```
Text(0.5, 1.0, 'Crime Rate VS. Median Home Price')
```



In [6]:
```
## To show the line that was fit I used the following code from
## https://plot.ly/matplotlib/linear-fits/
## It isn't the greatest fit... but it isn't awful either


import chart_studio.plotly as py
import plotly.graph_objs as go

# MatPlotlib
import matplotlib.pyplot as plt
from matplotlib import pylab

# Scientific libraries
from numpy import arange,array,ones
from scipy import stats


xi = arange(0,100)
A = array([ xi, ones(100)])

# (Almost) linear sequence
y = df['MedianHomePrice']
x = df['CrimePerCapita']

# Generated linear fit
slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)
line = slope*xi+intercept
```

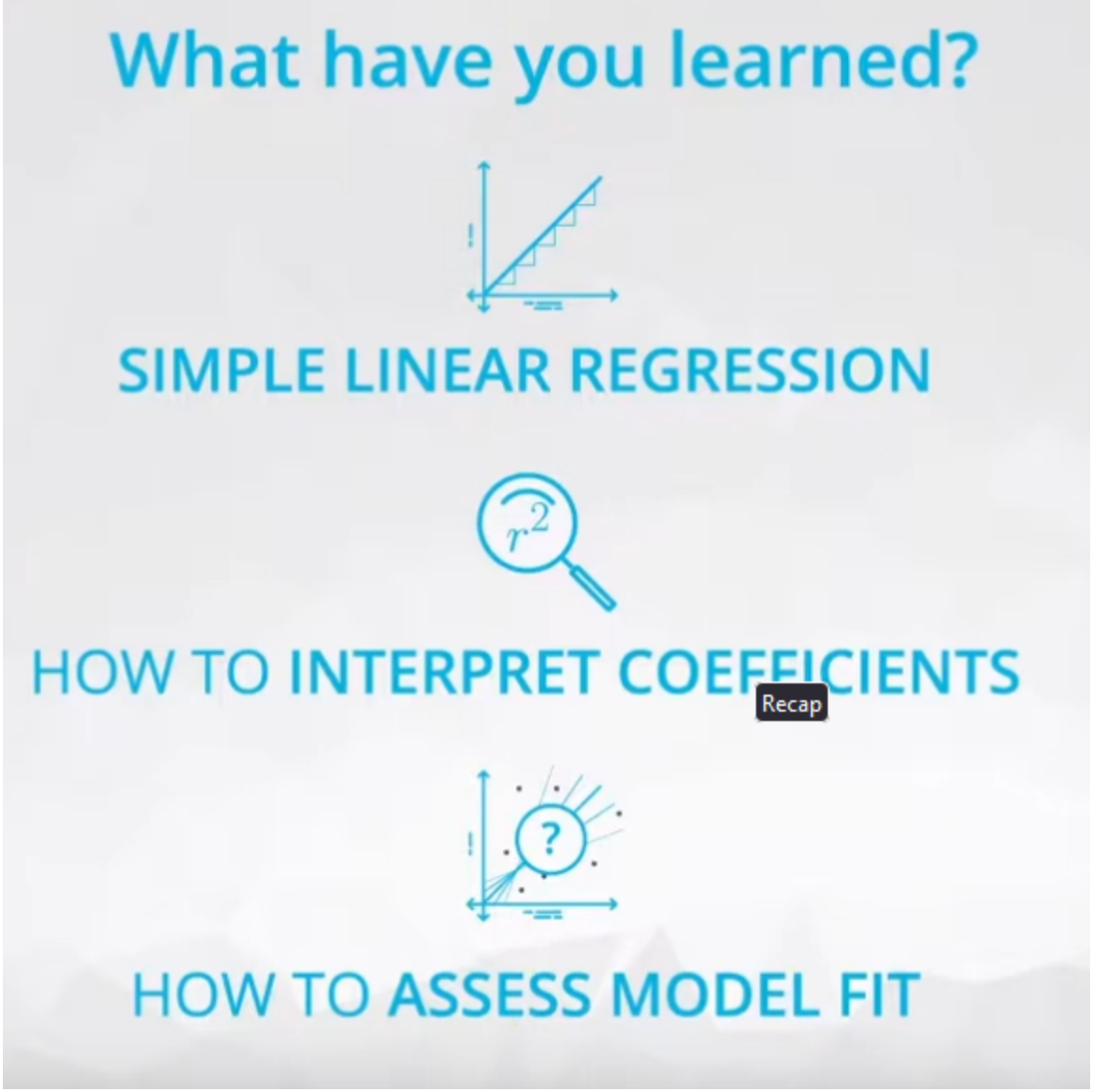
```
plt.plot(x,y,'o', xi, line);
plt.xlabel('Crime/Capita');
plt.ylabel('Median Home Price');
pylab.title('Median Home Price vs. CrimePerCapita');
```



Description	Value
For every 100% increase in crime per capita, the expected decrease in the median home price is _____ dollars.	412.80
If there was no crime, we would expect the median home price to be _____ dollars.	24016
_____% of the variability in price can be explained by the crime per capita.	14.9
The p-value of _____ associated with crime per capita suggests that it is statistically significant in providing information in predicting the median home values.	0.000

Real data is a bit messier, but can you tell based on your results from the linear model and your scatterplot what the value of the correlation coefficient should be? Mark your answer below.

- ○ 0.386
- ◉ -0.386 ✓
- ○ 0.149
- ○ -0.149



Multiple Linear Regression

In this lesson, you will be extending your knowledge of simple linear regression, where you were predicting a quantitative response variable using a quantitative explanatory variable. That is, you were using an equation that looked like this: $\hat{y} = b_0 + b_1 x_1$ In this lesson, you will learn about multiple linear regression. In these cases, you will be using both quantitative and categorical x-variables to predict a quantitative response. That is, you will be creating equations that like this to predict your response:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4$$

Fitting A Multiple Linear Regression Model

In [7]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm;

df = pd.read_csv('house_prices.csv')
df.head()
```

Out[7]:

	house_id	neighborhood	area	bedrooms	bathrooms	style	price
0	1112	B	1188	3	2	ranch	598291
1	491	B	3512	5	3	victorian	1744259
2	5952	B	1134	3	2	ranch	571669
3	3525	A	1940	4	2	ranch	493675
4	5108	B	2208	6	4	victorian	1101539

`1.` Using statsmodels, fit three individual simple linear regression models to predict price. You should have a model that uses **area**, another using **bedrooms**, and a final one using **bathrooms**. You will also want to use an intercept in each of your three models.

In [14]:
```
df['intercept'] = 1
lm_bath = sm.OLS(df['price'], df[['intercept', 'bathrooms']])
results_bath = lm_bath.fit()
results_bath.summary()
```

Out[14]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.541
Model:	OLS	**Adj. R-squared:**	0.541
Method:	Least Squares	**F-statistic:**	7116.
Date:	Mon, 03 Jul 2023	**Prob (F-statistic):**	0.00
Time:	18:27:51	**Log-Likelihood:**	-85583.
No. Observations:	6028	**AIC:**	1.712e+05
Df Residuals:	6026	**BIC:**	1.712e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	4.314e+04	9587.189	4.500	0.000	2.43e+04	6.19e+04
bathrooms	3.295e+05	3905.540	84.358	0.000	3.22e+05	3.37e+05

Omnibus:	915.429	**Durbin-Watson:**	2.003
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1537.531
Skew:	1.010	**Prob(JB):**	0.00
Kurtosis:	4.428	**Cond. No.**	5.84

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [15]:
```
lm_bed = sm.OLS(df['price'], df[['intercept', 'bedrooms']])
results_bed = lm_bed.fit()
results_bed.summary()
```

Out[15]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.553
Model:	OLS	**Adj. R-squared:**	0.553
Method:	Least Squares	**F-statistic:**	7446.
Date:	Mon, 03 Jul 2023	**Prob (F-statistic):**	0.00
Time:	18:28:08	**Log-Likelihood:**	-85509.
No. Observations:	6028	**AIC:**	1.710e+05
Df Residuals:	6026	**BIC:**	1.710e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	-9.485e+04	1.08e+04	-8.762	0.000	-1.16e+05	-7.36e+04
bedrooms	2.284e+05	2646.744	86.289	0.000	2.23e+05	2.34e+05

Omnibus:	967.118	**Durbin-Watson:**	2.014
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1599.431
Skew:	1.074	**Prob(JB):**	0.00
Kurtosis:	4.325	**Cond. No.**	10.3

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [16]:

```
lm_area = sm.OLS(df['price'], df[['intercept', 'area']])
results_area = lm_area.fit()
results_area.summary()
```

Out[16]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.678
Model:	OLS	**Adj. R-squared:**	0.678
Method:	Least Squares	**F-statistic:**	1.269e+04
Date:	Mon, 03 Jul 2023	**Prob (F-statistic):**	0.00
Time:	18:28:24	**Log-Likelihood:**	-84517.
No. Observations:	6028	**AIC:**	1.690e+05
Df Residuals:	6026	**BIC:**	1.691e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	9587.8878	7637.479	1.255	0.209	-5384.303	2.46e+04
area	348.4664	3.093	112.662	0.000	342.403	354.530

Omnibus:	368.609	**Durbin-Watson:**	2.007
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	349.279
Skew:	0.534	**Prob(JB):**	1.43e-76
Kurtosis:	2.499	**Cond. No.**	4.93e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.93e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

the coefficients had positive and negative values. Therefore, we can interpret each coefficient as the predicted increase or decrease in the response for every one-unit increase in the explanatory variable, holding all other variables in the model constant.

However, in general, coefficients might be positive or negative. Therefore, each coefficient is the predicted change in the response for every one-unit increase in the explanatory variable, holding all other variables in the model constant.

`2.` Now that you have looked at the results from the simple linear regression models, let's try a multiple linear regression model using all three of these variables at the same time. You will still want an intercept in this model.

In [20]:

```
mlr = sm.OLS(df['price'], df[['intercept', 'area', 'bedrooms', 'bathrooms']])
results_mlr = mlr.fit()
results_mlr.summary()
```

Out[20]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.678
Model:	OLS	**Adj. R-squared:**	0.678
Method:	Least Squares	**F-statistic:**	4230.
Date:	Mon, 03 Jul 2023	**Prob (F-statistic):**	0.00
Time:	18:31:36	**Log-Likelihood:**	-84517.
No. Observations:	6028	**AIC:**	1.690e+05
Df Residuals:	6024	**BIC:**	1.691e+05
Df Model:	3		
Covariance Type:	nonrobust		

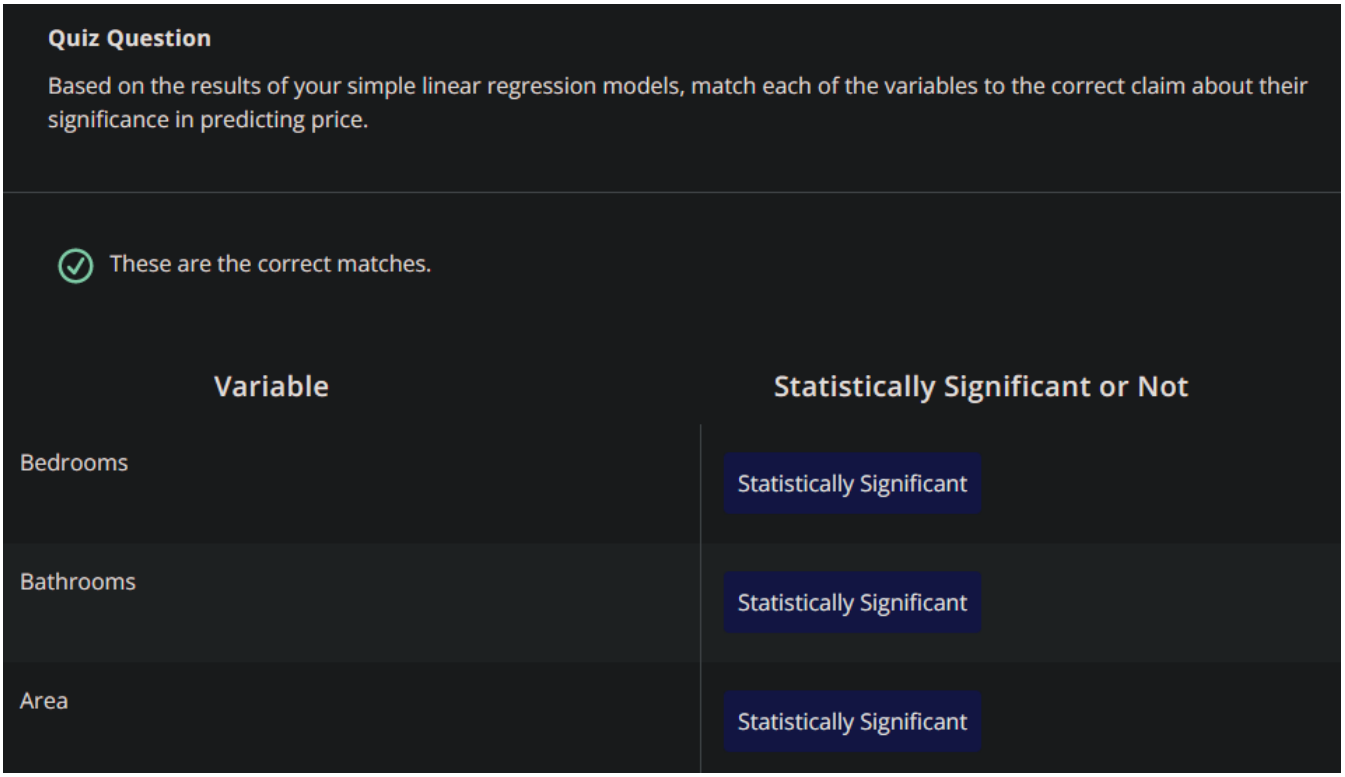
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
intercept	1.007e+04	1.04e+04	0.972	0.331	-1.02e+04	3.04e+04
area	345.9110	7.227	47.863	0.000	331.743	360.079
bedrooms	-2925.8063	1.03e+04	-0.285	0.775	-2.3e+04	1.72e+04
bathrooms	7345.3917	1.43e+04	0.515	0.607	-2.06e+04	3.53e+04

Omnibus:	367.658	**Durbin-Watson:**	2.007
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	350.116
Skew:	0.536	**Prob(JB):**	9.40e-77
Kurtosis:	2.503	**Cond. No.**	1.16e+04

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.16e+04. This might indicate that there are strong multicollinearity or other numerical problems.



if p_value < 0.05 MEANS is it Statistically Significant

In the context of regression analysis, if the p-value associated with a coefficient is less than 0.05, it means that the coefficient is statistically significant. This implies that there is strong evidence to suggest that the corresponding independent variable has a non-zero effect on the dependent variable. On the other hand, if the p-value is greater than 0.05, it suggests that there is not enough evidence to conclude that the coefficient is significantly different from zero.

Therefore, if p < 0.05, it indicates that the results are statistically significant, and the null hypothesis is rejected in favor of the alternative hypothesis. This is generally considered a good thing because it suggests that there is substantial evidence to support the presence of a relationship or effect being tested in the study.

Review Null & Alternative hypo in arabic :

الفرضية الصفر (Null Hypothesis):
الفرضية الصفر هي الفرضية التي يتم تقييمها واختبارها في الدراسة الإحصائية. تفترض الفرضية الصفر عدم وجود أو عدم وجود تأثير أو اختلاف بين المتغيرات أو العينات المدروسة. على سبيل المثال، إذا كنا ندرس تأثير علاج معين، فإن الفرضية الصفر تفترض أن ليس هناك تأثير فعلي للعلاج.

الفرضية البديلة (Alternative Hypothesis):
الفرضية البديلة هي الفرضية التي يتم اختبارها كبديل للفرضية الصفر. تتضمن الفرضية البديلة الافتراض بوجود تأثير أو فروق بين المتغيرات أو العينات المدروسة. بالاستمرار في المثال السابق، الفرضية البديلة يمكن أن تكون أن العلاج له تأثير فعلي.

Quiz Question

Based on the results of your multiple linear regression model using all of the variables, match each of the variables to the correct claim about their significance in predicting price.

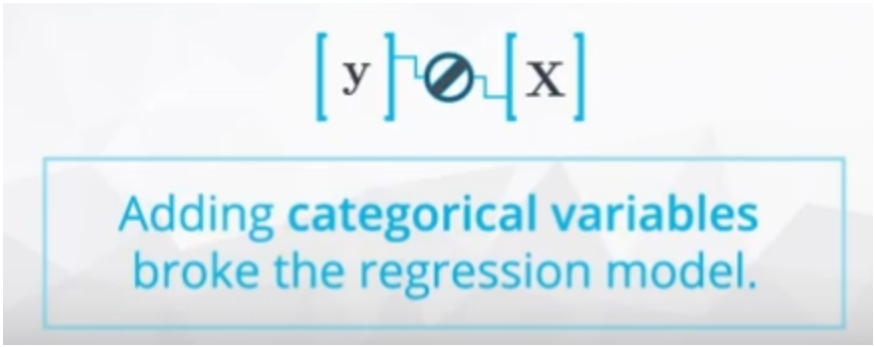
These are the correct matches.

Variable	Statistically Significant or Not
Bedrooms	Not Statistically Significant
Bathrooms	Not Statistically Significant
Area	Statistically Significant

Quiz Question

When adding **style** to the multiple linear regression model, the following happens:

- We get another coefficient associated with style added to the model.
- We get a coefficient for each level of the style added to the model.
- The model is fit, but no output is provided in the summary.
- There is an error because an object cannot be added to the multiple linear regression model.



Dummy Variables

The way that we add categorical variables into our multiple linear regression models is by using dummy variables.

The most common way dummy variables are added is through 1, 0 encoding. In this encoding method, you create a new column for each level of a category (in this case A, B, or C). Then our new columns either hold a 1 or 0 depending on the presence of the level in the original column.

When we add these dummy variables to our multiple linear regression models, we always drop one of the columns. The column you drop is called the baseline. when you create dummy variables using 0, 1 encodings, you always need to drop one of the columns from the model to make sure your matrices are full rank (and that your solutions are reliable from Python).

Quiz Question

If you are interested in adding a categorical variable to a regression model that tells us the colors of a street light: **red**, **yellow**, and **green**, how many dummy variable columns will be added to your linear regression model?

- ○ 1
- ○ 2 ✓

Quiz Question

If I have a categorical variable with two levels **yes** or **no**, how many dummy variables would needed to be added to a linear model to use this variable?

- ○ 1 ✓
- ○ 2
- ○ 3

Quiz Question

Imagine you own a restaurant, and you have a ratings scale of: 'great', 'good', 'okay', 'poor', or 'awful'. You would like to understand the tip given based on this rating, so you build a linear model, using dummy variables to represent the ratings. **How many total coefficients are in your model?**

- ○ 3
- ○ 4
- ○ 5 ✓
- ○ 6

You would have the intercept plus 4 dummy variables for the other variables for a total of 5 coefficients.

Quiz Question

Which of the below are true regarding the dummy variables we add to our multiple linear regression models? Let X be the X matrix as defined in the previous Screencast. **Mark all that are true.**

- ☑ There should always be as many dummy variables added to your X matrix as the number of levels of each categorical variable minus 1. ✓
- ☑ The reason for dropping a dummy variable is to assure that all of our columns are linearly independent. ✓
- ☑ The reason for dropping a dummy variable is to assure that the dot product of X'X is invertible. ✓
- ☑ The reason for dropping a dummy variable is to assure that your X matrix is full rank. ✓

Dummy Variables in Python

In [47]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm;
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('house_prices.csv')
df.head()
```

Out[47]:

	house_id	neighborhood	area	bedrooms	bathrooms	style	price
0	1112	B	1188	3	2	ranch	598291
1	491	B	3512	5	3	victorian	1744259
2	5952	B	1134	3	2	ranch	571669
3	3525	A	1940	4	2	ranch	493675
4	5108	B	2208	6	4	victorian	1101539

`1.` Use the pd.get_dummies documentation to assist you with obtaining dummy variables for the **neighborhood** column. Then use join to add the dummy variables to your dataframe, **df**, and store the joined results in **df_new**.

Fit a linear model using **all three levels** of **neighborhood** to predict the price. Don't forget an intercept.

Use your results to answer quiz 1 below.

In [48]:
```
neighborhood_dummies = pd.get_dummies(df['neighborhood'])
df_new = df.join(neighborhood_dummies)
df_new.head()
```

Out[48]:

	house_id	neighborhood	area	bedrooms	bathrooms	style	price	A	B	C
0	1112	B	1188	3	2	ranch	598291	0	1	0
1	491	B	3512	5	3	victorian	1744259	0	1	0
2	5952	B	1134	3	2	ranch	571669	0	1	0
3	3525	A	1940	4	2	ranch	493675	1	0	0
4	5108	B	2208	6	4	victorian	1101539	0	1	0

In [50]:
```
df_new['intercept'] = 1
lm = sm.OLS(df_new['price'], df_new[['intercept', 'A', 'B', 'C']])
results = lm.fit()
results.summary()
```

Out[50]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.246
Model:	OLS	**Adj. R-squared:**	0.246
Method:	Least Squares	**F-statistic:**	983.1
Date:	Tue, 04 Jul 2023	**Prob (F-statistic):**	0.00
Time:	00:11:53	**Log-Likelihood:**	-87082.
No. Observations:	6028	**AIC:**	1.742e+05
Df Residuals:	6025	**BIC:**	1.742e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	5.381e+05	4439.653	121.210	0.000	5.29e+05	5.47e+05
A	3001.8311	8650.726	0.347	0.729	-1.4e+04	2e+04
B	5.325e+05	7894.313	67.448	0.000	5.17e+05	5.48e+05
C	2669.4717	8925.271	0.299	0.765	-1.48e+04	2.02e+04

Omnibus:	689.315	**Durbin-Watson:**	1.999
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1154.155
Skew:	0.793	**Prob(JB):**	2.39e-251
Kurtosis:	4.442	**Cond. No.**	7.82e+15

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.32e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

2. Now, fit an appropriate linear model for using **neighborhood** to predict the price of a home. Use **neighborhood A** as your baseline. (And remember that the values shown in the results for the other neighborhoods will be based on comparisons with this baseline neighborhood A then.) Use your resulting model to answer the questions in Quiz 2 and Quiz 3 below.

In [43]:
```
df_new['intercept'] = 1

lm = sm.OLS(df_new['price'], df_new[['intercept', 'B', 'C']])
result = lm.fit()
result.summary()
```

Out[43]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.246
Model:	OLS	**Adj. R-squared:**	0.246
Method:	Least Squares	**F-statistic:**	983.1
Date:	Tue, 04 Jul 2023	**Prob (F-statistic):**	0.00
Time:	00:08:52	**Log-Likelihood:**	-87082.
No. Observations:	6028	**AIC:**	1.742e+05
Df Residuals:	6025	**BIC:**	1.742e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	5.411e+05	1.05e+04	51.537	0.000	5.21e+05	5.62e+05
B	5.295e+05	1.4e+04	37.870	0.000	5.02e+05	5.57e+05
C	-332.3594	1.52e+04	-0.022	0.983	-3.01e+04	2.94e+04

Omnibus:	689.315	**Durbin-Watson:**	1.999
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1154.155
Skew:	0.793	**Prob(JB):**	2.39e-251
Kurtosis:	4.442	**Cond. No.**	3.88

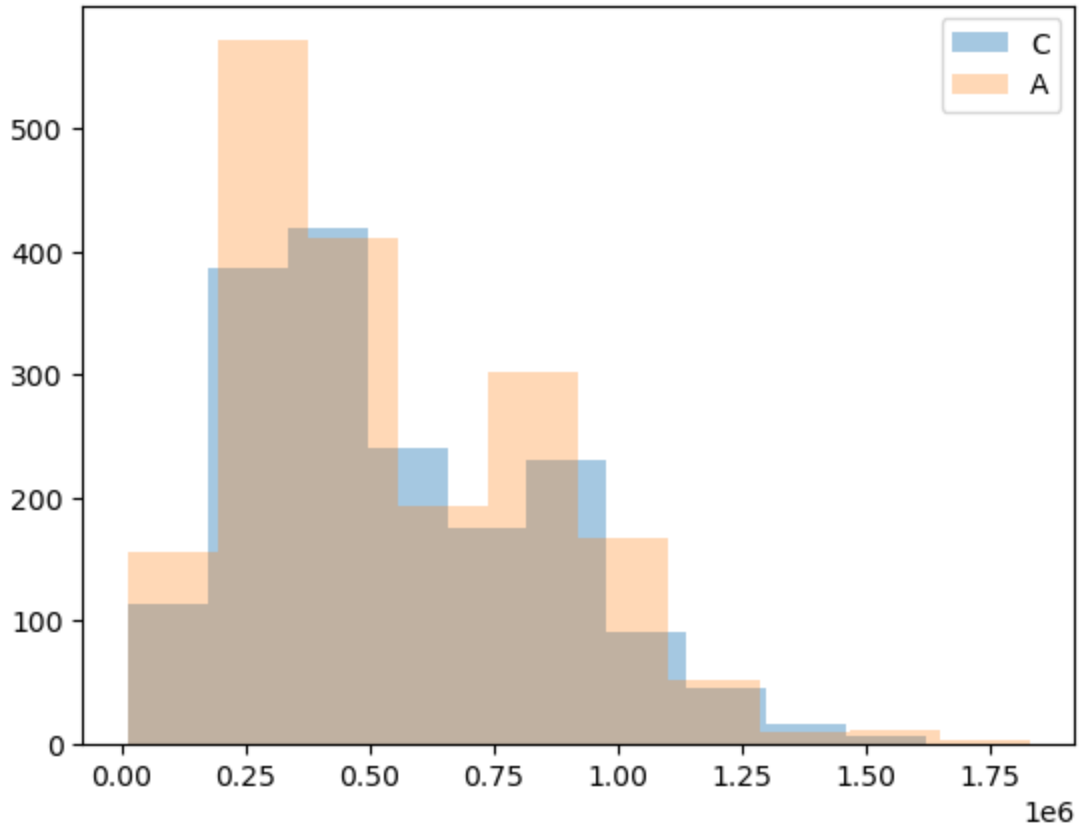
Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

`3.` Run the two cells below to look at the home prices for the A and C neighborhoods. Add neighborhood B. This creates a glimpse into the differences that you found in the previous linear model.

In [44]:

```
plt.hist(df_new.query("C == 1")['price'], alpha = 0.4, label = 'C');
plt.hist(df_new.query("A == 1")['price'], alpha = 0.3, label = 'A');

plt.legend();
```



4. Now, add dummy variables for the **style** of house. Create a new linear model using these new dummies, as well as the previous **neighborhood** dummies. Use **ranch** as the baseline for the **style**. Additionally, add **bathrooms** and **bedrooms** to your linear model. Don't forget an intercept. Use the results of your linear model to answer the last two questions below. **Home prices are measured in dollars, and this dataset is not real.**

To minimize scrolling, it might be useful to open another browser window to this concept to answer the quiz questions.

In [45]:
```
type_dummies = pd.get_dummies(df['style'])
df_new = df_new.join(type_dummies)
df_new.head()
```

Out[45]:

	house_id	neighborhood	area	bedrooms	bathrooms	style	price	A	B	C	intercept	lodge	ranch
0	1112	B	1188	3	2	ranch	598291	0	1	0	1	0	1
1	491	B	3512	5	3	victorian	1744259	0	1	0	1	0	0
2	5952	B	1134	3	2	ranch	571669	0	1	0	1	0	1
3	3525	A	1940	4	2	ranch	493675	1	0	0	1	0	1
4	5108	B	2208	6	4	victorian	1101539	0	1	0	1	0	0

In [46]:
```
lm3 = sm.OLS(df_new['price'], df_new[['intercept', 'B', 'C', 'lodge', 'victorian', 'bedr
results3 = lm3.fit()
results3.summary()
```

Out[46]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.809
Model:	OLS	**Adj. R-squared:**	0.809
Method:	Least Squares	**F-statistic:**	4250.
Date:	Tue, 04 Jul 2023	**Prob (F-statistic):**	0.00
Time:	00:08:53	**Log-Likelihood:**	-82944.
No. Observations:	6028	**AIC:**	1.659e+05
Df Residuals:	6021	**BIC:**	1.659e+05
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	-3.833e+05	1.2e+04	-31.995	0.000	-4.07e+05	-3.6e+05
B	5.229e+05	7040.928	74.271	0.000	5.09e+05	5.37e+05
C	-7168.6285	7639.254	-0.938	0.348	-2.21e+04	7807.045
lodge	1.685e+05	9906.629	17.012	0.000	1.49e+05	1.88e+05
victorian	7.056e+04	8337.790	8.463	0.000	5.42e+04	8.69e+04
bedrooms	1.732e+05	7677.152	22.558	0.000	1.58e+05	1.88e+05
bathrooms	9.996e+04	1.09e+04	9.164	0.000	7.86e+04	1.21e+05

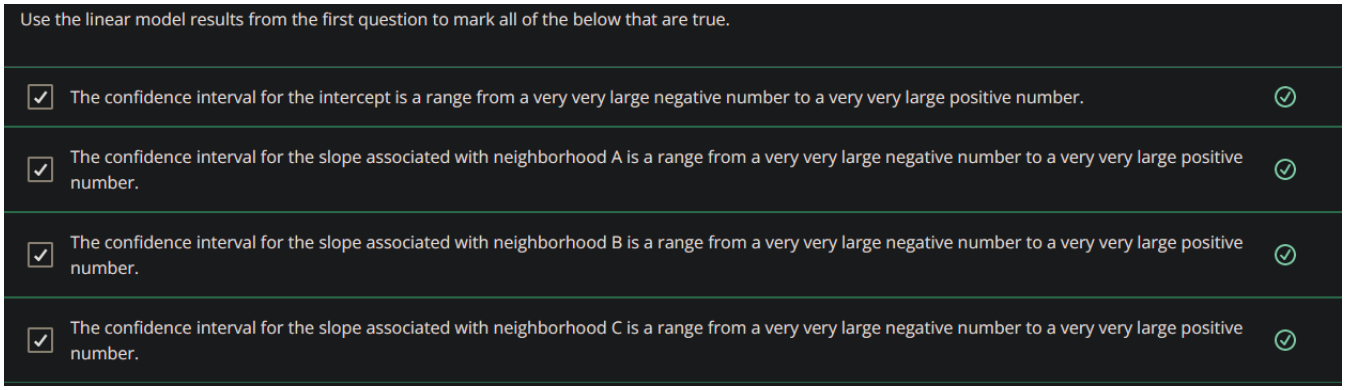
Omnibus:	978.611	**Durbin-Watson:**	1.993
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2926.472
Skew:	0.848	**Prob(JB):**	0.00
Kurtosis:	5.962	**Cond. No.**	25.9

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Confidence Interval is based on [0.025....0.975]

NOTE: neighborhood B is wrong not True



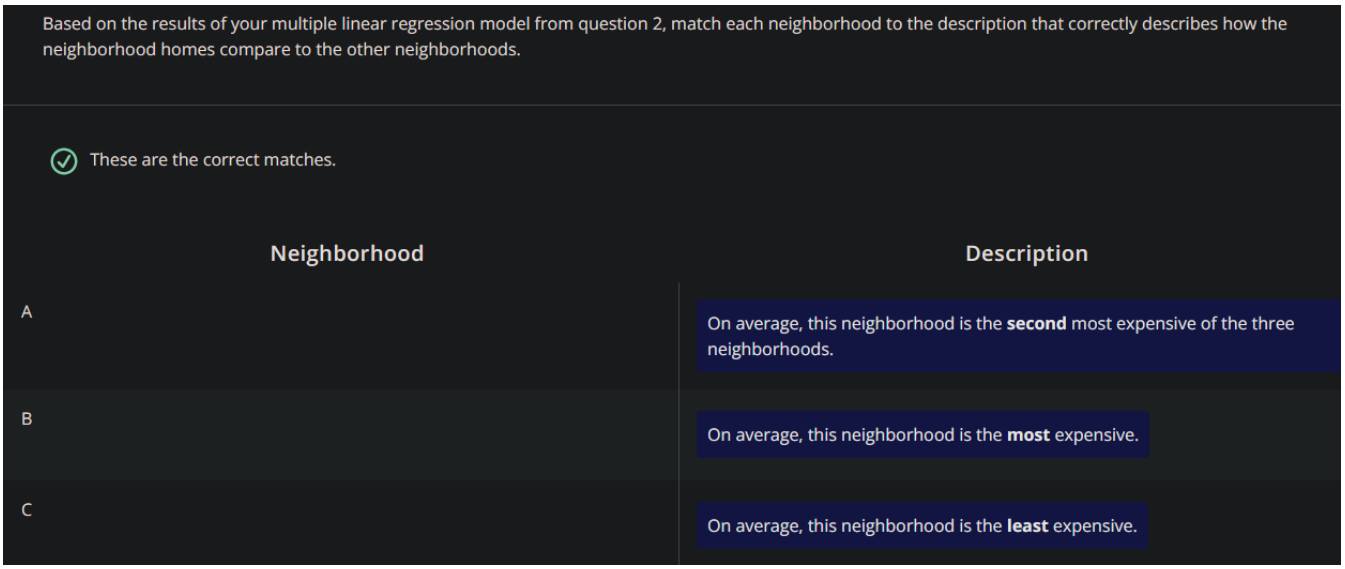
The Price is based on the coef

Below, the intercept means that if our neighborhood is A,

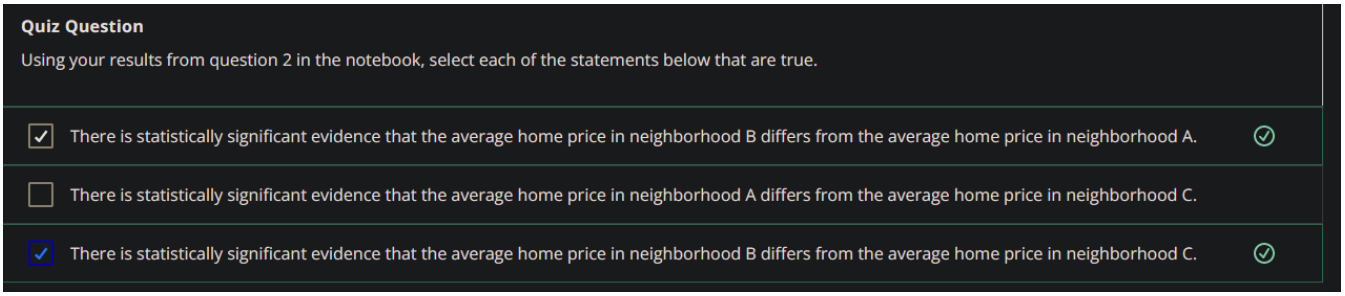
we predict its price to be 5.411e+05 dollars

and B is predicted to be 5.295e+05 more than a A.

And similarly,C is predicted to be 332.3594 less than A



You can look at the p-values to compare to neighborhood A. In order to compare neighborhood B to neighborhood C, you can compare the confidence intervals. Since the confidence intervals for B and C do not overlap, we have have evidence they differ as well



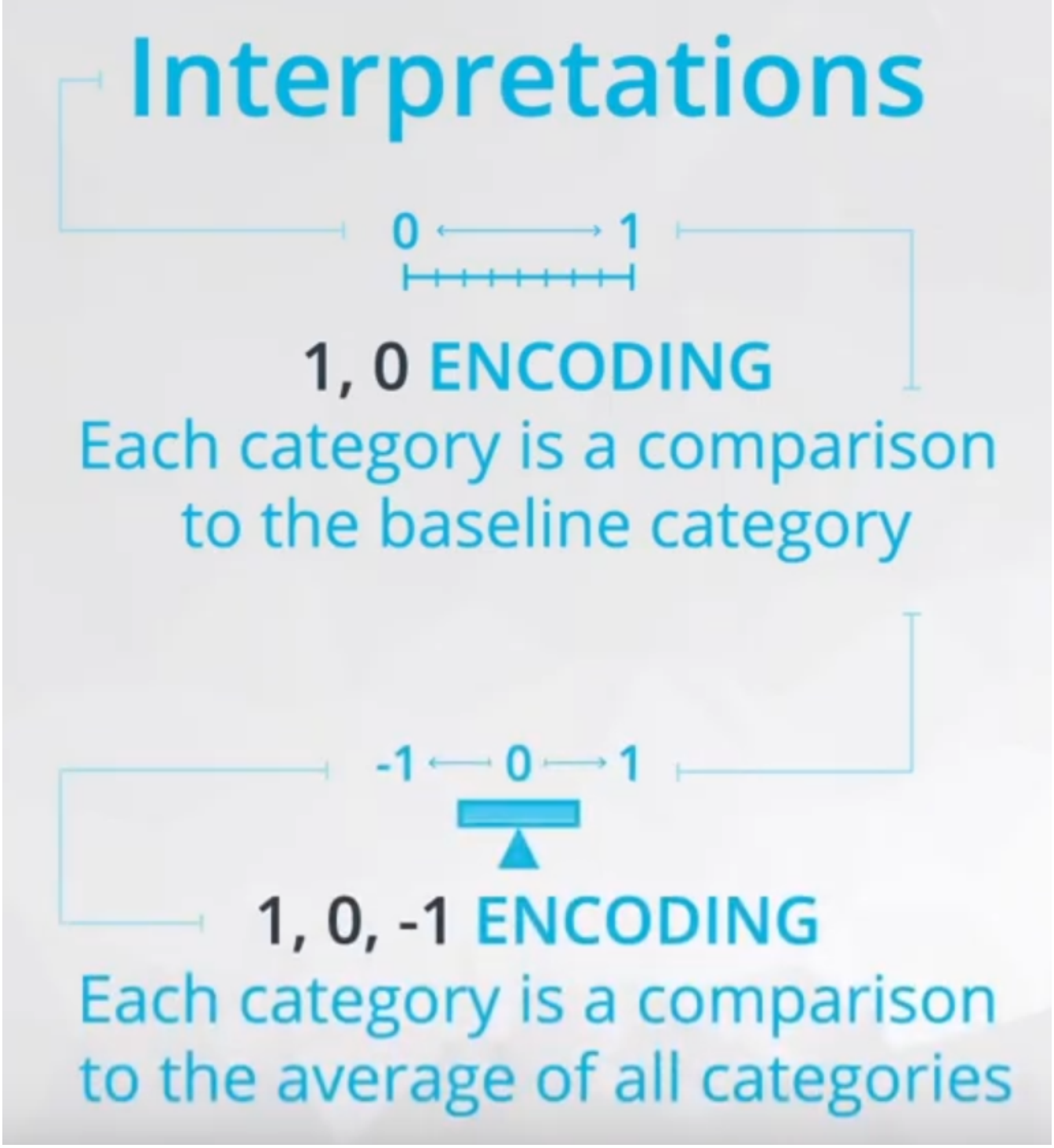
Check coef

Quiz Question

Use the results from your multiple linear regression model in question 4 above to match the values to the appropriate spot in the descriptions below.

These are the correct matches.

Description	Value
_____% of the variability in price can be explained by the linear model built using bedrooms, bathrooms, neighborhood, and home style.	80.9
For every additional bedroom a home has we can expect the price to increase by _____, holding all other variables constant.	173200
For every additional bathroom a home has we can expect the price to increase by _____, holding all other variables constant.	99960
We expect that a victorian house will cost _____ more than a _____ house, all else being equal.	70560, ranch
We expect that a house in neighborhood C will cost _____ less than a neighborhood _____ house, all else being equal.	7168, A



Potential Problems

When building multiple linear regression models, there are a number of problems that may arise. However, assessing a "problem" will depend on your use case.

- Is the focus understanding how all variables are related?
- Are you using regression to make predictions about the response variable?
- Is determining which variables are most useful for predicting the response, your goal?

The problems that can arise are:

1. A linear relationship may not exist
2. Correlated errors

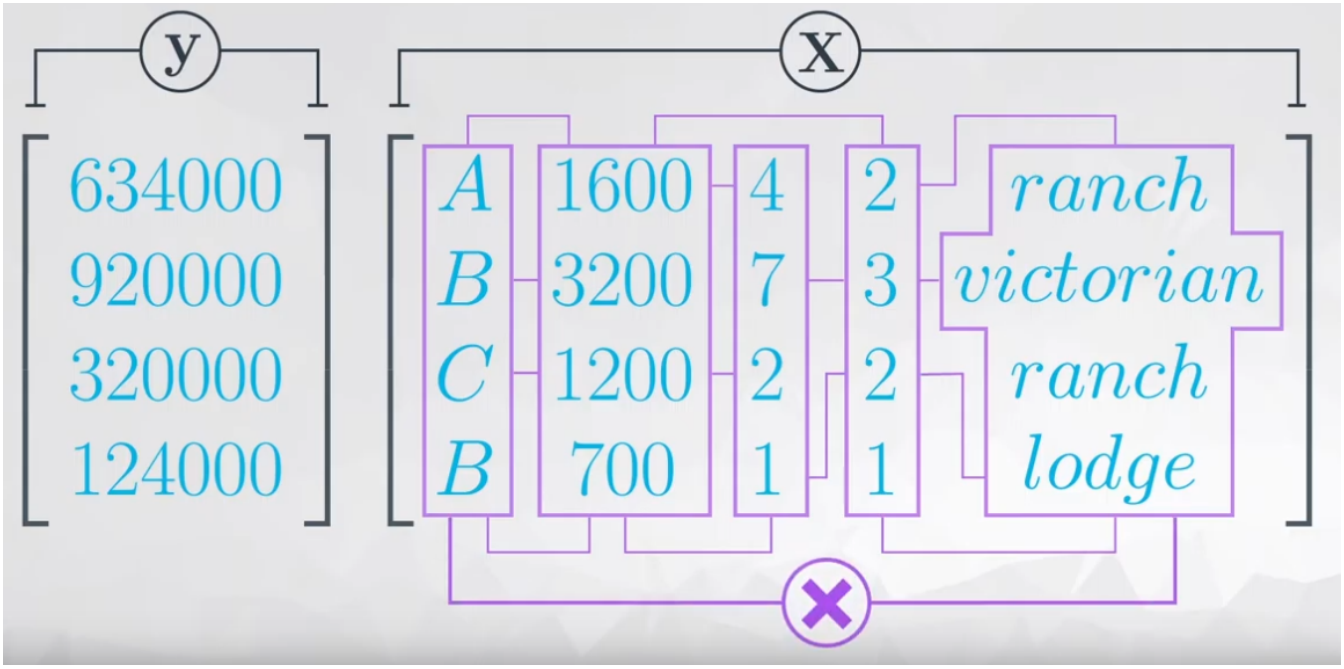
nt variance

4. Outliers
5. Multicollinearity

Multicollinearity & VIFs

Two different ways of identifying multicollinearity:

1. We can look at the correlation of each explanatory variable with each other explanatory variable (with a plot or the correlation coefficient).
2. We can look at Variance Inflation Factors (VIFs) for each variable. This calculation will be shown in more detail in the next video.



We would like x-variables to be related to the response, but not to be related to one another. When our x-variables are correlated with one another, this is known as multicollinearity. Multicollinearity has two potential negative impacts.

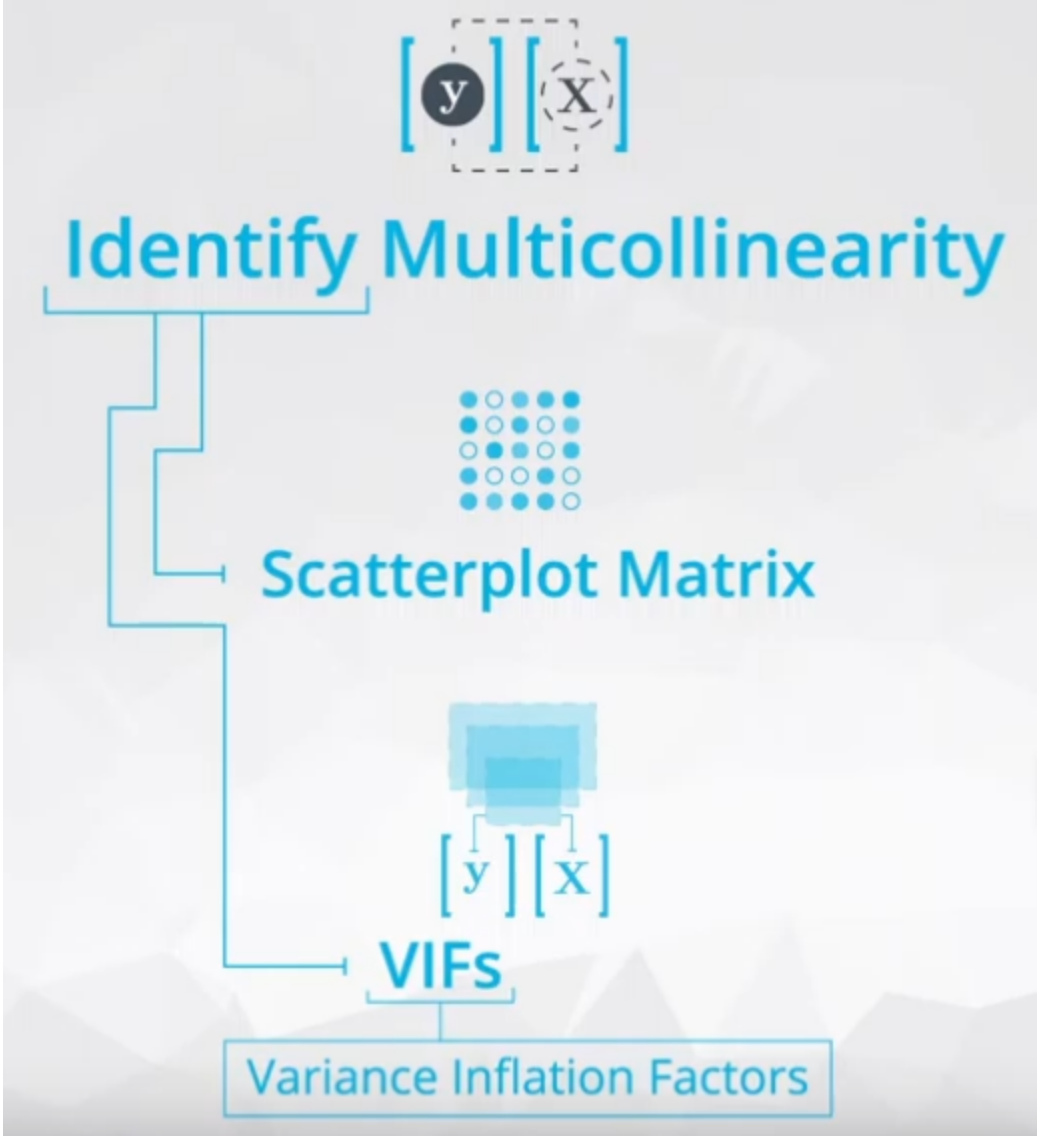
1. The expected relationships between your x-variables and the response may not hold when multicollinearity is present. That is, you may expect a positive relationship between the explanatory variables and the response (based on the bivariate relationships), but in the multiple linear regression case, it turns out the relationship is negative.

2. Our hypothesis testing results may not be reliable. It turns out that having correlated explanatory variables means that our coefficient estimates are less stable. That is, standard deviations (often called standard errors) associated with your regression coefficients are quite large. Therefore, a particular variable might be useful for predicting the response, but because of the relationship it has with other x-variables, you will no longer see this association.

different ways of identifying multicollinearity:

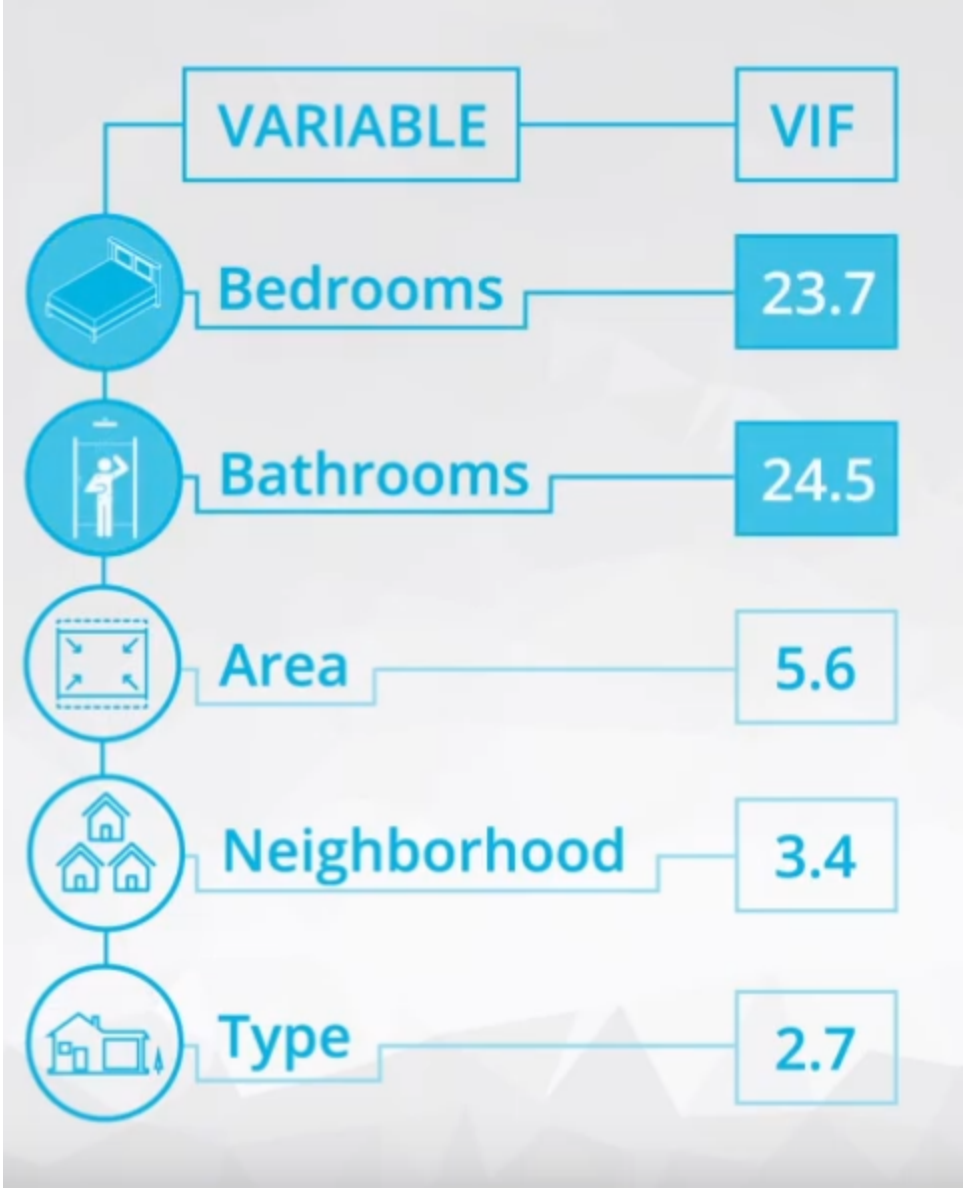
1. Looking at the correlation of each explanatory variable with each other explanatory variable (with a plot or the correlation coefficient).

VIFs for each variable.



When VIFs are greater than 10, this suggests that multicollinearity is certainly a problem in your model. Some experts even suggest that VIFs greater than 5 can be problematic. In most cases, not just one VIF is high, but rather many VIFs are high, as these are measures of how related variables are with one another.

The most common way of working with correlated explanatory variables in a multiple linear regression model is simply to remove one of the variables that is most related to the other variables. Choosing an explanatory variable that you aren't interested in, or isn't as important to you, is a common choice.



Multicollinearity & VIFs with Python

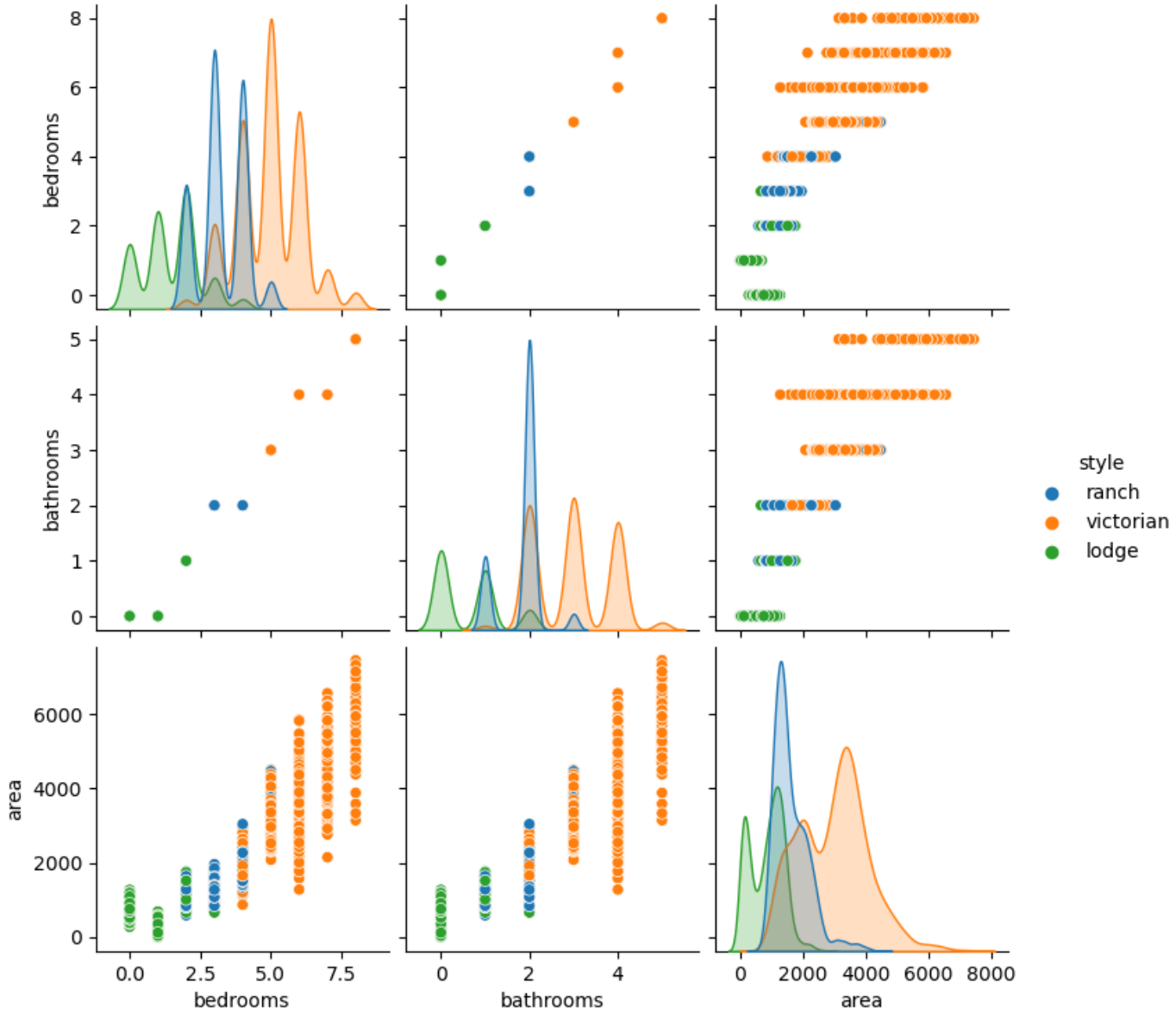
In [51]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from patsy import dmatrices
import statsmodels.api as sm;
from statsmodels.stats.outliers_influence import variance_inflation_factor
%matplotlib inline

df = pd.read_csv('house_prices.csv')
df.head()
```

Out[51]:

	house_id	neighborhood	area	bedrooms	bathrooms	style	price
0	1112	B	1188	3	2	ranch	598291
1	491	B	3512	5	3	victorian	1744259
2	5952	B	1134	3	2	ranch	571669
3	3525	A	1940	4	2	ranch	493675
4	5108	B	2208	6	4	victorian	1101539

1. Use seaborn to look at pairwise relationships for all of the quantitative, explanatory variables in the dataset by running the cell below. You might also investigate how to add color (**hue**) for the house style or neighborhood. Use the plot to answer the first quiz questions below.

In [64]:
```
sns.pairplot(df[['bedrooms', 'bathrooms', 'area', 'style']], hue='style');
```



2. Earlier, you fit linear models between each individual predictor variable and price, as well as using all of the variables and the price in a multiple linear regression model. Each of the individual models showed a positive relationship - that is, when bathrooms, bedrooms, or area increase, we predict the price of a home to increase.

Fit a linear model to predict a home **price** using **bedrooms**, **bathrooms**, and **area**. Use the summary to answer the second quiz question below. **Don't forget an intercept.**

In [65]:
```
df['intercept'] = 1

lm = sm.OLS(df['price'], df[['intercept', 'bedrooms', 'bathrooms', 'area']])
result = lm.fit()
result.summary()
```

Out[65]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.678
Model:	OLS	**Adj. R-squared:**	0.678
Method:	Least Squares	**F-statistic:**	4230.
Date:	Tue, 04 Jul 2023	**Prob (F-statistic):**	0.00
Time:	21:06:48	**Log-Likelihood:**	-84517.
No. Observations:	6028	**AIC:**	1.690e+05
Df Residuals:	6024	**BIC:**	1.691e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	1.007e+04	1.04e+04	0.972	0.331	-1.02e+04	3.04e+04
bedrooms	-2925.8063	1.03e+04	-0.285	0.775	-2.3e+04	1.72e+04
bathrooms	7345.3917	1.43e+04	0.515	0.607	-2.06e+04	3.53e+04
area	345.9110	7.227	47.863	0.000	331.743	360.079

Omnibus:	367.658	**Durbin-Watson:**	2.007
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	350.116
Skew:	0.536	**Prob(JB):**	9.40e-77
Kurtosis:	2.503	**Cond. No.**	1.16e+04

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.16e+04. This might indicate that there are strong multicollinearity or other numerical problems.

`3.` Calculate the VIFs for each variable in your model. Use quiz 3 below to provide insights about the results of your VIFs. Here is the helpful post again, in case you need it!

In [70]:

```
# get y and X dataframes based on this regression:
y, X = dmatrices('price ~ area + bedrooms + bathrooms' ,  df, return_type='dataframe')

# For each X, calculate VIF and save in dataframe
vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["features"] = X.columns

vif
```

Out[70]:

	VIF Factor	features
0	7.327102	Intercept
1	5.458190	area
2	20.854484	bedrooms
3	19.006851	bathrooms

4. Remove bathrooms from your above model. Refit the multiple linear regression model and re-compute the VIFs. Use the final quiz below to provide insights about your results.

In [71]:
```
lm = sm.OLS(df['price'], df[['intercept', 'bedrooms', 'area']])
results = lm.fit()
results.summary()
```

Out[71]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.678
Model:	OLS	**Adj. R-squared:**	0.678
Method:	Least Squares	**F-statistic:**	6345.
Date:	Tue, 04 Jul 2023	**Prob (F-statistic):**	0.00
Time:	21:17:47	**Log-Likelihood:**	-84517.
No. Observations:	6028	**AIC:**	1.690e+05
Df Residuals:	6025	**BIC:**	1.691e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	7857.4509	9425.288	0.834	0.405	-1.06e+04	2.63e+04
bedrooms	1626.8306	5191.702	0.313	0.754	-8550.763	1.18e+04
area	346.4458	7.152	48.443	0.000	332.426	360.466

Omnibus:	368.146	**Durbin-Watson:**	2.007
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	350.224
Skew:	0.535	**Prob(JB):**	8.91e-77
Kurtosis:	2.502	**Cond. No.**	6.45e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.45e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [76]:
```
y, X = dmatrices('price ~ area + bedrooms', df, return_type= 'dataframe')

vif = pd.DataFrame()
vif['VIF Factor'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['features'] = X.columns

vif
```

Out[76]:

	VIF Factor	features
0	6.063895	Intercept
1	5.345400	area
2	5.345400	bedrooms

Quiz Question

Based on the scatterplot matrix in the first question, select all the below statements that are true.

- [x] It appears that the predictor variables are correlated with one another. ✓
- [x] The variables that appear to be most correlated are the number of bedrooms and bathrooms. ✓
- [ ] There should not be a problem with multicollinearity in using all three of these variables to predict price.

Quiz Question

Select all that are true about the coefficients in your multiple linear regression model.

- [ ] As the number of bedrooms increases, we predict the price to increase.
- [x] As the number of bathrooms increases, we predict the price to increase. ✓
- [x] As the area of the home increases, we predict the price to increase. ✓

Quiz Question

Which one of the statements below reflects the action that should be taken based on the VIFs?

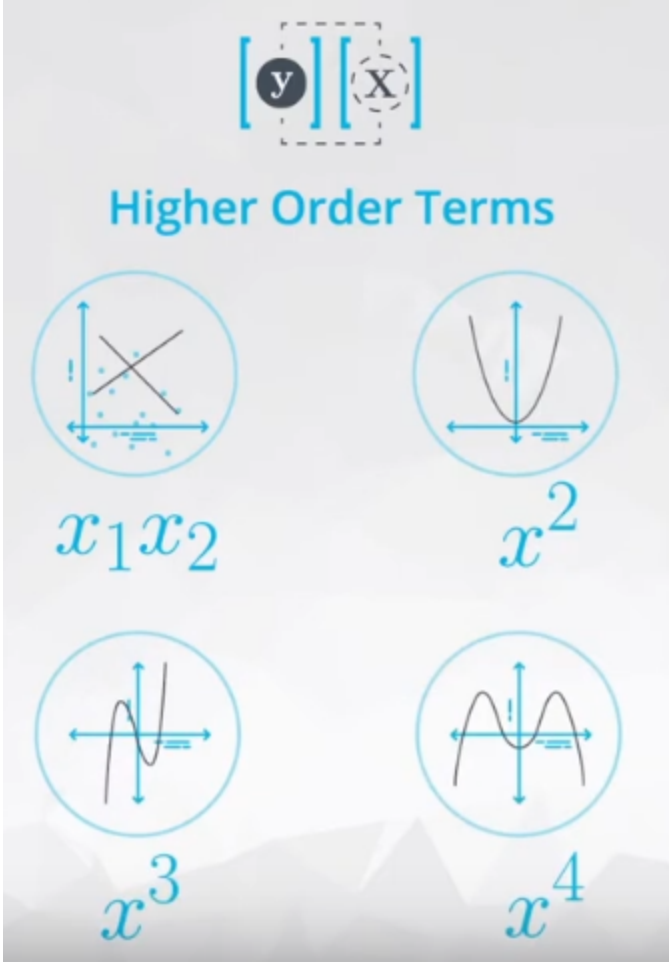
- ( ) We need to remove all of the variables from the model because they all have VIFs greater than 10.
- ( ) We should remove both bedrooms and bathrooms, because they both have VIFs greater than 10.
- (•) We should remove either bedrooms or bathrooms, because they both have VIFs greater than 10. ✓
- ( ) Keeping all the variables in the model is okay despite having large VIFs because removing one of the variables will reduce our Rsquared.

Quiz Question

Mark all of the statements below that are true with regard to your final results.

- [x] All VIFs are now below 10. ✓
- [x] All of the coefficients are now positive, as we would expect. ✓
- [x] To three digits the R-squared value stayed the same, suggesting we didn't really need both bedrooms and bathrooms in the model. ✓

Higher Order Terms



It's possible to fit linear models that look like non-linear models by adding higher order terms, like:

- Interactions
- Quadratics
- Cubics
- Higher order values

While these higher order terms might allow you to **better predict** your response, adding these terms makes interpreting your results more complex. In addition, interpretations for lower order terms like slope, are not easily interpreted.

How to Identify Higher-Order Terms?

Higher-order terms in linear models are created when multiplying two or more x-variables by one another. Common higher-order terms include **quadratics** (x_1^2) and **cubics** (x_1^3), where an x-variable is multiplied by itself, as well as **interactions** ($x_1 x_2$), where two or more x-variables are multiplied by one another.

In a model with no higher order terms, you might have an equation like:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2$$

Then we might decide the linear model can be improved with higher order terms. The equation might change to:

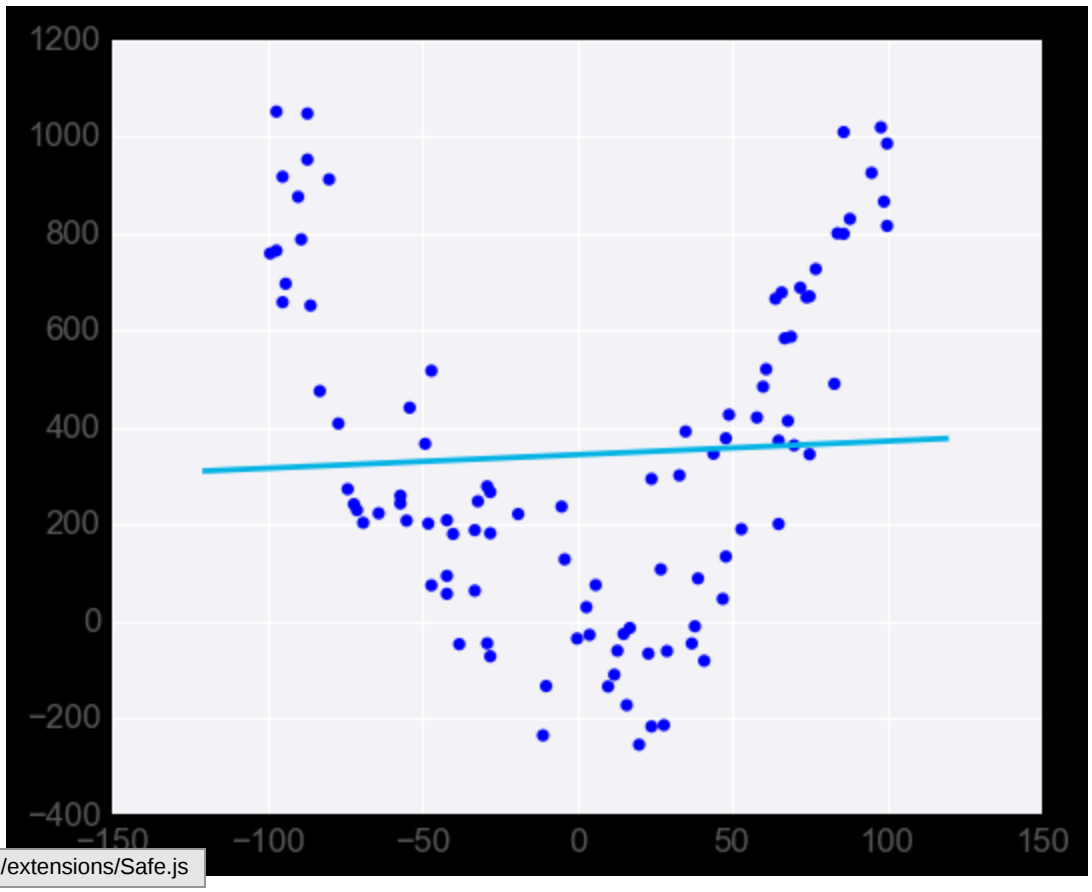
$$\hat{y} = b_0 + b_1 x_1 + b_2 x_1^2 + b_3 x_2 + b_4 x_1 x_2$$

Here, we have introduced a quadratic ($b_2 x_1^2$) and an interaction ($b_4 x_1 x_2$) term into the model.

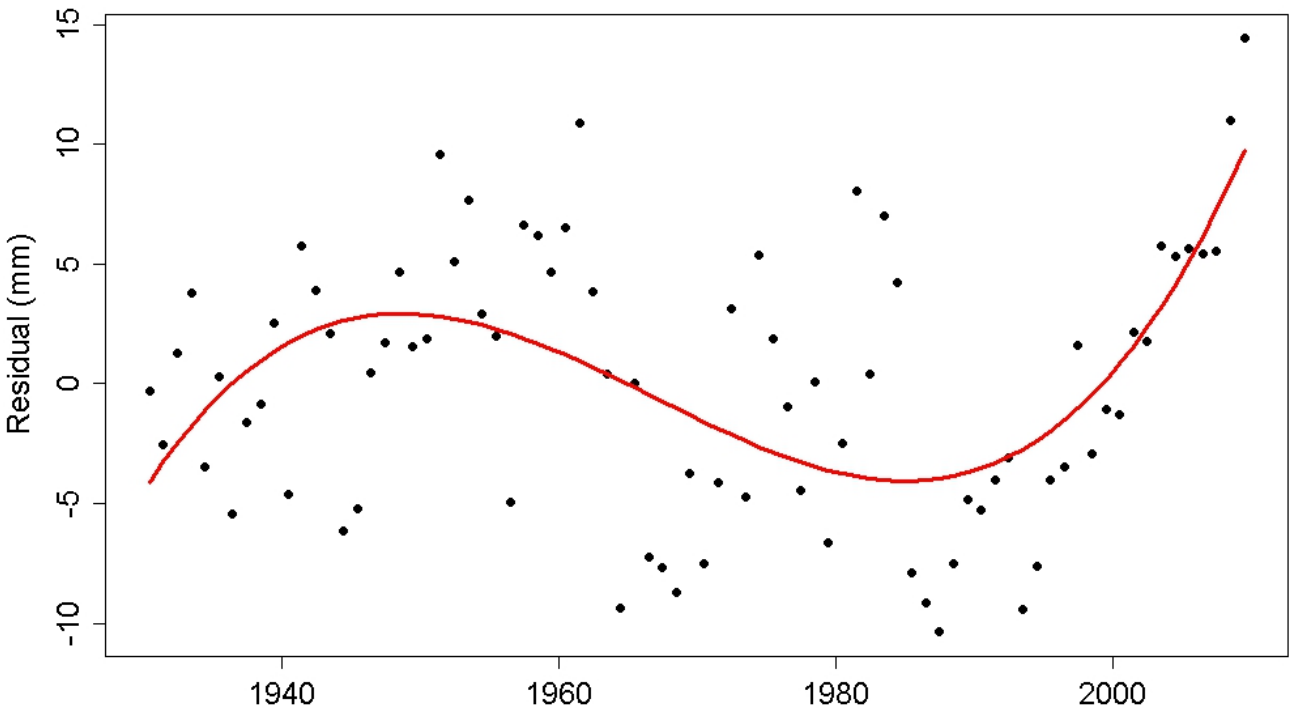
In general, these terms can help you fit more complex relationships in your data. However, they also take away from the ease of interpreting coefficients, as we have seen so far. You might be wondering: "How do I identify if I need one of these higher-order terms?"

When creating models with quadratic, cubic, or even higher orders of a variable, we are essentially looking at how many curves there are in the relationship between the explanatory and response variables.

If there is one curve, like in the plot below, then you will want to add a quadratic. Clearly, we can see a line isn't the best fit for this relationship.



Then, if we want to add a cubic relationship, it is because we see two curves in the relationship between the explanatory and response variable. An example of this is shown in the plot below.



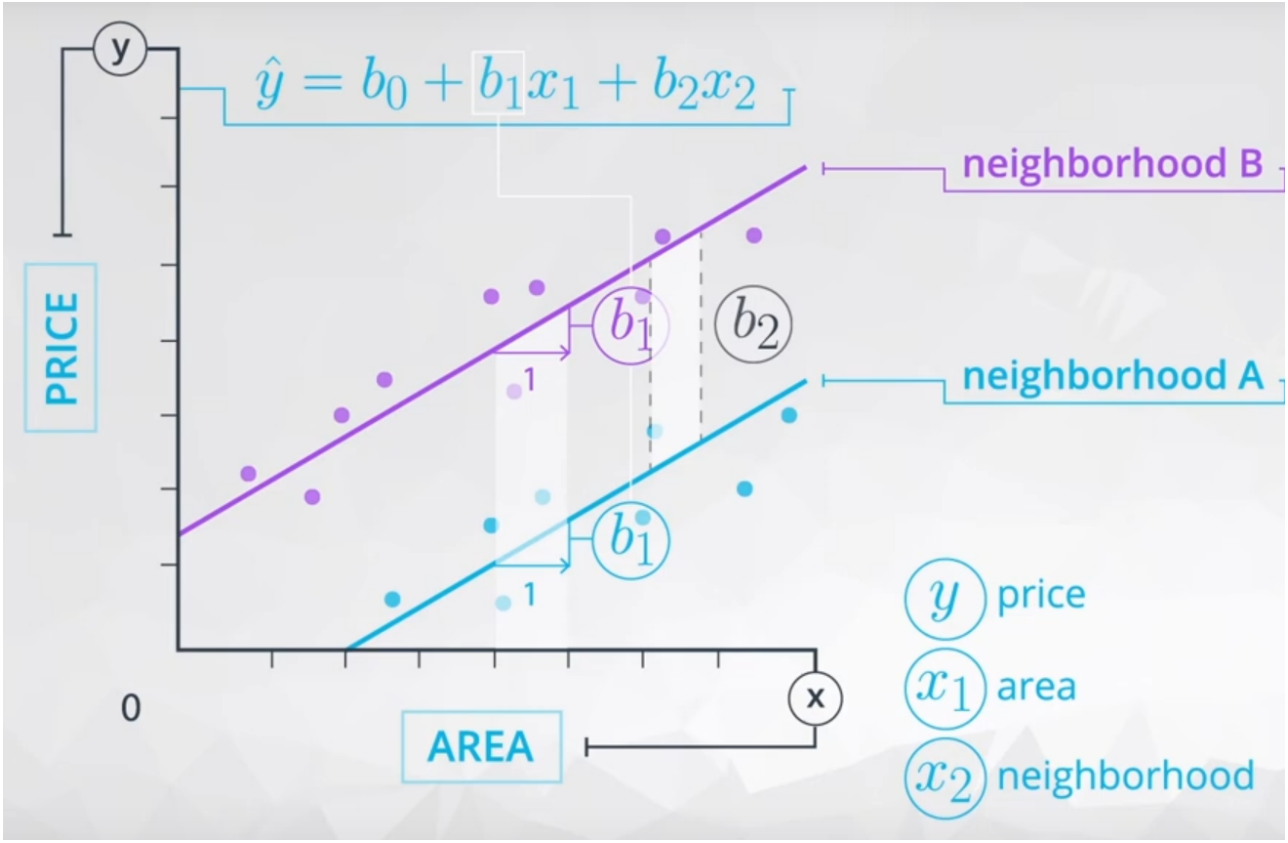
Interpreting Interactions

When working with higher order terms, each time a higher order term is added to a model, a lower order term should be added as well. You've learned that using quadratics and cubics is useful when the relationship between response and explanatory variables is non-linear and represented by a curve or curves.

When to use interaction terms (x_1x_2)?

Interaction terms tell us how a variable (x_1) is related to the response variable (y) and is dependent on the value of another variable (x_2). In the video example, the response variable (y), price, has a positive and linear relationship to the total square footage of a house (x_1), dependent on the neighborhood (x_2) that house is in. Depending on the neighborhood, the slope of the (x_1, y) lines may be different.

- If the slopes of explanatory variable lines are equal or close to equal, interaction terms are not added.



- If the slopes are not equal it is an indication that an interaction term should be added to the model.

Interpreting Model Coefficients with Python

It is important that not only can you fit complex linear models, but that you then know which variables you can interpret.

In this notebook, you will fit a few different models and use the quizzes below to match the appropriate interpretations to your coefficients when possible.

In [80]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm;

df = pd.read_csv('house_prices.csv')
df.head()
```

Out[80]:

	house_id	neighborhood	area	bedrooms	bathrooms	style	price
0	1112	B	1188	3	2	ranch	598291
1	491	B	3512	5	3	victorian	1744259
2	5952	B	1134	3	2	ranch	571669
3	3525	A	1940	4	2	ranch	493675
4	5108	B	2208	6	4	victorian	1101539

We will be fitting a number of different models to this dataset throughout this notebook. For each model, there is a quiz question that will allow you to match the interpretations of the model coefficients to the corresponding values. If there is no 'nice' interpretation, this is also an option!

Model 1

`1.` For the first model, fit a model to predict `price` using `neighborhood`, `style`, and the `area` of the home. Use the output to match the correct values to the corresponding interpretation in quiz 1 below. Don't forget an intercept! You will also need to build your dummy variables, and don't forget to drop one of the columns when you are fitting your linear model. It may be easiest to connect your interpretations to the values in the first quiz by creating the baselines as neighborhood C and home style **lodge**.

In [82]:
```
df_dummies = pd.get_dummies(df[['neighborhood', 'style']])

df_new = df.join(df_dummies)

df_new['intercept'] = 1

df_new.head()
```

Out[82]:

	house_id	neighborhood	area	bedrooms	bathrooms	style	price	neighborhood_A	neighborhood_B
0	1112	B	1188	3	2	ranch	598291	0	1
1	491	B	3512	5	3	victorian	1744259	0	1
2	5952	B	1134	3	2	ranch	571669	0	1
3	3525	A	1940	4	2	ranch	493675	1	0
4	5108	B	2208	6	4	victorian	1101539	0	1

In [84]:
```
lm = sm.OLS(df_new['price'], df_new[['intercept', 'neighborhood_A','neighborhood_B', 'st
result = lm.fit()
result.summary()
```

Out[84]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.919
Model:	OLS	**Adj. R-squared:**	0.919
Method:	Least Squares	**F-statistic:**	1.372e+04
Date:	Tue, 04 Jul 2023	**Prob (F-statistic):**	0.00
Time:	22:18:56	**Log-Likelihood:**	-80348.
No. Observations:	6028	**AIC:**	1.607e+05
Df Residuals:	6022	**BIC:**	1.607e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	-1.983e+05	5540.744	-35.791	0.000	-2.09e+05	-1.87e+05
neighborhood_A	-194.2464	4965.459	-0.039	0.969	-9928.324	9539.832
neighborhood_B	5.243e+05	4687.484	111.844	0.000	5.15e+05	5.33e+05
style_ranch	-1974.7032	5757.527	-0.343	0.732	-1.33e+04	9312.111
style_victorian	-6262.7365	6893.293	-0.909	0.364	-1.98e+04	7250.586
area	348.7375	2.205	158.177	0.000	344.415	353.060

Omnibus:	114.369	**Durbin-Watson:**	2.002
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	139.082
Skew:	0.271	**Prob(JB):**	6.29e-31
Kurtosis:	3.509	**Cond. No.**	1.12e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.12e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Model 2

`2.` Now let's try a second model for predicting price. This time, use `area` and `area squared` to predict price. Also use the `style` of the home, but not `neighborhood` this time. You will again need to use your dummy variables, and add an intercept to the model. Use the results of your model to answer quiz questions 2 and 3.

In [87]:
```
df_new['area squared'] = df_new['area']*df_new['area']
```

In [89]:
```
lm = sm.OLS(df_new['price'], df_new[['intercept', 'style_ranch', 'style_victorian', 'are
results = lm.fit()
results.summary()
```

Out[89]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.678
Model:	OLS	**Adj. R-squared:**	0.678
Method:	Least Squares	**F-statistic:**	3173.
Date:	Tue, 04 Jul 2023	**Prob (F-statistic):**	0.00
Time:	22:20:50	**Log-Likelihood:**	-84516.
No. Observations:	6028	**AIC:**	1.690e+05
Df Residuals:	6023	**BIC:**	1.691e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	1.855e+04	1.26e+04	1.467	0.142	-6229.316	4.33e+04
style_ranch	9917.2547	1.27e+04	0.781	0.435	-1.5e+04	3.48e+04
style_victorian	2509.3957	1.53e+04	0.164	0.870	-2.75e+04	3.25e+04
area squared	0.0029	0.002	1.283	0.199	-0.002	0.007
area	334.0146	13.525	24.696	0.000	307.501	360.528

Omnibus:	375.220	**Durbin-Watson:**	2.009
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	340.688
Skew:	0.519	**Prob(JB):**	1.05e-74
Kurtosis:	2.471	**Cond. No.**	4.33e+07

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 4.33e+07. This might indicate that there are strong multicollinearity or other numerical problems.

In [90]:

```
# Trying one more model, with neighborhoods also, responding to
#   prompt in question 3
lm = sm.OLS(df_new['price'], df_new[['intercept', 'neighborhood_A', 'neighborhood_B', 's
results = lm.fit()
results.summary()
```

Out[90]:

OLS Regression Results

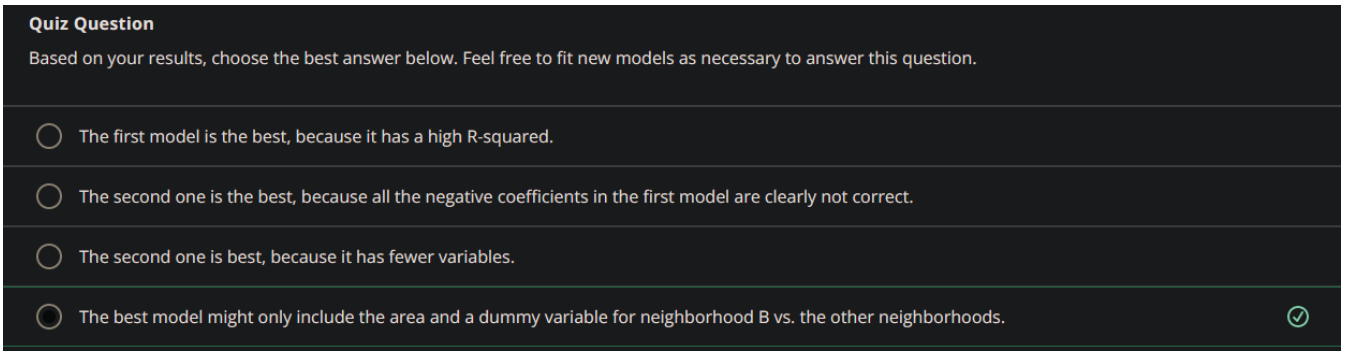
Dep. Variable:	price	**R-squared:**	0.919
Model:	OLS	**Adj. R-squared:**	0.919
Method:	Least Squares	**F-statistic:**	1.144e+04
Date:	Tue, 04 Jul 2023	**Prob (F-statistic):**	0.00
Time:	22:21:54	**Log-Likelihood:**	-80345.
No. Observations:	6028	**AIC:**	1.607e+05
Df Residuals:	6021	**BIC:**	1.608e+05
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	-1.881e+05	7023.287	-26.776	0.000	-2.02e+05	-1.74e+05
neighborhood_A	-248.3127	4963.602	-0.050	0.960	-9978.750	9482.124
neighborhood_B	5.242e+05	4685.714	111.877	0.000	5.15e+05	5.33e+05
style_ranch	4458.3334	6361.432	0.701	0.483	-8012.351	1.69e+04
style_victorian	1650.2605	7654.606	0.216	0.829	-1.34e+04	1.67e+04
area squared	0.0027	0.001	2.374	0.018	0.000	0.005
area	333.5383	6.772	49.256	0.000	320.264	346.813

Omnibus:	57.788	**Durbin-Watson:**	2.002
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	70.025
Skew:	0.168	**Prob(JB):**	6.23e-16
Kurtosis:	3.407	**Cond. No.**	4.33e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.33e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

Notice the R-squared value is 0.919 again, same as in model #1, and the area-squared coefficient remains very low



Logistic Regression

In the previous lesson, you learned to take both categorical and quantitative variables to predict a quantitative response. In this lesson, you will use logistic regression results to make a prediction about the relationship between categorical dependent variables and predictors.

Logistic regression predicts categorical responses when there are only two possible outcomes.

Fitting Logistic Regression

Logistic regression – the predicted response variable is limited to a probability between 0 and 1

Linear regression – the predicted response variable can take any value and is unconstrained

Fitting Logistic Regression in Python

Steps for Fitting Logistic Regression in Python

1. Load libraries. Similar to working with linear regression models, the statsmodels.api library can be used to run logistic regression models.
2. Use dummy variables for categorical columns
3. Impute missing values
4. Use statsmodels Logit() method and pass in the response variable, the intercept, and your explanatory variable, x.
5. Fit the model
6. Get summary results – Note: the code for this is currently results.summary2()

In [100…
```
import numpy as np
import pandas as pd
import statsmodels.api as sm


df = pd.read_csv('fraud_dataset.csv')
df.head()
```

Out[100]:

	transaction_id	duration	day	fraud
0	28891	21.302600	weekend	False
1	61629	22.932765	weekend	False
2	53707	32.694992	weekday	False
3	47812	32.784252	weekend	False
4	43455	17.756828	weekend	False

`1.` As you can see, there are two columns that need to be changed to dummy variables. Replace each of the current columns to the dummy version. Use the 1 for `weekday` and `True`, and 0 otherwise. Use the first quiz to answer a few questions about the dataset.

In [105…
```
df['weekday'] = pd.get_dummies(df['day'])['weekday']
# df[['weekday','weekend']] = pd.get_dummies(df['day'])
#df = df.drop('weekend', axis=1)

#The two code are equivalent and achieve the same result.!

df[['not_fraud','fraud']] = pd.get_dummies(df['fraud'])
df = df.drop('not_fraud', axis=1)
```

In [106…
```
#The proportion of fraudulent transactions.
print(df['fraud'].mean())
#The proportion of weekday transactions.
print(df['weekday'].mean())
#The average duration for fraudulent &  non-fraudulent transaction.
print(df.groupby('fraud').mean()['duration'])
```

```
0.012168770612987604
0.3452746502900034
fraud
0    30.013583
1     4.624247
Name: duration, dtype: float64
```

`2.` Now that you have dummy variables, fit a logistic regression model to predict if a transaction is fraud using both day and duration. Don't forget an intercept! Use the second quiz below to assure you fit the model correctly. Also remember to use the `.summary2() method to get your summary results.

In [107…
```
df['intercept'] = 1
log_mod = sm.Logit(df['fraud'], df[['intercept', 'weekday', 'duration']])
results = log_mod.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: inf
         Iterations 16
```

```
C:\Users\Tsgts\anaconda3\lib\site-packages\statsmodels\discrete\discrete_model.py:1819:
RuntimeWarning: overflow encountered in exp
  return 1/(1+np.exp(-X))
C:\Users\Tsgts\anaconda3\lib\site-packages\statsmodels\discrete\discrete_model.py:1872:
RuntimeWarning: divide by zero encountered in log
  return np.sum(np.log(self.cdf(q*np.dot(X,params))))
C:\Users\Tsgts\anaconda3\lib\site-packages\statsmodels\base\model.py:592: HessianInversi
onWarning: Inverting hessian failed, no bse or cov_params available
  warnings.warn('Inverting hessian failed, no bse or cov_params '
C:\Users\Tsgts\anaconda3\lib\site-packages\statsmodels\base\model.py:592: HessianInversi
onWarning: Inverting hessian failed, no bse or cov_params available
  warnings.warn('Inverting hessian failed, no bse or cov_params '
```

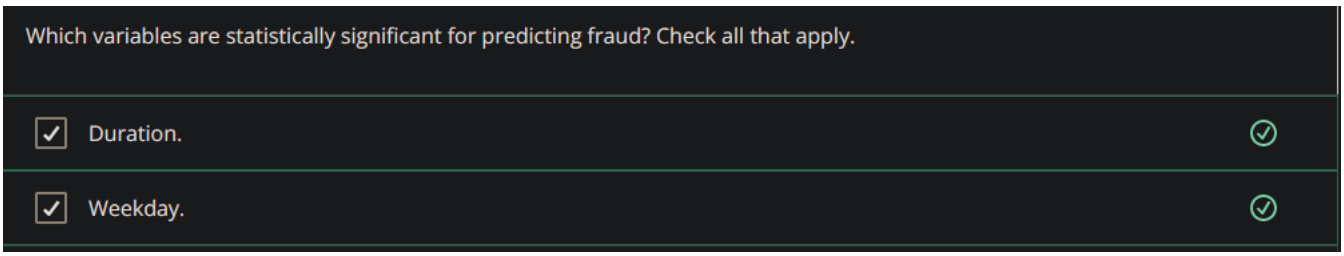
Out[107]:

Logit Regression Results

Dep. Variable:	fraud	**No. Observations:**	8793
Model:	Logit	**Df Residuals:**	8790
Method:	MLE	**Df Model:**	2
Date:	Thu, 06 Jul 2023	**Pseudo R-squ.:**	inf
Time:	18:18:03	**Log-Likelihood:**	-inf
converged:	True	**LL-Null:**	0.0000
Covariance Type:	nonrobust	**LLR p-value:**	1.000

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	9.8709	1.944	5.078	0.000	6.061	13.681
weekday	2.5465	0.904	2.816	0.005	0.774	4.319
duration	-1.4637	0.290	-5.039	0.000	-2.033	-0.894

Possibly complete quasi-separation: A fraction 0.98 of observations can be perfectly predicted. This might indicate that there is complete quasi-separation. In this case some parameters will not be identified.



Interpreting Results

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	9.8709	1.944	5.078	0.000	6.061	13.691
duration	-1.4637	0.290b	0.290	0.000	-2.033	-0.894
weekday	2.5465	0.904	0.904	0.000	0.774	4.319

Quantitative interpretations for every one unit increase in the explanatory variable x_1, we expect a multiplicative change in the odds of being in the 1 category of e^{b_1}, holding all over variables constant.

Categorical interpretations when in category x_1, we expect a multiplicative change in the odds of being in the 1 category of e^{b_1}, compared to baseline.

In [108…
```
# coef of weekday & duration
np.exp(2.5465), np.exp(-1.4637)
```

Out[108]:
```
(12.762357271496972, 0.2313785882117941)
```

As we see above,

In weekday fraud is 12.76 times than weekends

For each 1 unit increase in duration, fraud is 0.23 times as likely holding all else constant

Note – When multiplicative changes are less than 1, like duration = 0.231, it is usually useful to calculate the reciprocal. This changes the direction from a unit increase to a unit decrease. So, the result for duration could also be interpreted as:

In [110…
```
1/np.exp(-1.4637)
```

Out[110]:
```
4.321921089278333
```

For each unit decrease in duration, fraud is 4.32 times as likely holding all else constant

Interpret Results in Python

The dataset contains four variables: `admit`, `gre`, `gpa`, and `prestige`:

- `admit` is a binary variable. It indicates whether or not a candidate was admitted into UCLA (admit = 1) our not (admit = 0).
- `gre` is the GRE score. GRE stands for Graduate Record Examination.
- `gpa` stands for Grade Point Average.
- `prestige` is the prestige of an applicant alta mater (the school attended before applying), with 1 being the highest (highest prestige) and 4 as the lowest (not prestigious).

To start, let's read in the necessary libraries and data.

In [112…
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
```

```
df = pd.read_csv("admissions.csv")
df.head()
```

Out[112]:

	admit	gre	gpa	prestige
0	0	380	3.61	3
1	1	660	3.67	3
2	1	800	4.00	1
3	1	640	3.19	4
4	0	520	2.93	4

There are a few different ways you might choose to work with the `prestige` column in this dataset. For this dataset, we will want to allow for the change from prestige 1 to prestige 2 to allow a different acceptance rate than changing from prestige 3 to prestige 4.

1. With the above idea in place, create the dummy variables needed to change prestige to a categorical variable, rather than quantitative, then answer quiz 1 below.

In [113…
```
df[['prest_1', 'prest_2', 'prest_3','prest_4',]] = pd.get_dummies(df['prestige'])
```

In [114…
```
df.head()
```

Out[114]:

	admit	gre	gpa	prestige	prest_1	prest_2	prest_3	prest_4
0	0	380	3.61	3	0	0	1	0
1	1	660	3.67	3	0	0	1	0
2	1	800	4.00	1	1	0	0	0
3	1	640	3.19	4	0	0	0	1
4	0	520	2.93	4	0	0	0	1

In [115…
```
df['prestige'].astype(str).value_counts()
```

Out[115]:
```
2    148
3    121
4     67
1     61
Name: prestige, dtype: int64
```

`2.` Now, fit a logistic regression model to predict if an individual is admitted using `gre`, `gpa`, and `prestige` with a baseline of the prestige value of `1`. Use the results to answer quiz 2 and 3 below. Don't forget an intercept.

In [116…
```
df['intercept'] = 1

logit_mod = sm.Logit(df['admit'], df[['intercept','gre', 'gpa', 'prest_2', 'prest_3', 'p
results = logit_mod.fit()
results.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.573854
         Iterations 6
```

Out[116]:

Model:	Logit	Pseudo R-squared:	0.082
Dependent Variable:	admit	AIC:	467.6399
Date:	2023-07-06 18:46	BIC:	491.5435
No. Observations:	397	Log-Likelihood:	-227.82
Df Model:	5	LL-Null:	-248.08
Df Residuals:	391	LLR p-value:	1.1761e-07
Converged:	1.0000	Scale:	1.0000
No. Iterations:	6.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
intercept	-3.8769	1.1425	-3.3934	0.0007	-6.1161	-1.6376
gre	0.0022	0.0011	2.0280	0.0426	0.0001	0.0044
gpa	0.7793	0.3325	2.3438	0.0191	0.1276	1.4311
prest_2	-0.6801	0.3169	-2.1459	0.0319	-1.3013	-0.0589
prest_3	-1.3387	0.3449	-3.8819	0.0001	-2.0146	-0.6628
prest_4	-1.5534	0.4175	-3.7211	0.0002	-2.3716	-0.7352

In [117…
```
np.exp(results.params)
```

Out[117]:
```
intercept    0.020716
gre          1.002221
gpa          2.180027
prest_2      0.506548
prest_3      0.262192
prest_4      0.211525
dtype: float64
```

In [118…
```
1/_
```

Out[118]:
```
intercept    48.272116
gre           0.997784
gpa           0.458710
prest_2       1.974147
prest_3       3.813995
prest_4       4.727566
dtype: float64
```

In [119…
```
df.groupby('prestige').mean()['admit']
```

Out[119]:
```
prestige
1    0.540984
2    0.358108
3    0.231405
4    0.179104
Name: admit, dtype: float64
```

Select all the below statements that are true regarding the admissions dataset. Select all that apply.

- [ ] We will need to add 4 dummy variables to our logistic regression model for `prestige`.
- [x] We will need to add 3 dummy variables to our logistic regression model for `prestige`. ✓
- [x] The most common prestige value is 2. ✓
- [ ] The most common prestige value is 3.

Which statement is true about the variables in your logistic regression model that appear to have statistically significant relationships with being admitted?

- ( ) All the variables are statistically significant. ✓
- ( ) Some of the variables are statistically significant.
- ( ) None of the variables are statistically significant.

Use the results to match each value to the correct corresponding interpretation.

✓ These are the correct matches.

Interpretation	
If an individual attended the most prestigious alma mater, they are _____ more likely to be admitted than if they attended the least prestigious, holding all other variables constant.	4.73 times
If an individual attended the most prestigious alma mater, they are _____ more likely to be admitted than if they attended the second lowest in prestigious-ness, holding all other variables constant.	3.81 times
If an individual attended the most prestigious alma mater, they are _____ more likely to be admitted than if they attended the second most prestigious, holding all other variables constant.	1.97 times
For every one point increase in gpa, an individual is _____ more likely to be admitted, holding all other variables constant.	2.18 times

Notice that in order to compare the lower prestigious values to the most prestigious (the baseline), we took one over the exponential of the coefficients. However, for a 1 unit increase, we could use the exponential directly.

Model Diagnostics + Performance Metrics

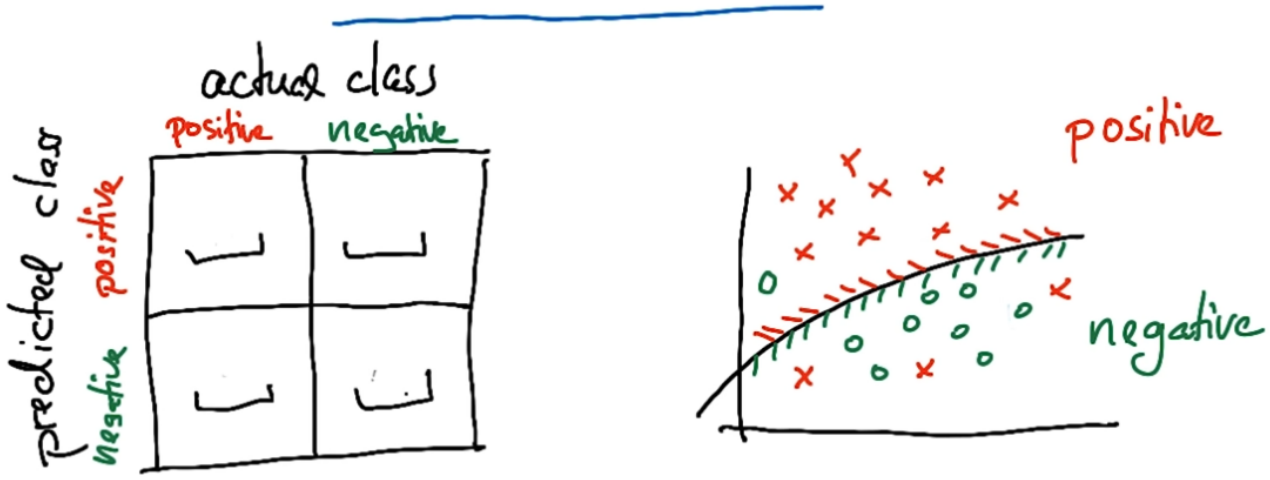
When determining how well your model is performing, the most common measure to use is accuracy.

$$Accuracy = \frac{number\ of\ correctly\ labeled\ rows}{number\ of\ total\ rows\ in\ dataset}$$

However, accuracy is not always the

best measure, especially for unbalanced datasets. In the next few pages, we will cover other types of metrics that will help us determine if our models are performing well.

Filling in a Confusion Matrix



Match the numbers to the correct matrix category, based on the scatterplot in the PHOTO.

Matrix Category	Number
(Positive, Positive)	9
(Positive, Negative)	3
(Negative, Negative)	8
(Negative, Positive)	1

Precision and Recall

Recall: True Positive / (True Positive + False Negative). Out of all the items that are truly positive, how many were correctly classified as positive. Or simply, how many positive items were 'recalled' from the dataset.

Precision: True Positive / (True Positive + False Positive). Out of all the items labeled as positive, how many truly belong to the positive class.

		Predicted						
True/ Actual	Ariel Sharon	13	4	1	1	0	0	1
	Colin Powell	0	55	0	8	0	0	0
	Donald Rumsfeld	0	1	25	8	0	0	2
	George W Bush	0	3	0	123	0	0	1
	Gerhard Schroeder	0	1	0	7	14	0	4
	Hugo Chavez	0	3	0	2	1	10	0
	Tony Blair	0	0	1	7	0	0	26

The **recall rate** on Hugo Chavez is the probability of the algorithm classifying an image as Hugo Chavez and the person is Hugo Chavez, here that was 10/16 or 0.625.

Hugo Chavez	0	3	0	2	1	10	0

The **precision rate** on Hugo Chavez is the probability that the algorithm correctly observes that the image is Hugo Chavez, here that was 10/10 or 1.

0
0
0
0
0
10
0

Practicing TP, FP, FN with Rumsfeld

Quiz Question

What is the number of true positives (Pos, Pos) for Tony Blair?

- ○ 26 ✓
- ○ 34
- ○ 68
- ○ Not enough information provided.

Quiz Question

What is the number of false positives (Neg, Pos) for Tony Blair?

- 8 ✓
- 34
- 26
- Not enough information provided.

1
0
2
1
4
0

is the sum of the predicted

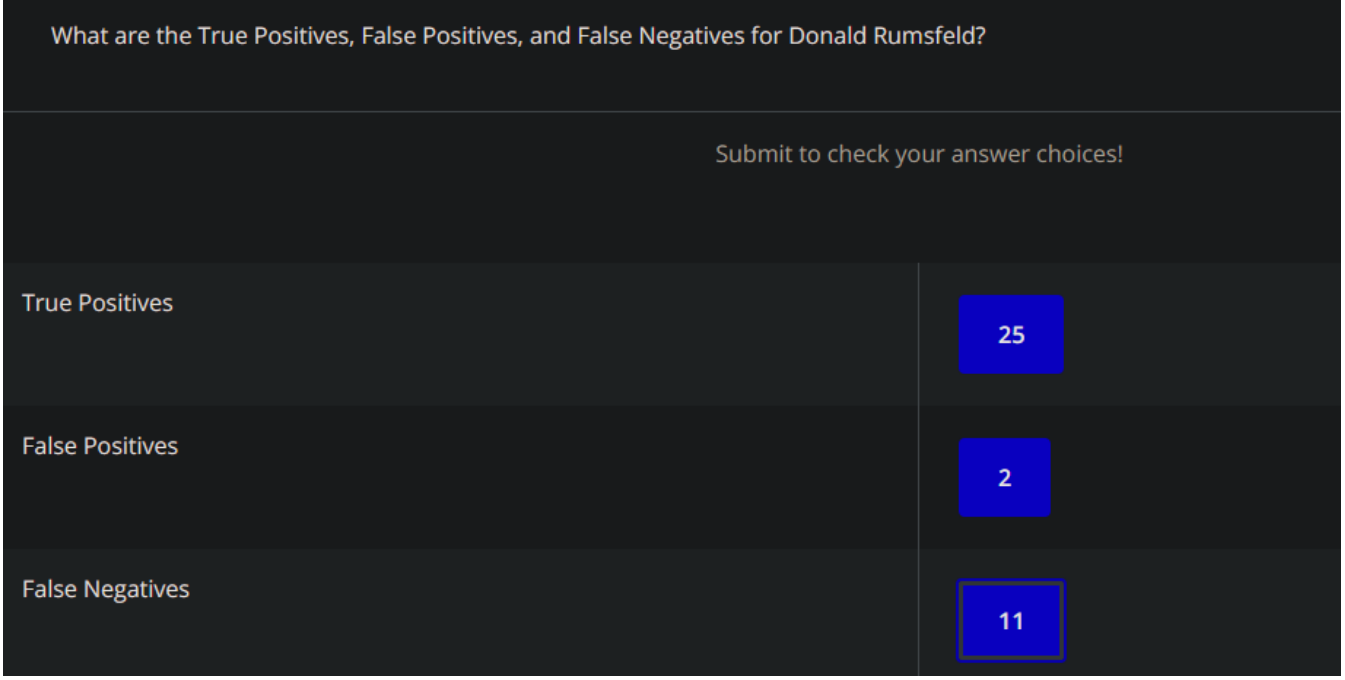
Quiz Question

What is the number of False Negatives (Neg, Neg) for Tony Blair?

- 8 ✓
- 34
- 26
- Not enough information provided.

is the sum of the row of True

0	0	1	7	0	0



Equation for Precision

$$\frac{True\ Positives}{True\ Positives + False\ Positives}$$

Equation for Recall

$$\frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Model Diagnostics in Python

In [120…
```
#read in libraries
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, precision_score, recall_score, accuracy_sc
from sklearn.model_selection import train_test_split
np.random.seed(42)
```

In [121…
```
#read in dataset
df = pd.read_csv('admissions.csv')
df.head()
```

Out[121]:

	admit	gre	gpa	prestige
0	0	380	3.61	3
1	1	660	3.67	3
2	1	800	4.00	1
3	1	640	3.19	4
4	0	520	2.93	4

In [122…
```
#create response and explanatory variables
y = df['admit']#the target variable
df[['level1','level2','level3','level4']] = pd.get_dummies(df['prestige'])
X = df[['gre', 'gpa','level1','level2','level3']]
```

In [123…
```
#create the train test split
X_train, X_test, y_train, y_test =train_test_split(X,y,test_size=0.10, random_state=42)
```

In [124…
```
#instantiate logistic regression model
log_mod = LogisticRegression()
```

In [125…
```
#fit the model to the training data
log_mod.fit(X_train, y_train)
```

```
C:\Users\Tsgts\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[125]:
```
LogisticRegression()
```

In [126…
```
#create predictions using test data
y_preds = log_mod.predict(X_test)
```

In [128…
```
#print summary statistics
print(precision_score(y_test, y_preds))
print(recall_score(y_test, y_preds))
print(accuracy_score(y_test, y_preds))
confusion_matrix(y_test, y_preds)
```

```
0.3333333333333333
0.0625
0.575
```

Out[128]:
```
array([[22,  2],
       [15,  1]], dtype=int64)
```

	Predicted	
Actual	0	1
0	22	2
1	15	1

	Predicted Negative	Predicted Positive
egative	TN	FP

```
Actual Positive      FN                  TP
```

- TN (True Negative): The number of instances correctly predicted as negative.
- FP (False Positive): The number of instances incorrectly predicted as positive.
- FN (False Negative): The number of instances incorrectly predicted as negative.
- TP (True Positive): The number of instances correctly predicted as positive.

    - Therefore, there are 22 non-admitted that we predict to be non-admitted.
    - There are 15 admitted that we predicted to be non-admitted.
    - There is 2 non-admitted that we predict to be admitted.
    - There are 1 admitted that we predict to be admitted.

    In general, it is useful to split your data into training and testing data to assure your model can predict well not only with the data it was fit to, but also on data that the model has never seen before. Proving the model performs well on test data assures that you have a model that will do well in future use cases - whether that be future students, future transactions, or any other future predictions you might want to make.

Model Diagnostics in Python Quiz

First let's read in the necessary libraries and the dataset.

In [129…
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, precision_score, recall_score, accuracy_sc
from sklearn.model_selection import train_test_split
np.random.seed(42)

df = pd.read_csv('admissions.csv')
df.head()
```

Out[129]:

	admit	gre	gpa	prestige
0	0	380	3.61	3
1	1	660	3.67	3
2	1	800	4.00	1
3	1	640	3.19	4
4	0	520	2.93	4

`1.` Change prestige to dummy variable columns that are added to `df`. Then divide your data into training and test data. Create your test set as 20% of the data, and use a random state of 0. Your response should be the `admit` column. Here are the docs, which can also find with a quick google search if you get stuck.

In [130…
```
df[['prest_1', 'prest_2', 'prest_3', 'prest_4']] = pd.get_dummies(df['prestige'])
X = df.drop(['admit', 'prestige', 'prest_1'] , axis=1)
y = df['admit']
X_train, X_test, y_train, y_test = train_test_split(
          X, y, test_size=0.20, random_state=0)
```

`2.` Now use sklearn's Logistic Regression to fit a logistic model using `gre`, `gpa`, and 3 of your `prestige` dummy variables. For now, fit the logistic regression model without changing any of the hyperparameters.

The usual steps are:

- Instantiate
- Fit (on train)
- Predict (on test)
- Score (compare predict to test)

As a first score, obtain the confusion matrix. Then answer the first question below about how well your model performed on the test data.

In [131…
```
log_mod = LogisticRegression()
log_mod.fit(X_train, y_train)
preds = log_mod.predict(X_test)
confusion_matrix(y_test, preds)
```

```
C:\Users\Tsgts\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[131]:
```
array([[56,  0],
       [20,  4]], dtype=int64)
```

`3.` Now, try out a few additional metrics: precision, recall, and accuracy are all popular metrics, which you saw with Sebastian. You could compute these directly from the confusion matrix, but you can also use these built in functions in sklearn.

Another very popular set of metrics are ROC curves and AUC. These actually use the probability from the logistic regression models, and not just the label. This is also a great resource for understanding ROC curves and AUC.

Try out these metrics to answer the second quiz question below. I also provided the ROC plot below. The ideal case is for this to shoot all the way to the upper left hand corner. Again, these are discussed in more detail in the Machine Learning Udacity program.

In [132…
```
precision_score(y_test, preds)
```

Out[132]:
```
1.0
```

In [133…
```
recall_score(y_test, preds)
```

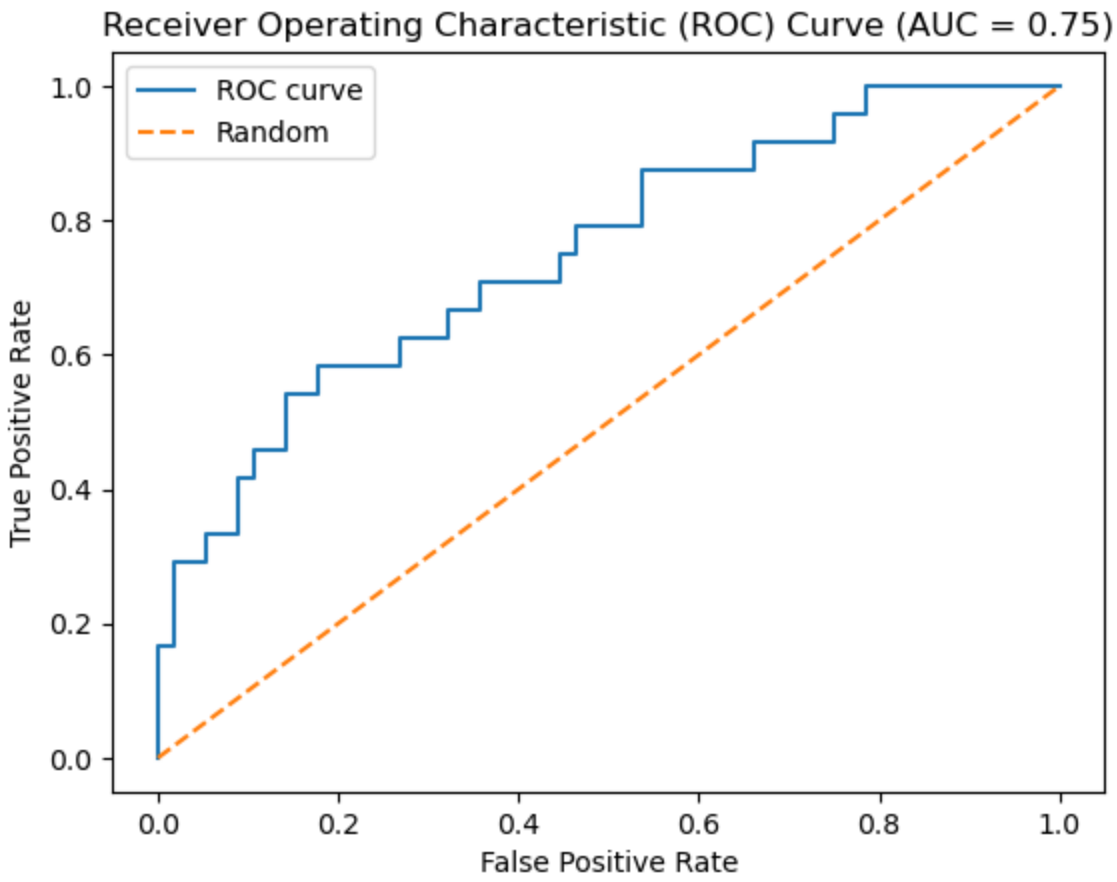
Out[133]:
```
0.16666666666666666
```

In [134…
```
accuracy_score(y_test, preds)
```

Out[134]:
```
0.75
```

In [137…

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

preds = log_mod.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, preds)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve (AUC = {:.2f})'.format(roc_auc)
plt.legend(['ROC curve', 'Random'])
plt.show()
```



This code allows you to plot the Receiver Operating Characteristic (ROC) curve, which illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) for different classification thresholds. The AUC value provides a measure of the model's ability to distinguish between the positive and negative classes, with a **higher AUC indicating better predictive performance**.

Final Thoughts On Shifting to Machine Learning

Looking back at the models we fit using scikit-learn, you will notice that we did not get as much information back regarding coefficients in terms of the p-values. Even if you pull the coefficients, they do not match what was retrieved using the statsmodels.api library.

This represents the shift from statistics to machine learning:

- **Statistics** – Determine relationships and understand the driving mechanisms. Are relationships due to chance?

- **Supervised Machine Learning** – Work to predict as well as possible, often without regard to why it works well.

In [ ]: