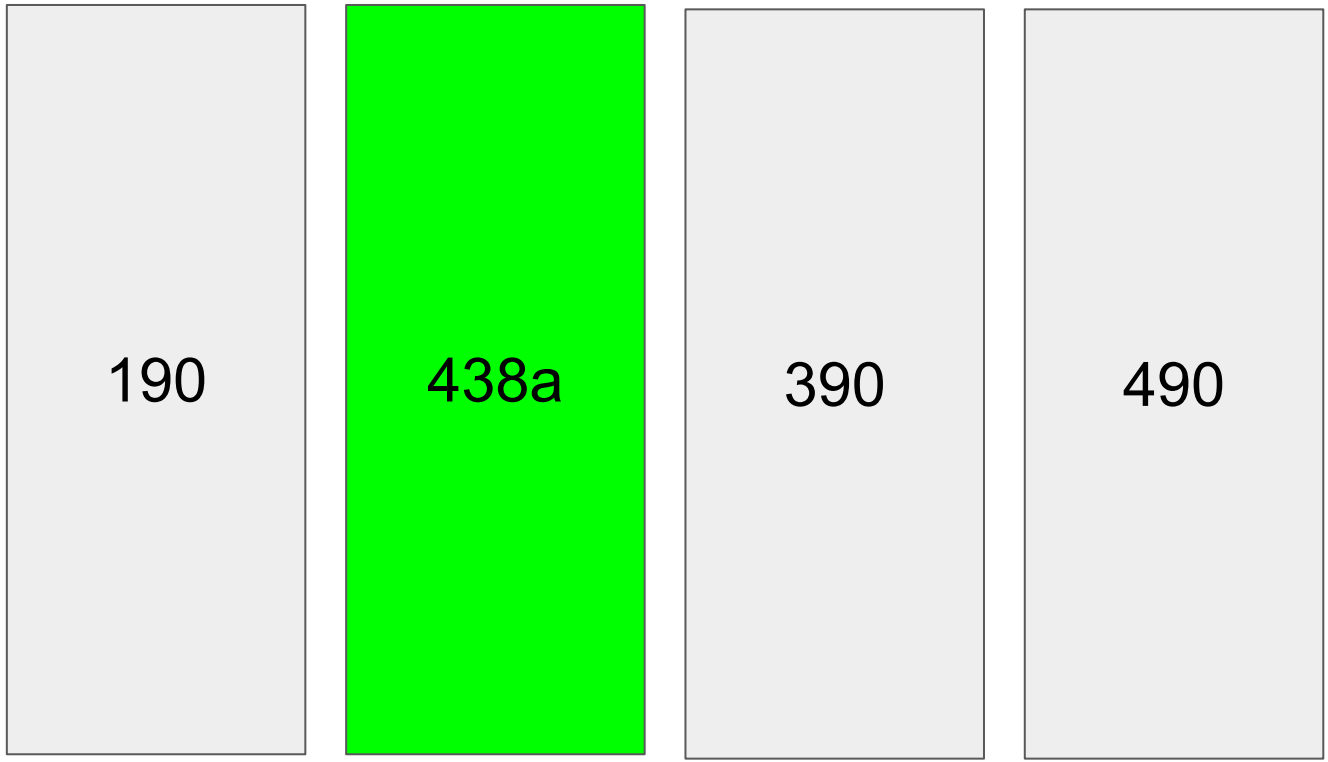
Applied Quantitative Analysis

BMGT438A/ENES489A

Instructors

Joe Bailey

Jessica Strongin (Q36)


190
438a
390
490

190

- Design Thinking
- Quality/Voice of the Customer
- Prototyping
- Physical products, digital products, and process improvements
- Iteration
- Continuous Improvement
- DMAIC
- Garvin’s 8 Dimensions

438a

- Framing the problem
- Development of hypotheses
- Identification, collection, ingestion, and analysis of data
- Bais
- Numbers and narrative

390

- Disruptive technologies
- Extrapolation
- Culture and context
- Thinking globally
- Feedback loops
- System Dynamics
- Change management
- Embracing complexity

490

- Capstone QUEST Experience
- Your toolkit is heavy but still incomplete
- What tools will you use?

How can we listen to the ‘voice’ of the customer using data science?

Why 438a after 190?

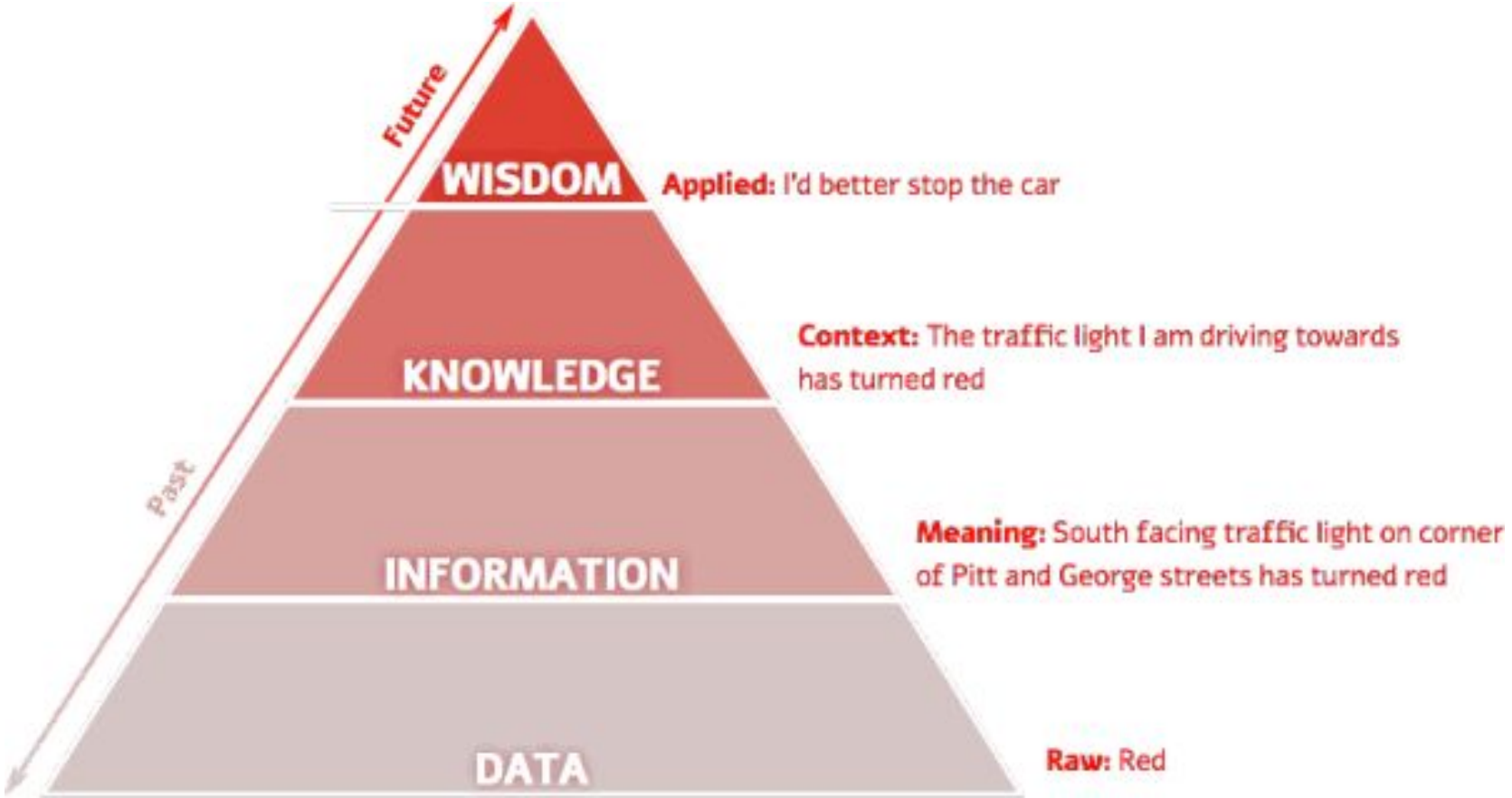
- You understand the importance of the voice of the customer but you may not know how that voice translates into data
- You have designed innovative products and processes but you need to know how to continually improve these things using data
- DMAIC is a data-driven concept

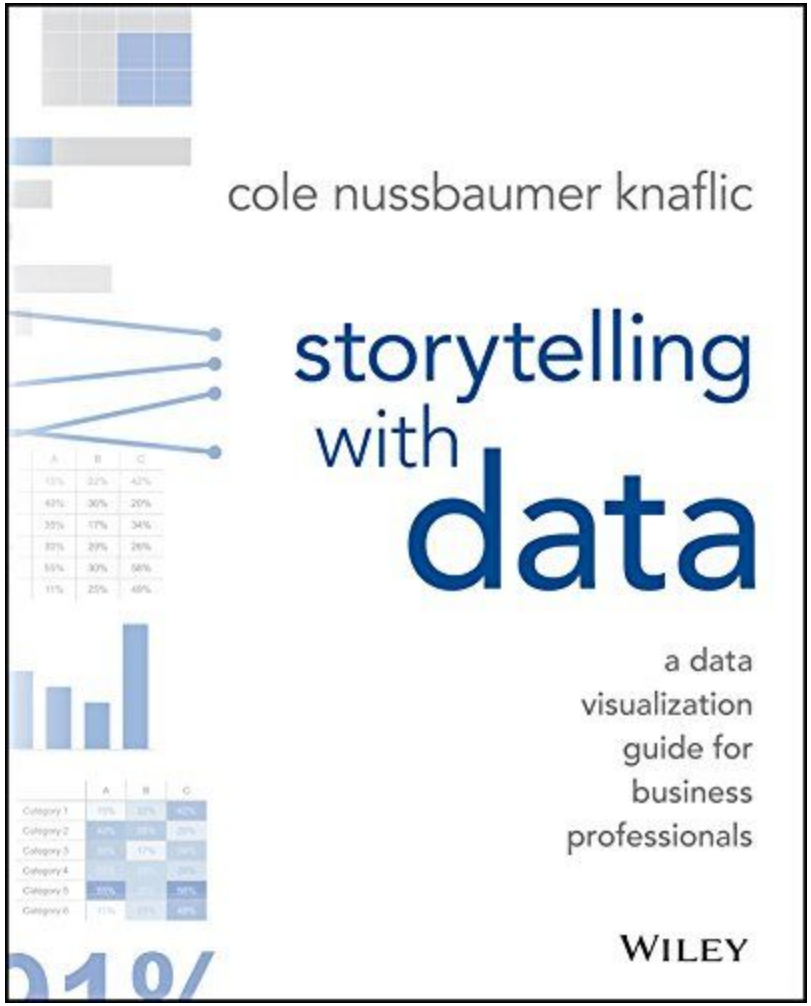
Why learn data science?

- It is important to know
- It spans the QUEST disciplines of business, engineering, and science
- It will help you in your future QUEST classes
  - 390 – building models using python
  - 390 – designing systems where data is an important feedback mechanism
  - 490 – all of the capstone projects have elements of data science
- It will be important to your career

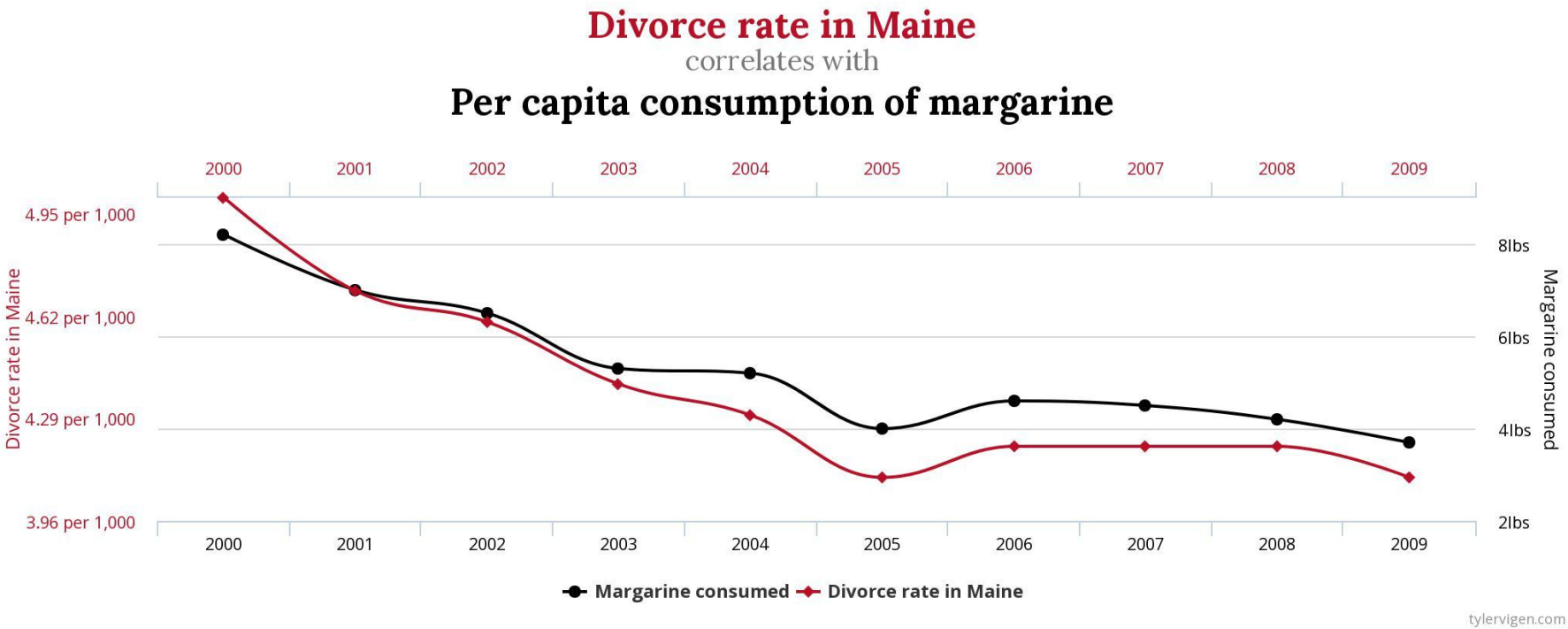
1. Data Science

1. Frame the Problem
2. Collect Raw Data
3. Process the Data
4. Explore the Data
5. Analyze the Data
6. Communicate the Insights

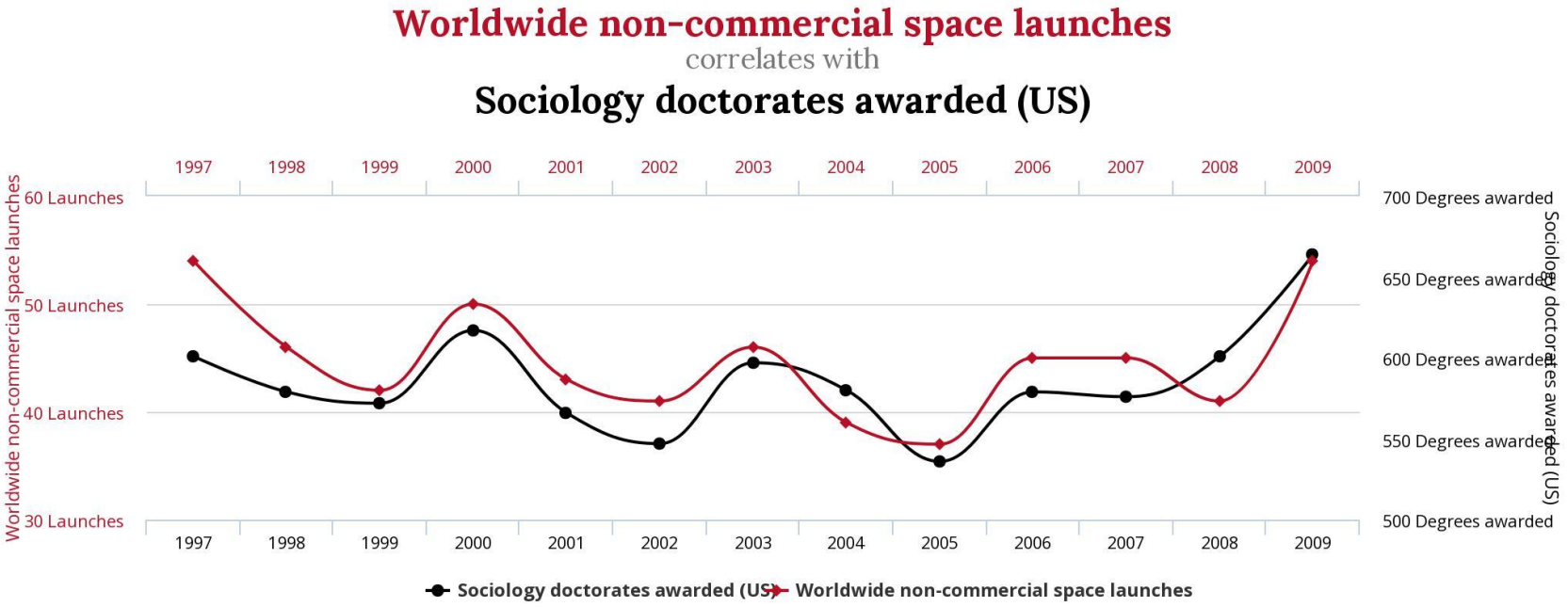

Future
Past
WISDOM
Applied: I'd better stop the car
KNOWLEDGE
Context: The traffic light I am driving towards has turned red
INFORMATION
Meaning: South facing traffic light on corner of Pitt and George streets has turned red
DATA
Raw: Red


cole nussbaumer knaflic
storytelling with data
a data visualization guide for business professionals
WILEY

Spurious correlations



Spurious correlations



2. Framing the Opportunity

Develop good client communication norms

- Introduce yourselves in an email
- Set expectations
  - Communication norms (email/phone/text?)
  - Email response time (one business day)
  - Don’t let them push around (stay in scope)
- Have weekly meetings (around 30 minutes)
  - Set an agenda (facilitator)
  - Take notes (notetaker)
  - Send a recap
- Ask lots of questions
- Communicate intermediate and draft deliverables

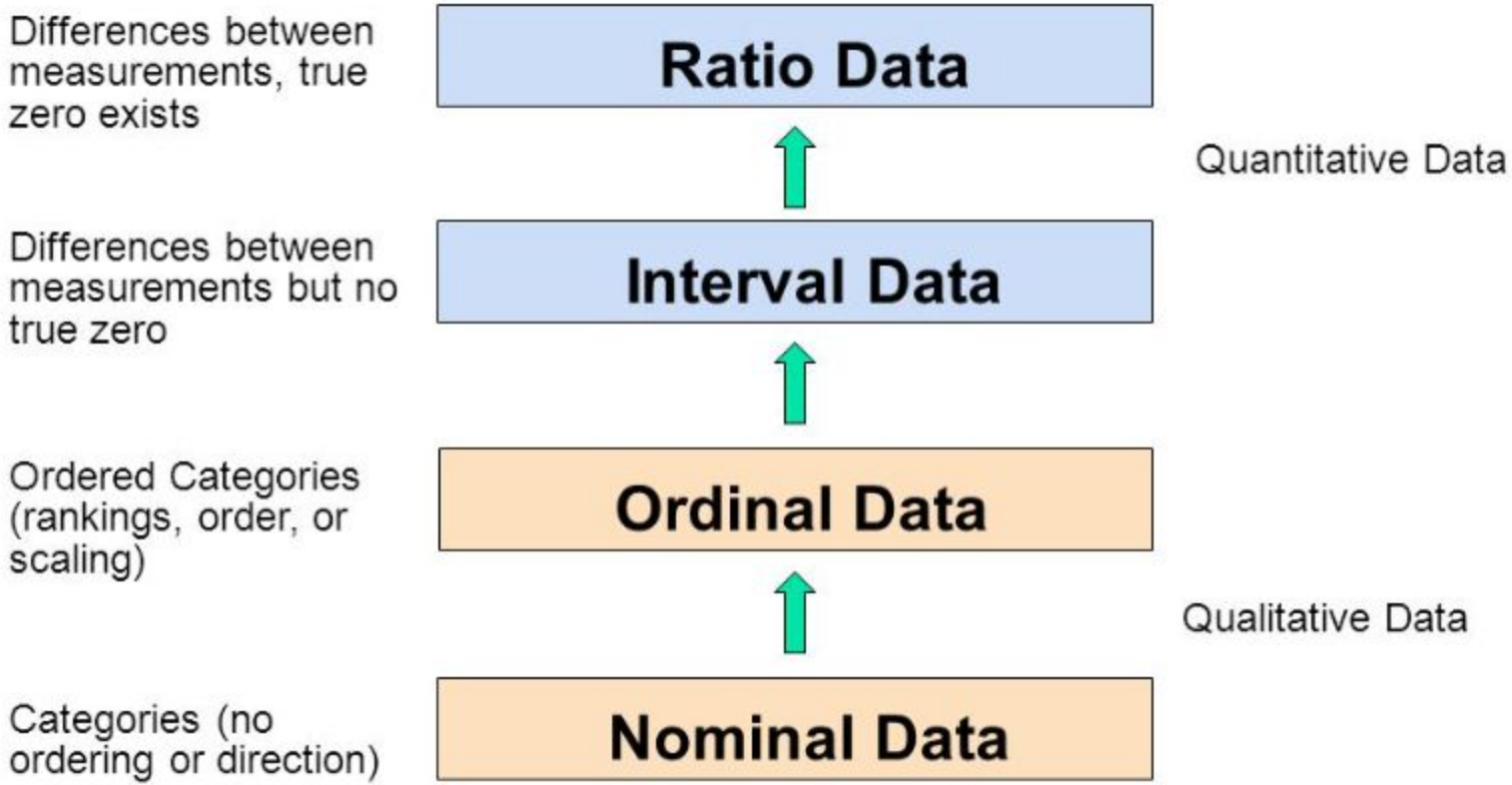
QUEST Honors Program:
Statement of Confidentiality

I acknowledge that as part of my participation in this QUEST course, I will be given access to data and information that is of a confidential nature. I agree to:

- Hold all confidential data and information in trust and strict confidence and only use the information for the purposes required to fulfill course obligations.
- Keep any confidential data and information in my control or possession in a physically secure location to which only I and other persons who have signed a confidentiality agreement with QUEST have access.
- Not to publicly present confidential data and information unless I have obtained pre-authorization from my project champion.
- Delete all confidential information immediately at the conclusion of the QUEST course.

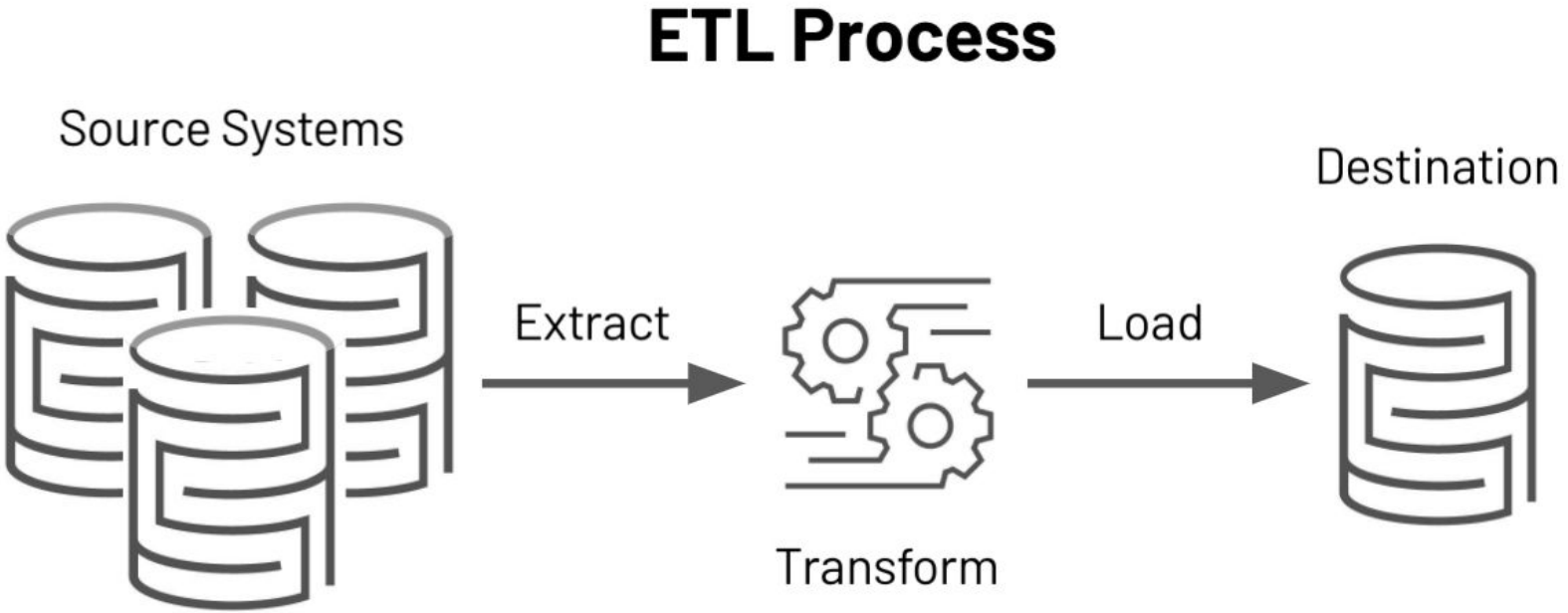
Questions to Consider

- What variables do you have?
  - independent variables vs. dependent variables
- What is your unit of analysis
  - Individual, group, company, market, country
- Is your data collected at the same time
  - Cross-sectional vs. time series
- How many observations do you have?
  - 10 vs. 100 vs. 1,000 vs. 10k vs. 100k +
- What unobserved heterogeneity do you have?


Differences between measurements, true zero exists
Ratio Data
Quantitative Data
Differences between measurements but no true zero
Interval Data
Ordered Categories (rankings, order, or scaling)
Ordinal Data
Qualitative Data
Categories (no ordering or direction)
Nominal Data

3. Data Quality

Prompt: If Chipotle wanted to see how well each store performed, what data would it collect?



ETL: Extract, Transform, and Load

Data Exploration Steps

- Univariable study - understand each of the key variables
- Multivariate study - look for relationships in the data
- Test assumptions - does our data meet the assumptions of multivariate methods?

Useful starting points for cleaning the data

- Confirm data type and number of non-null in each column: use `info()`
- Look for invalid values (numerical data) - use `describe()`
- Remove duplicate elements: `unique()`
- Check validity of values in columns (ex. number of students in a class must be >=0)
- Visually inspect snapshot of values with `df.head()`

.info() example

```
nba.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126314 entries, 0 to 126313
Data columns (total 23 columns):
gameorder        126314 non-null int64
game_id          126314 non-null object
lg_id            126314 non-null object
_iscopy          126314 non-null int64
year_id          126314 non-null int64
date_game        126314 non-null object
seasongame       126314 non-null int64
is_playoffs      126314 non-null int64
team_id          126314 non-null object
fran_id          126314 non-null object
pts              126314 non-null int64
elo_i            126314 non-null float64
elo_n            126314 non-null float64
win_equiv        126314 non-null float64
opp_id           126314 non-null object
opp_fran         126314 non-null object
opp_pts          126314 non-null int64
opp_elo_i        126314 non-null float64
opp_elo_n        126314 non-null float64
game_location    126314 non-null object
game_result      126314 non-null object
forecast         126314 non-null float64
notes            5424 non-null object
dtypes: float64(6), int64(7), object(10)
memory usage: 22.2+ MB
```

Univariate Study: Descriptive Statistics

- `describe()` calculates descriptive statistics
- `size()` returns the number of elements
- `mean()` returns the sample mean
- `median()` returns the sample median
- `range()` returns the largest and smallest values
- `quantile()` calculates percentiles
- `count(), min(), max(), std()`

Descriptive Statistics

Using `describe()` on the dataframe (applied to all numeric variables) and on individual columns, both numerical and categorical.

[4]
```
import pandas as pd
df = pd.read_csv('nba_data.csv')
```

[5]
```
df.describe()
```

	Number	Age	Weight	Salary
count	457.000000	457.000000	457.000000	4.460000e+02
mean	17.678337	26.938731	221.522976	4.842684e+06
std	15.966090	4.404016	26.368343	5.229238e+06
min	0.000000	19.000000	161.000000	3.088800e+04
25%	5.000000	24.000000	200.000000	1.044792e+06
50%	13.000000	26.000000	220.000000	2.839073e+06
75%	25.000000	30.000000	240.000000	6.500000e+06
max	99.000000	40.000000	307.000000	2.500000e+07

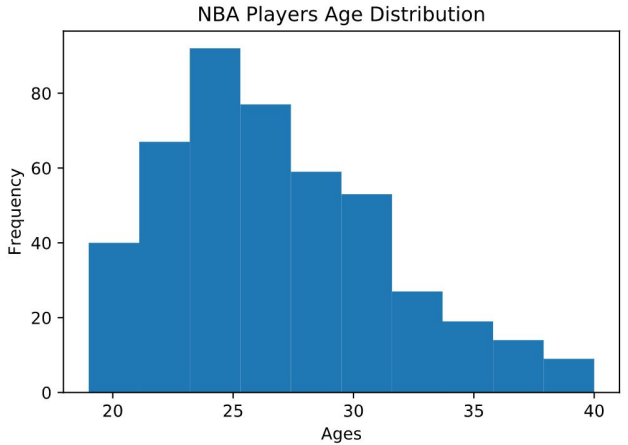
```
df['Salary'].describe()
```

```
count    4.460000e+02
mean     4.842684e+06
std      5.229238e+06
min      3.088800e+04
25%      1.044792e+06
50%      2.839073e+06
75%      6.500000e+06
max      2.500000e+07
Name: Salary, dtype: float64
```

```
df['College'].describe()
```

```
count           373
unique          118
top        Kentucky
freq             22
Name: College, dtype: object
```

Univariate Study: Frequency distributions



Create a dataframe for numerical variables

```
df_num=df.select_dtypes(include=['float64','int64'])
```

Explore df.plot.hist(), df.plot.bar()

- Can name axes with

```
df.set(xlabel="...", ylabel="...", title="...")
```

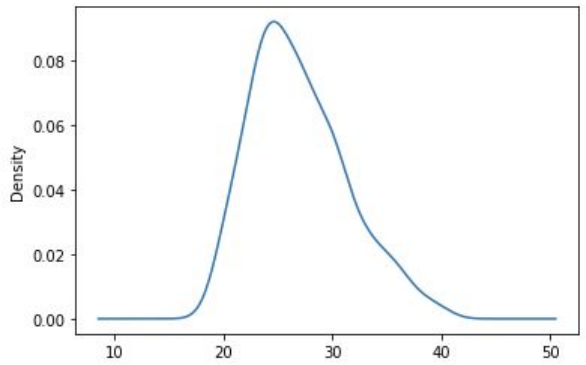
Univariate Study: Kernel Density Estimation

Creates a continuous curve based on histogram of data

Can be used to estimate the probability density function

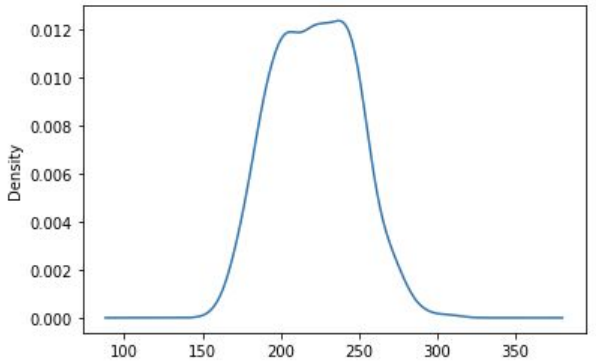
```
df['Age'].plot.kde()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11dd680d0>
```




```
df['Weight'].plot.kde()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a20a01750>
```



Agency Name	Investment Title	Project ID	Agency Project ID	Project Name	Project Description	Start Date	...	Lifecycle Cost	Schedule Variance (in days)	Schedule Variance (%)	Cost Variance ($ M)
Department of Agriculture	AMS Infrastructure WAN and DMZ (AMSWAN)	656.0	NaN	Operations	Annual Agency Operations.	01/10/2011	...	15.2970	0.0	0.00	0.0000
Department of Agriculture	AMS Infrastructure WAN and DMZ (AMSWAN)	657.0	NaN	Virtualization	Program Areas will migrate their data over to ...	01/10/2011	...	0.1790	-153.0	-84.07	0.0000
Department of Agriculture	AMS Infrastructure WAN and DMZ (AMSWAN)	658.0	NaN	Refresh	Programs Areas will replace 1/3 of their compu...	01/04/2012	...	1.4600	0.0	0.00	0.0000
Department of Agriculture	APHIS Electronic Permits System (ePermits)	661.0	NaN	ePermits O&M FY11 Part 1.	Production Support including Analysis, Softwar...	01/04/2011	...	1.8205	0.0	0.00	0.3641
Department of Agriculture	APHIS Electronic Permits System (ePermits)	662.0	NaN	ePermits O&M FY12 Part 1	Production Support including Analysis, Softwar...	01/04/2012	...	1.7130	0.0	0.00	0.0000

Schema (how the data is constructed)

- INT
- FLOAT
- DATE (what time zone is it in?)
- STRING
- BOOLEAN

```
1796,15,4,Clark,3,1
1797,16,3.08,Smith,4.5,1
1798,15,3.74,Smith,2,0
1799,60,3.93,CMNS,4.5,1
1801,16,3.83,Smith,3,1
1802,14,4,Clark,4.5,1
1803,17,3.95,CMNS,5,1
1805,15,3.59,Clark,4.5,1
1807,15,4,CMNS,5,1
1809,16,3.63,Smith,3.5,1
1810,16,3.57,CMNS,3.5,1
1812,15,4,Clark,3.5,1
1814,16,3.92,Clark,0,1
1816,16,3.5,CMNS,4,1
1818,18,4,CMNS,5,1
```

In [4]:
```
import pandas as pd
```

In [6]:
```
df = pd.read_csv('contact_info.csv')
```

In [7]:
```
df
```

Out[7]:

	id	name	pronouns	email	cell
0	1	Joe Bailey	He/Him/His	jpbailey@umd.edu	(240) 464-8096
1	2	Tania Arya	She/Her/Hers	tarya@terpmail.umd.edu	NaN
2	3	Vivek Banerjee	He/Him/His	vivekjyoti24@gmail.com	NaN
3	4	Utsa Santhosh	She/Her/Hers	utsa.santhosh@gmail.com	NaN

In [8]:
```
df['email_hash']=df['email'].apply(hash)
```

In [9]:
```
df
```

Out[9]:

	id	name	pronouns	email	cell	email_hash
0	1	Joe Bailey	He/Him/His	jpbailey@umd.edu	(240) 464-8096	-8720133959821803574
1	2	Tania Arya	She/Her/Hers	tarya@terpmail.umd.edu	NaN	8835033280126197767
2	3	Vivek Banerjee	He/Him/His	vivekjyoti24@gmail.com	NaN	4854794598247253376
3	4	Utsa Santhosh	She/Her/Hers	utsa.santhosh@gmail.com	NaN	-2460795020607407826

field	suggested action	notes
Name	keep	looks like a unique identifier for the transaction
Email	mask	uniquely identify a customer but make it anonymous
Financial Status	keep	
Paid at	keep	
Fulfillment Status	keep	
Fulfilled at	keep	
Accepts Marketing	keep	
Currency	keep	
Subtotal	keep	
Shipping	keep	
Taxes	keep	
Total	keep	
Discount Code	keep	
Discount Amount	keep	
Shipping Method	keep	
Created at	keep	
Lineitem quantity	keep	
Lineitem name	keep	
Lineitem price	keep	
Lineitem compare at price	keep	
Lineitem sku	keep	
Lineitem requires shipping	keep	
Lineitem taxable	keep	
Lineitem fulfillment status	keep	
Billing Name	drop	PII concerns
Billing Street	drop	PII concerns
Billing Address1	drop	PII concerns
Billing Address2	drop	PII concerns
Billing Company	drop	PII concerns
Billing City	keep	
Billing Zip	keep	
Billing Province	keep	
Billing Country	keep	

In [ ]:
```
import pandas as pd
```

In [ ]:
```
df = pd.read_csv('full_data.csv')
```

In [ ]:
```
df.head()
```

In [ ]:
```
df['email_hash'] = df['Email'].apply(hash)
```

In [ ]:
```
df_mapping=df[['Email', 'email_hash']]
df_mapping.head()
```

In [ ]:
```
df_mapping.to_csv("email_hash_mapping_full_data.csv")
```

In [ ]:
```
del df['Email']
del df['Billing Name']
del df['Billing Street']
del df['Billing Address1']
del df['Billing Address2']
del df['Billing Company']
del df['Billing Phone']
del df['Shipping Name']
del df['Shipping Street']
del df['Shipping Address1']
del df['Shipping Address2']
del df['Shipping Company']
del df['Shipping Phone']
del df['Phone']
```

In [ ]:
```
df.to_csv("full_data_scrubbed.csv")
```

What's wrong here?

	A	B	C	D	E	F
1	Name	Applicants total	Admissions total	Enrolled total	Percent of freshmen submitting SAT scores	Percent of freshn SA
2	Alabama A & M University	6142	5521	1104	15	88
3	University of Alabama at Birmingham	5689	4934	1773	6	93
4	Amridge University					
5	University of Alabama in Huntsville	2054	1656	651	34	94
6	Alabama State University	10245	5251	1479	18	87
7	The University of Alabama	30975	17515	6454	23	76
8	Athens State University					
9	Auburn University at Montgomery	1958	1639	579	0	54
10	Auburn University	15745	13027	3726	17	83
11	Birmingham Southern College	1931	1240	356	32	82
12	Concordia College Alabama					
13	Faulkner University					
14	Huntingdon College	1470	923	261	16	93
15	Jacksonville State University	3083	2567	1158	18	89
16	Judson College	268	198	68	10	94
17	University of West Alabama	462	460	380	0	100
18	Miles College					
19	University of Mobile	866	617	259	13	87
20	University of Montevallo	1385	1209	531	3	96
21	University of North Alabama	2542	2057	970	3	96
22	Oakwood University	2728	937	409	47	58
23	Samford University	3447	2653	765	36	84
24	University of South Alabama	4814	4142	1878	8	92
25	Spring Hill College	6596	3052	431	20	92
26	Stillman College	4121	1772	257	6	59
27	Talladega College					
28	Troy University	5946	4056	2087	7	38
29	Tuskegee University	10022	3519	650	43	75
30	University of Alaska Anchorage					

What's wrong here?

	A	AH	AI
1	Name	Total price for in-state students	Total price for out-of-state students l
2	United States Merchant Marine Academy	9768	9768
3	Brigham Young University-Idaho	11398	11398
4	Southwestern Christian College	15085	15085
5	Rust College	15400	154oo
6	Oklahoma Panhandle State University	15888	15888
7	Southern University at New Orleans	16191	16191
8	Minot State University	16196	16196
9	Chadron State College	16440	16470
10	Peru State College	16556	16556
11	Brigham Young University-Hawaii	16564	16564
12	Donnelly College	16,833	16833
13	Goddard College	17018	17018
14	Brigham Young University-Provo	17496	17496
15	Blue Mountain College	17550	17550
16	Concordia College Alabama	18330	18330
17	Dickinson State University	16044	18396
18	Le Moyne-Owen College	18540	18540
19	Mississippi Valley State University	18552	18.552
20	Mayville State University	16445	18767
21	Delta State University	18887	18887

Date?

Start Date	Completion D	Planned Proj	Projected/Ac	L
1/10/11	2012-30-09			
1/10/11	31/03/2012	31/03/2012	31/03/2012	
1/4/12	30/09/2012			
1/4/11	30/09/2011	30/09/2011	30/09/2011	
1/4/12	30/09/2012	30/09/2012	30/09/2012	
1/10/12	31/03/2013			
31/12/2010	31/03/2011	31/03/2011	31/03/2011	
1/10/10	30/09/2011	30/09/2011	20/09/2011	
15/08/2011	30/04/2012	30/04/2012	30/04/2012	
6/9/11	31/12/2012			
6/9/11	21/12/2012	2/3/12	2/3/12	
3/10/11	7/9/12	7/9/12	7/9/12	
1/11/11	31/05/2012	31/05/2012	31/05/2012	
1/3/12	24/08/2012	24/08/2012	24/08/2012	
1/10/11	30/09/2012			
1/10/02	28/09/2018	28/09/2012	28/09/2012	
1/10/02	28/09/2018	28/09/2012	28/09/2012	
1/10/02	28/09/2018			
1/10/02	28/09/2018	28/09/2012	28/09/2012	
1/10/02	28/09/2018	28/09/2012	28/09/2012	
1/7/11	30/09/2012	30/09/2012	30/09/2012	
1/7/11	30/09/2012	30/09/2012	30/09/2012	
1/7/11	30/09/2012	30/09/2012	30/09/2012	
1/10/04	30/09/2012	30/09/2012	30/09/2012	
1/7/11	30/09/2012			

Data Quality Issues

- Date validation
- Number format
- Text ⇔ numbers
- Missing data
- Duplicate data
- Redundant data
- Combined data and separated data

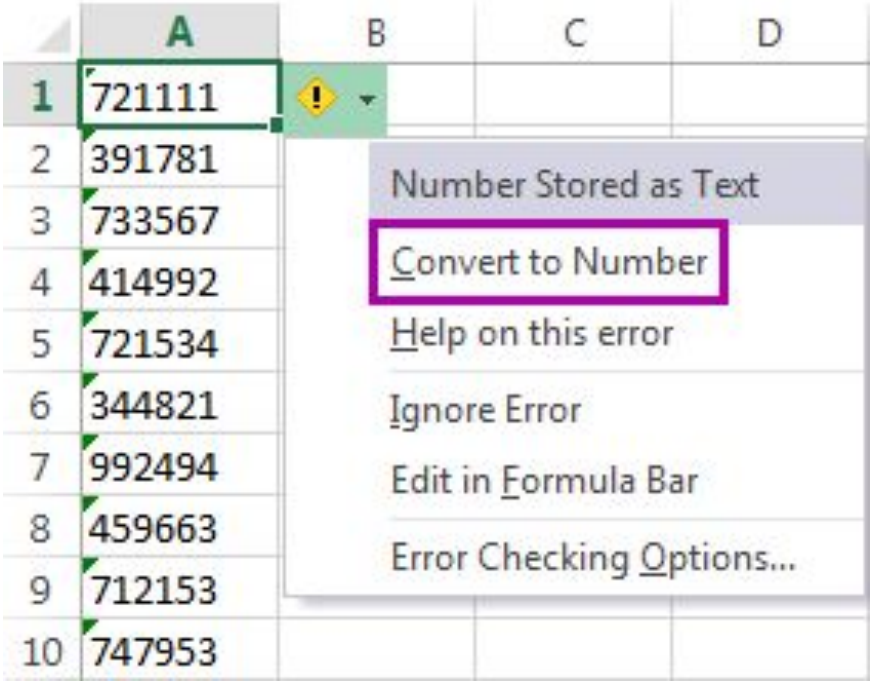
Date validation

- What month and day is the date: 30/01/2019?
- How about the year 04/05/2019?
- How about 2019-04-05?
- Example:
  - df[‘date’] = pd.to_datetime(df['string_date'])

Number format

- If the price for one item is 5.00 and another is 4.95, are they the same price?
- What happens if you round the price for each item to 5?
- What happens if the price is reported as an integer once (5) and a decimal another place (4.95) for the same item?
- Example:
  - round(4.95, 0) => 5.0
  - int(4.95) => 4
  - int(round(4.95,0)) => 5

Text ⇔ numbers




```
id="123456789"
print(type(id)) => str
id_num = int(id)
print(type(id_num)) => int
```

Common Causes of Outliers

- Data entry errors (human errors)
- Measurement errors (instrument errors)
- Data processing errors (data manipulation or data set unintended mutations)
- Sampling errors (extracting or mixing data from wrong or various sources)
- Natural (not an error, novelties in data)

When to drop outliers

When it is obvious that it is due to incorrectly entered or measured data (e.g., age is 231)

Otherwise, keep outliers for now, but make note of them; we will need to address them in our analysis.

Outliers often affect both results and assumptions of analysis.

4. Data Transformations

You are analyzing donor data and you see the following two donations were made in 2019:

Name	Email	Address	Amount	Date
John Smith	jksmith@startup.com	124 Main St.; College Park, MD	$1,000	1/15/2019
John K. Smith	jksmith01@umd.edu	124 Main St.; College Park, MD	$100	6/1/2019

Is this the same person?

Types of transformations

Expanding and collapsing rows and columns

Mathematical transformations

Transforming words

Transforming images

Collapsing and Expanding Columns

- Collapsing
  - Combine two pieces of data into one
  - Example:
    - Df[‘name’] = df[‘first_name’] + “ “ + df[‘last_name’]
- Expanding
  - Parse an observation based on a delimiter
  - Example:
    - df[‘name’] = “John Smith” into df[‘first_name’] = “John and df[‘last_name’] = “Smith”

Collapsing rows

Order	Customer	Amount
1	A	$20
2	A	$20
3	B	$5
4	A	$15
5	B	$55

Customer	Amount	Orders
A	$55	3
B	$60	2

Expanding rows

Order	Customer	Amount	Date
1	A	$20	1/15/22
2	A	$20	2/20/22
3	B	$5	2/25/22
4	A	$15	3/30/22
5	B	$55	3/30/22

Customer	Month	Spending
A	1	$20
B	1	$0
A	2	$20
B	2	$5
A	3	$15
B	3	$55

Questions to ask when collapsing and expanding

What is the unit of analysis?

Should I transform the data to work with a different unit of analysis?

When an observation is missing, does it tell me something?

Are there variables that work better together or separate?

What's wrong here?

id	name
1	Jay
2	May
2	Kay

id	number
1	1234567
1	464-8096
2	(301) 983-2928
2	293-9827
3	973-696-9627

Missing data

- If we have a database with gender information and some are marked as ‘f’ and some are marked as ‘m’, what can we infer when the data is missing?
- If we are looking at temperature over time and have observations every 5 minutes, but one of the observations is missing, what do we do?

2pm	2:05pm	2:10pm	2:15pm	2:20pm
65	65		65	64

```
temps=[65, 65, '', 65, 64]
```

```
count = 0
while count < len(temps):
    if temps[count] == '':
        temps[count] = (temps[count-1] + temps [count+1])/2
    count = count +1
temps
```

```
[65, 65, 65.0, 65, 64]
```

Duplicate data

- What happens if we have two people with the same name in our database? Does it matter if the name is “John Smith” vs. “Fifer Testudo”?
- If two people on the same QUEST team submit the team deliverable, which one should we accept?

Submission 1	Submission 2
10/1/2020	10/1/2020
11:05pm	11:30pm
15 pages	10 pages
pdf	docx

Redundant data

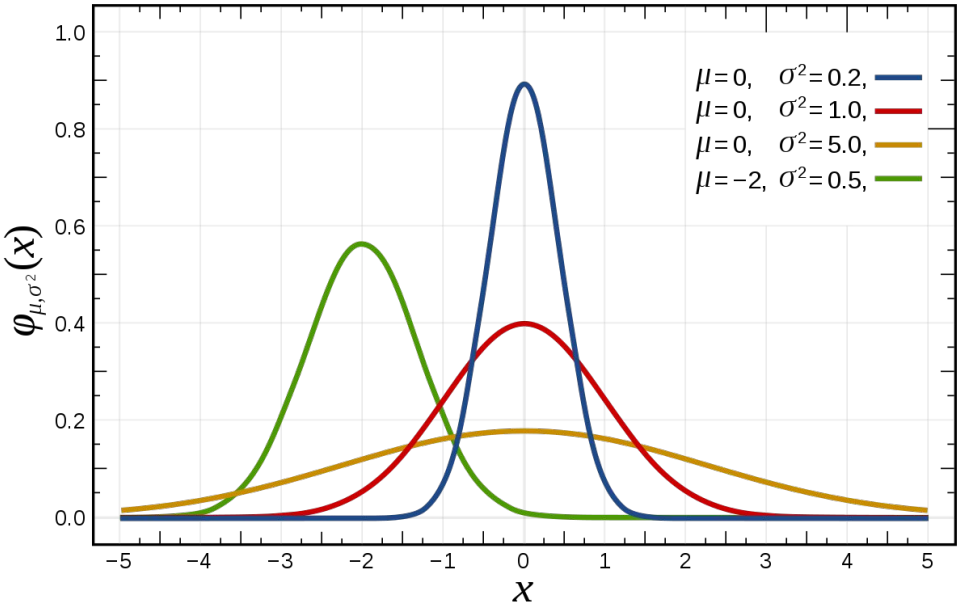
- If we have birth date and age in a database, do we need to keep both?
  - If no, which one do we keep?

	id	name	birth	weight
0	1	Joe Bailey	10-16-1970	NaN
1	1	NaN	NaN	183.6
2	2	John Smith	2-28-1965	NaN
3	2	NaN	NaN	170.2

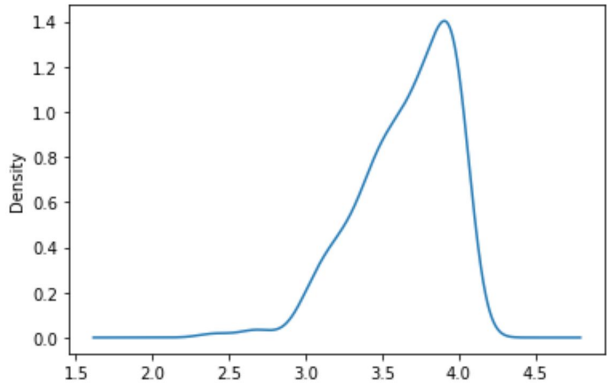
```
df = pd.read_csv('process.csv')

for index, row in df.iterrows():
    if pd.isnull(row['weight']):
        df.iloc[index,3] = df.iloc[(index+1),3]
df
```

	id	name	birth	weight
0	1	Joe Bailey	10-16-1970	183.6
1	1	NaN	NaN	183.6
2	2	John Smith	2-28-1965	170.2
3	2	NaN	NaN	170.2

Normality of Observation



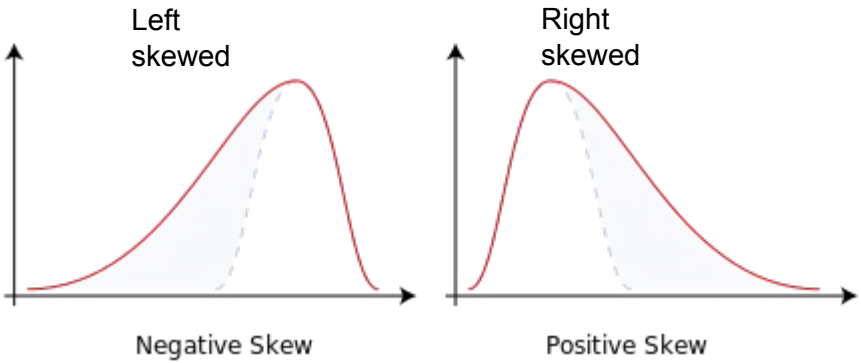
```
In [12]: df_filtered['gpa'].plot.kde()
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f837b643100>
```

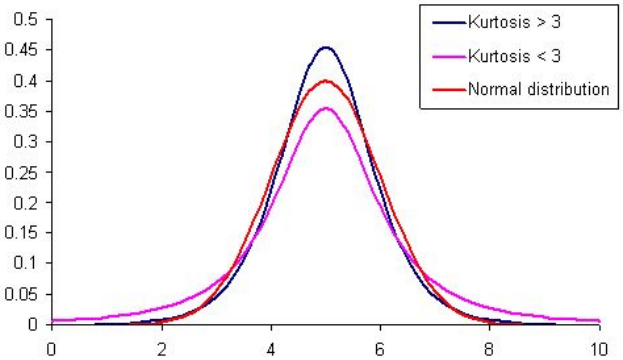



```
In [13]: df_filtered['gpa'].skew()
Out[13]: -0.9544798519591109

In [14]: df_filtered['gpa'].kurtosis()
Out[14]: 0.7988453164694445
```

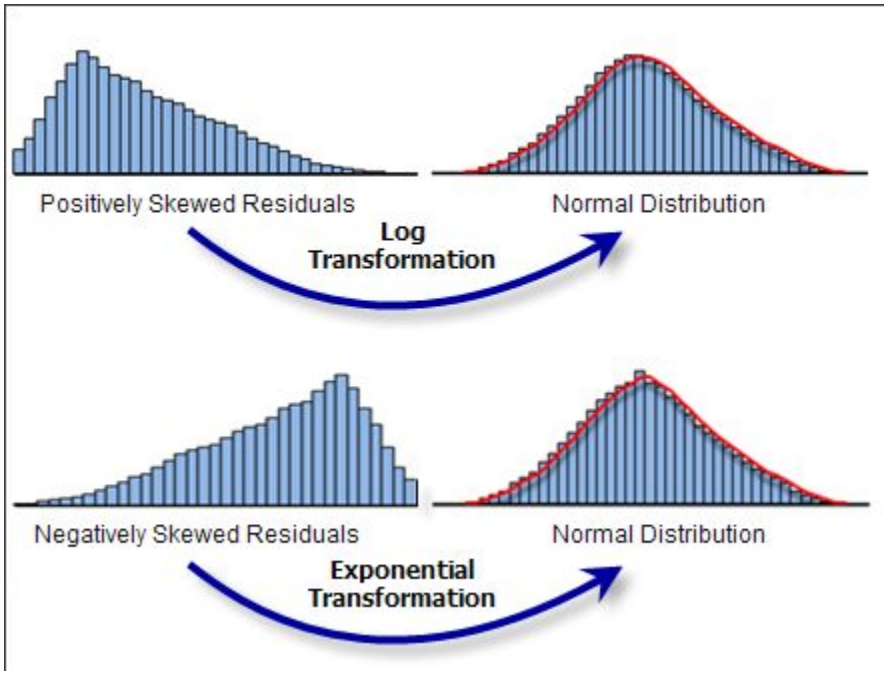
rule of thumb -0.8 to 0.8 for skewness and -3.0 to 3.0 for kurtosis.

Skewness, Kurtosis





Log and Exponential Transformations



```
import pandas as pd
df = pd.read_csv('process.csv')

df['first_name']=""
df['last_name']=""

for index, row in df.iterrows():
    if (type(row['name']) == str):
        first_name = (str.split(df.iloc[index]['name'])[0])
        last_name = (str.split(df.iloc[index]['name'])[1])
        df.iloc[index,4] = first_name
        df.iloc[index,5] = last_name

df
```

	id	name	birth	weight	first_name	last_name
0	1	Joe Bailey	10-16-1970	NaN	Joe	Bailey
1	1	NaN	NaN	183.6		
2	2	John Smith	2-28-1965	NaN	John	Smith
3	2	NaN	NaN	170.2		

Image transformations

Patterns for matching

Handwriting recognition

Optical Character Recognition

```
import pandas as pd
```

```
df = pd.read_csv("~/Desktop/questselections18.csv")
```

```
dummies = pd.get_dummies(df["school"])
df = pd.concat([df, dummies], axis=1)
```

```
df.head()
```

	id	credits	gpa	school	score	interview	CMNS	Clark	Smith
0	1757	15	3.55	CMNS	3.5	1	1	0	0
1	1760	15	4.00	CMNS	4.0	1	1	0	0
2	1761	15	3.45	Clark	4.5	1	0	1	0
3	1762	18	3.40	Clark	4.5	1	0	1	0
4	1763	19	3.38	Smith	2.5	0	0	0	1

```
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: Futur
  import pandas.util.testing as tm
```

[2]
```
df = pd.read_csv('questselections18.csv')
```

[3]
```
# we need to clean up the dataframe to remove instances where gpa == 0
df = df[df.gpa != 0]
```

[4]
```
# moving the categorical variables to dummy variables
df = df.join(pd.get_dummies(df['school'], drop_first=True))
df = df.drop('school',axis = 1)
```

Dummy Variables

- Make sure to use dummy variables for categorical variables
  - Number of dummies = # of groups -1
  - E.g. if year = {freshman, sophomore, junior, and senior}, then you need three dummy variables
- Often used in control variables
  - E.g. controlling for gender

Dummy Variables

- Make sure to use dummy variables for categorical variables
  - Number of dummies = # of groups -1
  - E.g. if year = {freshman, sophomore, junior, and senior}, then you need three dummy variables
- Often used in control variables
  - E.g. controlling for gender

```
import pandas as pd
```

```
df = pd.read_csv("~/Desktop/questselections18.csv")
```

```
dummies = pd.get_dummies(df["school"])
df = pd.concat([df, dummies], axis=1)
```

```
df.head()
```

	id	credits	gpa	school	score	interview	CMNS	Clark	Smith
0	1757	15	3.55	CMNS	3.5	1	1	0	0
1	1760	15	4.00	CMNS	4.0	1	1	0	0
2	1761	15	3.45	Clark	4.5	1	0	1	0
3	1762	18	3.40	Clark	4.5	1	0	1	0
4	1763	19	3.38	Smith	2.5	0	0	0	1

Natural Language Processing

How can a new product development team use Amazon reviews to design new products?

```
[4]  from textblob import TextBlob
```

```
[8]  sentences = [
         "bad ",
         "okay ",
         "awful",
         "good",
         "like",
     ]
```

```
for each in sentences:
  my_string = TextBlob(each)
  print(my_string.sentiment.polarity)
```

```
-0.6999999999999998
0.5
-1.0
0.7
0.0
```

General Techniques of NLP

- Remove stopwords
- Create stems of words (i.e. remove prefix and suffix)
- Lemmatize the words (i.e. move synonyms to the same root)
- Consider tuples of words
- Transform words into numbers (dummy encoding, gensim, …)

5. Team Presentations

Team Presentation Content

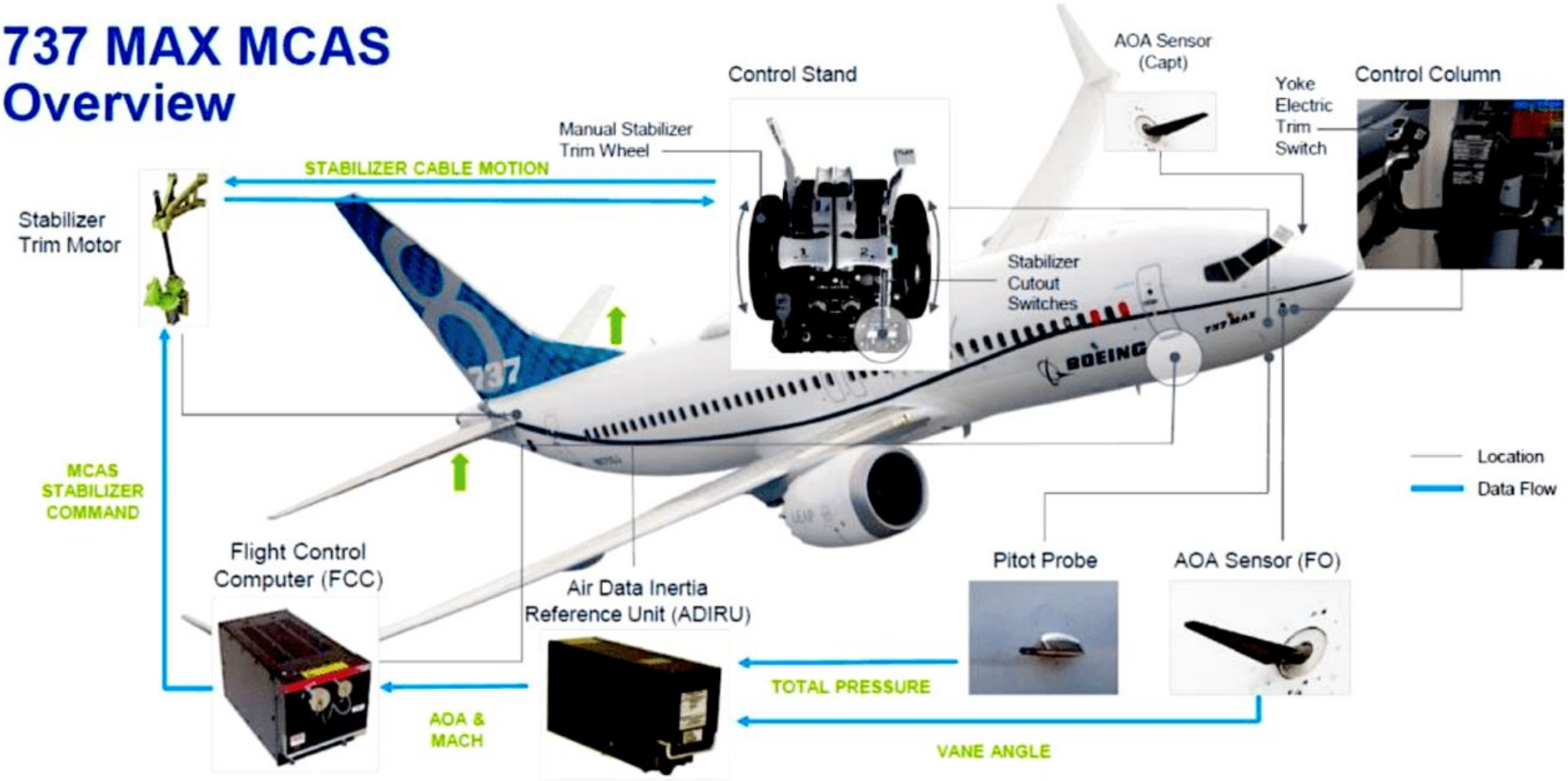
- Who is your client?
- How does your client use data?
- What questions does your client want to answer?
- What type of data will your client share with you?
- Have you encountered any data quality issues?
- Can you share with us any descriptive statistics?
- How does the qualitative data mesh with the quantitative data?

6. Data Exploration

Visit from Tuscany Consulting

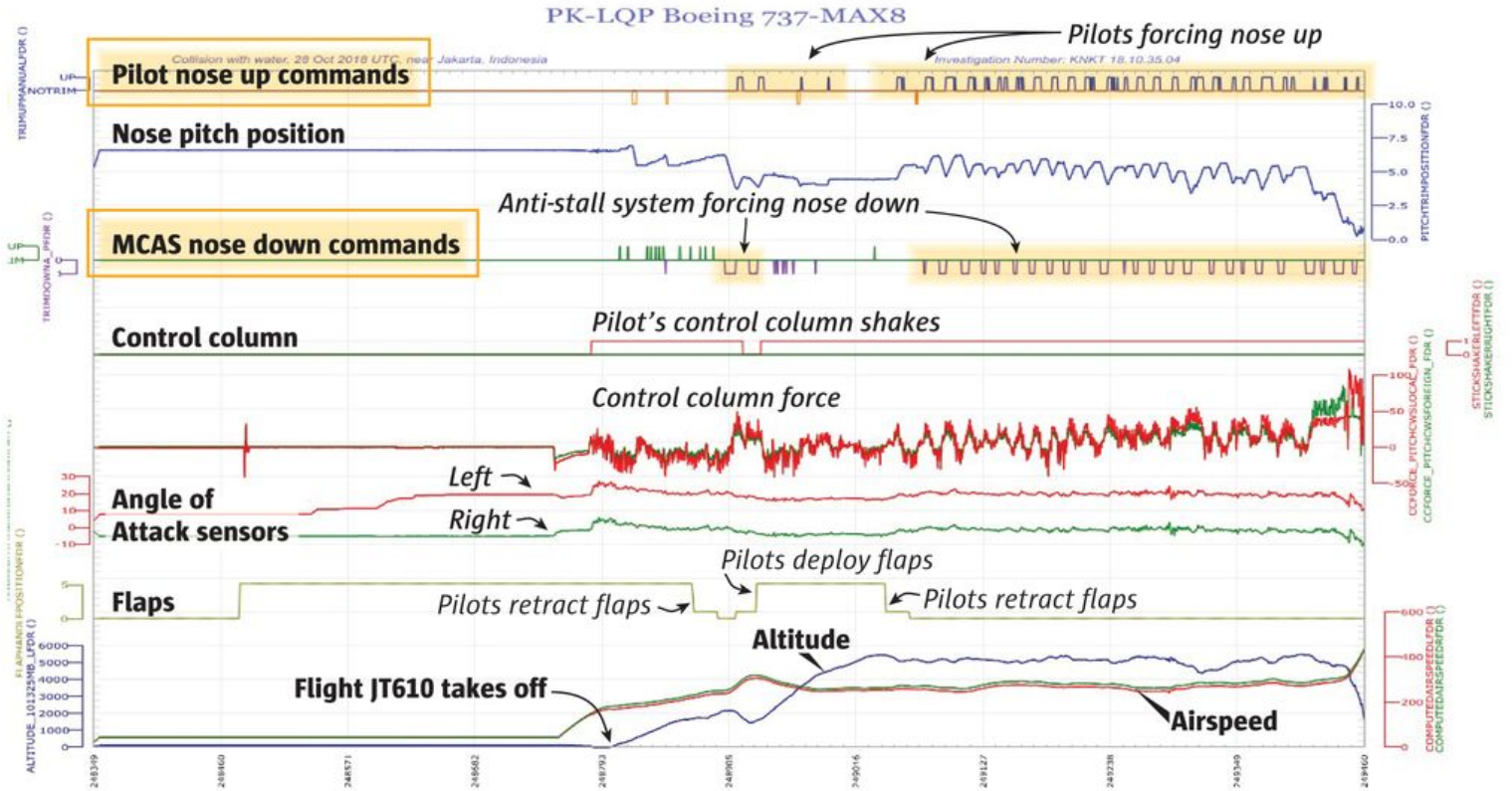
- How can you consider several variables at once in creating a ranking
- Warm-up: How can you look at the markets to sell snow shovels?
  - Weather
  - Demand
  - Competition
  - …
- Exercise: Where should a nursing school expand to a new location?
  - Need
  - Supply
  - Demographics
  - …

737 MAX MCAS Overview



The jet's nose is repeatedly pushed down

The new anti-stall system on the Boeing 737 MAX forced the nose of Lion Air JT610 down 26 times in 10 minutes before the pilots lost control and the plane dived into the sea.



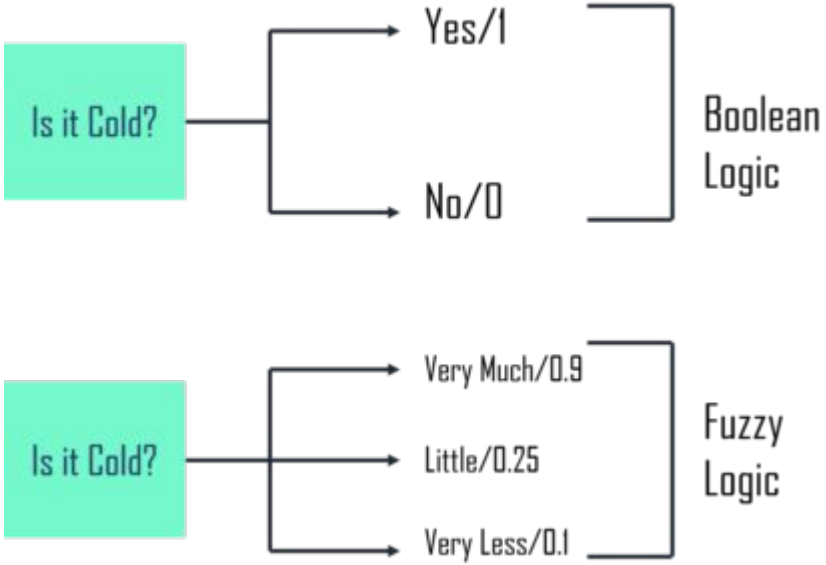
Sources: Indonesian safety regulators, black box flight recorder data

MARK NOWLIN / THE SEATTLE TIMES

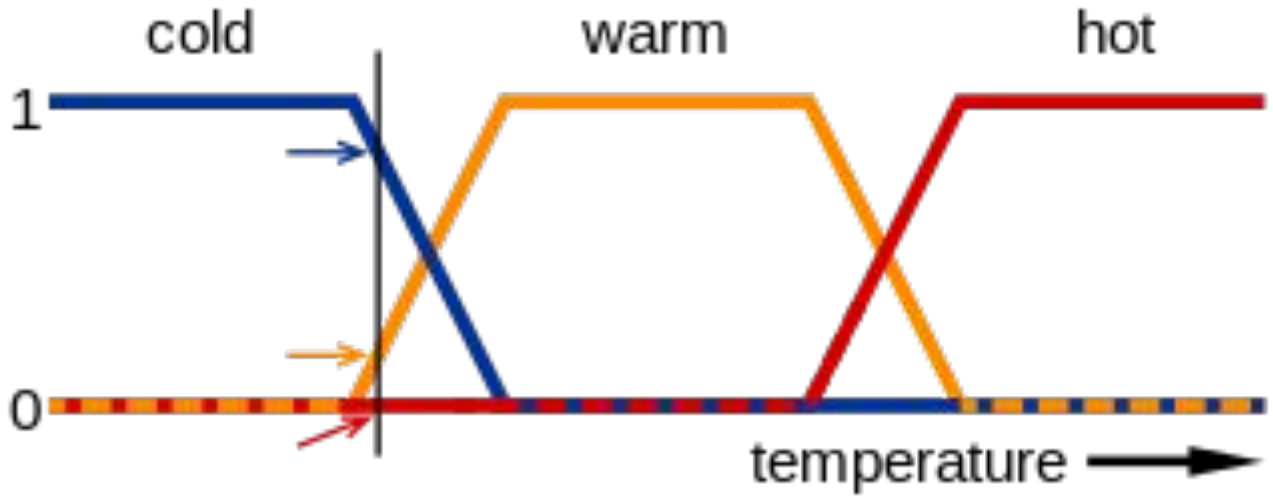
Fuzzy Logic

- How certain are you that your data is correct?
- What types of quality filters can we put on our data?
- How can we error-check the data?

Fuzzy Logic


Is it Cold?
Yes/1
No/0
Boolean Logic
Is it Cold?
Very Much/0.9
Little/0.25
Very Less/0.1
Fuzzy Logic

Fuzzy Logic

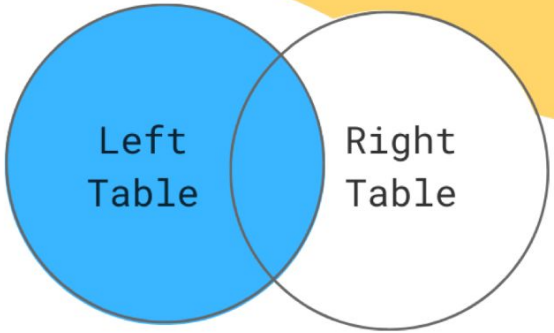


7. Data Integration

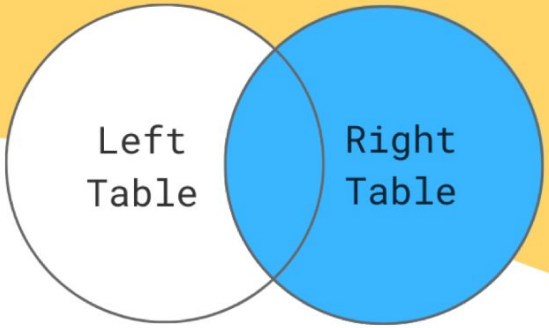
Visit from CapTech

- Data science in the real world
- Hypothesis development and testing
- Value of a model
- Use of qualitative and quantitative data

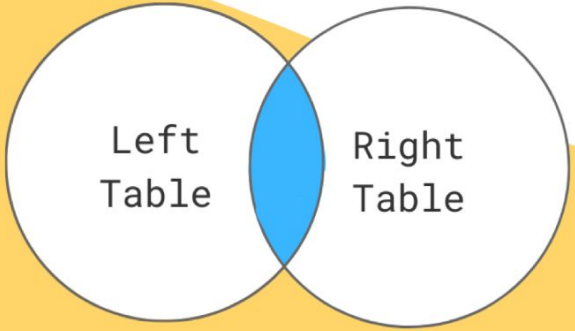
LEFT JOIN



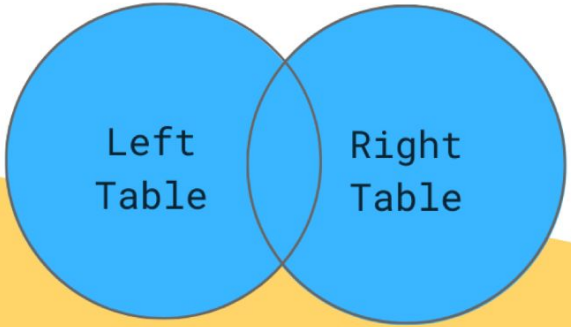
RIGHT JOIN



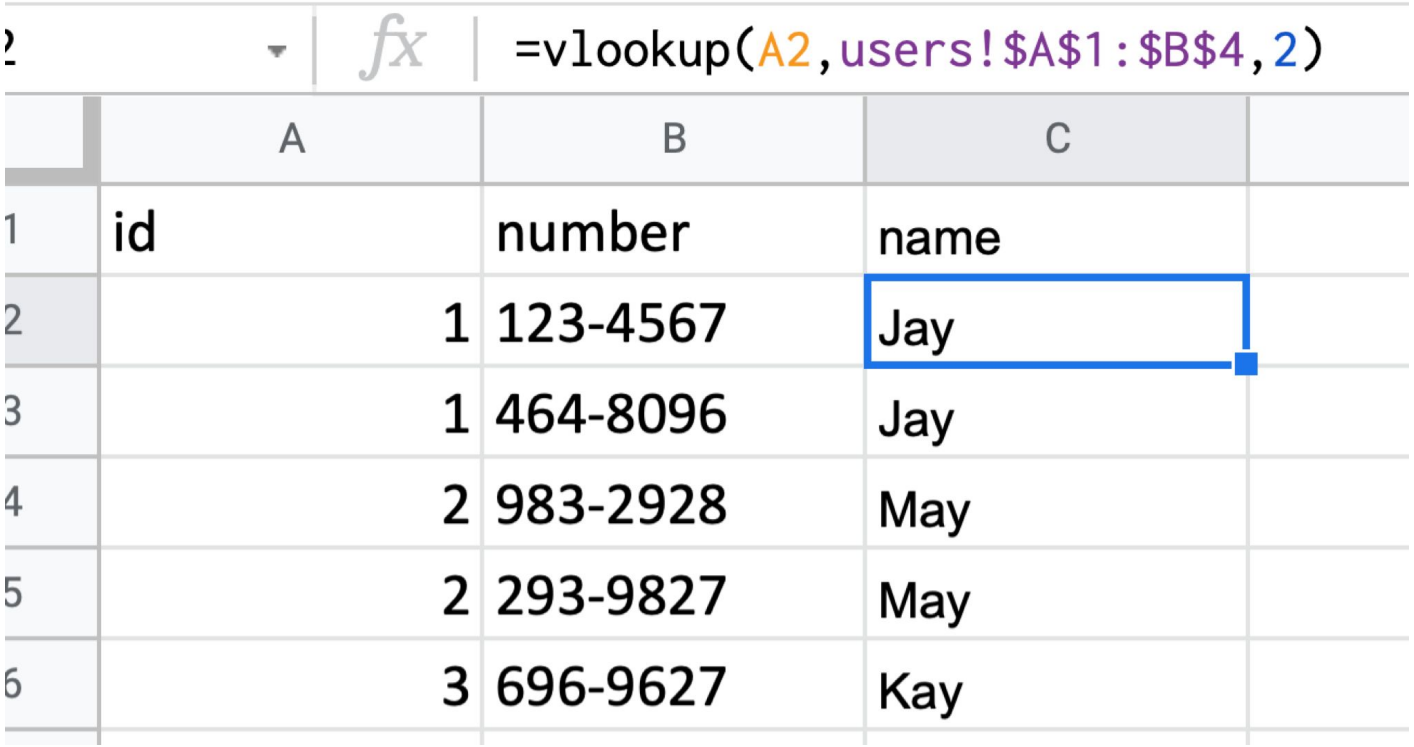
INNER JOIN



FULL JOIN



LearnSQL.com

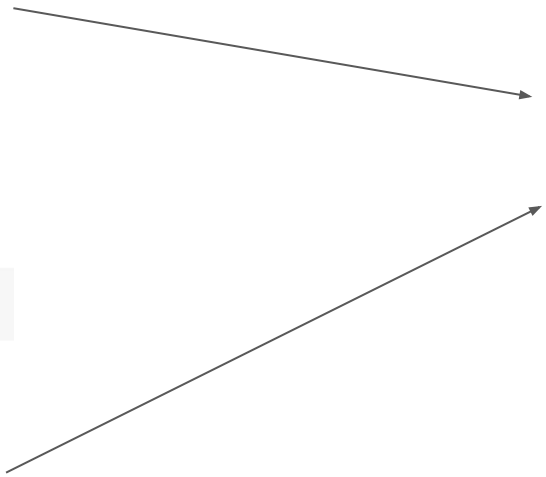

=vlookup(A2,users!$A$1:$B$4,2)
A
B
C
id
number
name
1
123-4567
Jay
1
464-8096
Jay
2
983-2928
May
2
293-9827
May
3
696-9627
Kay

```
df_users
```

	id	name
0	1	Jay
1	2	May
2	3	Kay

```
df_phone
```

	id	number
0	1	123-4567
1	1	464-8096
2	2	983-2928
3	2	293-9827
4	3	696-9627



```
df = pd.merge(df_users, df_phone, left_on='id', right_on='id')
```

```
df
```

	id	name	number
0	1	Jay	123-4567
1	1	Jay	464-8096
2	2	May	983-2928
3	2	May	293-9827
4	3	Kay	696-9627

Before you join data...

- Clean up each dataset first
- Make sure the mappings between two tables make sense
- Determine how to handle data that may be missing in one table

Take a look at the columns

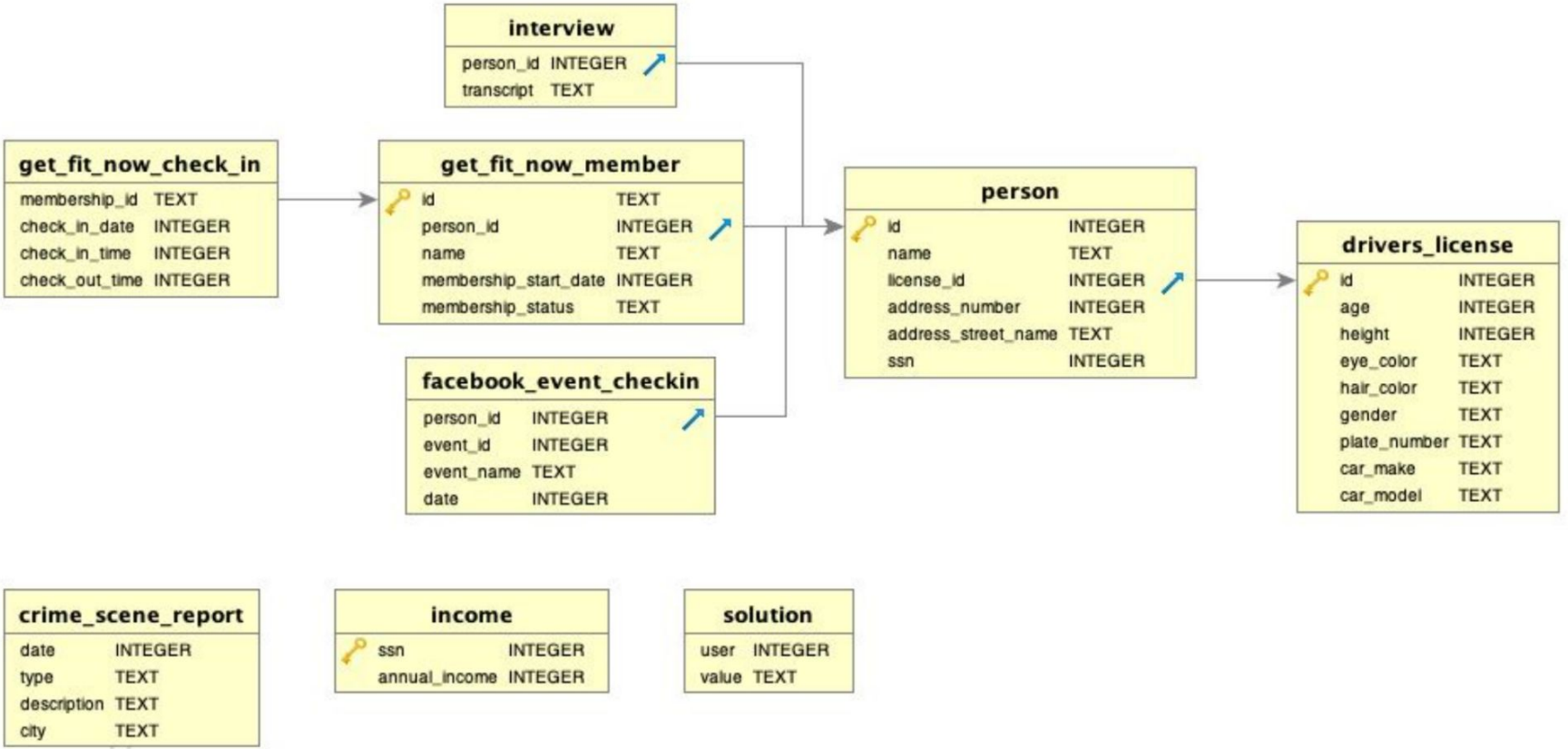
- Does the data have the right format?
- Does the data have the right range?
- Consider a transformation of the variable while preserving the original variable

Take a look at the rows

- Are there any duplicates?
- If yes, should you collapse them? If yes, how?
- Should any of the rows be dropped?
- Consider building some logic to examine each row and score each row to determine whether or not you should keep it

Prompt: Go to https://mystery.knightlab.com/ and see if you can navigate this database.

SQL


interview
person_id INTEGER
transcript TEXT
get_fit_now_check_in
membership_id TEXT
check_in_date INTEGER
check_in_time INTEGER
check_out_time INTEGER
get_fit_now_member
id TEXT
person_id INTEGER
name TEXT
membership_start_date INTEGER
membership_status TEXT
person
id INTEGER
name TEXT
license_id INTEGER
address_number INTEGER
address_street_name TEXT
ssn INTEGER
drivers_license
id INTEGER
age INTEGER
height INTEGER
eye_color TEXT
hair_color TEXT
gender TEXT
plate_number TEXT
car_make TEXT
car_model TEXT
facebook_event_checkin
person_id INTEGER
event_id INTEGER
event_name TEXT
date INTEGER
crime_scene_report
date INTEGER
type TEXT
description TEXT
city TEXT
income
ssn INTEGER
annual_income INTEGER
solution
user INTEGER
value TEXT

In [8]:
```
# now that we are down to our two last suspects, we can use their
# member id to find their person_id
SQL_Query = pd.read_sql_query("SELECT person_id, name FROM get_fit_now_member WHERE id='48Z7A' or id='48Z55'", con)
df = pd.DataFrame(SQL_Query, columns=['person_id', 'name'])

print(df)
```

```
   person_id           name
0      67318  Jeremy Bowers
1      28819   Joe Germuska
```

In [9]:
```
# need to go from the person_id to a license id
SQL_Query = pd.read_sql_query("SELECT id, license_id FROM person WHERE id=67318 or id=28819", con)
df = pd.DataFrame(SQL_Query, columns=['id', 'license_id'])

print(df)
```

```
      id  license_id
0  28819      173289
1  67318      423327
```

```
SQL_String = """
SELECT person.id, person.license_id
FROM get_fit_now_member
INNER JOIN person ON get_fit_now_member.person_id=person.id
WHERE get_fit_now_member.id='48Z7A' or get_fit_now_member.id='48Z55'
"""

SQL_Query = pd.read_sql_query(SQL_String, con)
df = pd.DataFrame(SQL_Query, columns=['id', 'license_id'])
print(df)
```

```
      id  license_id
0  67318      423327
1  28819      173289
```

8. Correlations and Clustering


COLM GLAD
17:34

Prompt

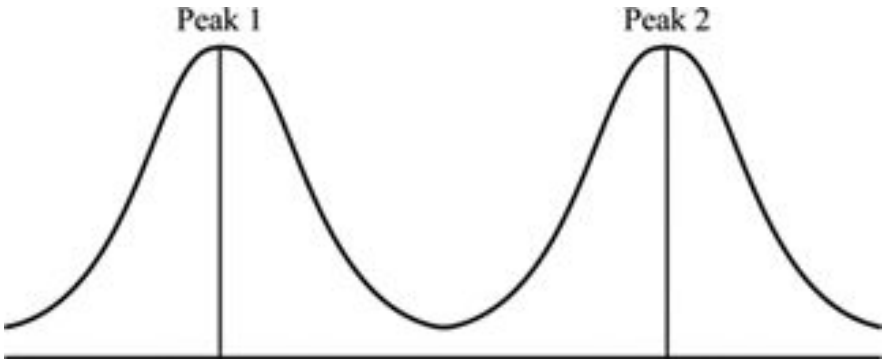
If your client hands you a bunch of data and asks you to help you make sense of it, should you find a dependent variable? Yes? No? Maybe? Both?

Is there a dependent variable?

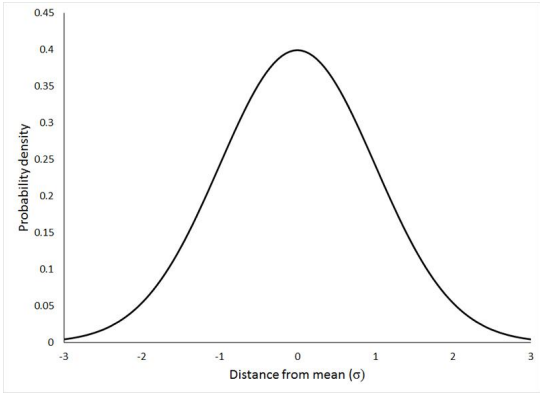
- Yes
  - Some objective function or definition of success has been defined
  - Based on the framing, a dependent variable can be inferred
  - For time series data, you can identify cause-and-effect relationships
- No
  - No one has yet defined a clear objective or definition of success
  - The data is complex and the cause-and-effect relationships are not evident from the outset
  - For time series data it is difficult to identify what comes first

Which distribution would you prefer if you were looking at customer satisfaction?

Bimodal distribution



Normal distribution



Multivariate Study: Scatterplot

```
plt.scatter(df["Age"], df["Salary"],  alpha=0.5)
plt.title("Salary Versus Age")
plt.xlabel("Age")
plt.ylabel("Salary")
plt.show()
```

Based on this scatter plot, can we assume that there is a correlation between a person's salary and their age?

Do you think outliers exist?

Multivariate Study: Categorical variables

Pivot Tables (cross-tabs)

Row Labels	C	PF	PG	SF	SG	Grand Total
Atlanta Hawks	3	4	2	2	4	15
Boston Celtics	3	3	4	1	4	15
Brooklyn Nets	2	4	3		6	15
						100
Charlotte Hornets	3	3	3	1	5	15
Chicago Bulls	2	5	2	2	4	15
Cleveland Cavaliers	3	2	3	2	5	15

Cross-tabs using Pandas

pd.crosstab(df[‘col1’], df[‘col2’]....)

```
vacation_by_season =  pd.crosstab(vacation_data["Vacation City"], vacation_data["Season"])
vacation_by_season
```

Season	Fall	Spring	Summer	Winter
Vacation City				
Capetown	19	12	25	23
London	11	26	23	40
Moscow	10	19	14	48
Paris	11	16	24	49
Sydney	8	14	13	37
Tokyo	9	7	9	33

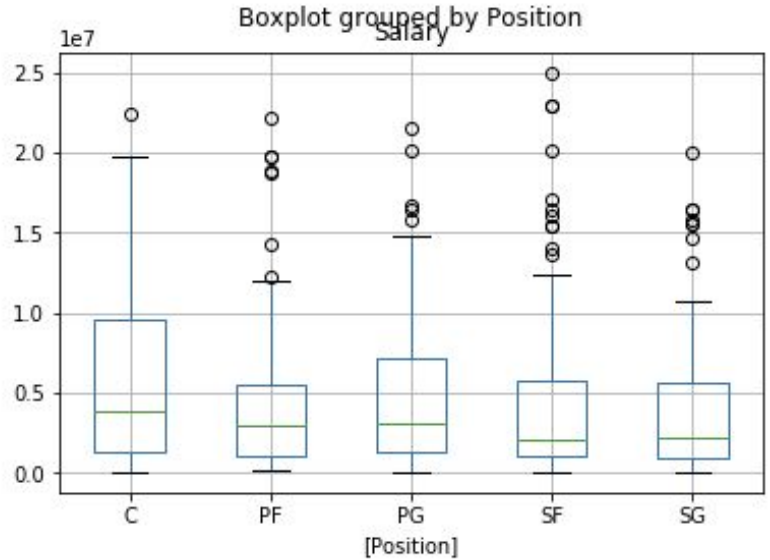
The vacation data is details about vacations people have taken to different cities in different seasons.

What might this crosstab tell you about what seasons people tend to visit some cities?

Boxplots: Compare categorical distributions

(using a categorical and a numerical variable)

```
boxplot = df.boxplot(column=['Salary'], by=['Position'])
```

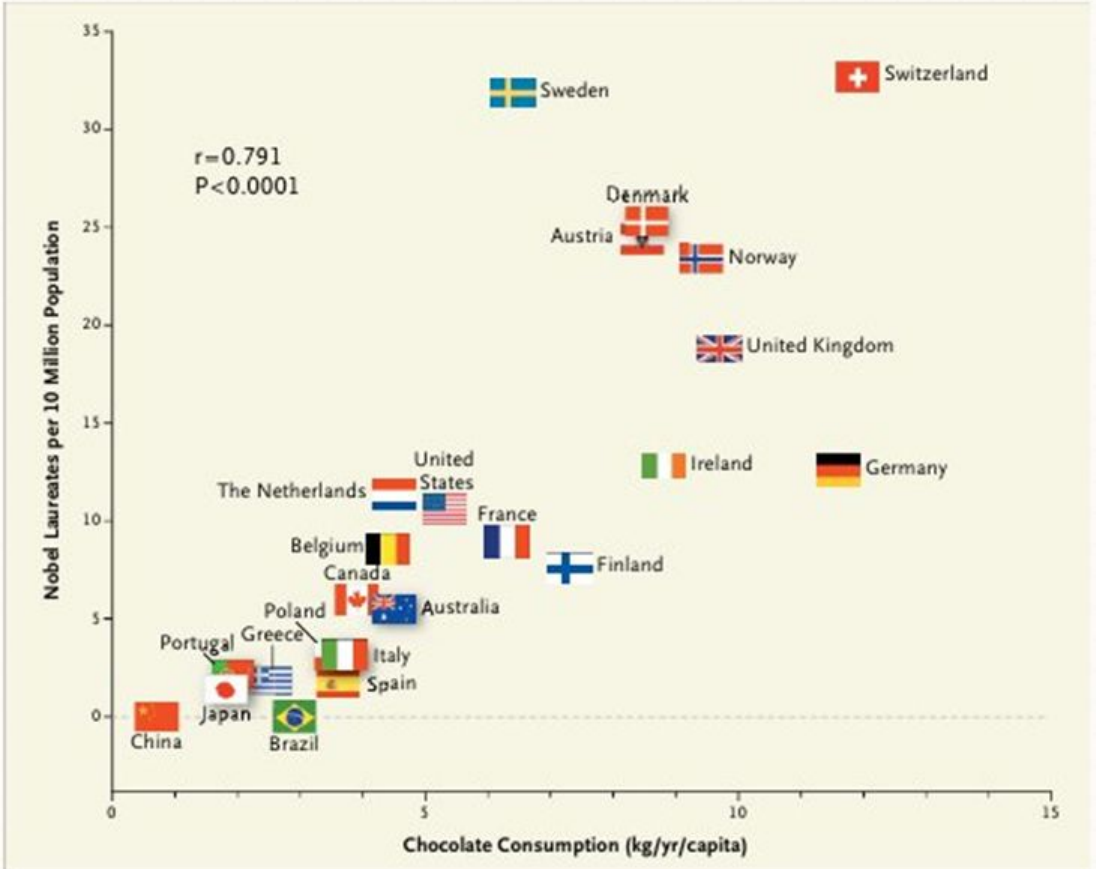


Which positions have to most similar distributions?

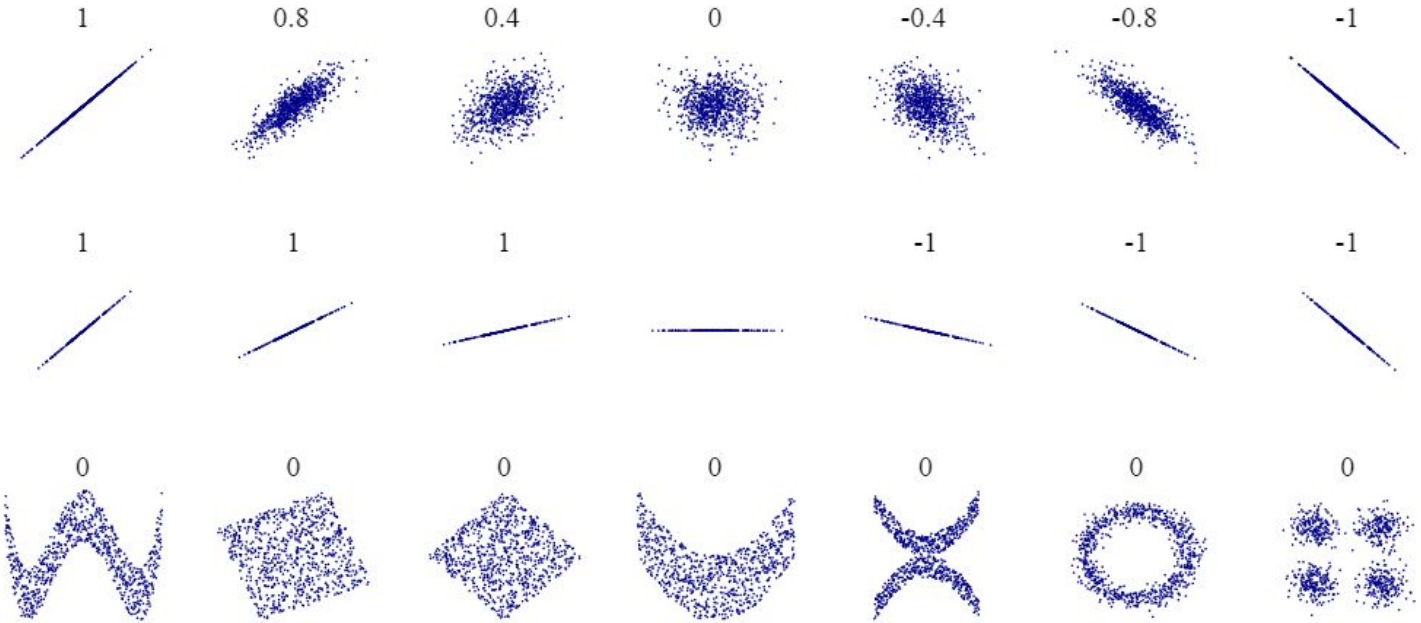
What questions might you ask based on your observations in this graph?

Correlation vs. Causation

The number of Nobel Prizes won by a country (adjusted for population) correlates well with the
per capita chocolate consumption (New England Journal of Medicine)



Correlations



Correlation Matrix

```
df[["Salary","Age"]].corr()
```

	Salary	Age
Salary	1.000000	0.213459
Age	0.213459	1.000000

This is the correlation between a player's salary and their age.

What are some conclusions we can draw based on this correlation matrix?

Can we say that a player's salary is dependent on their age?

Full correlation matrix

```
corrmat = df.corr()
print(corrmat)
```

	Number	Age	Weight	Salary
Number	1.000000	0.028724	0.206921	-0.112386
Age	0.028724	1.000000	0.087183	0.213459
Weight	0.206921	0.087183	1.000000	0.138321
Salary	-0.112386	0.213459	0.138321	1.000000

```
[5] df.corr()
```

	id	credits	gpa	score	interview	Clark	Smith
id	1.000000	0.039472	-0.228626	-0.233167	-0.298226	-0.121388	0.091343
credits	0.039472	1.000000	0.021185	0.071874	0.098077	0.063498	-0.066088
gpa	-0.228626	0.021185	1.000000	0.138119	0.181762	0.015697	-0.032025
score	-0.233167	0.071874	0.138119	1.000000	0.596856	0.142954	-0.097852
interview	-0.298226	0.098077	0.181762	0.596856	1.000000	0.111113	-0.077083
Clark	-0.121388	0.063498	0.015697	0.142954	0.111113	1.000000	-0.513159
Smith	0.091343	-0.066088	-0.032025	-0.097852	-0.077083	-0.513159	1.000000

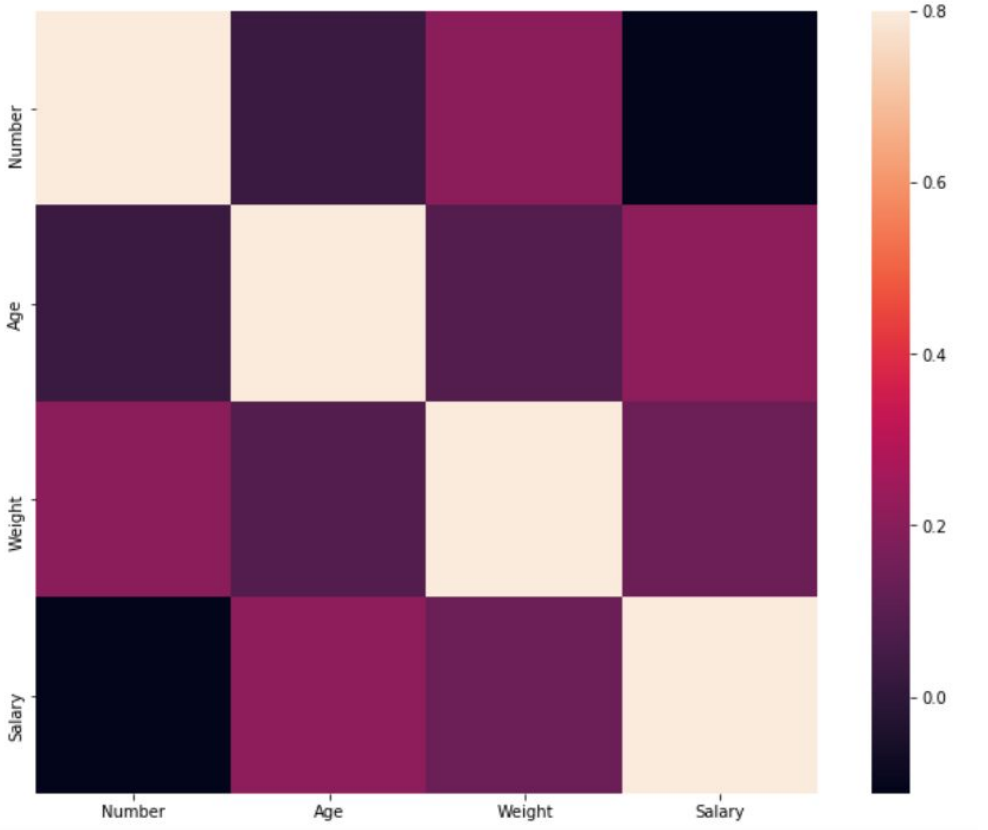
Heatmaps

Heatmaps are possible with the Matplotlib or Seaborn libraries, but can be tricky to set up … Here are some links to explore...

https://towardsdatascience.com/heatmap-basics-with-pythons-seaborn-fb92ea280a6c

https://blog.quantinsti.com/creating-heatmap-using-python-seaborn/

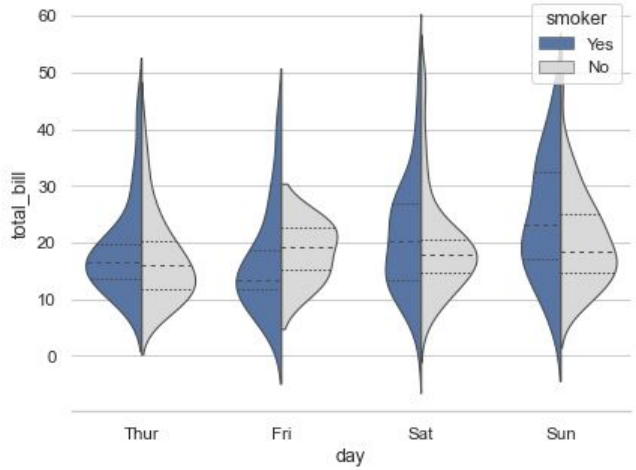
Heatmaps

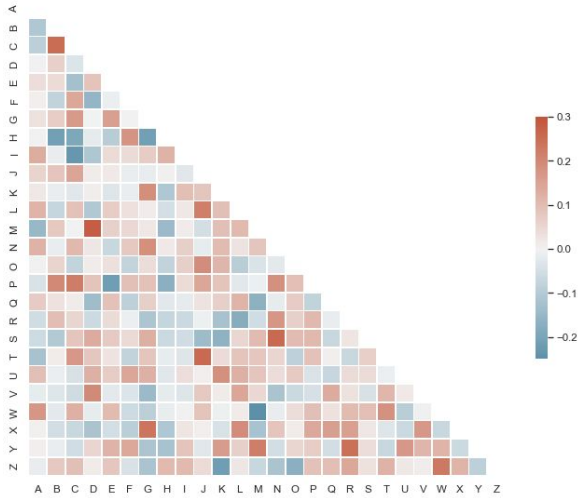


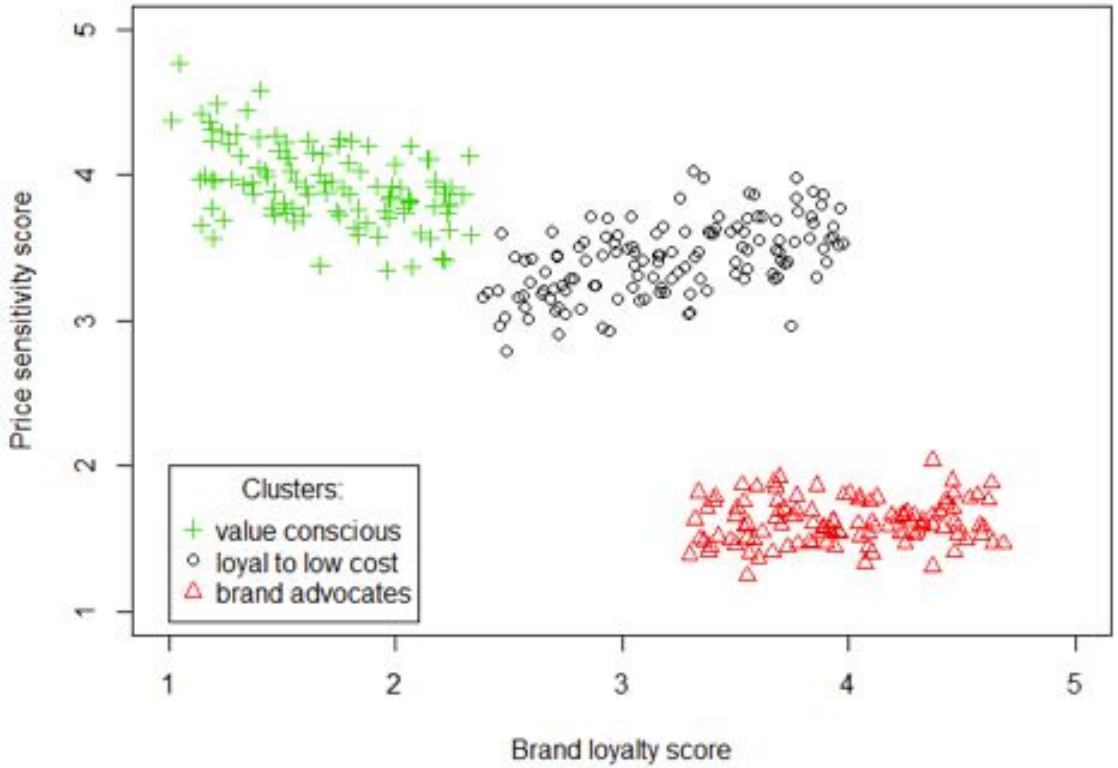

```
import seaborn as sns

corrmat = df.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
```

Other Visualizations

Python has tons of visualization libraries and tools...check out **Matplotlib** and **Seaborn** for some good ones!






Price sensitivity score
Brand loyalty score
Clusters:
value conscious
loyal to low cost
brand advocates
1
2
3
4
5

Clustering ⇔ Design Thinking

- The customer may not speak with one voice so how can we aggregate customers into representative groups (e.g. clusters)?
- Customer variation helps us drive the number of clusters and the differences between customer segments?
- A centroid is defined as the center point of the cluster and is a great way to develop a user persona.
- We can return to empathy interviews after we understand the centroids.

Focused Concept Miner (FCM): Interpretable Deep Learning for Text Exploration

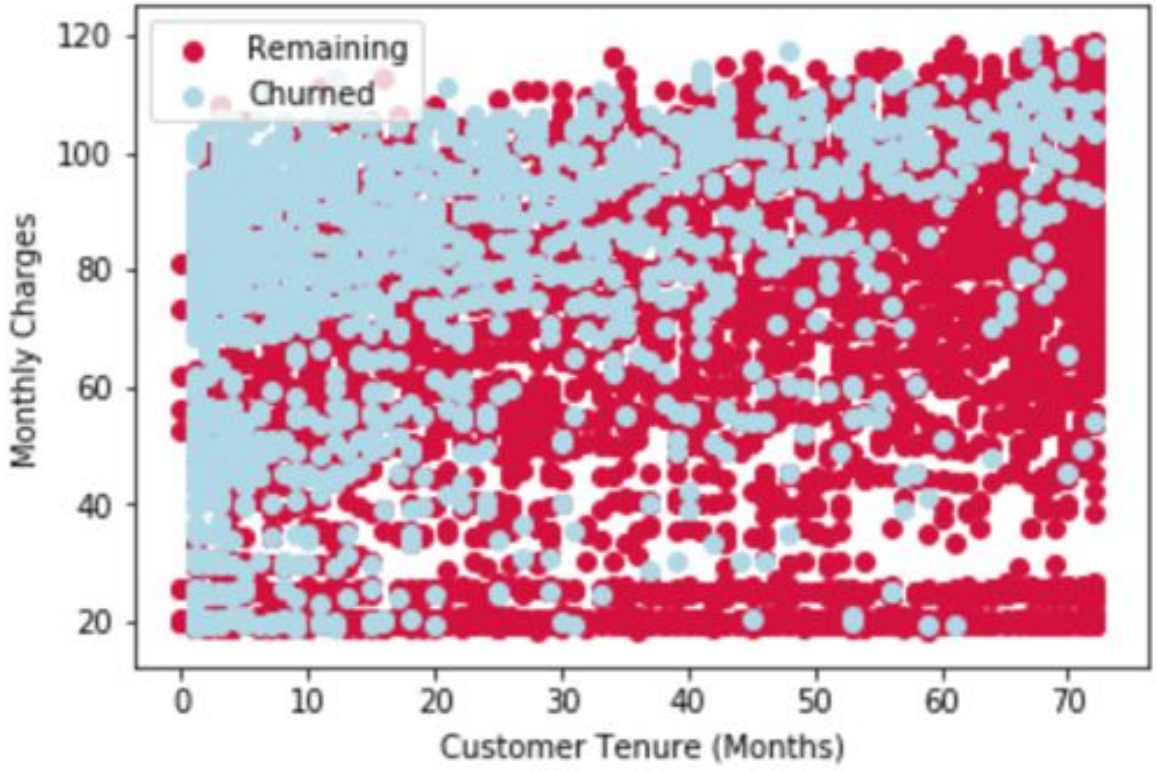
Dokyun Lee, Emaad Ahmed Manzoor, Zhaoqi Cheng
{Dokyun, Emaad, Zhaoqi}@cmu.edu
Carnegie Mellon University*

Previous Versions May 2018, Oct 2018, Oct 2019, Dec 2019, May 2020.
This Version Sept 2020.

	Concept	
1	Aesthetics	bright, clean, design, gorgeous, look, lovely, nice, pretty, simple, stylish
2	Conformance	decent, different, disappointed, excellent, expect, faulty, feature, refund, use, work
3	Feature	assemble, compact, fit, function, handy, lightweight, mount, powerful, strong, versatile
4	Brand	dyson, htc, ipad, iphone, lg, microsoft, samsung, sony, windows, xbox
5	Price (Value)	bargain, cheap, expensive, fit, inexpensive, money, price, purchase, reasonable, worst
6	Serviceability	break, durable, flimsy, poor, problem, quality, size, solid, strong, sturdy

Table 2: **Operationalized concept words for Garvin's concepts.**

- Concept Cluster A:
  - small nice space little use design clean size love look
  - look assemble fit sturdy nice quality room lovely small item
  - use clean work small time love excellent little look job
- Concept Cluster B:
  - money quality cheap poor instruction price fine ok overall work
  - happy excellent definitely pleased far purchase need recommend worth time
  - use clean time price excellent work simple small money far
  - clean use feel like love price new old quality recommend

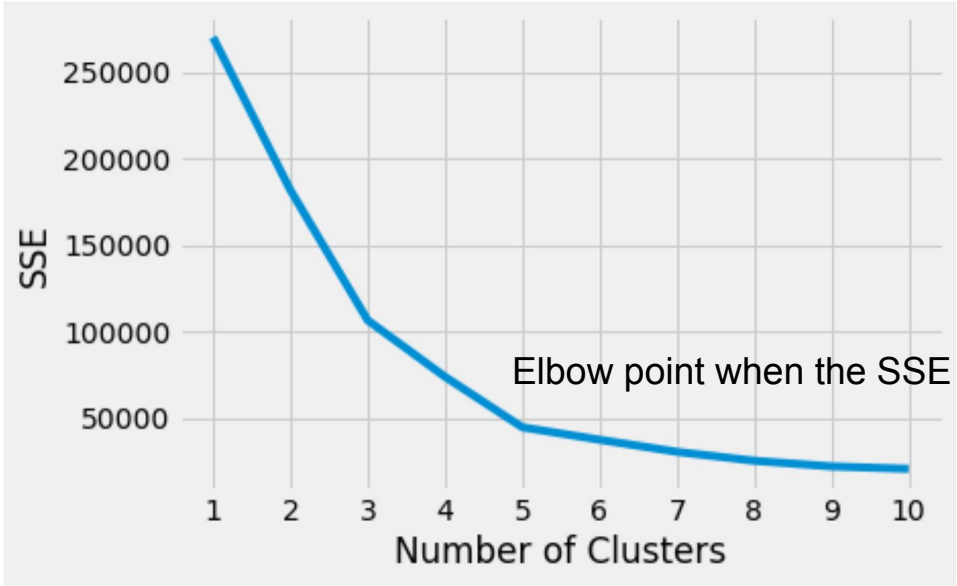

Remaining
Churned
Monthly Charges
120
100
80
60
40
20
0
10
20
30
40
50
60
70
Customer Tenure (Months)

```
sse = []
for k in range(1, 11):
    model = KMeans(n_clusters=k)
    model.fit(X)
    sse.append(model.inertia_)
```

Mall customer data

(SSE = sum of squares error)

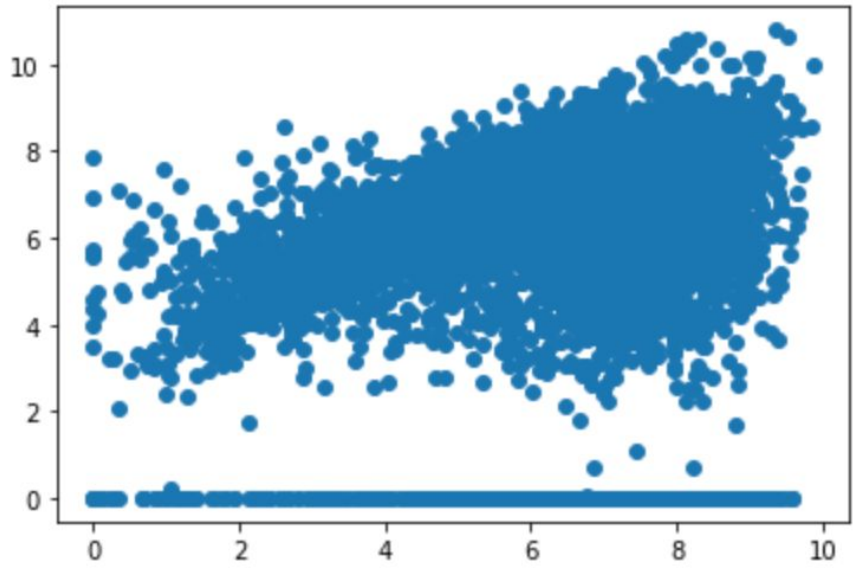
```
plt.style.use("fivethirtyeight")
plt.plot(range(1, 11), sse)
plt.xticks(range(1, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()
```



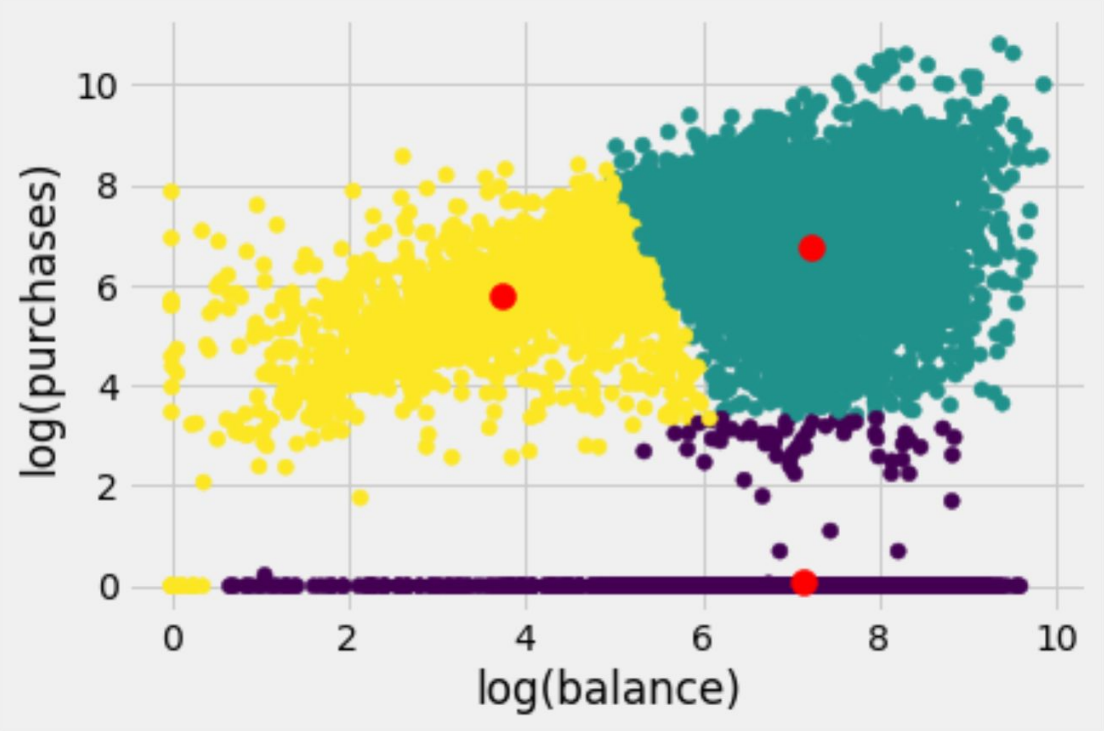
Elbow point when the SSE starts to level off

8636 rows × 18 columns

In [6]:
```
# let's explore the variables to see what might be interesting
# are there any variables that need to be transformed because
# they are not normally distributed?  (e.g. balance, purchases)
df['ln_balance'] = np.log(df['BALANCE'] + 1)
df['ln_purchases'] = np.log(df['PURCHASES'] + 1)
plt.scatter(df['ln_balance'], df['ln_purchases'])
plt.show()
```



```
# now we can plot the data along with the centroids
plt.scatter(dfc['ln_balance'],dfc['ln_purchases'], c=model.labels_)
plt.scatter(clusters[:,0], clusters[:,1], s=100, color="red") # Show the centroids
plt.xlabel('log(balance)')
plt.ylabel('log(purchases)')
plt.show()
```



Clustering limitations

- The definition of the clusters is determined by patterns in the data, not a well-framed problem
- Biases that exist in the data may lead to problematic clusters (e.g. codifying biases in the data)
- Sometimes the elbow in the graph is hard to determine

9. Linear Regression

Warm Up

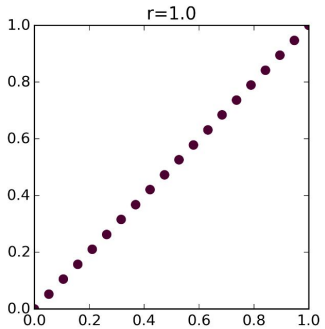
If a college is trying to predict the GPA of a high school applicant, how can they factor in both SAT score and GPA along with other criteria? How can they separate the effect of each of these independent variables?

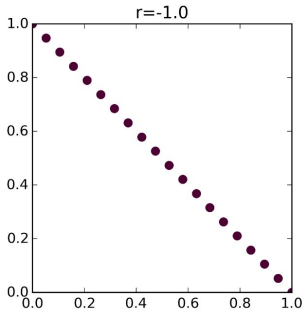
Is there a dependent variable?

- Yes
  - Some objective function or definition of success has been defined
  - Based on the framing, a dependent variable can be inferred
  - For time series data, you can identify cause-and-effect relationships
- No
  - No one has yet defined a clear objective or definition of success
  - The data is complex and the cause-and-effect relationships are not evident from the outset
  - For time series data it is difficult to identify what comes first

Beyond Bivariate to Multivariate

- Bivariate analysis is independent of all other sources of variation
  - Bivariate correlation
    - 1: perfectly positive linear
    - 0: no relationship
    - -1: perfectly negative linear



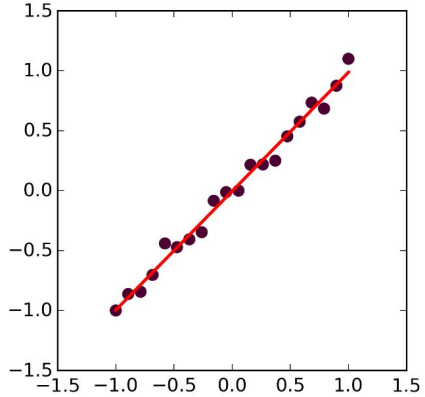


- Multivariate includes all independent variables
  - Each coefficient isolates out the effect of that independent variable
  - Partial derivative of the function

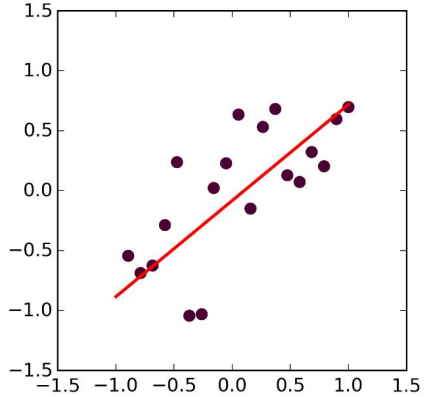
(Bi-variate) Linear Regression Assumptions

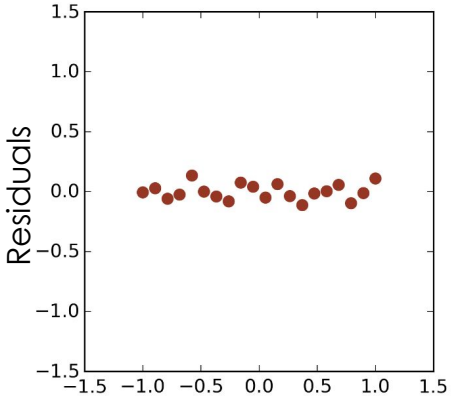
- The relationship between the variables is linear
- Errors (or residuals) are independent, normally distributed with mean zero and constant variance
- Variable being predicted is continuous

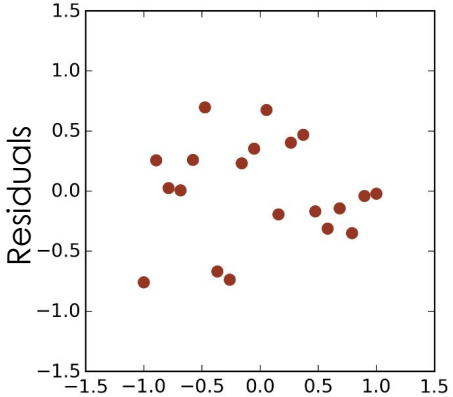
Linear, Strong



Linear, Weak

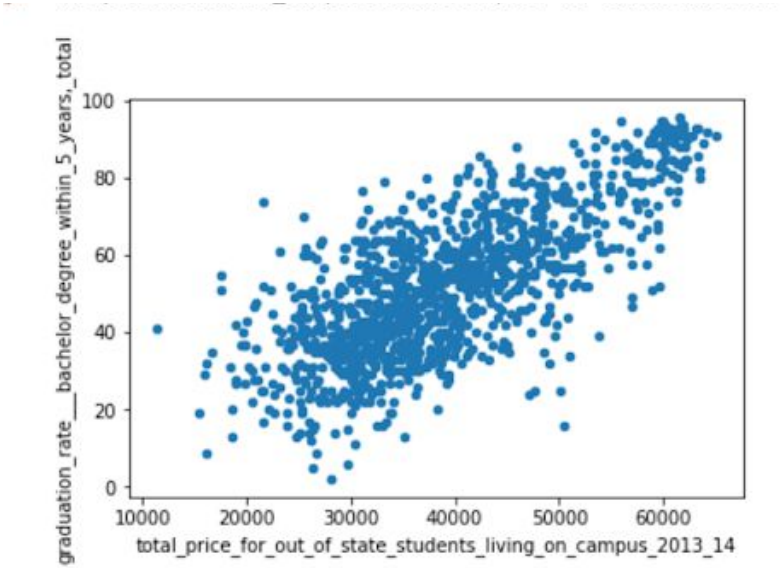






Linearity

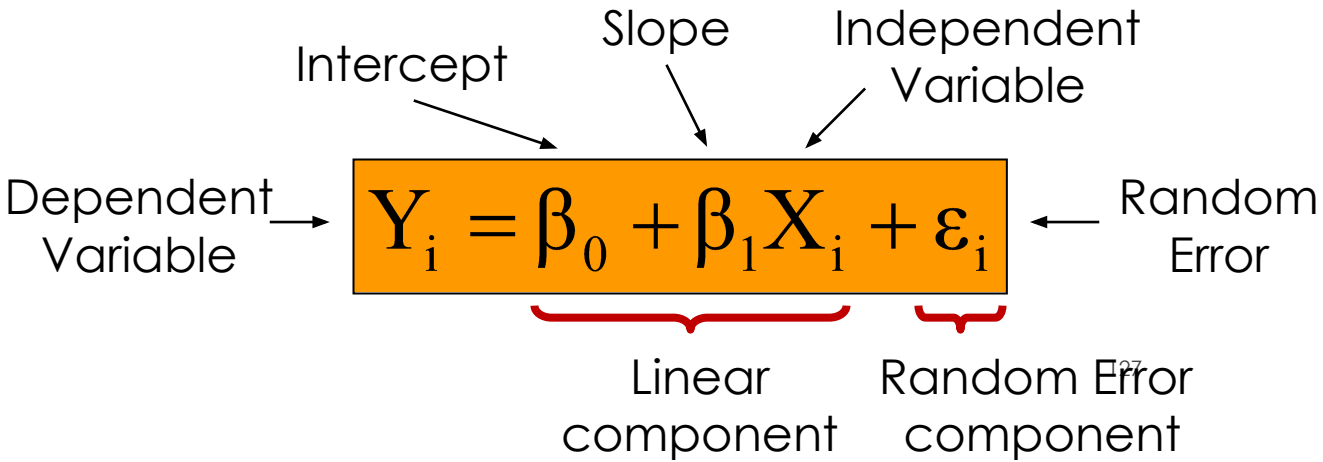
Plot each independent variable against the dependent variable



Regression Analysis

- Linear regression analysis means “fitting a straight line to data”
- Widely used technique to help model and understand real-world phenomena
- Allows prediction of one dependent variable from one or more independent variables

Linear Regression Model



```
[6]  YVar = df[["score"]] # this is our dependent variable
     XVar = df[["gpa"]] # these are our independent variables
     XVar = sm.add_constant(XVar)
     LinearModel = sm.OLS(YVar, XVar, missing='drop').fit()
     print(LinearModel.summary())
```

OLS Regression Results

Dep. Variable:	score	R-squared:	0.019
Model:	OLS	Adj. R-squared:	0.015
Method:	Least Squares	F-statistic:	4.298
Date:	Tue, 26 Oct 2021	Prob (F-statistic):	0.0393
Time:	15:09:55	Log-Likelihood:	-280.92
No. Observations:	223	AIC:	565.8
Df Residuals:	221	BIC:	572.7
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.9214	0.687	2.795	0.006	0.567	3.276
gpa	0.3885	0.187	2.073	0.039	0.019	0.758

Omnibus:	6.093	Durbin-Watson:	2.122
Prob(Omnibus):	0.048	Jarque-Bera (JB):	5.806
Skew:	-0.344	Prob(JB):	0.0548
Kurtosis:	3.389	Cond. No.	47.2

```
YVar = df[["score"]] # this is our dependent variable
XVar = df[["credits"]] # these are our independent variables
XVar = sm.add_constant(XVar)
LinearModel = sm.OLS(YVar, XVar, missing='drop').fit()
print(LinearModel.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  score   R-squared:                       0.005
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     1.148
Date:                Tue, 26 Oct 2021   Prob (F-statistic):              0.285
Time:                        15:09:55   Log-Likelihood:                -282.49
No. Observations:                 223   AIC:                             569.0
Df Residuals:                     221   BIC:                             575.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.2089      0.137     23.489      0.000       2.940       3.478
credits        0.0071      0.007      1.071      0.285      -0.006       0.020
==============================================================================
Omnibus:                        3.348   Durbin-Watson:                   2.061
Prob(Omnibus):                  0.187   Jarque-Bera (JB):                3.003
Skew:                          -0.271   Prob(JB):                        0.223
Kurtosis:                       3.169   Cond. No.                         48.6
==============================================================================
```

Multivariate Regression

- Instead of having one x term, we can have multiple independent variables predict one dependent variable

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + E$$

- The outcome variable is assumed to be a linear combination of the predictor variables (the inputs)

- R^2 measures the **proportion of variance** in the observed data that is **explained by the model**

Multiple Linear Regression Assumptions

- The relationship between the variables is linear
- Errors (or residuals) are independent, normally distributed with mean zero and constant variance
- Variable being predicted is continuous
- **Predictor variables are independent**

What if Assumptions are Violated?

- The relationship between the variables is linear. → *Use different regression model (e.g. exponential model)*
- Errors (or residuals) are independent, normally distributed with mean zero and constant variance. → *Non-linear data transformation (e.g. ln)*
- Variable being predicted is continuous. → *Use logistic regression*
- Predictor variables are independent → *Remove variables from analysis*

```
YVar = df[["score"]] # this is our dependent variable
XVar = df[["gpa", "credits", "id", "Clark", "Smith"]] # these are our independent
XVar = sm.add_constant(XVar)
LinearModel = sm.OLS(YVar, XVar, missing='drop').fit()
print(LinearModel.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  score   R-squared:                       0.081
Model:                            OLS   Adj. R-squared:                  0.060
Method:                 Least Squares   F-statistic:                     3.815
Date:                Tue, 26 Oct 2021   Prob (F-statistic):            0.00248
Time:                        15:09:55   Log-Likelihood:                -273.68
No. Observations:                 223   AIC:                             559.4
Df Residuals:                     217   BIC:                             579.8
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.6912      1.185      3.958      0.000       2.355       7.027
gpa            0.2485      0.188      1.320      0.188      -0.122       0.619
credits        0.0070      0.006      1.073      0.284      -0.006       0.020
id            -0.0012      0.000     -2.986      0.003      -0.002      -0.000
Clark          0.1996      0.148      1.349      0.179      -0.092       0.491
Smith         -0.0336      0.133     -0.253      0.800      -0.295       0.228
==============================================================================
Omnibus:                       14.654   Durbin-Watson:                   2.175
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               18.286
Skew:                          -0.490   Prob(JB):                     0.000107
Kurtosis:                       4.003   Cond. No.                     4.22e+04
==============================================================================
```

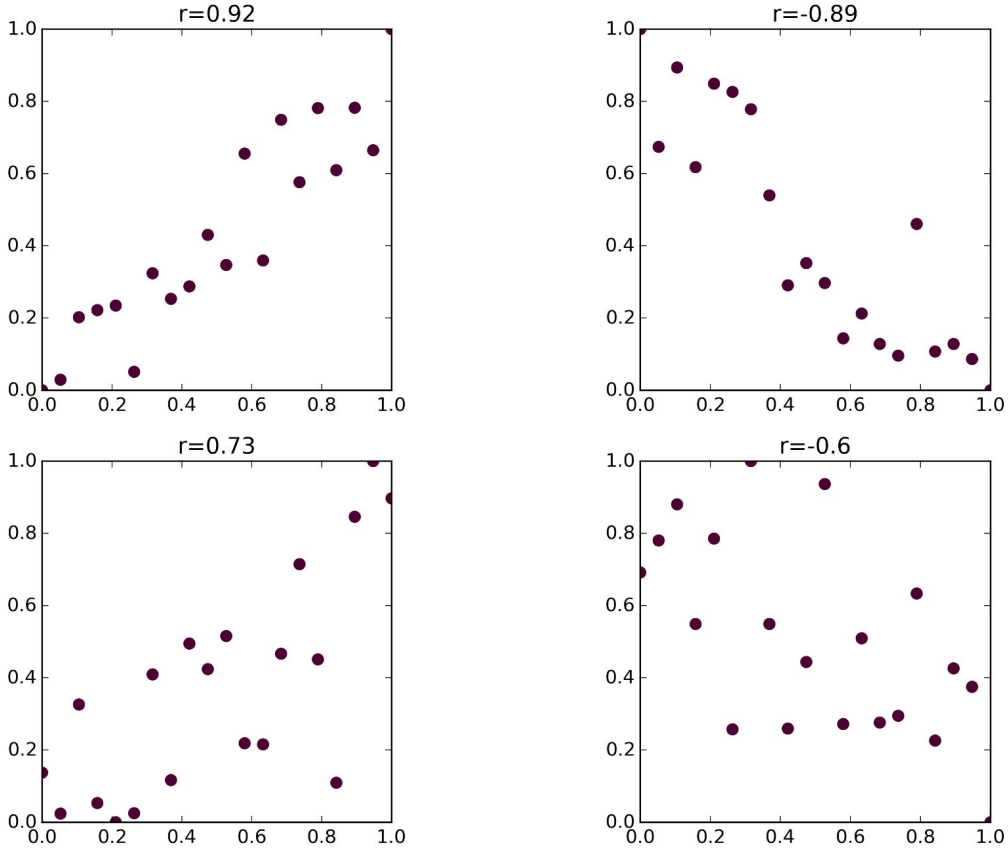
Check for Multicollinearity

Create a correlation matrix of the independent variables in the model

(subset of correlation matrix)

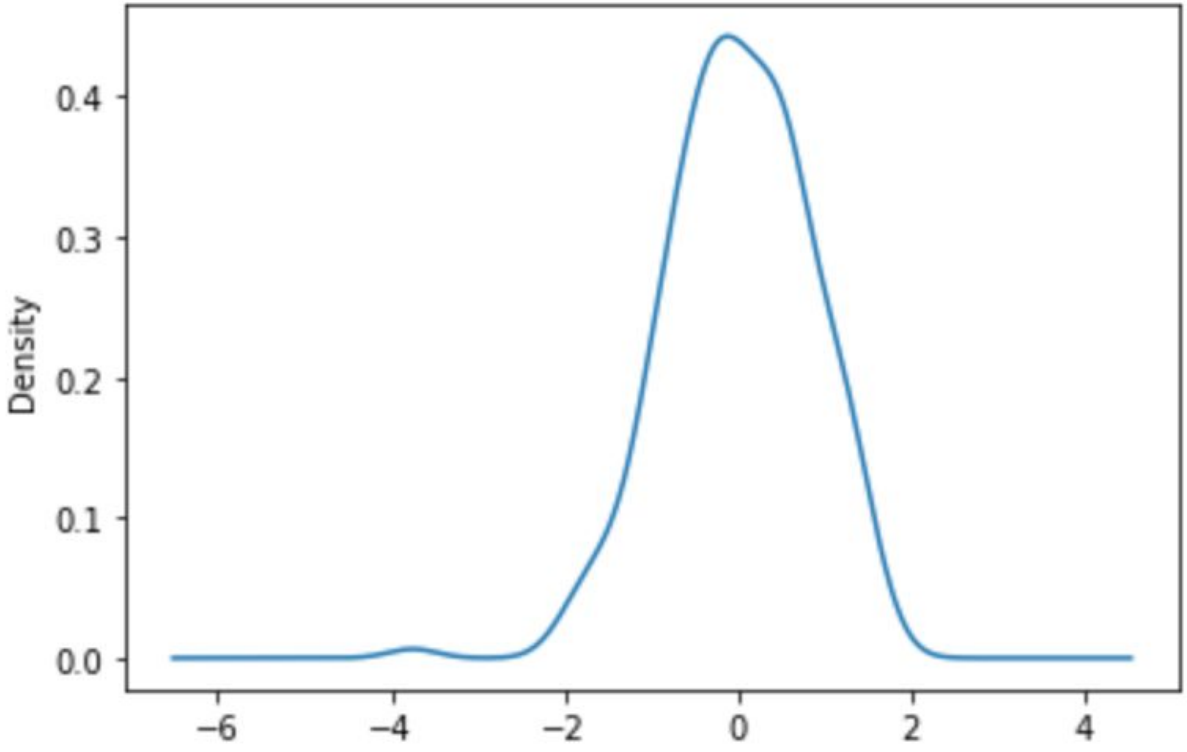
	applicants_total	admissions_total	enrolled_total
applicants_total	1.000000	0.850997	0.782896
admissions_total	0.850997	1.000000	0.881969
enrolled_total	0.782896	0.881969	1.000000
act_composite_75th_percentile_score	0.397111	0.242204	0.249229
estimated_undergraduate_enrollment,_total	0.743812	0.829953	0.957130
total_price_for_in_state_students_living_on_campus_2013_14	0.048229	-0.121052	-0.283698
total_price_for_out_of_state_students_living_on_campus_2013_14	0.309317	0.160058	0.022267

For variables with high correlation, try removing the less significant variable in the model


r=0.92
r=-0.89
r=0.73
r=-0.6
0.0
0.2
0.4
0.6
0.8
1.0

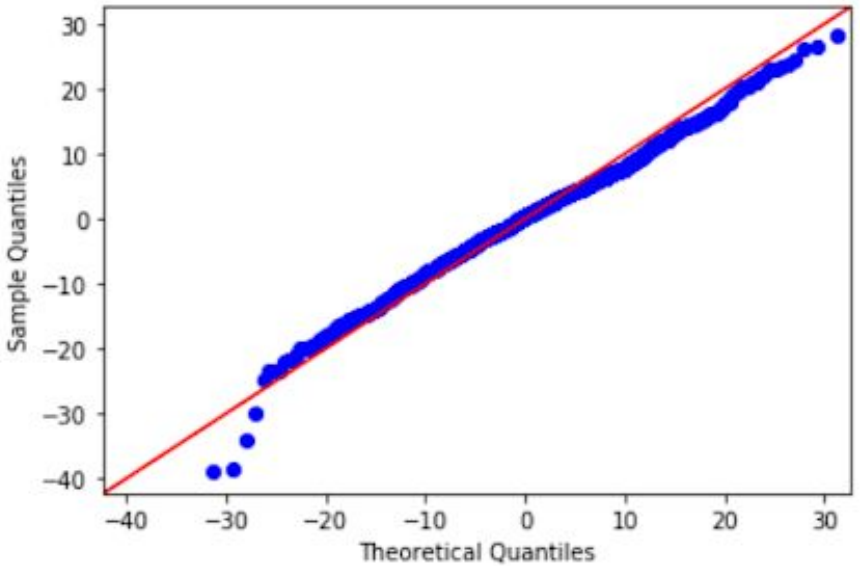
```
[10] LinearModel.resid.plot.kde()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f41f83
```



Q-Q Plot

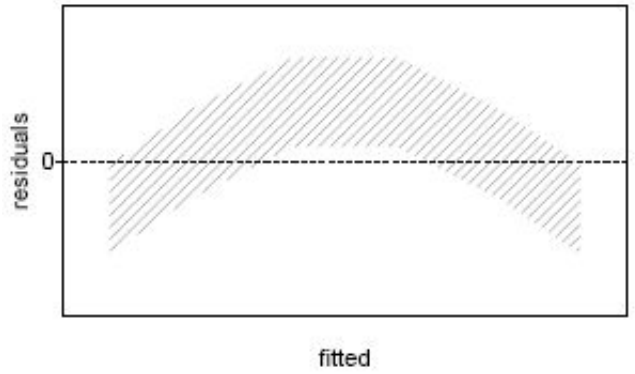
```
residuals=LinearModel.resid
sm.qqplot(residuals, scale=10, line ='45')
plt.show()
```



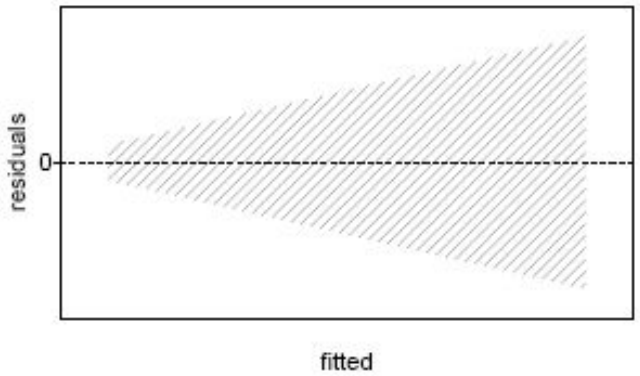
If the Q-Q Plot is not on the diagonal, the data is likely skewed in some way

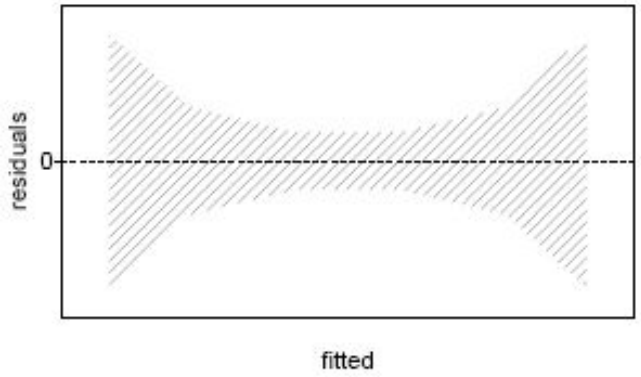
Bad Residuals

Bad
Mean



Bad
Variance





How Well Does our Model Fit?

- **Coefficient of Determination,** R^2 *:* Indicates how well data fit a statistical model or the proportion of total variation in outcome explained by the model
  - R^2 = 1: Regression line fits perfectly, independent variables explain all variance in outcome variable
  - R^2 = 0: Regression line does not fit at all; independent variables explain no variance in outcome variable

A Note about R^2

- WARNING: The R^2 statistic can be artificially inflated by adding *any* independent variable to the model.

  R^2 assumes that every independent variable in the model helps to explain variation in the Dependent Variable, so it tells you the percentage of explained variation as if All IVs in the model affect the DV.

  The adjusted R^2:
  - tells you the percentage of variation explained by only those IVs that truly affect the DV AND
  - penalizes you for adding independent variable(s) that do not belong in the model

- We can compare adjusted-R^2 values as a heuristic to tell if adding an additional independent variable really helps.

10. Status Presentations

Things to consider when presenting your status presentation:

- What have you learned so far?
- What are you still trying to build?
- What do you think your client should do differently?
- How will you test these draft recommendations?

Outline

1. Introduction Slide (client/opportunity/team)
2. Frame the Problem (cross-sectional vs. time-series/independent vs. dependent/
3. Questions (hypotheses)
4. Data processing (quality/joins/transformations)
5. Preliminary insights (graphs/charts/trends)
6. Model (design/results)
7. Next steps (improve model)
8. Draft recommendations
9. Backup slides (assumptions/correlations/additional graphs)

11. Classification

Core Courses

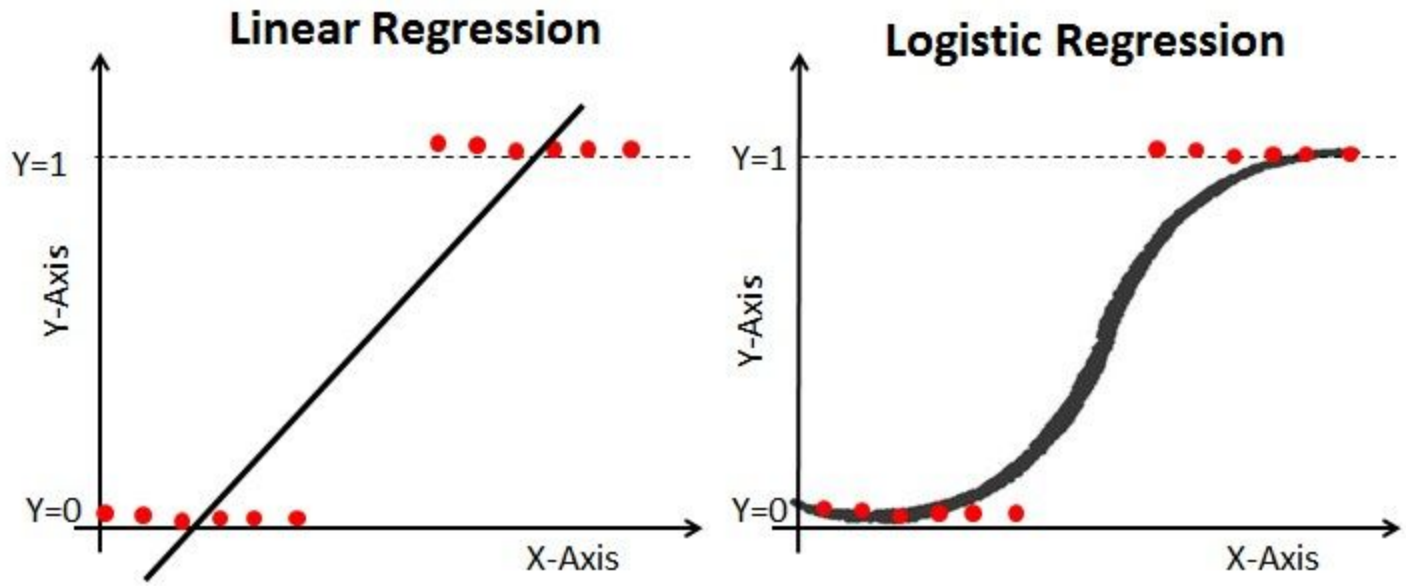
- BMGT/ENES190H: Tu Th 12:30-1:45pm, Tu 6-7:50pm (Testudo is correct)
- BMGT438A/ENES489A: Tu 5-7:40pm; blended format (To be updated on Testudo)
- BMGT/ENES390H: Mon 5-7:40pm (To be updated on Testudo)
- BMGT/ENES490H: Weds 5-7:40pm (Testudo is correct)

Predicting Event Probability

- You may want to predict a discrete value (will purchase, will be accepted)
- These all have a probability associated with them, but ultimately it will or won’t happen
- What factors influence that?

Warm Up

You are helping the University of Maryland build a tool to help predict whether or not a rising senior will live on campus in the fall? How could you build such a model? How can you assess how accurate it might be?

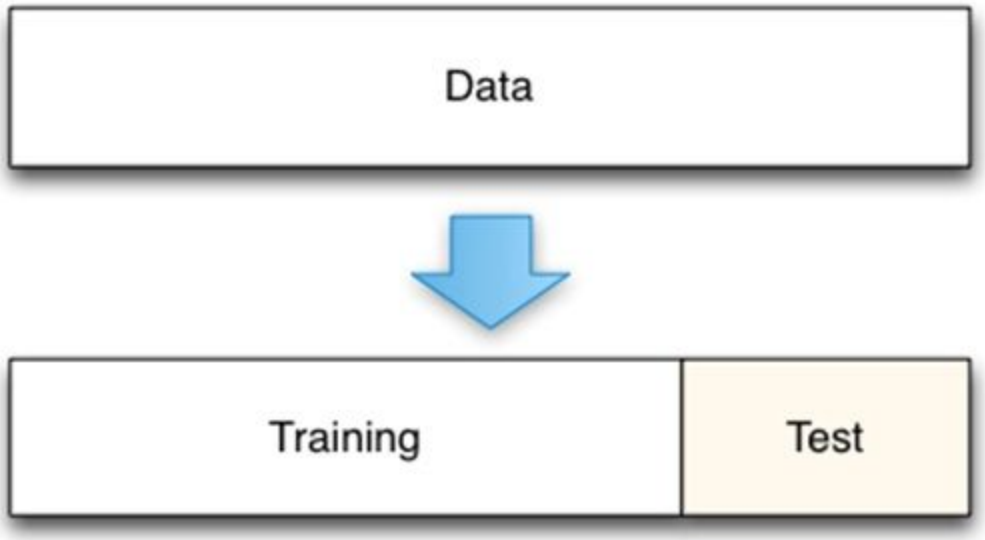

Linear Regression
Y=1
Y-Axis
Y=0
X-Axis
Logistic Regression
Y=1
Y-Axis
Y=0
X-Axis

```
YVar = df[["interview"]] # this is our dependent variable
XVar = df[["score", "credits", "gpa", "id", "Clark", "Smith"]] # these are our inc
XVar.corr()
LogisticModel = sm.Logit(YVar, XVar).fit() # this is the logistic regression
print(LogisticModel.summary()) # this prints out the results of the model
```

```
Optimization terminated successfully.
         Current function value: 0.404264
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:              interview   No. Observations:                  223
Model:                          Logit   Df Residuals:                      217
Method:                           MLE   Df Model:                            5
Date:                Tue, 26 Oct 2021   Pseudo R-squ.:                  0.3781
Time:                        15:09:55   Log-Likelihood:                -90.151
converged:                       True   LL-Null:                       -144.96
Covariance Type:            nonrobust   LLR p-value:                 4.935e-22
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
score          2.2582      0.340      6.635      0.000       1.591       2.925
credits        0.0235      0.023      1.011      0.312      -0.022       0.069
gpa            0.4291      0.438      0.979      0.327      -0.430       1.288
id            -0.0043      0.001     -4.987      0.000      -0.006      -0.003
Clark         -0.0212      0.503     -0.042      0.966      -1.007       0.964
Smith         -0.0186      0.433     -0.043      0.966      -0.868       0.831
==============================================================================
```

Interpret Logistic Regression

- Regression coefficients now represent a unit increase in the logit (log-odds)
  - Hard to reason about directly
  - Care more about significance and size relative to other predictors


Data
Training
Test
80%
20%

Training and Testing

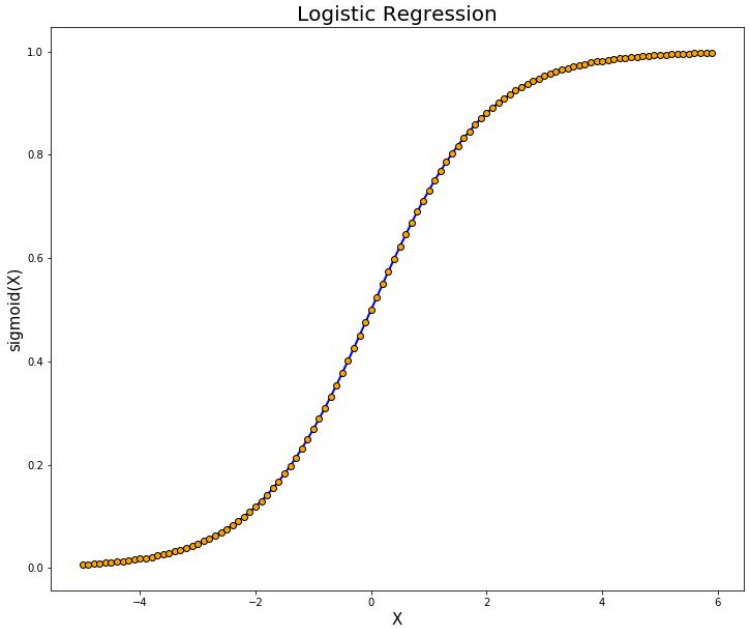
- Randomly select around 80% of the observations for training a model
- Build a model and then apply it to the 20% testing data (hold-out sample)
- See how well your model works on your hold-out sample and look at the errors

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Logistic Regression

- Separating groups into two categories is something we learned using logistic regression



```
# !pip install statsmodels --upgrade # Installing statsmodels if you don't have it already
import pandas as pd # we've seen pandas before
import statsmodels.api as sm # this is the modeling library that we will use
from sklearn.metrics import (confusion_matrix, accuracy_score)
from sklearn.model_selection import train_test_split
```

```
# be sure to have the csv file questselections18.csv in the same directory as this ipynb
df = pd.read_csv("questselections18.csv")
```

```
dummies = pd.get_dummies(df['school'], drop_first=True)
df = pd.concat([df, dummies], axis=1)
df
```

	id	credits	gpa	school	score	interview	Clark	Smith
0	1757	15	3.55	CMNS	3.5	1	0	0
1	1760	15	4.00	CMNS	4.0	1	0	0
2	1761	15	3.45	Clark	4.5	1	1	0
3	1762	18	3.40	Clark	4.5	1	1	0
4	1763	19	3.38	Smith	2.5	0	0	1
...	...	...	...	...	...	...	...	...
221	2266	15	3.20	Smith	3.0	0	0	1
222	2268	16	3.65	CMNS	3.5	0	0	0
223	2271	14	3.70	CMNS	2.0	0	0	0
224	2273	16	3.25	Smith	4.0	1	0	1
225	2277	16	4.00	Smith	2.5	1	0	1

226 rows × 8 columns

```
fn = ["score", "credits", "gpa", "Clark", "Smith"]
cn = ["interview"]
```

```
# let's take a look at the correlation matrix of independent variables
y_var = df[cn] # this is our dependent variable
x_var = df[fn] # these are our independent variables
x_var.corr()
```

	score	credits	gpa	Clark	Smith
score	1.000000	0.071874	0.138119	0.142954	-0.097852
credits	0.071874	1.000000	0.021185	0.063498	-0.066088
gpa	0.138119	0.021185	1.000000	0.015697	-0.032025
Clark	0.142954	0.063498	0.015697	1.000000	-0.513159
Smith	-0.097852	-0.066088	-0.032025	-0.513159	1.000000

```python
# let's now do a logistic regression where we try to predict whether or not the applicant
# gets an interview
LogisticModel = sm.Logit(y_var, x_var).fit() # this is the logistic regression
print(LogisticModel.summary()) # this prints out the results of the model
```

```
Optimization terminated successfully.
         Current function value: 0.471757
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:              interview   No. Observations:                  223
Model:                          Logit   Df Residuals:                      218
Method:                           MLE   Df Model:                            4
Date:                Mon, 22 Mar 2021   Pseudo R-squ.:                  0.2743
Time:                        18:09:53   Log-Likelihood:                -105.20
converged:                       True   LL-Null:                       -144.96
Covariance Type:            nonrobust   LLR p-value:                 2.206e-16
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
score          1.8954      0.286      6.633      0.000       1.335       2.455
credits        0.0017      0.020      0.086      0.932      -0.037       0.040
gpa           -1.4600      0.263     -5.560      0.000      -1.975      -0.945
Clark         -0.0189      0.467     -0.040      0.968      -0.935       0.897
Smith         -0.2556      0.395     -0.647      0.518      -1.030       0.519
==============================================================================
```

```
train, test = train_test_split(df, test_size = 0.2, stratify = df['interview'], random_state = 42)
```

```
x_train = train[fn]
y_train = train[cn]
x_test = test[fn]
y_test = test[cn]
```

```
LogisticModel = sm.Logit(y_train, x_train).fit() # this is the logistic regression
print(LogisticModel.summary()) # this prints out the results of the model
```

```
Optimization terminated successfully.
         Current function value: 0.475149
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:              interview   No. Observations:                  178
Model:                          Logit   Df Residuals:                      173
Method:                           MLE   Df Model:                            4
Date:                Tue, 23 Mar 2021   Pseudo R-squ.:                  0.2688
Time:                        12:36:52   Log-Likelihood:                -84.577
converged:                       True   LL-Null:                       -115.67
Covariance Type:            nonrobust   LLR p-value:                 1.004e-12
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
score          1.8034      0.302      5.981      0.000       1.212       2.394
credits       -0.0001      0.021     -0.007      0.995      -0.041       0.041
gpa           -1.3757      0.279     -4.926      0.000      -1.923      -0.828
Clark         -0.0374      0.522     -0.072      0.943      -1.060       0.986
Smith         -0.1753      0.434     -0.404      0.686      -1.026       0.676
==============================================================================
```

```
y_hat = LogisticModel.predict(x_test)
prediction = list(map(round, y_hat))

# confusion matrix
cm = confusion_matrix(y_test, prediction)
print ("Confusion Matrix : \n", cm)

# accuracy score of the model
print('Test accuracy = ', accuracy_score(y_test, prediction))
```

```
Confusion Matrix :
 [[ 6 10]
 [ 2 27]]
Test accuracy =  0.7333333333333333
```

```
compare = pd.concat([y_test, y_hat], axis=1)
compare = compare.rename(columns={0: 'y_hat'})
```

Can you predict who would survive the Titanic sinking?

- Think about the sources of variation
- Consider the context of the event

PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	male	Q	S	outcome
1	0	3	22.0	1	0	7.2500	1	0	1	died
2	1	1	38.0	1	0	71.2833	0	0	0	survived
3	1	3	26.0	0	0	7.9250	0	0	1	survived
4	1	1	35.0	1	0	53.1000	0	0	1	survived
5	0	3	35.0	0	0	8.0500	1	0	1	died
...	...	...	...	...	...	...	...	...	...	...
886	0	3	39.0	0	5	29.1250	0	1	0	died
887	0	2	27.0	0	0	13.0000	1	0	1	died
888	1	1	19.0	0	0	30.0000	0	0	1	survived
890	1	1	26.0	0	0	30.0000	1	0	0	survived
891	0	3	32.0	0	0	7.7500	1	1	0	died

714 rows × 10 columns

```python
# logistic regression - we have seen this before

LogisticModel = sm.Logit(y_train, x_train).fit()
print(LogisticModel.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.549689
         Iterations 5
                           Logit Regression Results
==============================================================================
Dep. Variable:                Survived   No. Observations:                  712
Model:                           Logit   Df Residuals:                      710
Method:                            MLE   Df Model:                            1
Date:                 Tue, 23 Mar 2021   Pseudo R-squ.:                  0.1743
Time:                         12:23:56   Log-Likelihood:                -391.38
converged:                        True   LL-Null:                       -473.99
Covariance Type:            nonrobust   LLR p-value:                 8.173e-38
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Pclass         0.1448      0.052      2.773      0.006       0.042       0.247
male          -1.8331      0.176    -10.426      0.000      -2.178      -1.488
==============================================================================
```

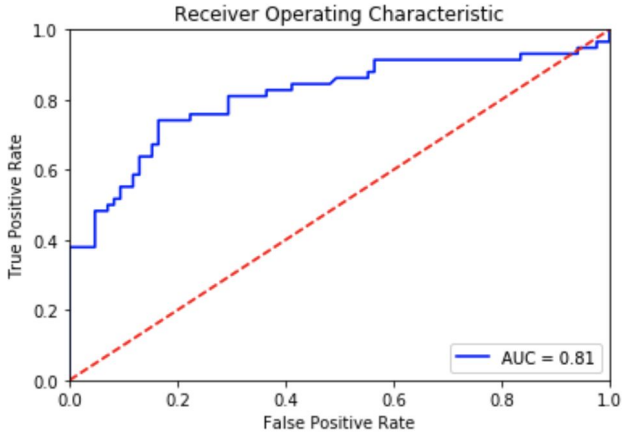
```
Confusion Matrix :
 [[94 16]
 [24 45]]
Test accuracy =  0.776536312849162
```

```
True positives:  43
True negatives:  70
False positives:  15
False negatives:  15
```

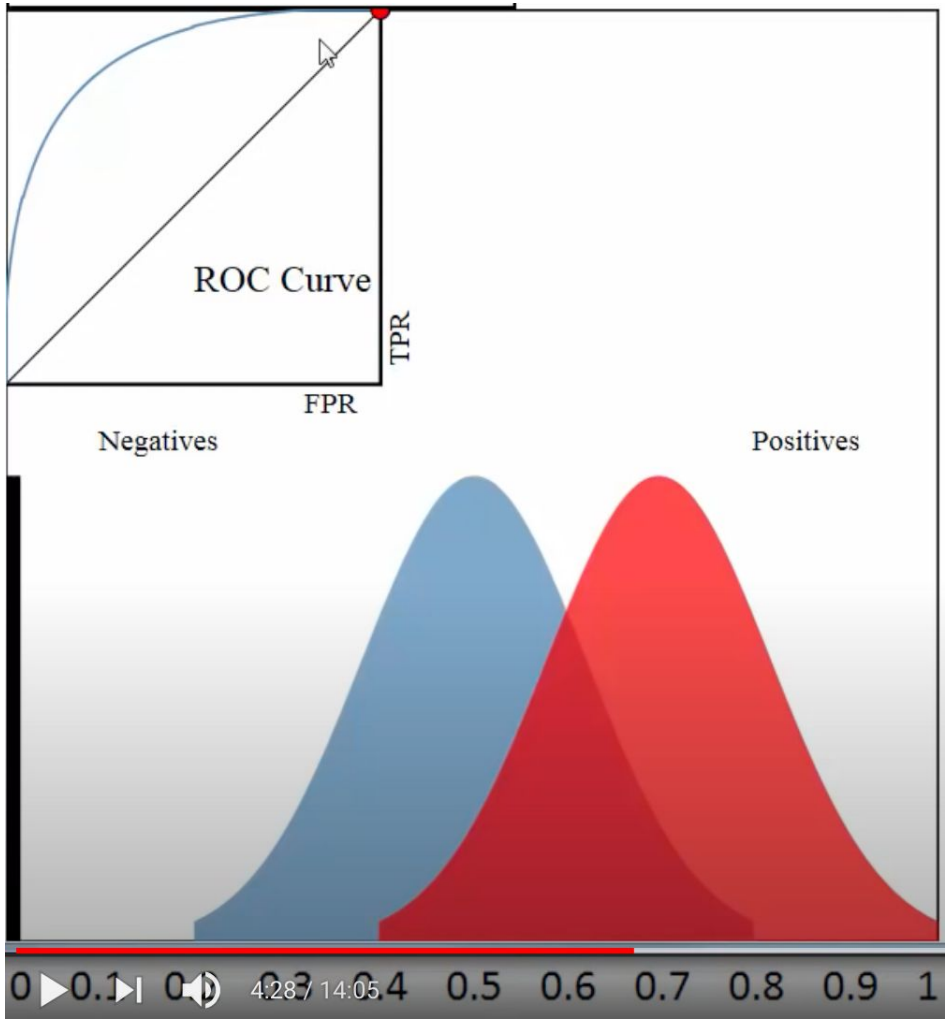
```
# we haven't seen this before, but we are going to measure
# the Receiver Operating Characteristic (ROC)
# https://en.wikipedia.org/wiki/Receiver_operating_characteristic

roc_auc_score(Yvar_test, YPred)
```

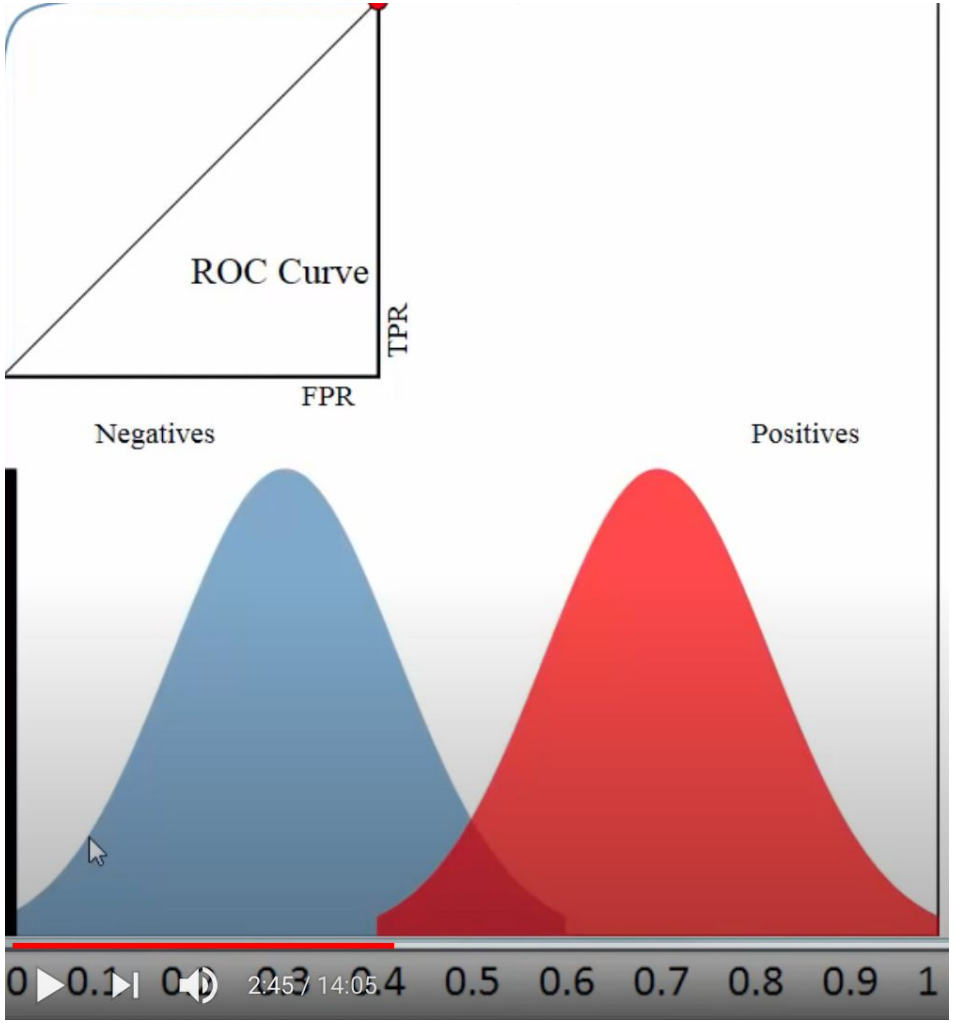
```
0.8100405679513185
```

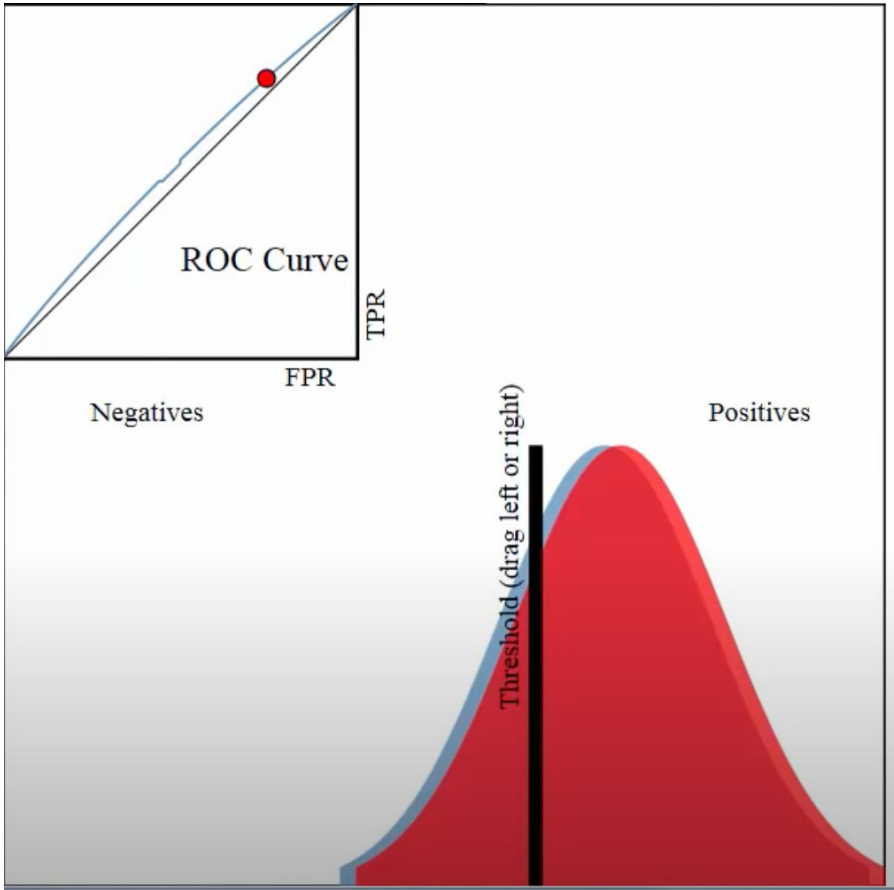
```
# we then take a look at the ROC curve

fpr, tpr, threshold = roc_curve(Yvar_test, YPred)
roc_auc = metrics.auc(fpr, tpr)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
# plt.plot(fpr[2], tpr[2], color='darkorange',
#          lw=lw, label='ROC curve (area = %0.2f)' % roc_auc[2])
```



https://youtu.be/OAl6eAyP-yo

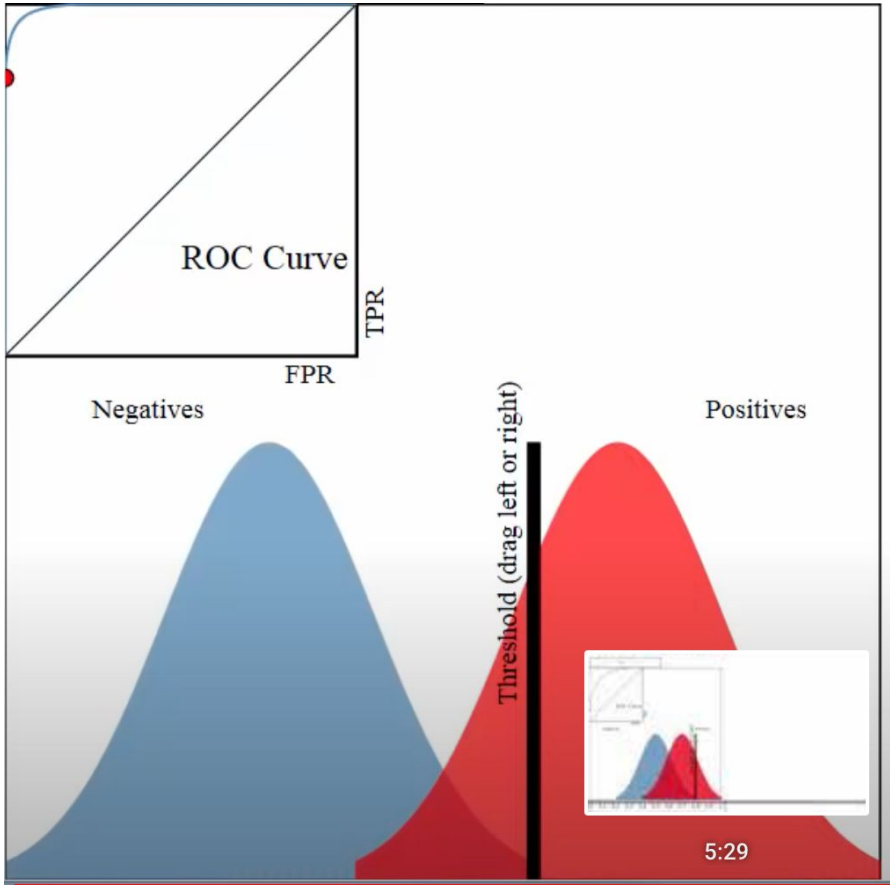



ROC Curve
TPR
FPR
Negatives
Positives
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
2:45 / 14:05



AUC = Area Under the Curve

0.5 = random guessing



AUC = Area Under the Curve

0.5 = random guessing

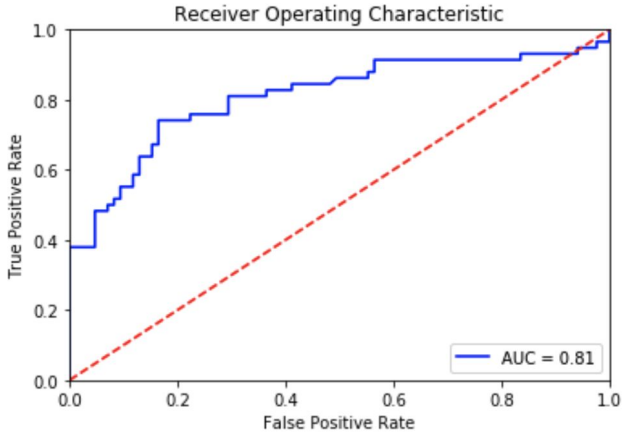
1.0 = perfect classifier

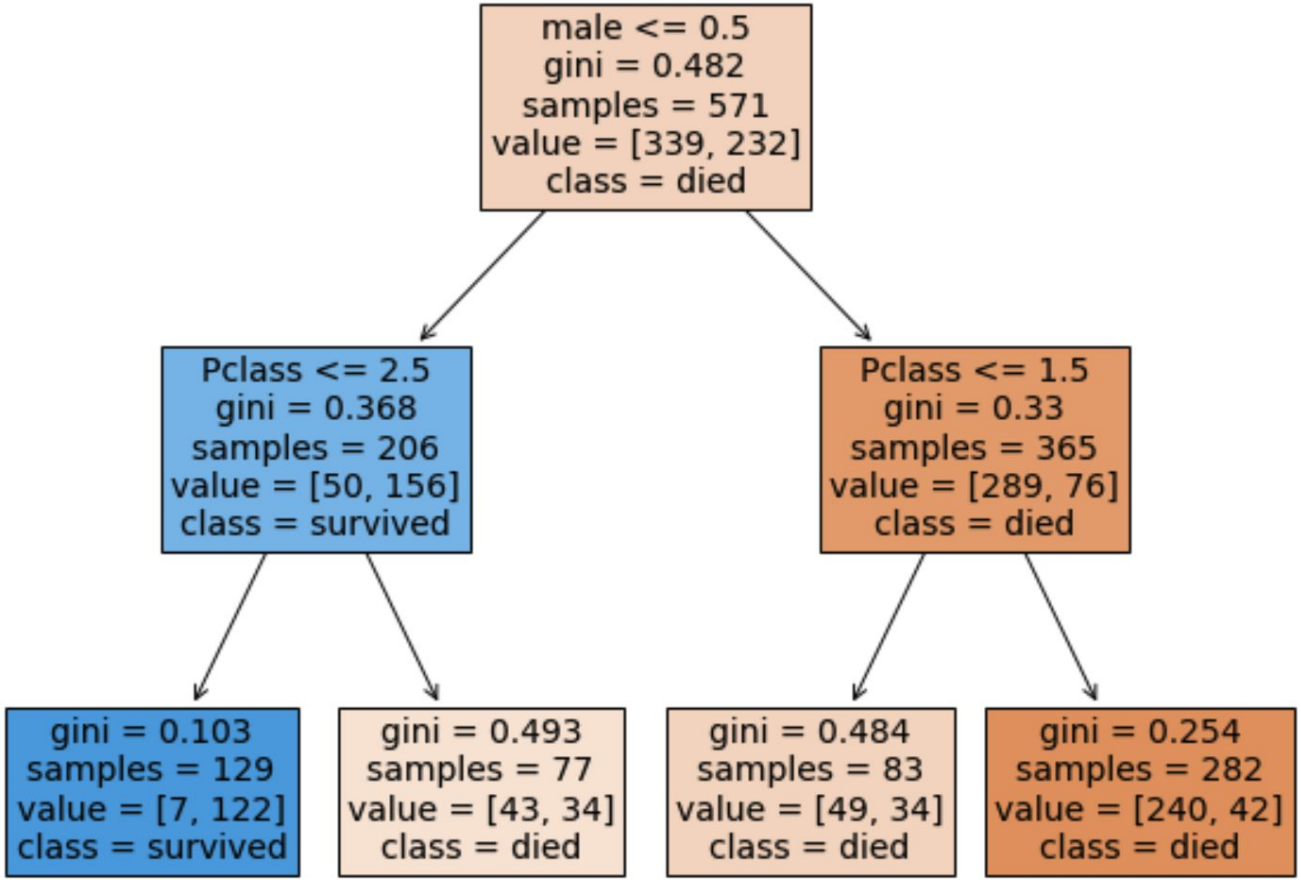
```
True positives:  43
True negatives:  70
False positives:  15
False negatives:  15
```

```
# we haven't seen this before, but we are going to measure
# the Receiver Operating Characteristic (ROC)
# https://en.wikipedia.org/wiki/Receiver_operating_characteristic

roc_auc_score(Yvar_test, YPred)
```

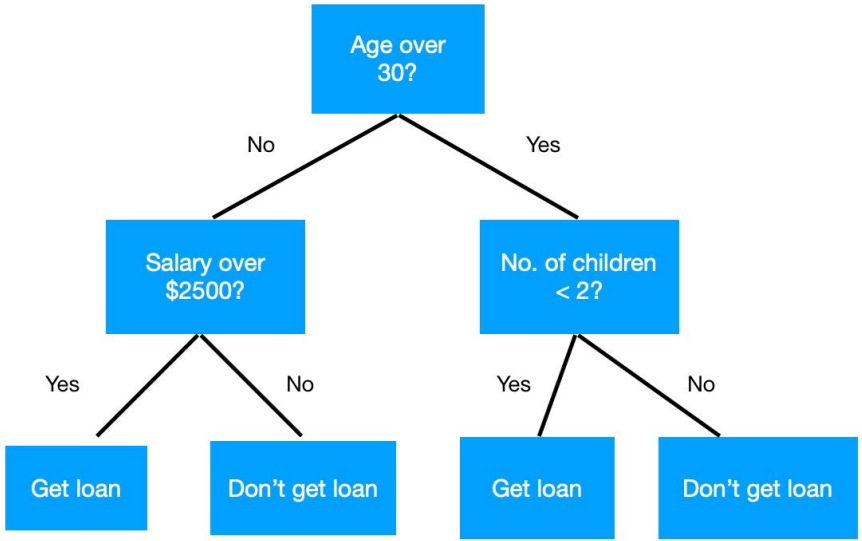
```
0.8100405679513185
```

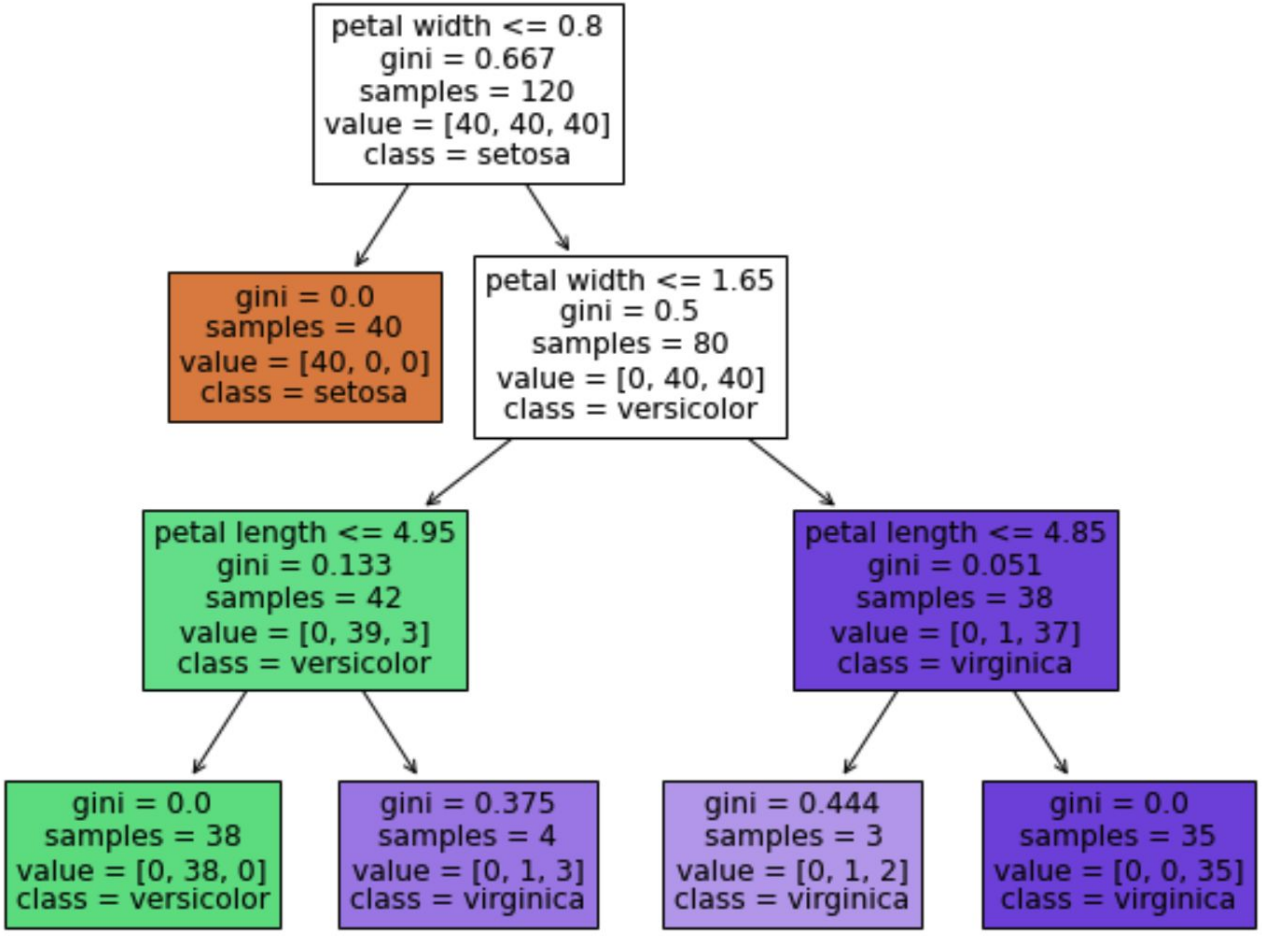
```
# we then take a look at the ROC curve

fpr, tpr, threshold = roc_curve(Yvar_test, YPred)
roc_auc = metrics.auc(fpr, tpr)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
# plt.plot(fpr[2], tpr[2], color='darkorange',
#          lw=lw, label='ROC curve (area = %0.2f)' % roc_auc[2])
```




male <= 0.5
gini = 0.482
samples = 571
value = [339, 232]
class = died
Pclass <= 2.5
gini = 0.368
samples = 206
value = [50, 156]
class = survived
Pclass <= 1.5
gini = 0.33
samples = 365
value = [289, 76]
class = died
gini = 0.103
samples = 129
value = [7, 122]
class = survived
gini = 0.493
samples = 77
value = [43, 34]
class = died
gini = 0.484
samples = 83
value = [49, 34]
class = died
gini = 0.254
samples = 282
value = [240, 42]
class = died

Decision Tree Classifier




petal width <= 0.8
gini = 0.667
samples = 120
value = [40, 40, 40]
class = setosa
gini = 0.0
samples = 40
value = [40, 0, 0]
class = setosa
petal width <= 1.65
gini = 0.5
samples = 80
value = [0, 40, 40]
class = versicolor
petal length <= 4.95
gini = 0.133
samples = 42
value = [0, 39, 3]
class = versicolor
petal length <= 4.85
gini = 0.051
samples = 38
value = [0, 1, 37]
class = virginica
gini = 0.0
samples = 38
value = [0, 38, 0]
class = versicolor
gini = 0.375
samples = 4
value = [0, 1, 3]
class = virginica
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 35
value = [0, 0, 35]
class = virginica

12. Model Improvements

Thought question

Engineer is building a model to look at 10^6 variables that impact the deflection of a building. How can she reduce the number of variables into a more manageable set?

Multicollinearity

```
df['SAT Critical Reading 25th percentile score'].corr(df['SAT Math 25th percentile score'])
```

```
0.9463850891202771
```

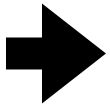
Principal Component Analysis (PCA)

- Examine the patterns among the variables
- Generate new variables that are defined by
  - Maximizing the uniqueness of each variable
  - i.e. minimizing the correlation between the components

4 dimensions

	sepallength	sepalwidth	petallength	petalwidth	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns



2 dimensions

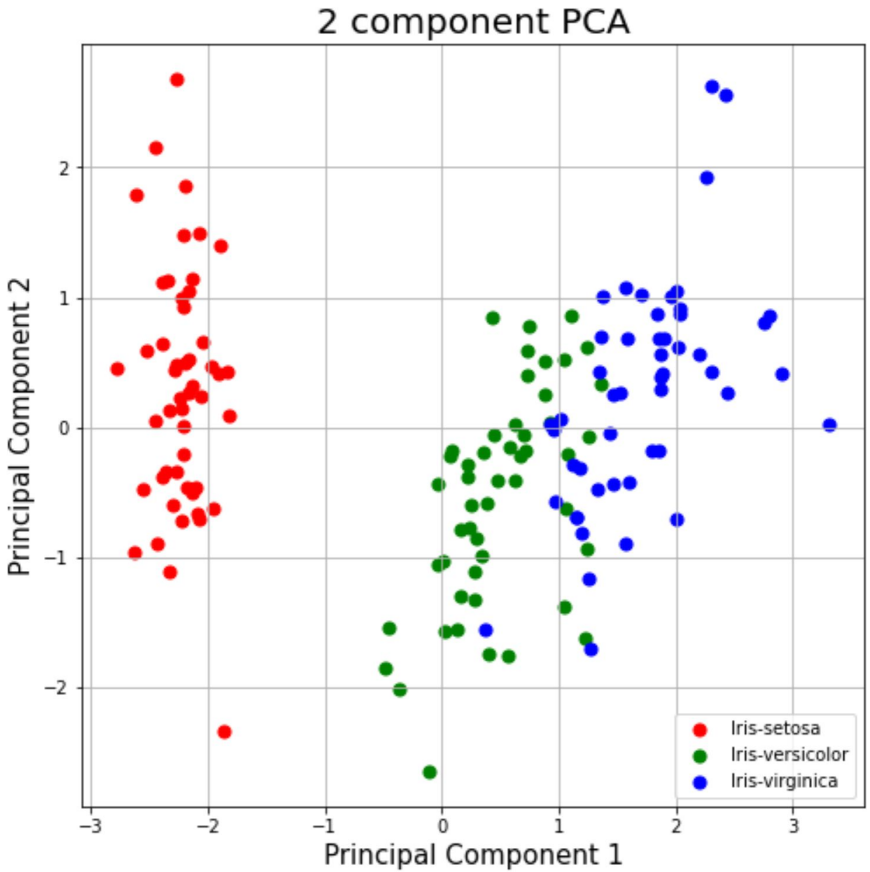
	Principal component 1	Principal component 2	target
0	-2.264703	0.480027	0
1	-2.080961	-0.674134	0
2	-2.364229	-0.341908	0
3	-2.299384	-0.597395	0
4	-2.389842	0.646835	0

```
#PCA is an unsupervised algorithm so it doesn't require the labels (although we have access to it)

#normalize data to NormalDist (mean =0, stdev=1)
x = StandardScaler().fit_transform(X)

#specify the number of components if you know ahead of time. Can leave empty
pca = PCA(n_components=2) #update can tweak this around

#compute the principal components
principalComponents = pca.fit_transform(x)
```

```
print(pca.components_)
```

```
[[ 0.52106591 -0.26934744  0.5804131   0.56485654]
 [ 0.37741762  0.92329566  0.02449161  0.06694199]]
```

	Principal component 1	Principal component 2
Principal component 1	1.000000e+00	5.666012e-17
Principal component 2	5.666012e-17	1.000000e+00


2 component PCA
Principal Component 2
Principal Component 1
Iris-setosa
Iris-versicolor
Iris-virginica
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

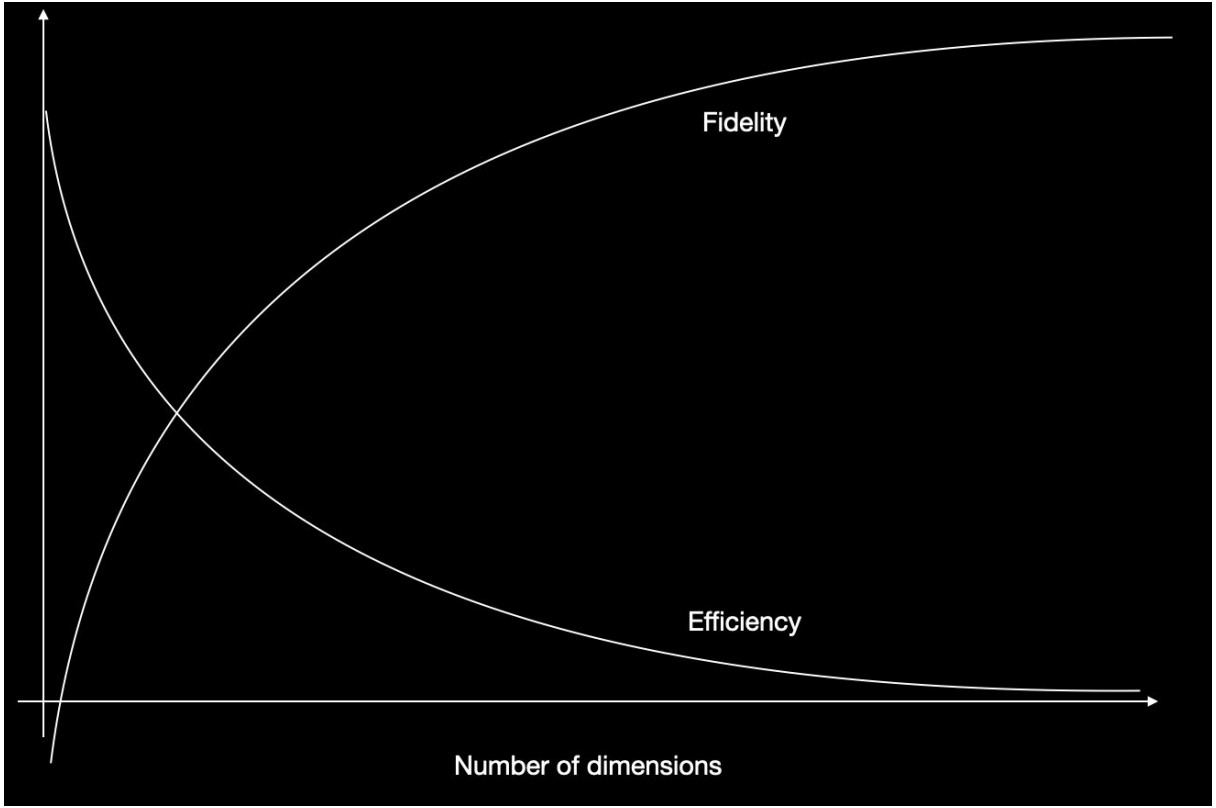
Dimensions

- What do we mean by looking at the number of dimensions?
- These are 10 dimensional vectors:
  - Vector1 = [1, 0, 1, 0, 0, 0, 1, 0, 0, 0]
  - Vector 2 = [1, 0, 0, 0, 0, 0, 1, 1, 0, 1]
  - …
  - Vector n = [0, 0, 1, 0, 0, 0, 0, 1, 1, 1]
- Can we collapse them into fewer number of dimensions based on patterns?

Intuition

- The things that are measured may be formed into a variable that cannot be measured but more accurately represents underlying measures:
  - e.g. the person is fit or not fit based on the underlying measures of height, weight, body fat index, …
- The things that are measured have sufficient overlap they need to be decomposed into separate variables
  - e.g. the person’s weight contributes both to their fitness as well as looks

Dimensionality Trade-Off



In [17]:
```
sentences = [
    "it was a good movie",
    "it was a very good movie",
    "it was an bad movie",

    "it was a bad movie",
    "it was a terrible movie",
]
cv = CountVectorizer(
    min_df = 2, # absolute threshold
    max_df = 0.9, # ratio threshold,
    stop_words='english'
)
cv.fit(sentences)
word_sentence_matrix = cv.transform(sentences)
sentences_df = pd.DataFrame(word_sentence_matrix.toarray(), columns=cv.get_feature_names())
sentences_df
```

Out[17]:

	bad	good
0	0	1
1	0	1
2	1	0
3	1	0
4	0	0

In [11]:

```
sentences = [
    "it was a bad movie",
    "it was a okay movie",
    "it was an awful movie",

    "it was a good movie",
    "i like the movie",
    "i liked this movie okay",
]

sentiment = [
    0,
    0,
    0,
    1,
    1,
    1,
]

cv = CountVectorizer()
cv.fit(sentences)
word_sentence_matrix = cv.transform(sentences)
sentences_df = pd.DataFrame(word_sentence_matrix.toarray(), columns=cv.get_feature_names())
sentences_df
```

Out[11]:

	an	awful	bad	good	it	like	liked	movie	okay	the	this	was
0	0	0	1	0	1	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	1	1	0	0	1
2	1	1	0	0	1	0	0	1	0	0	0	1
3	0	0	0	1	1	0	0	1	0	0	0	1
4	0	0	0	0	0	1	0	1	0	1	0	0
5	0	0	0	0	0	0	1	1	1	0	1	0

In [31]:
```
model['cat']
```

Out[31]:
```
array([ 0.45281 , -0.50108 , -0.53714 , -0.015697,  0.22191 ,  0.54602 ,
       -0.67301 , -0.6891  ,  0.63493 , -0.19726 ,  0.33685 ,  0.7735  ,
        0.90094 ,  0.38488 ,  0.38367 ,  0.2657  , -0.08057 ,  0.61089 ,
       -1.2894  , -0.22313 , -0.61578 ,  0.21697 ,  0.35614 ,  0.44499 ,
        0.60885 , -1.1633  , -1.1579  ,  0.36118 ,  0.10466 , -0.78325 ,
        1.4352  ,  0.18629 , -0.26112 ,  0.83275 , -0.23123 ,  0.32481 ,
        0.14485 , -0.44552 ,  0.33497 , -0.95946 , -0.097479,  0.48138 ,
       -0.43352 ,  0.69455 ,  0.91043 , -0.28173 ,  0.41637 , -1.2609  ,
        0.71278 ,  0.23782 ], dtype=float32)
```

In [32]:
```
from sklearn.metrics.pairwise import cosine_similarity

cosine_similarity([model['cat']], [model['dog']])
```

Out[32]:
```
array([[0.92180055]], dtype=float32)
```

In [33]:
```
cosine_similarity([model['cat']], [model['laptop']])
```

Out[33]:
```
array([[0.29200307]], dtype=float32)
```

QUEST
HONORS PROGRAM
QUALITY ENHANCEMENT SYSTEMS AND TEAMS

notebook

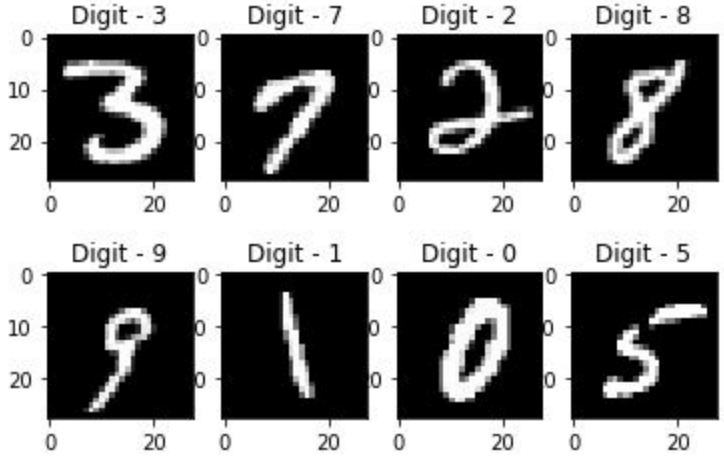
Linear operations with Word Embeddings

In [34]:
```
queen = model['king'] - model['man'] + model['woman']

cosine_similarity([queen], [model['queen']])
```

Out[34]:
```
array([[0.8609581]], dtype=float32)
```

Thought exercise

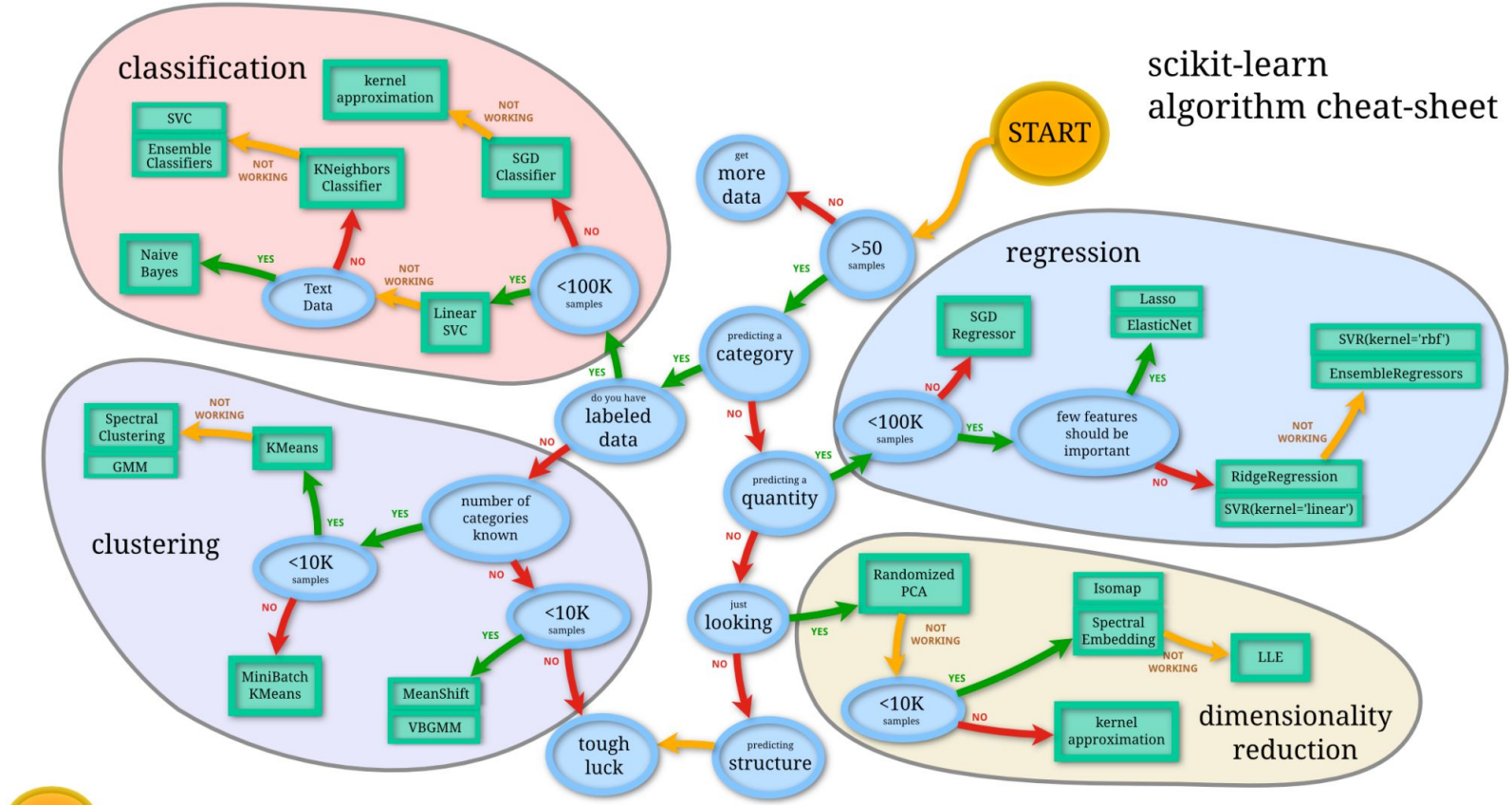
Campbell’s currently uses people to visually inspect tomatoes before they are used as an ingredient in soup. Could this be automated? What data science considerations are there?


Digit - 3
Digit - 7
Digit - 2
Digit - 8
Digit - 9
Digit - 1
Digit - 0
Digit - 5
0
10
20
0
20

Turning an image into numbers

- Easier than you might think when the image is digital already
- Facial recognition uses this

notebook


scikit-learn
algorithm cheat-sheet
START
get more data
NO
>50 samples
YES
predicting a category
YES
do you have labeled data
YES
<100K samples
YES
Linear SVC
NOT WORKING
Text Data
NO
Naive Bayes
YES
KNeighbors Classifier
NO
Ensemble Classifiers
NOT WORKING
SVC
kernel approximation
NOT WORKING
SGD Classifier
NO
classification
NO
predicting a quantity
YES
<100K samples
YES
few features should be important
YES
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
NOT WORKING
NO
RidgeRegression
SVR(kernel='linear')
SGD Regressor
NO
regression
NO
number of categories known
YES
<10K samples
YES
KMeans
NOT WORKING
Spectral Clustering
GMM
NO
MiniBatch KMeans
NO
<10K samples
YES
MeanShift
VBGMM
NO
clustering
NO
just looking
YES
Randomized PCA
NOT WORKING
<10K samples
YES
Isomap
Spectral Embedding
NOT WORKING
LLE
NO
kernel approximation
dimensionality reduction
NO
predicting structure
tough luck
Back
scikit learn

14. Storytelling with Data

15. The Future of Data Science

Prompt

How could a new product development company use data science to make bespoke products for consumers?


Webkinz
discover a virtual world
GANZ


Harrah's


ZARA


CVS
pharmacy®


TARGET®

Moving from development to production

- How to decide where to locate your data
- How to use distributed data for faster analytics
- How to think about big data
- How to think about advanced models (e.g. deep learning)
- How to build better tools


13:41:18
New York
18:41:18
London
01:41:18
Hong Kong
02:41:18
Tokyo
Most Up
Most Down
NFIB Small Business Optimism
JOLTS Job Openings
Americas
United States
Canada
Brazil (USD)
Colombia (USD)
Argentina (USD)
Mexico (USD)
EMEA
United Kingdom
France
Germany
Italy
Spain
Portugal
Sweden
Netherlands
Switzerland
Greece
Asia/Pacific
Japan
Australia
New Zealand
South Korea
Last hour movers (5)
Last hour laggers (5)
Remaining Securities (590)
Catastrophe Bonds Brace For Disaster
Global Macro News
Top Performers (10)
Bottom Performers (10)
Remaining Securities (485)
Erste Group Bank AG (Vienna)
Group by Region Sector
Bonds Can Lose


Overview
Patient Leads
Web Presence
Reputation
Dr. Katie Coleman
Overall Rating
4.0
+0.5 points from last month
3.6
July 2019
jan
apr
jul
oct
dec
Public Reviews
1 year
Total Reviews
3,431
Since PatientPop
+1,725
Reviews
Since PatientPop
400
200
0
jan
feb
mar
apr
may
jun
jul
aug
sep
oct
nov
dec
Patient Sentiment
Superb!
91% of patients satisfied with their visit.
Latest Reviews
Deena Timmons
5 hours ago from Yelp
I must once again praise Dr. Coleman for her outstanding advise and medical care. Her skills as a physician are stellar, and she will only recommend procedures that can enhance your physical beauty. The office is immaculate, colorful and inviting.
Sheila Lee
2 days ago from PatientPop
Dr. Coleman is the consumate professional. I have seen dermatologists in NYC and Beverly Hills, and she is by far the most knowledeable. As a physician, her primary concern is health, skin care, and screening.
Sarah Doyle
5 days ago from Facebook
Dr. Coleman clearly cares about her patients and spent time walking me through my skin's health and things I can do to stay looking my best.
Online Requests
9,245
Requests
New: 5.9k
Returning: 3.1k
Phone Leads
3,271
Leads
New: 2.6k
Returning: 671
Website Traffic
Unique visitors
26,751
Direct
14,750
55%
Search Position Increases
dermatology
6,721
botox
4,320

What comes next in 390?

- Interpolation vs. Extrapolation
- Descriptive vs. Normative
- Current vs. Future