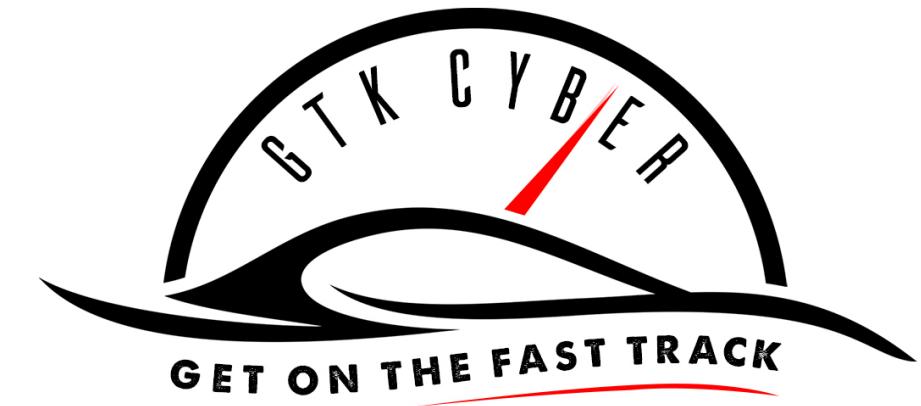


GT K CYBER

GET ON THE FAST TRACK

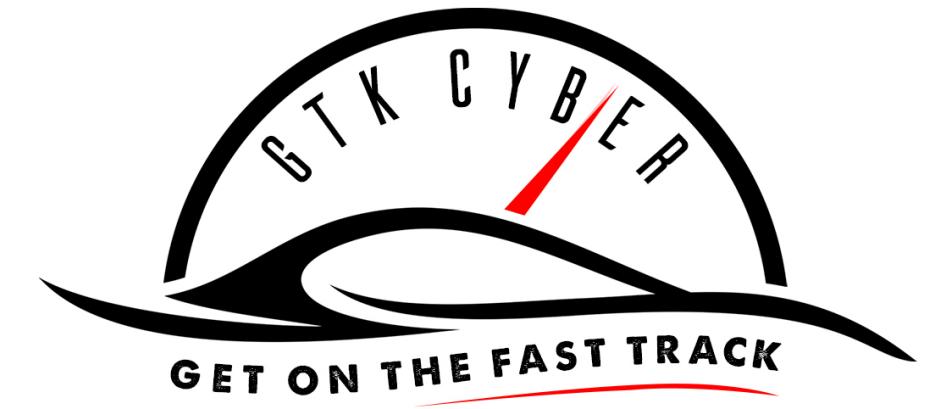
Data Science for Security Professionals - Day 1

Module 1: Introduction & Data Preparation with Pandas

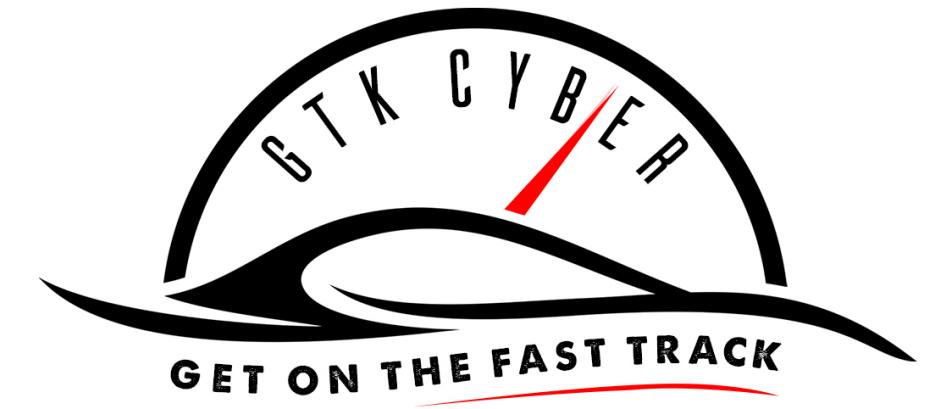


Expectations

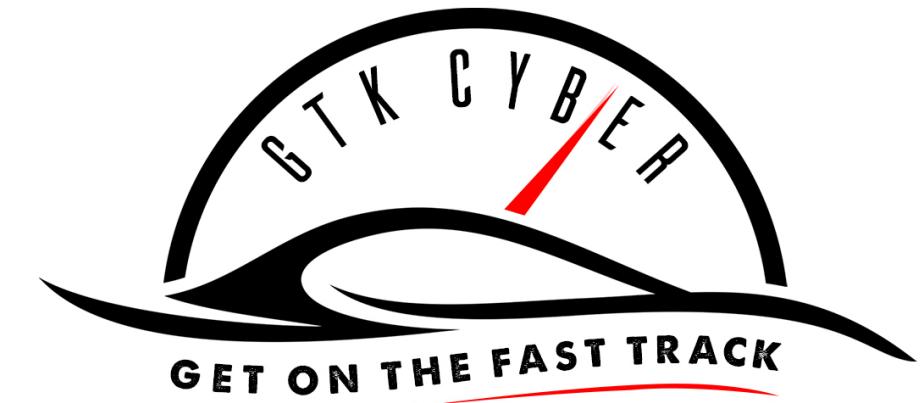
- Please participate and **ask questions**.
- Please follow along and **TRY OUT** the examples yourself during the class
- All the answers are in the slide decks or GitHub repository, but please try to complete the exercises **without looking at the answers**.
- Have fun!



Introduction



Who am I?

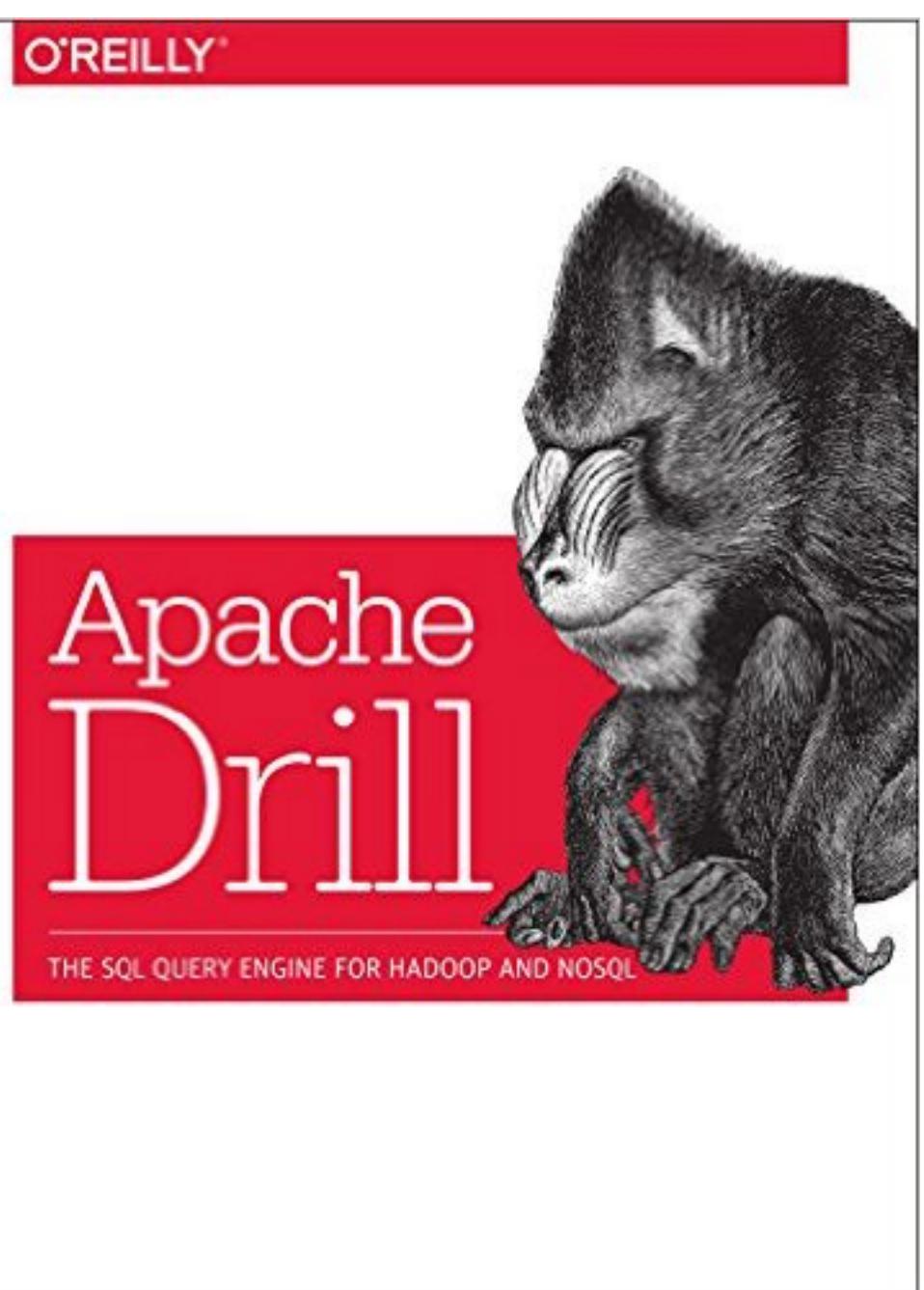


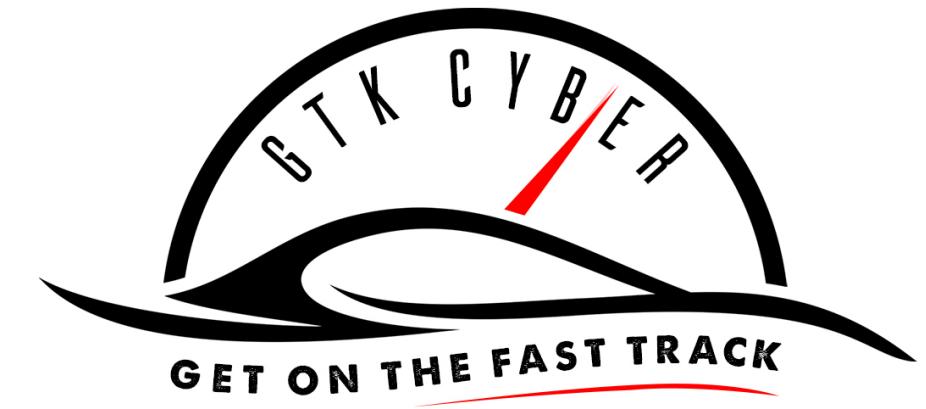
Charles Givre

- Senior Lead Data Scientist @ Booz Allen
- 5 Years @ CIA
- Working on Apache Drill Book
- Masters Degree in Middle Eastern Studies
- Undergraduate in Comp.Sci & Music

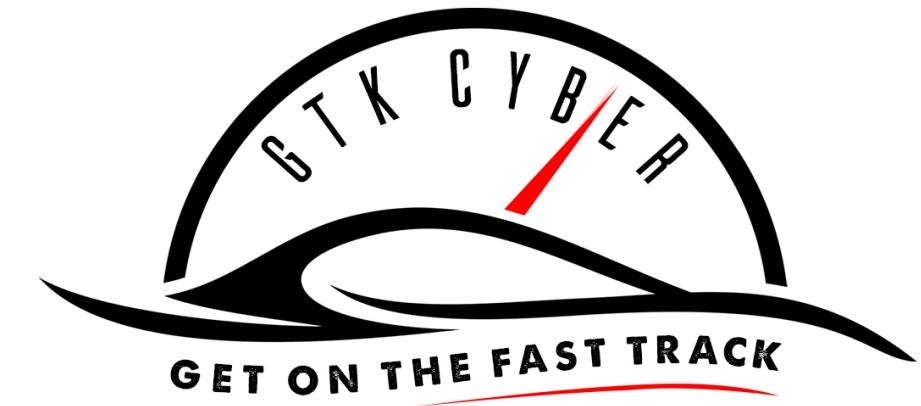


Booz | Allen | Hamilton
100 YEARS



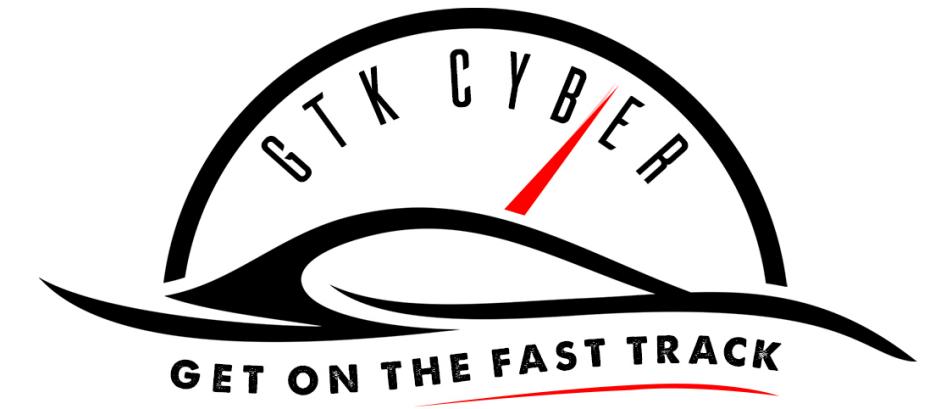


What is Data Science?

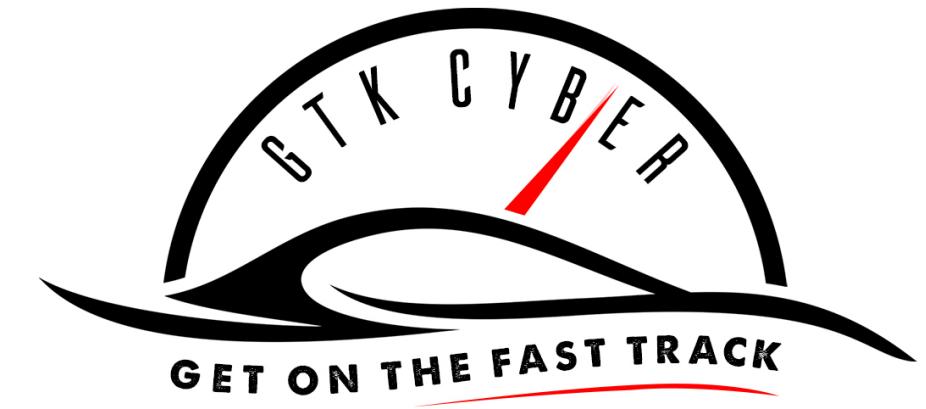


Data Science is the art of **turning data into actions**. This is accomplished through the creation of data products, which provide actionable information without exposing decision makers to the underlying data or analytics

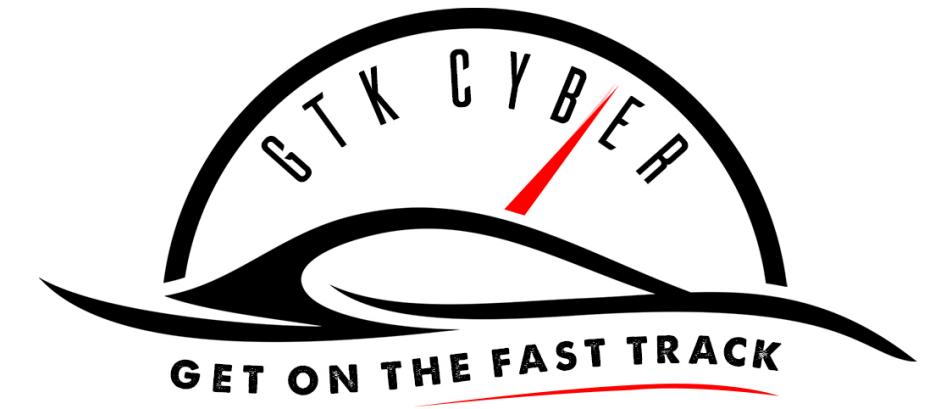
Booz Allen Hamilton, Field Guide to Data Science, Pg. 17



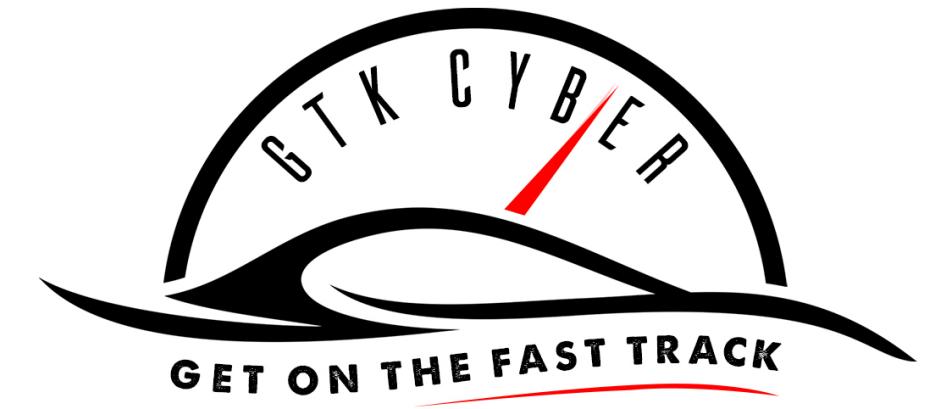
Extracting **useful** information



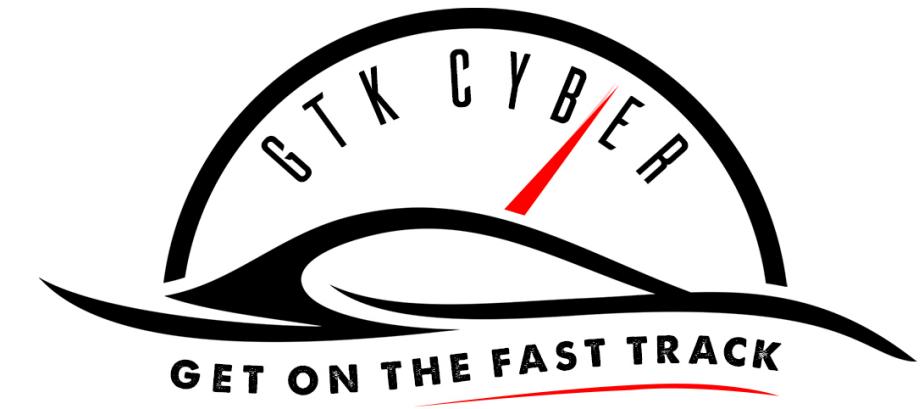
Extracting useful information **from data**



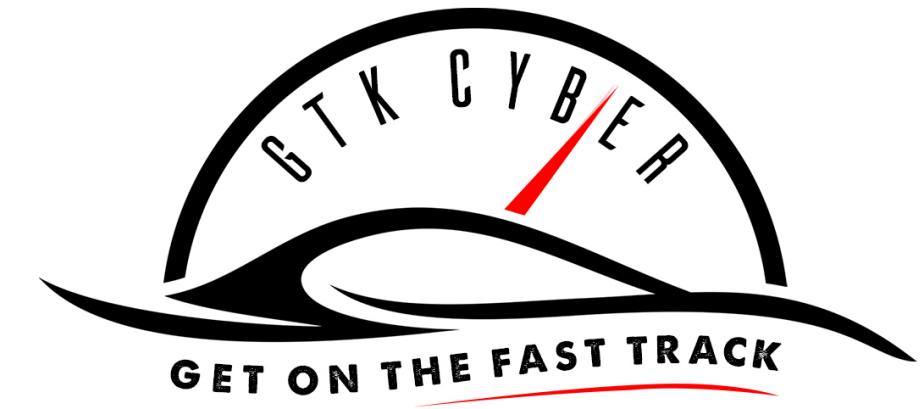
Answering questions



Answering **business questions**



Answering business questions **with data**



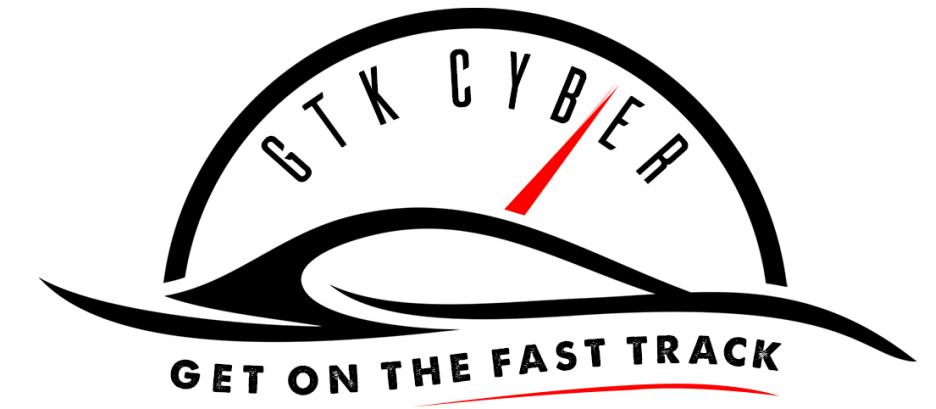
Answering business questions with data

- Know what you want to know



Answering business questions with data

- Know what you want to know
- Have the **technical skills** to get it



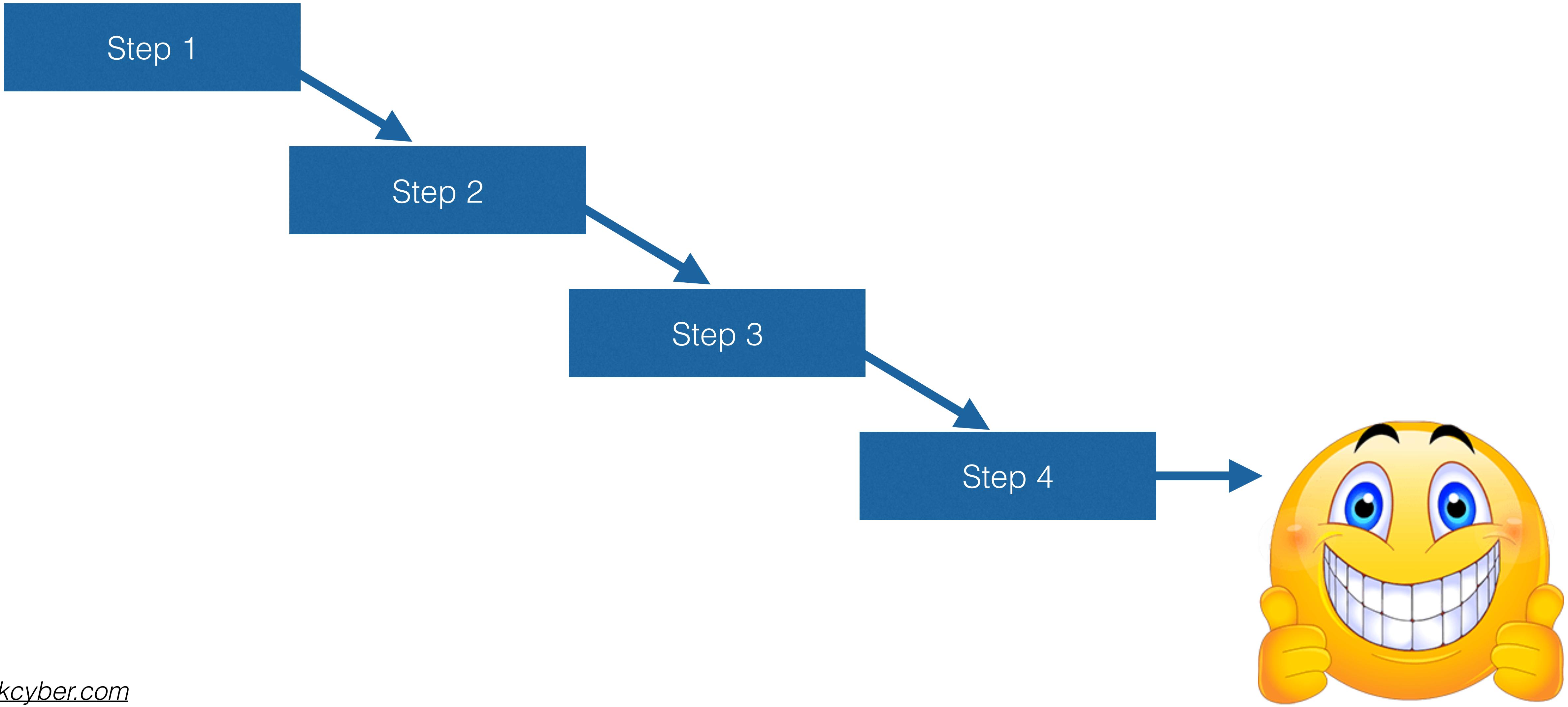
Analyst ← → Developer



Analyst + Developer

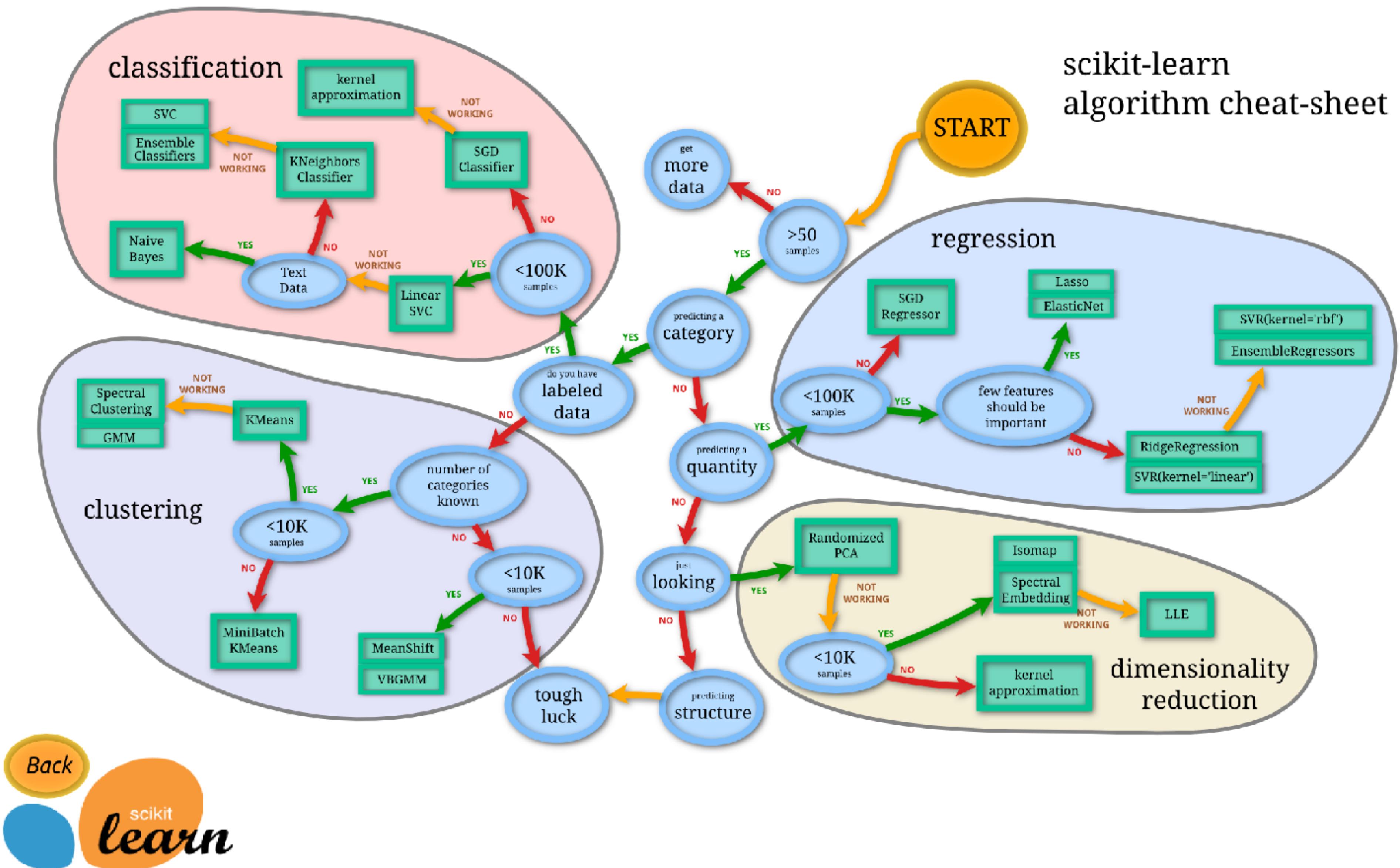


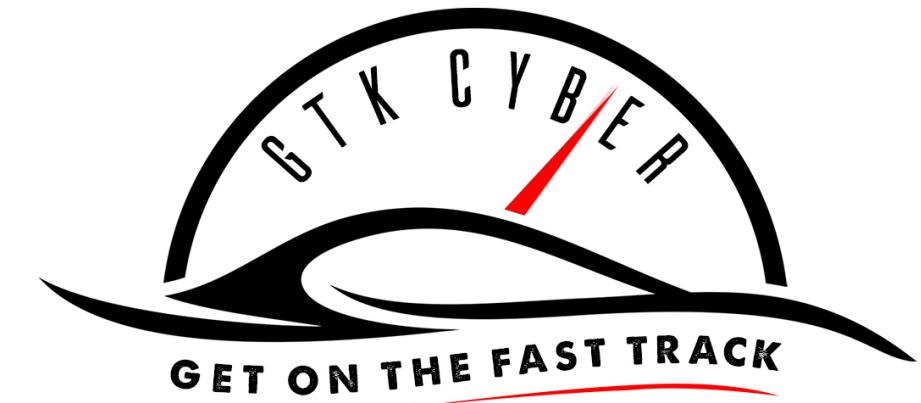
What Data Science is Not



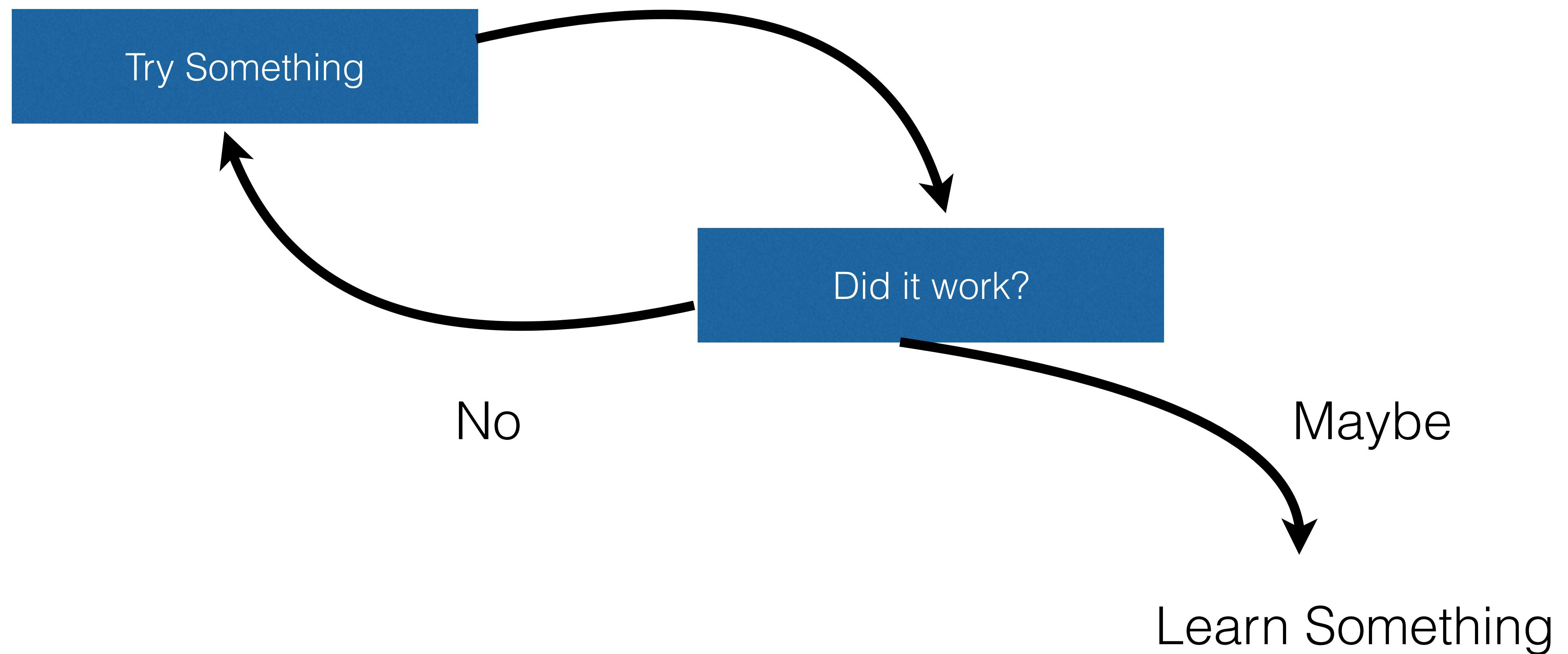


What Data Science is Not



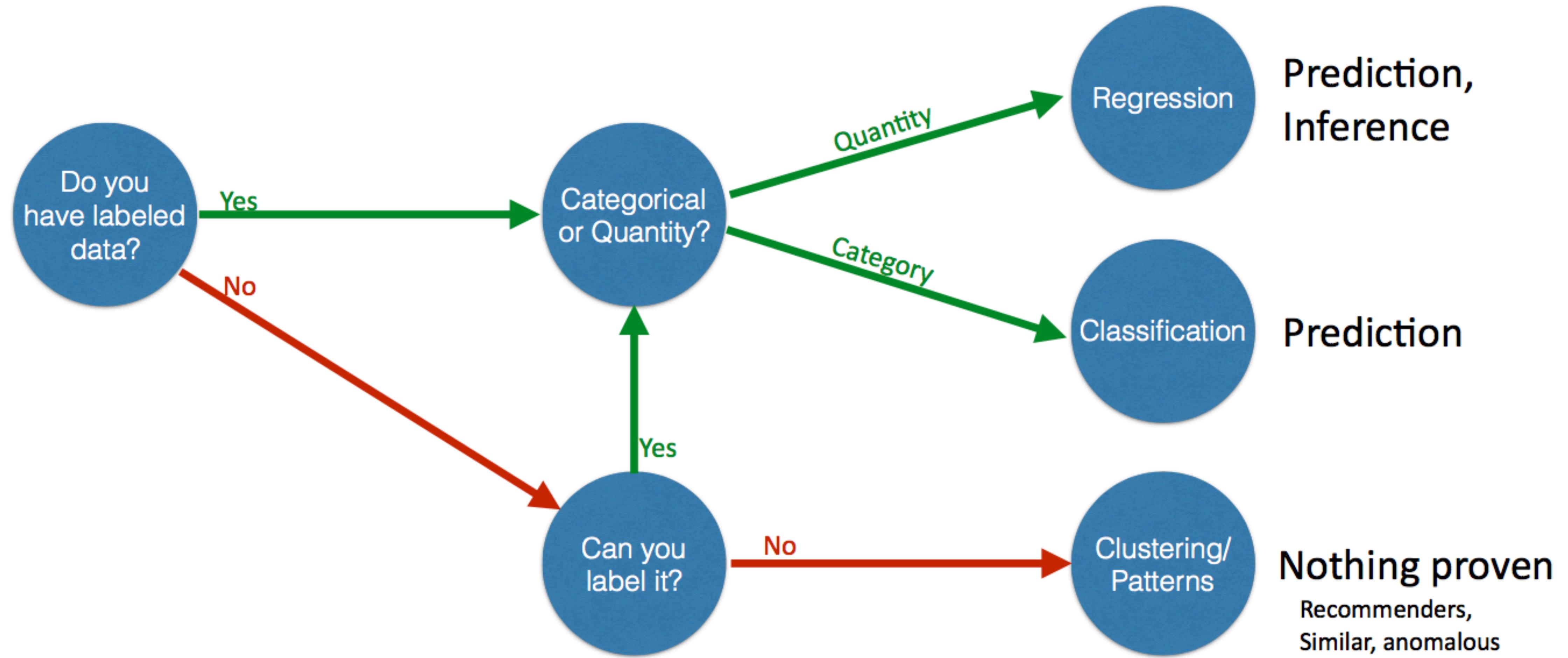


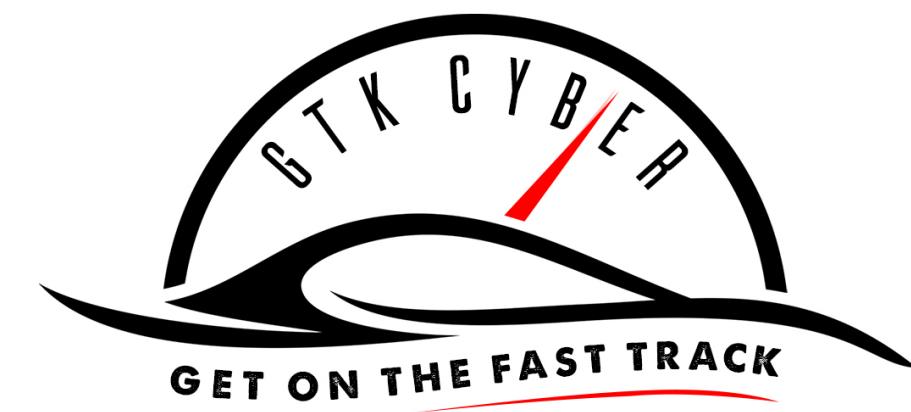
What Data Science is



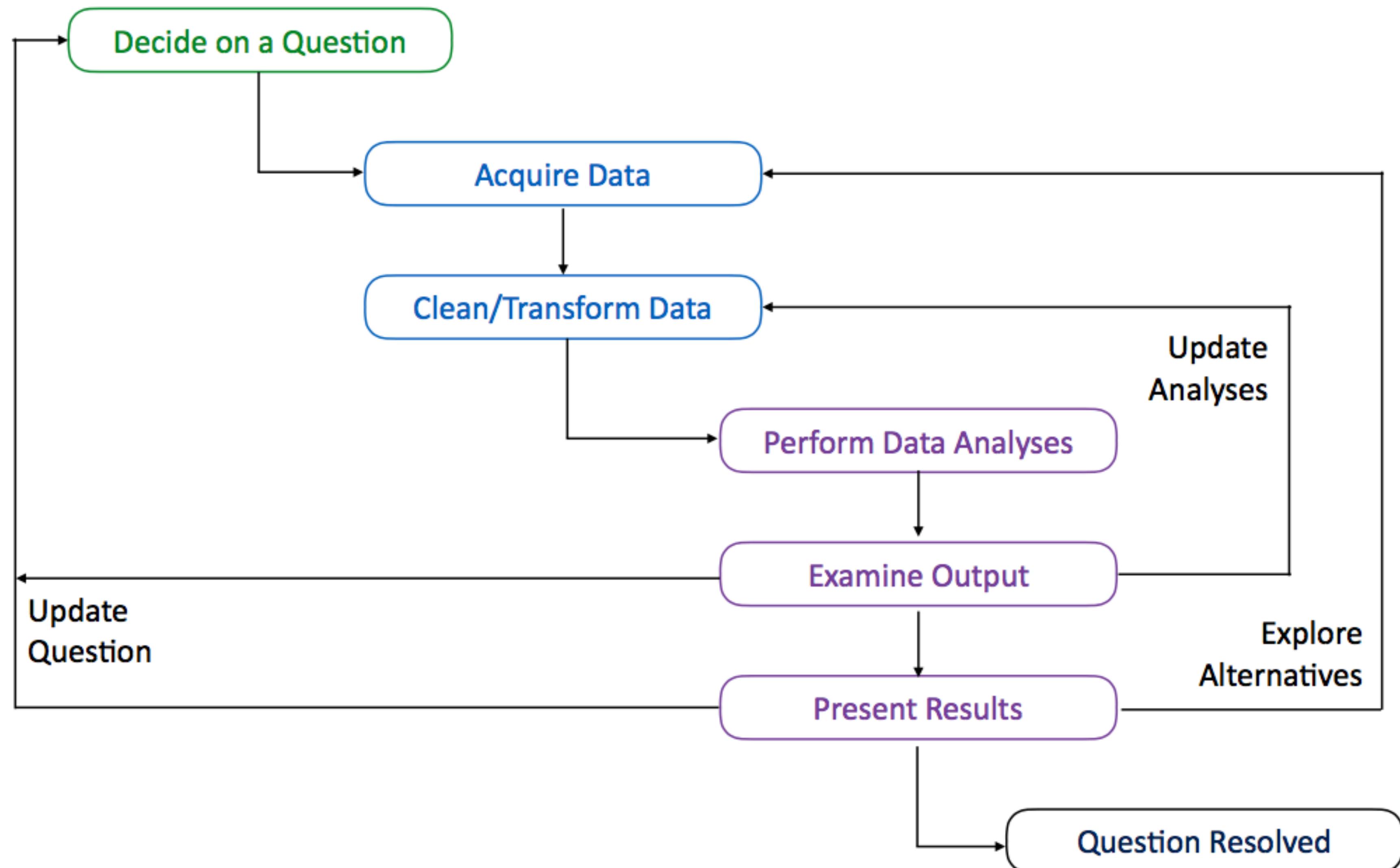


What Data Science might be...





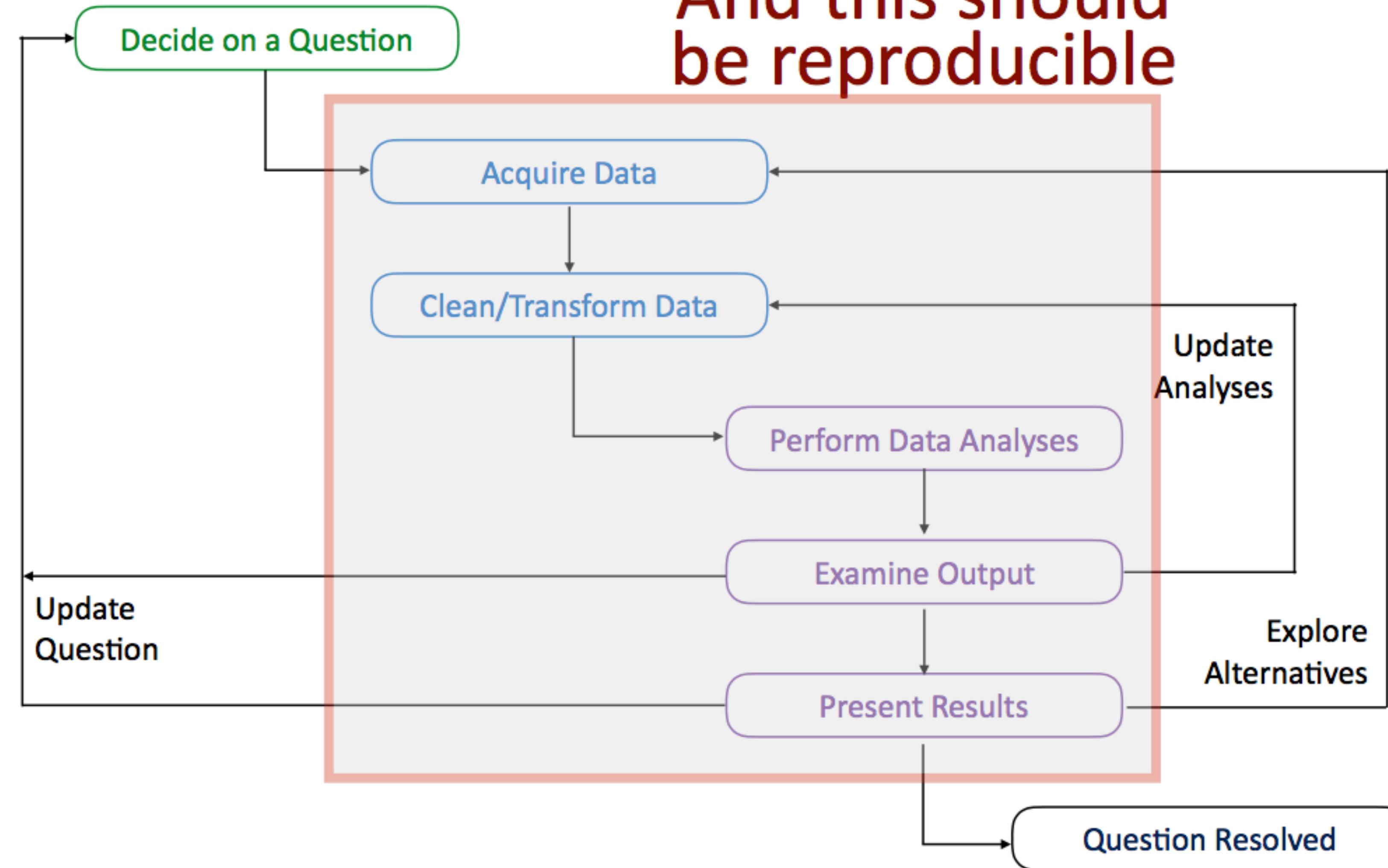
Research Process

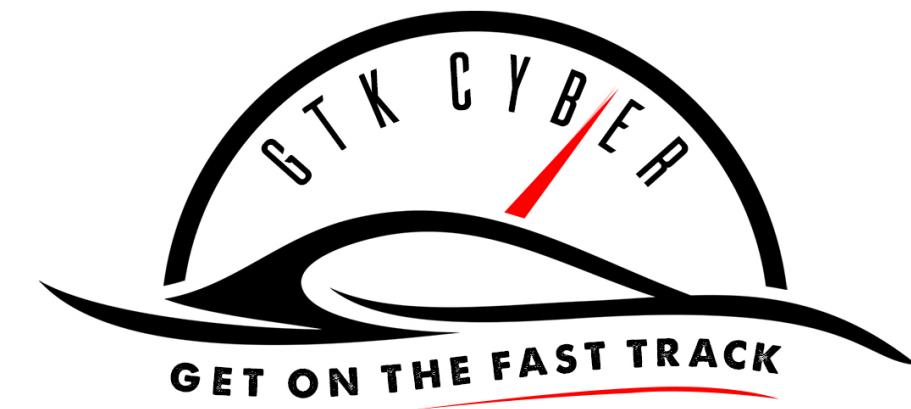




Research Process

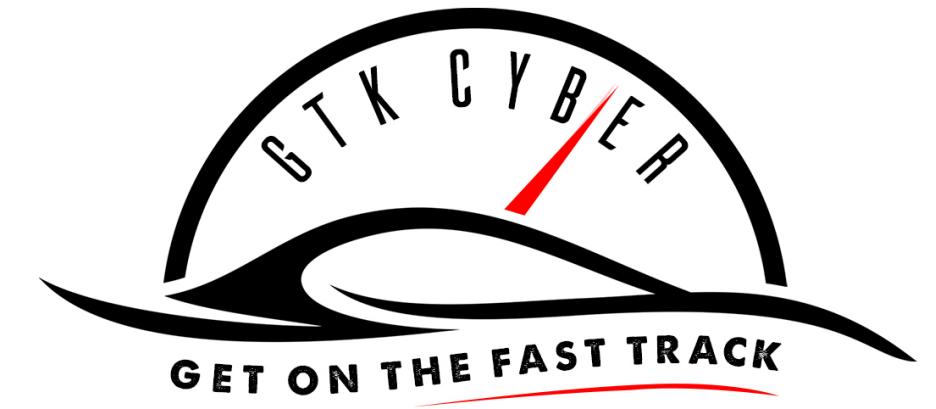
And this should
be reproducible



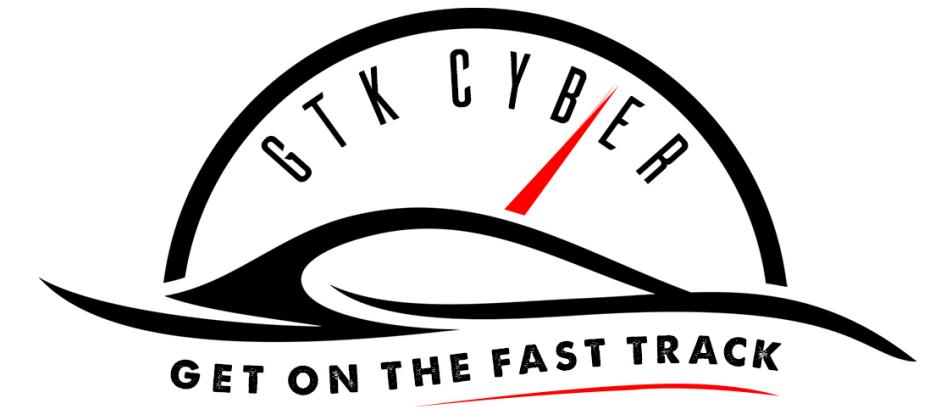


“The term "data scientist" will subside and may well sound dated five years from now. **The skills will become more commonplace and commoditized. When that happens, the real boom will begin**, because the technology will become widely adopted and thus more useful. ... **Instead, we need self-service tools that empower smart and tenacious business people to perform Big Data analysis themselves.**

–Andrew Brust, “Data scientists don't scale”, <http://www.zdnet.com/article/data-scientists-dont-scale/>



Time to Insight



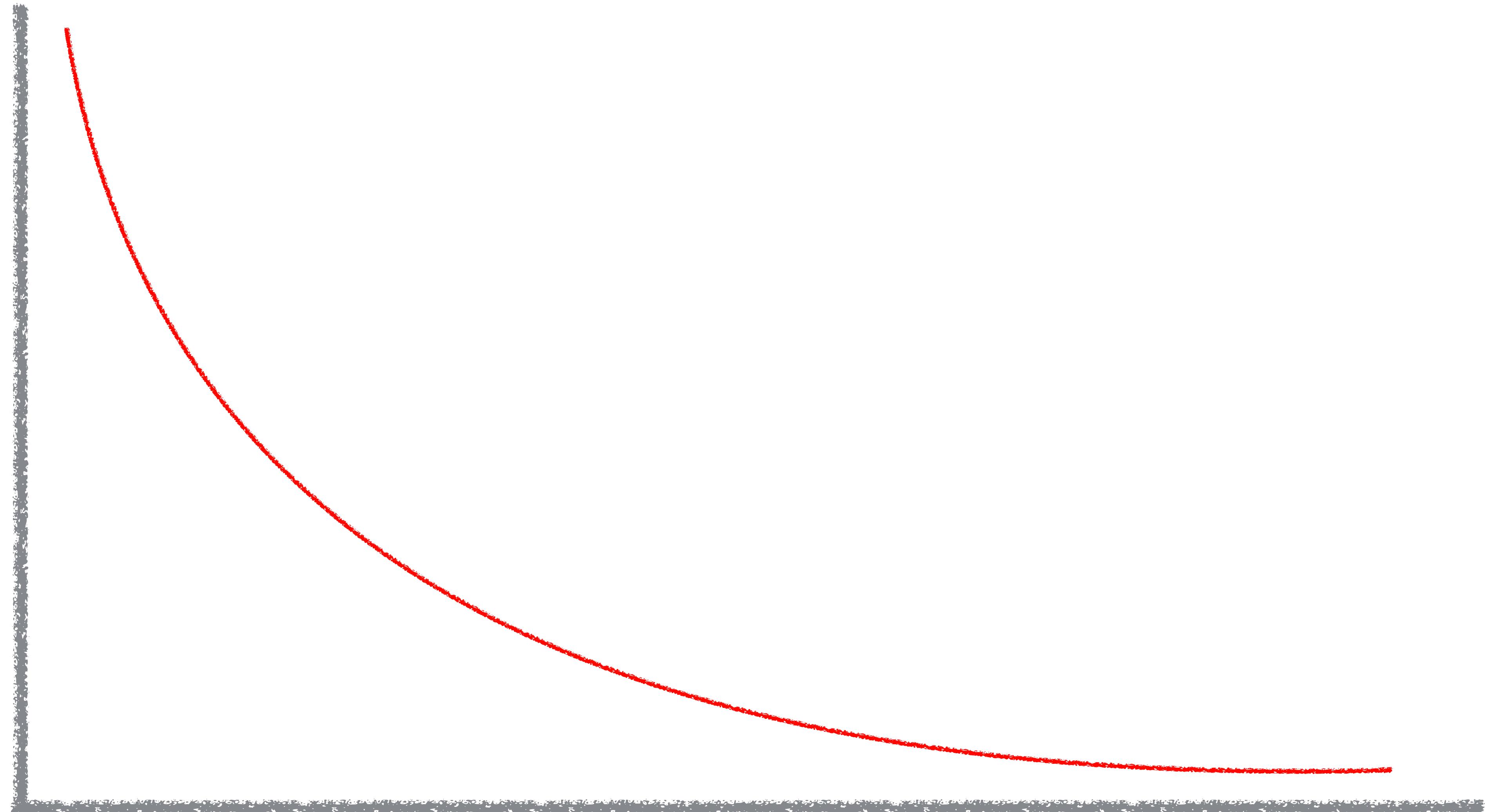
Time to Insight

Time = \$\$



Value of Insights over Time

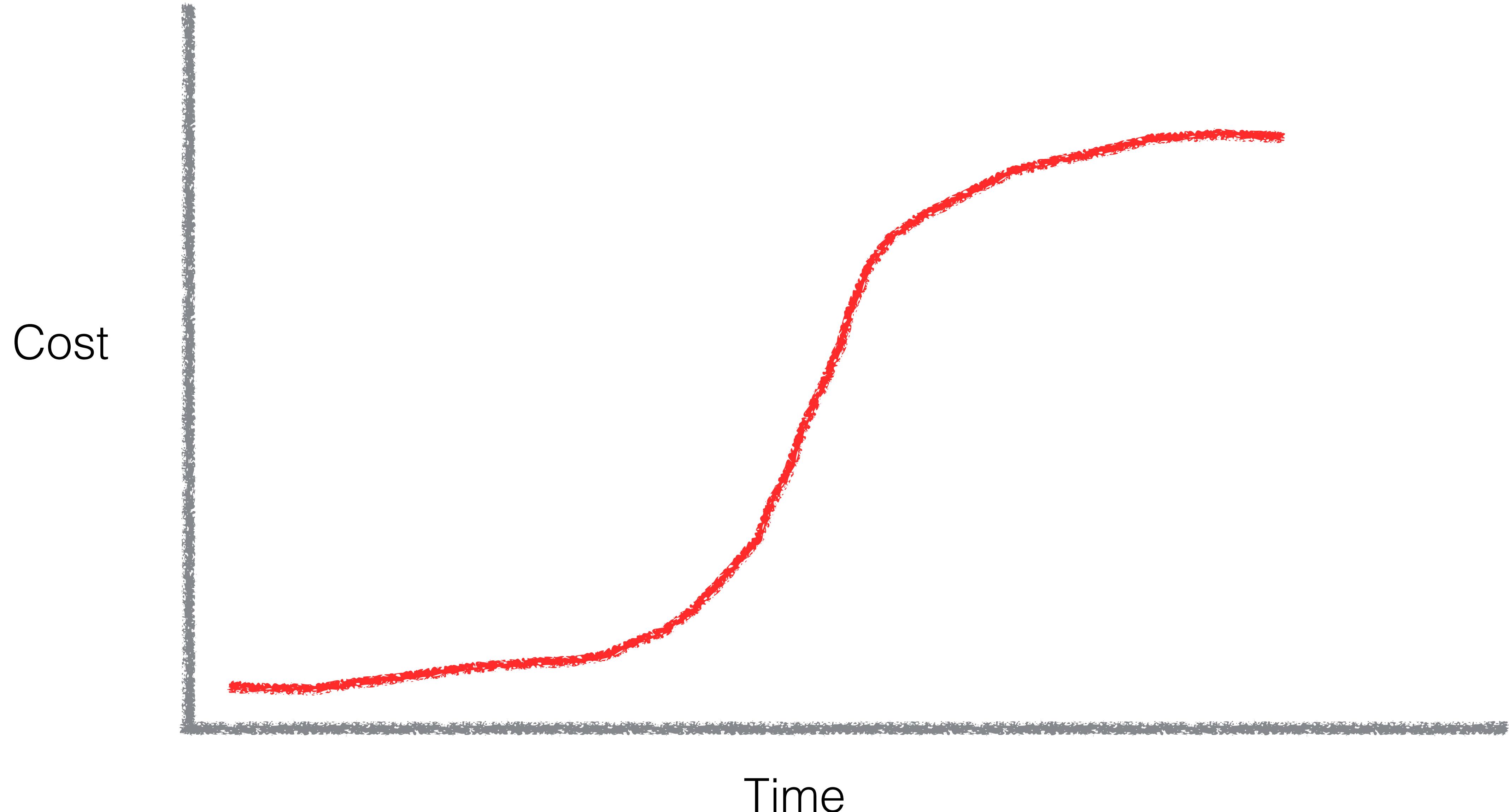
Value



Time

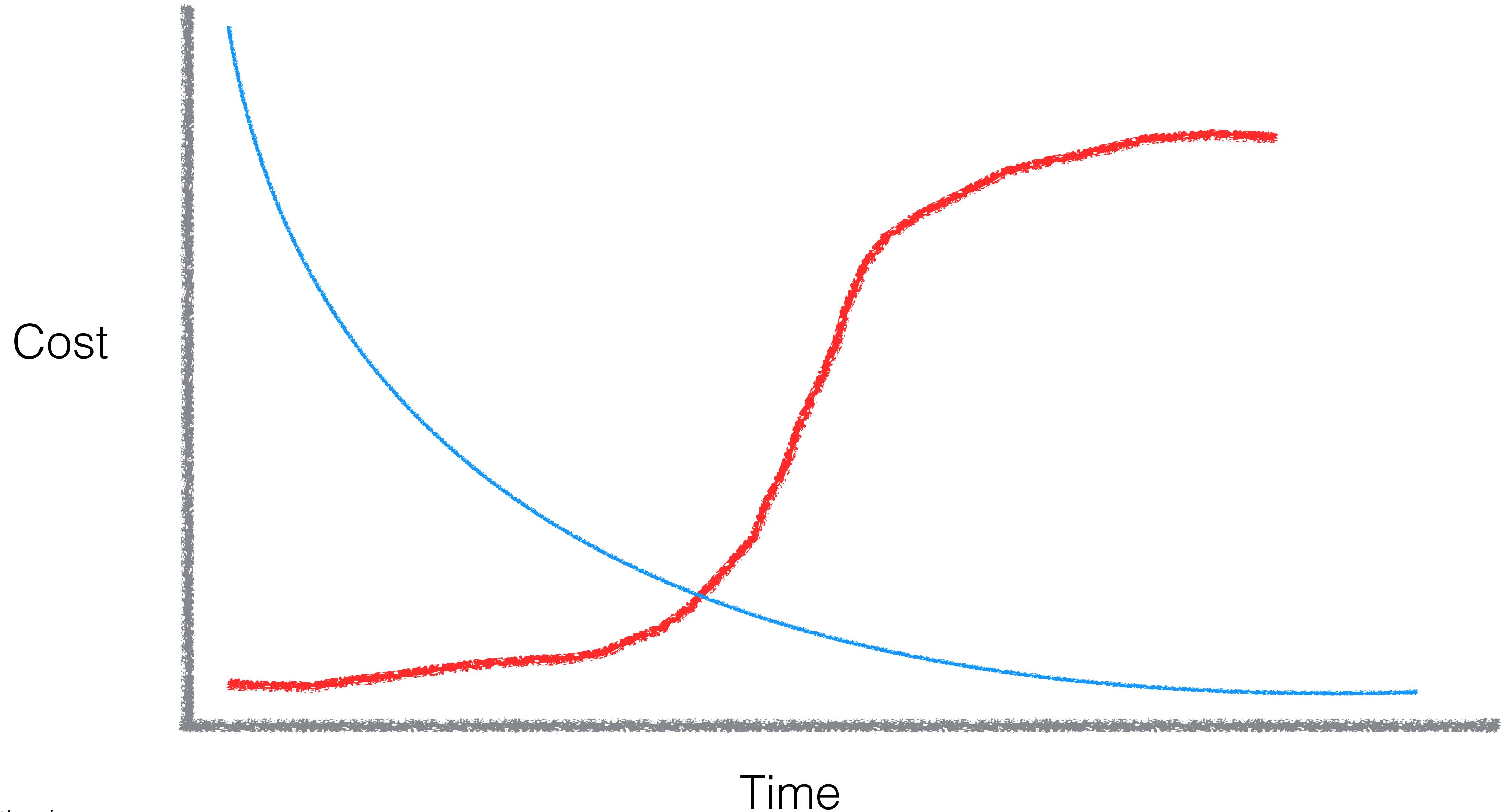


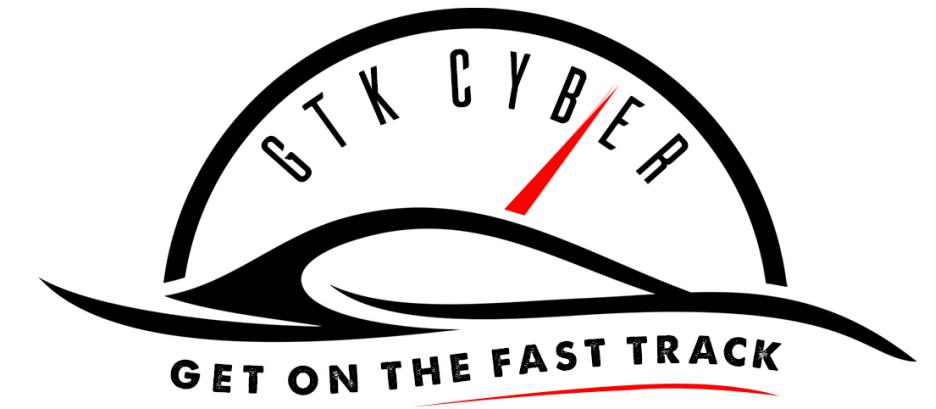
Cost of Insights over Time



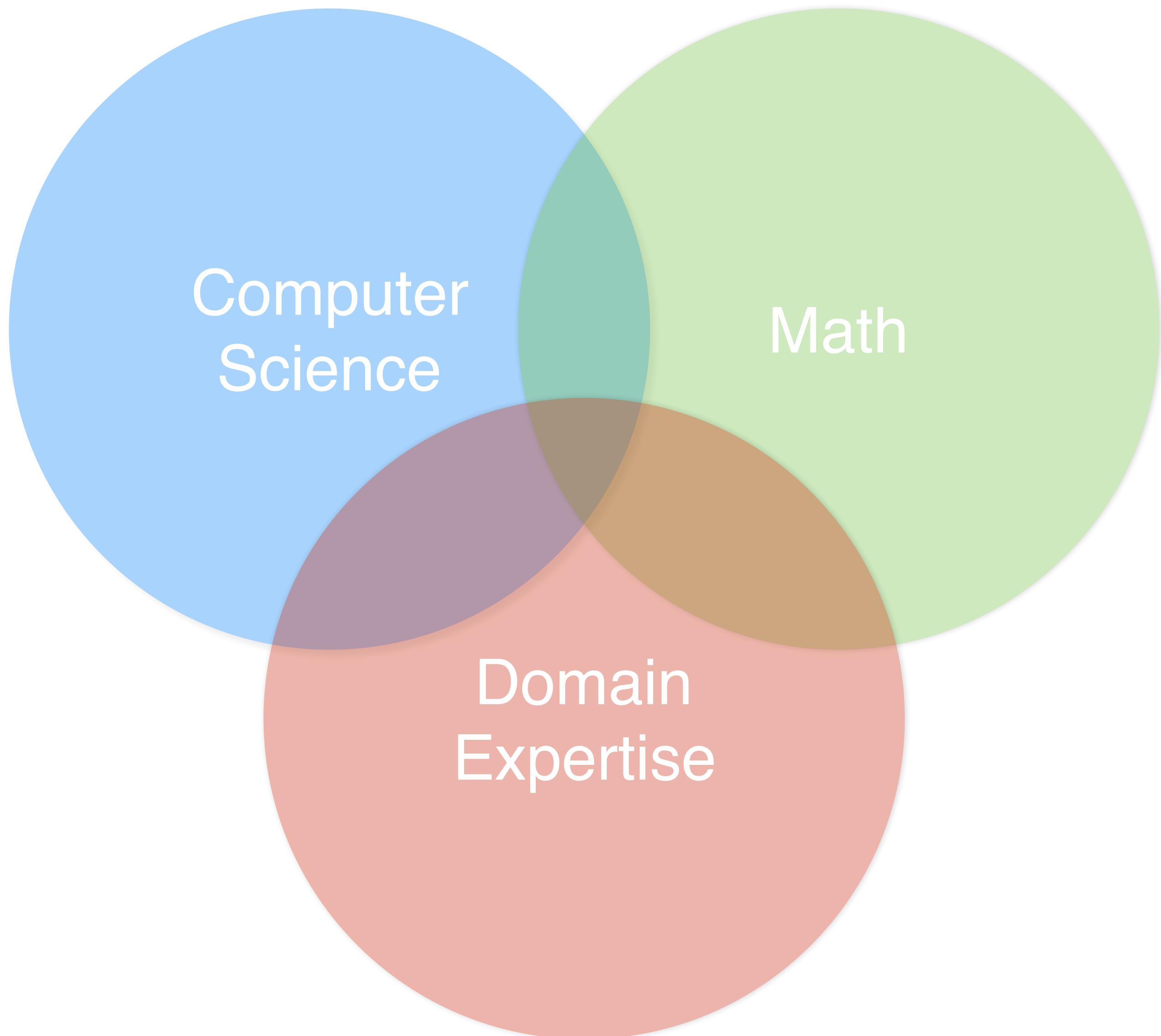
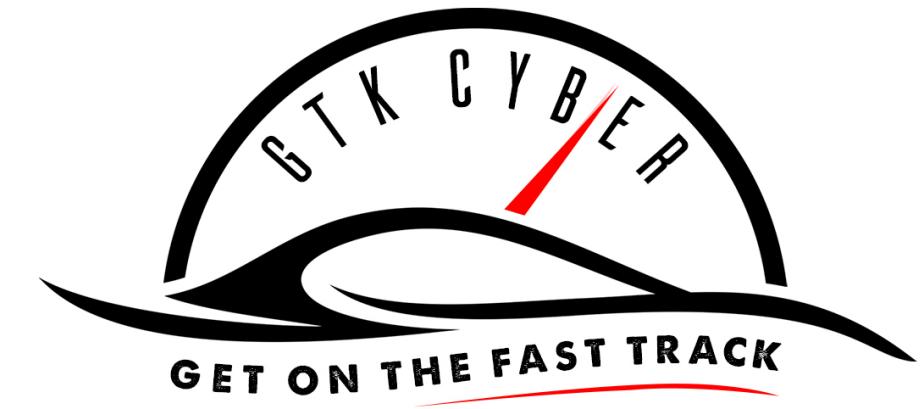


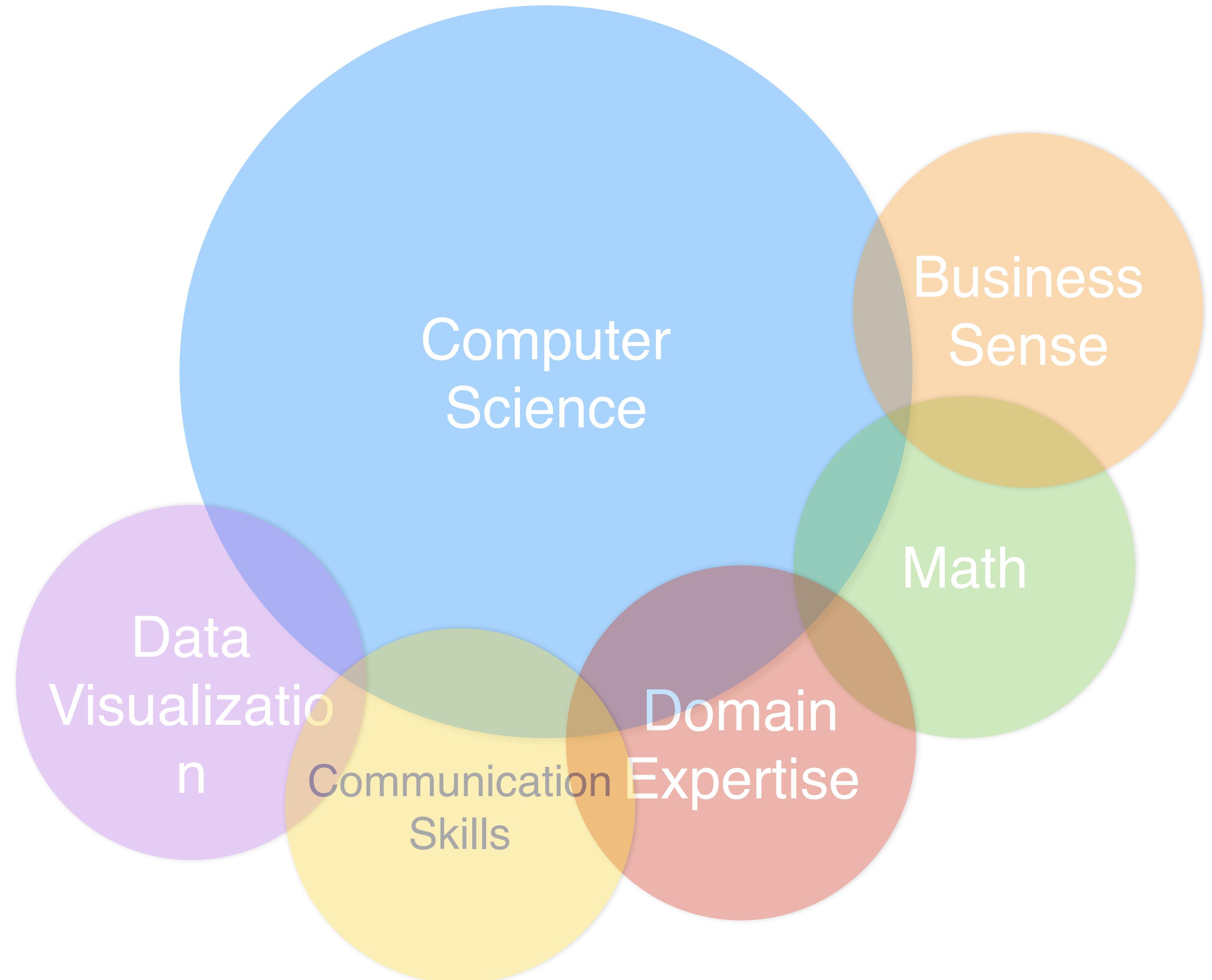
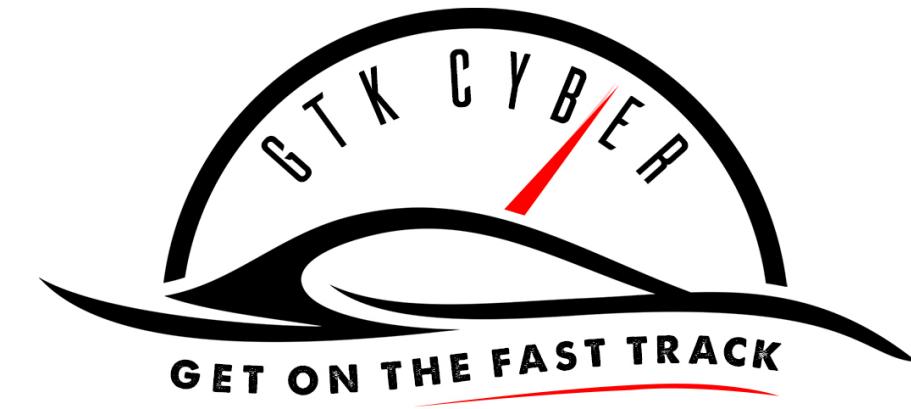
Cost of Insights over Time

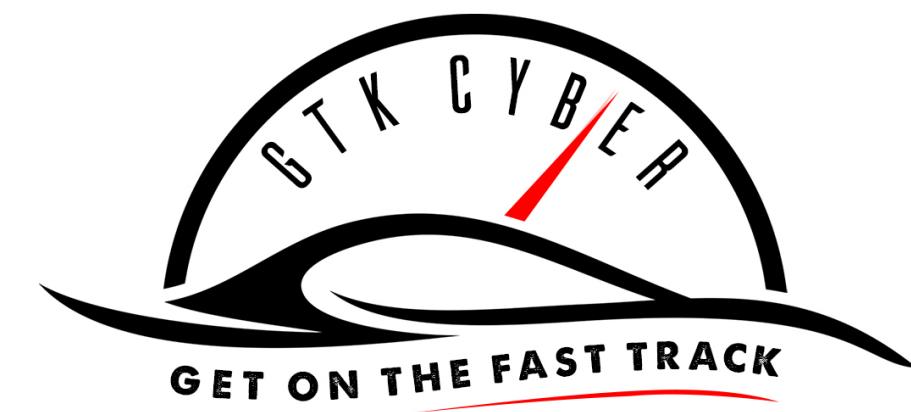




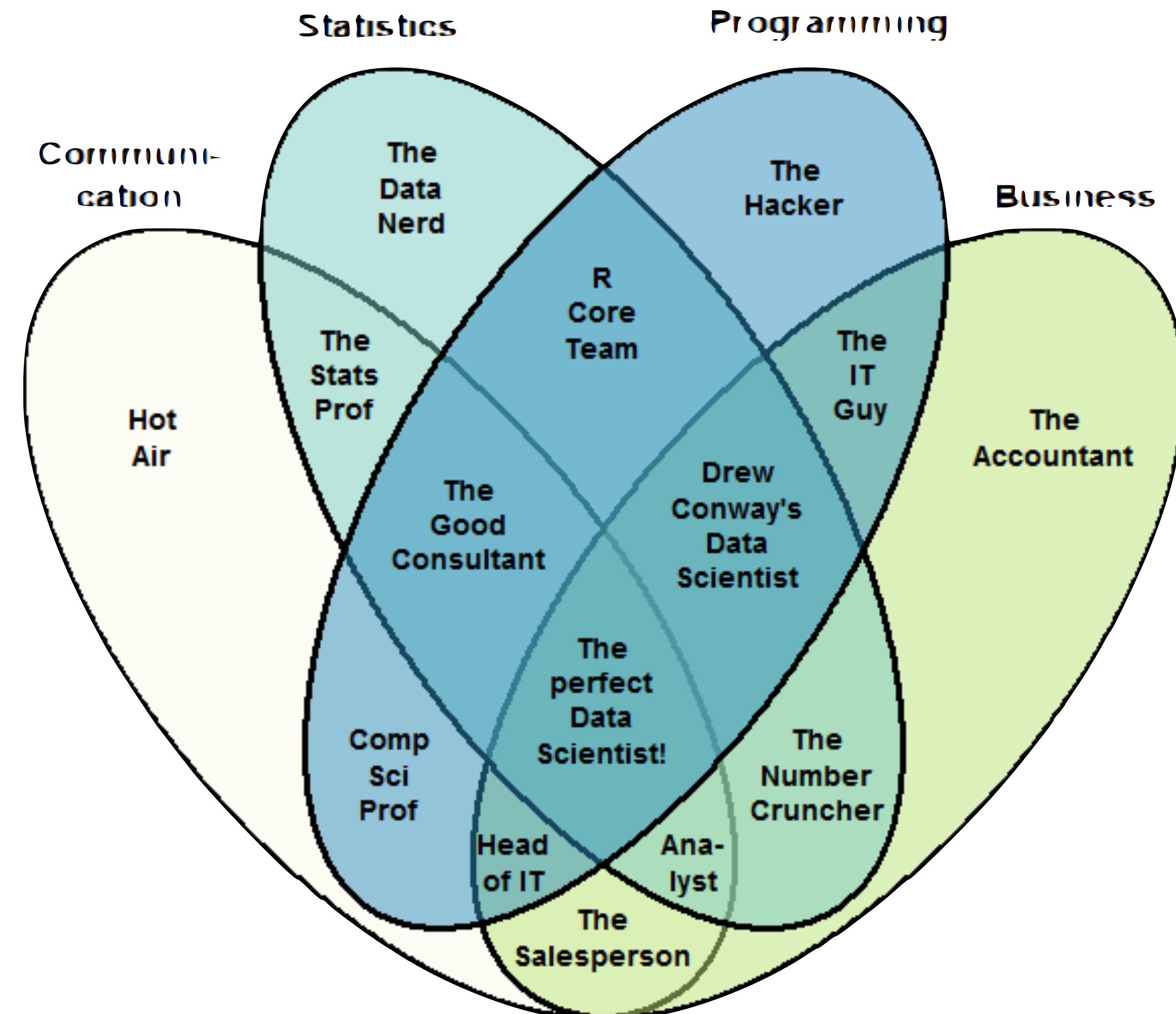
What skills does a data scientist need?

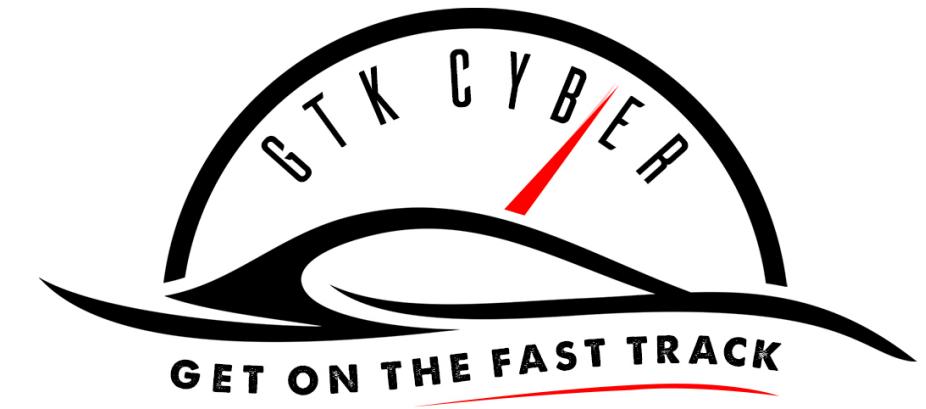




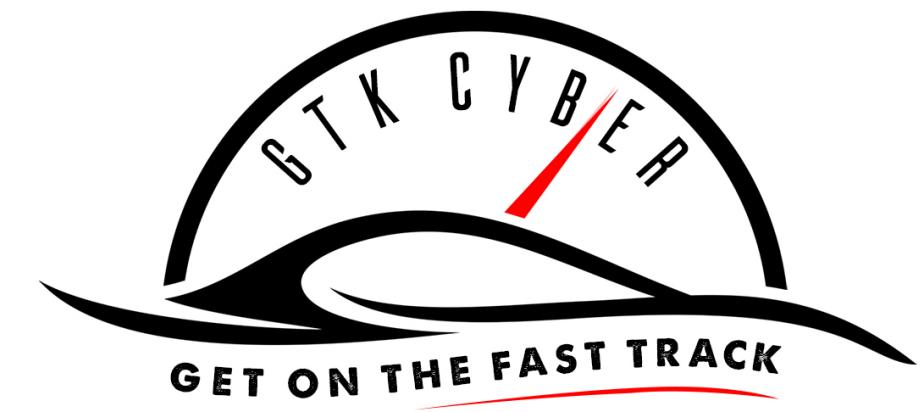


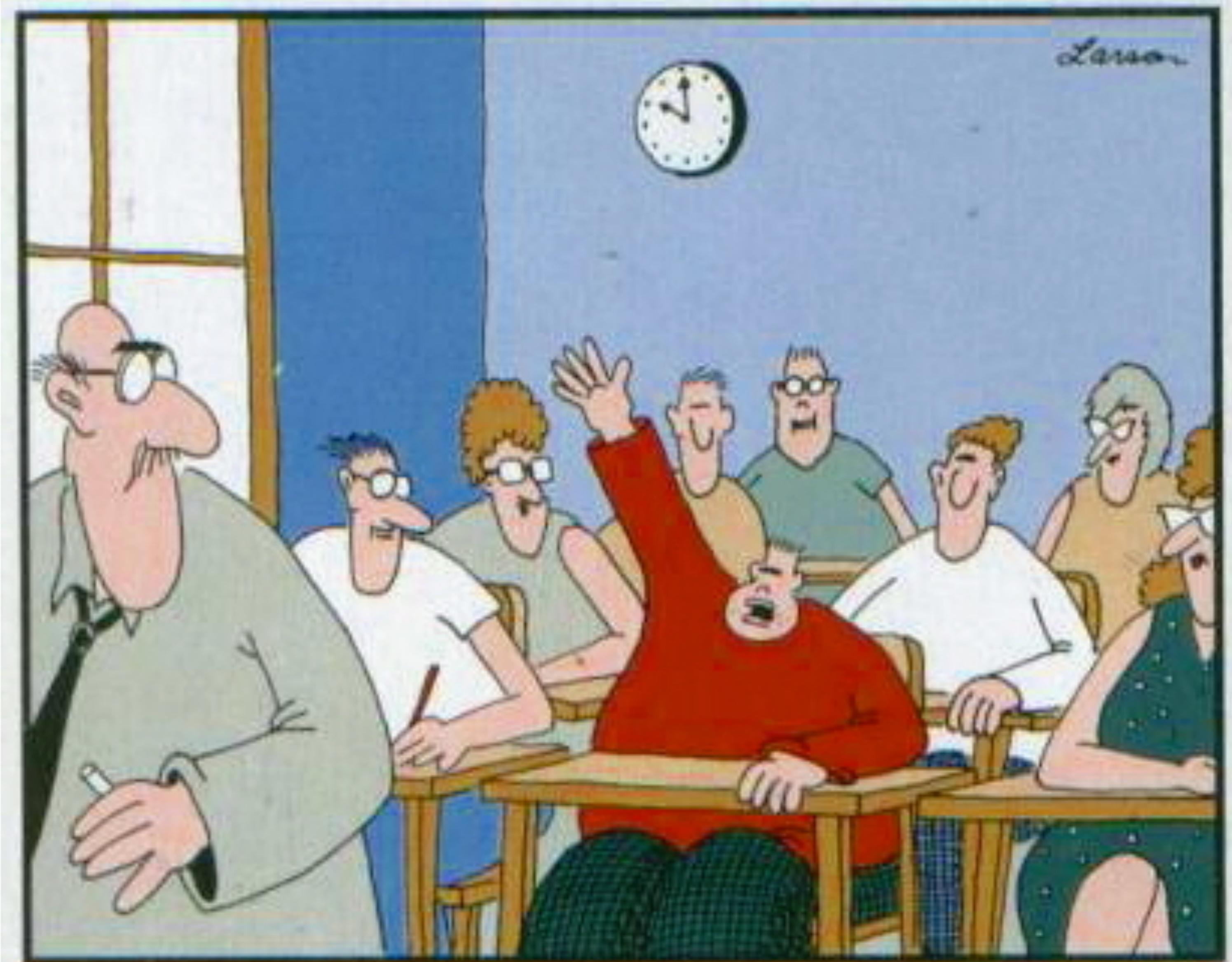
The Data Scientist Venn Diagram



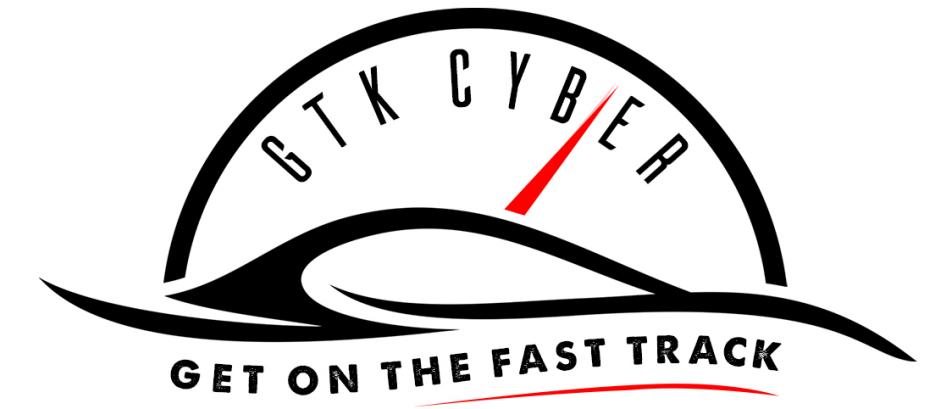


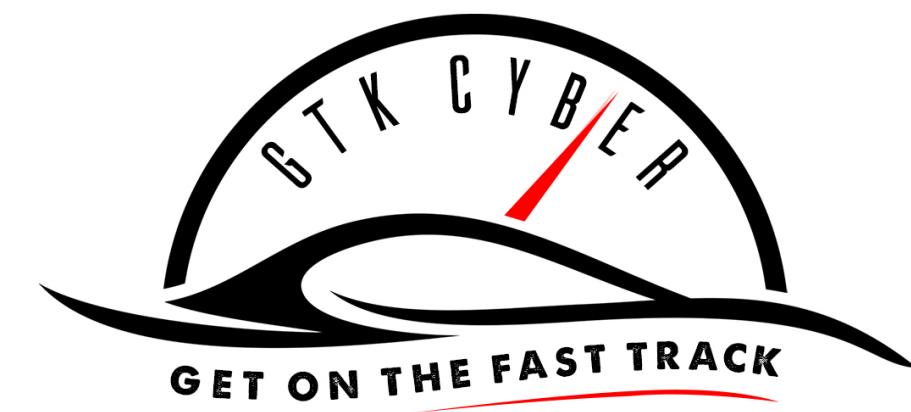
Data Scientists spend
50-90% of their time being...



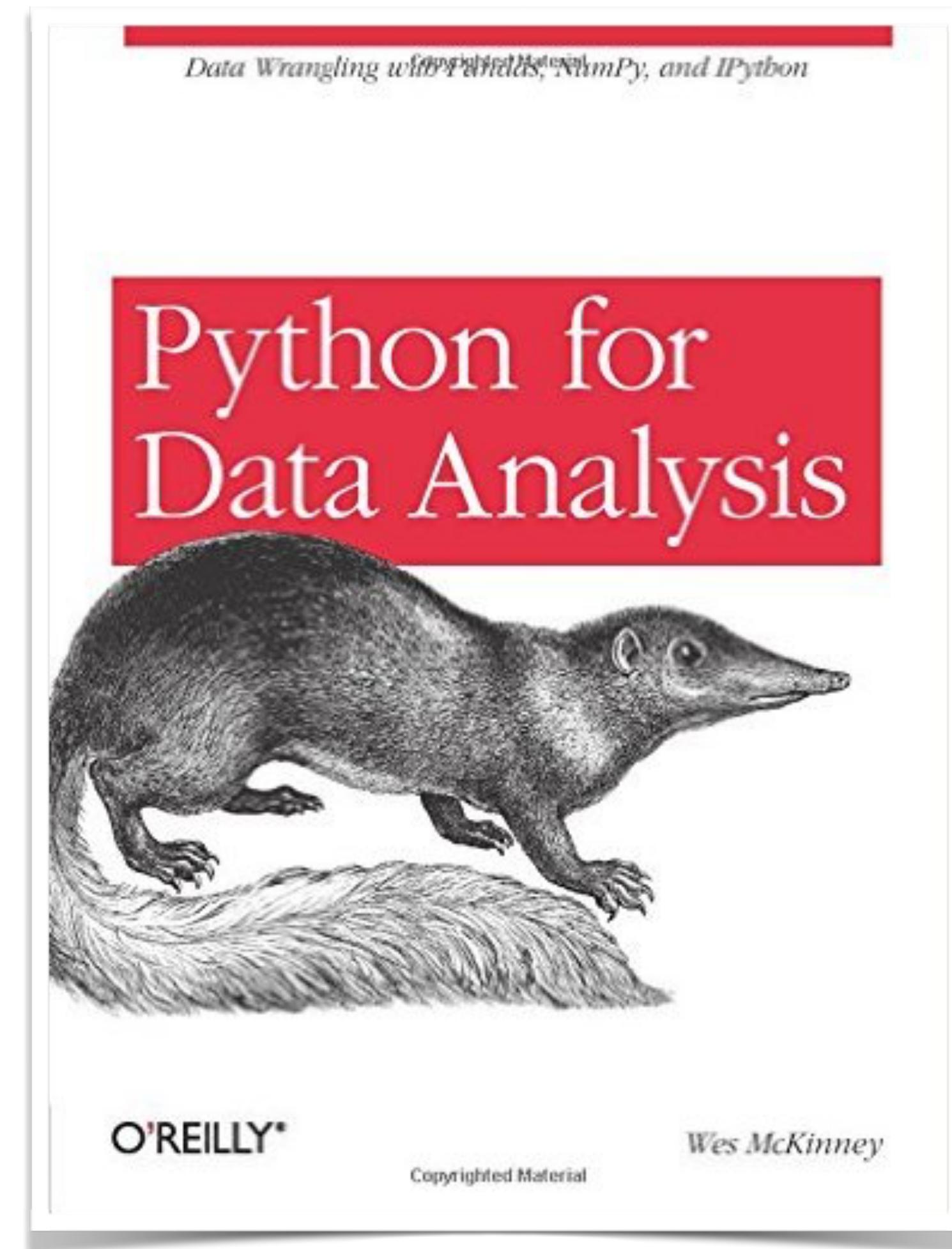


**“Mr. Osborne, may I be excused?
My brain is full.”**



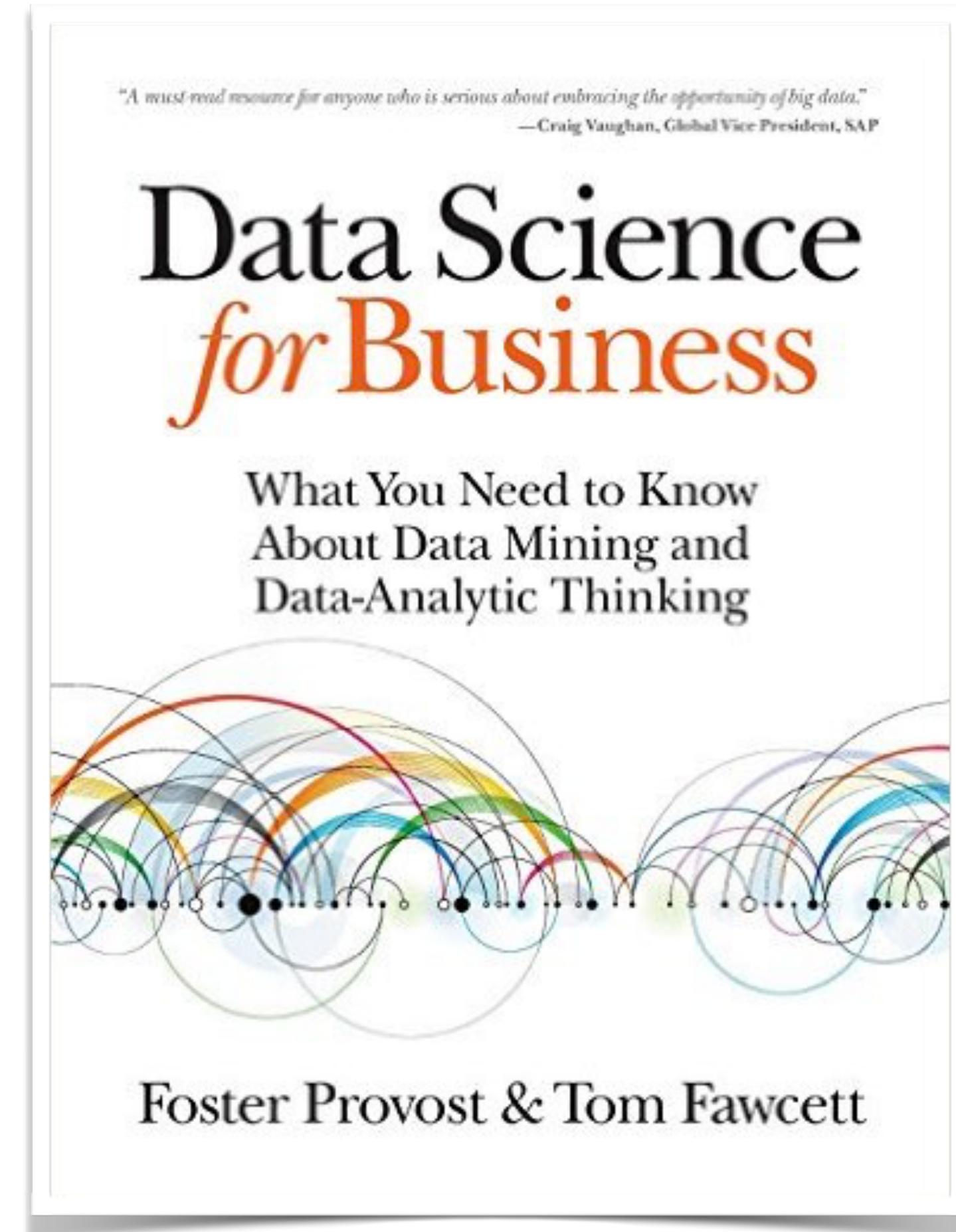


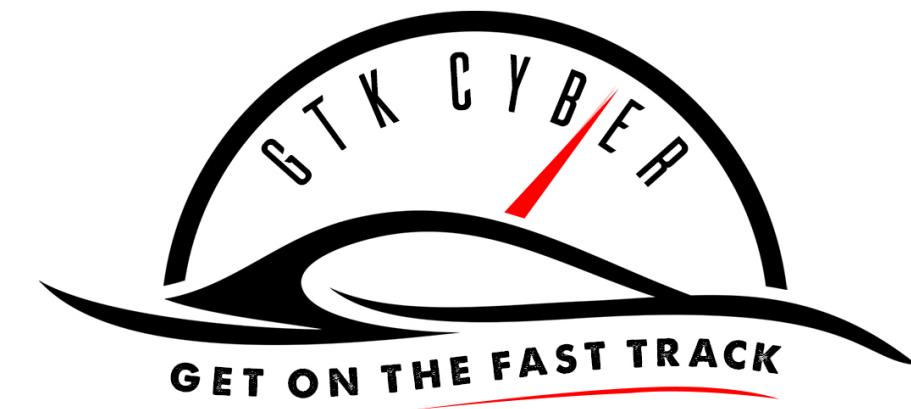
Recommended Reading



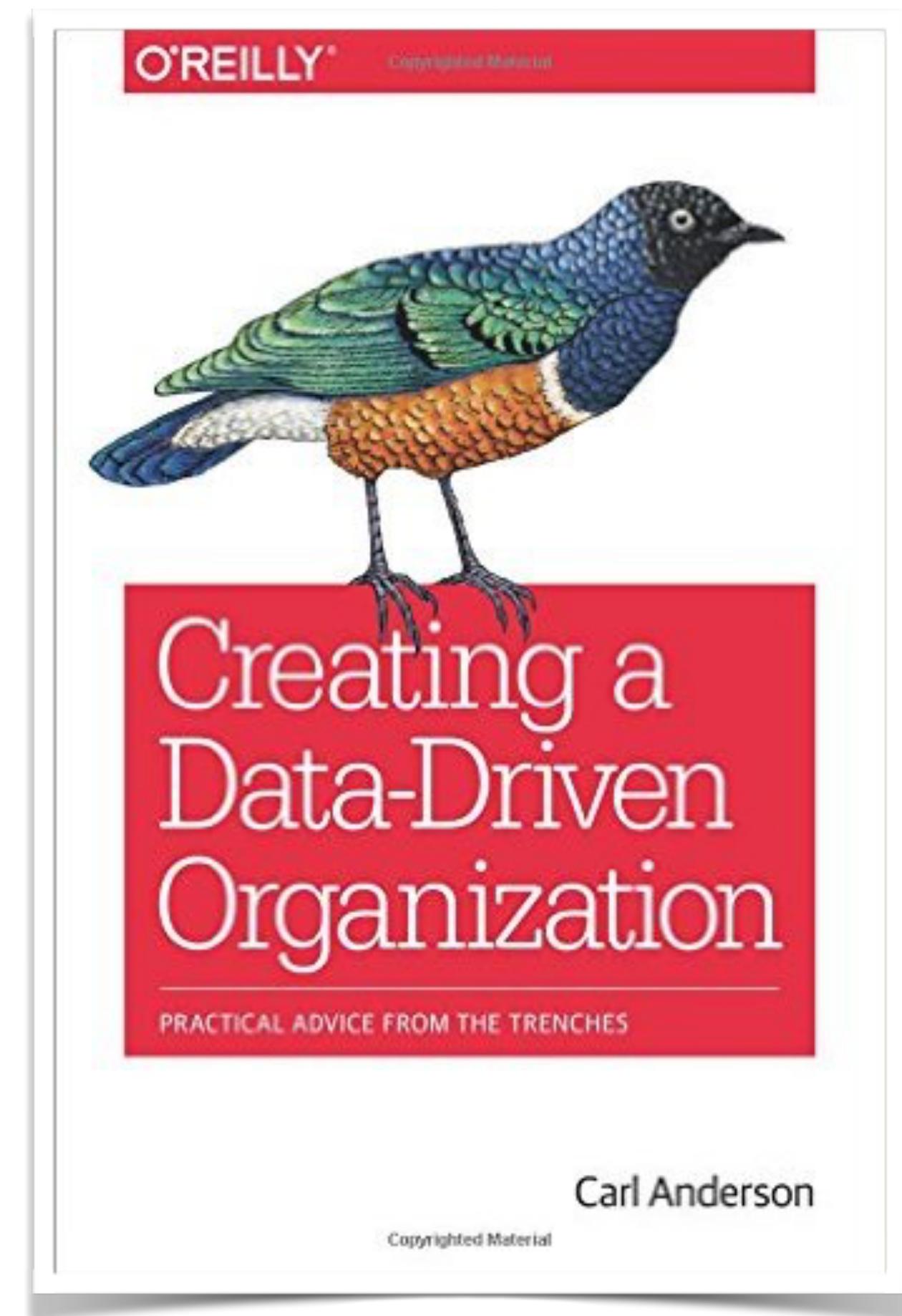


Recommended Reading



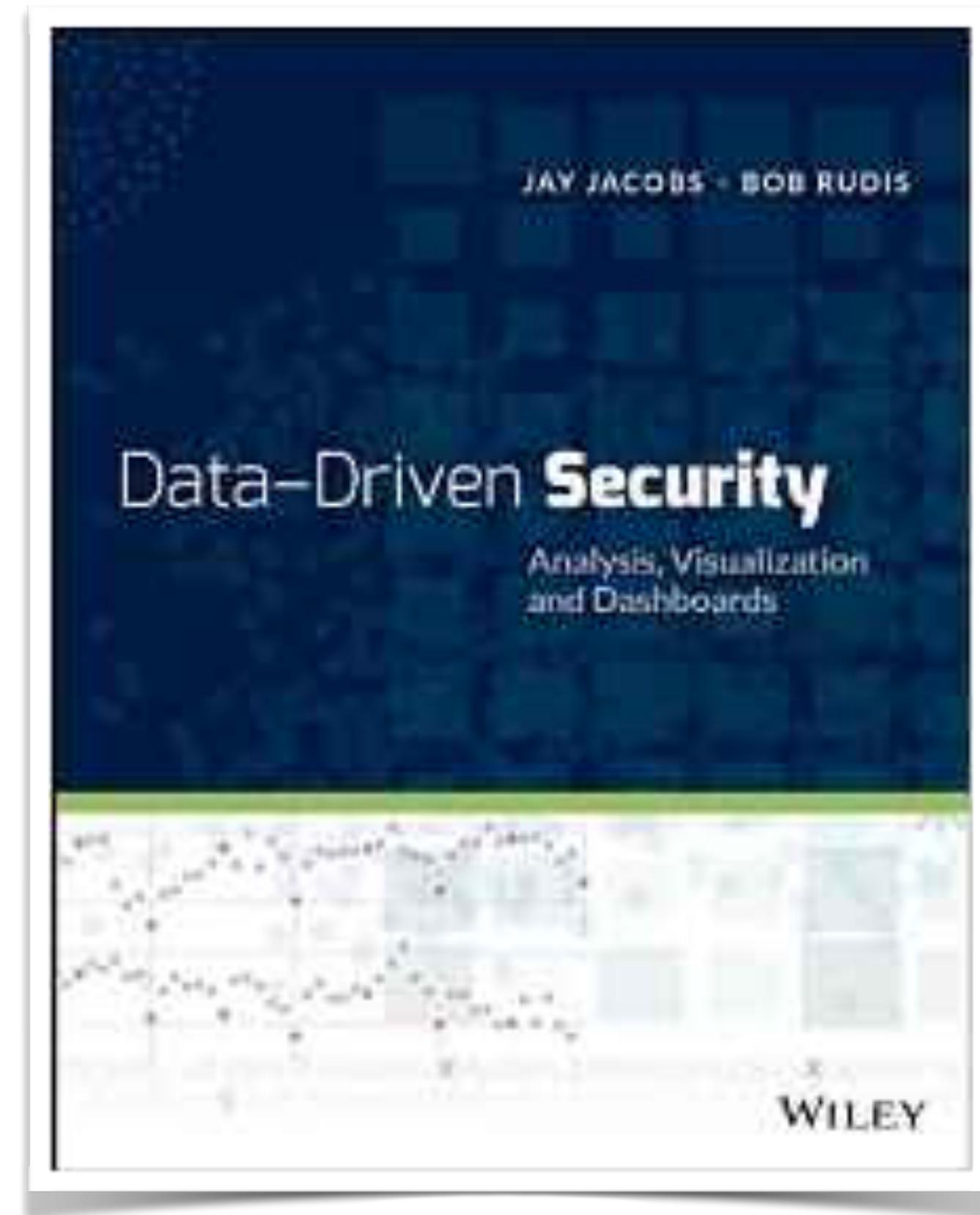


Recommended Reading

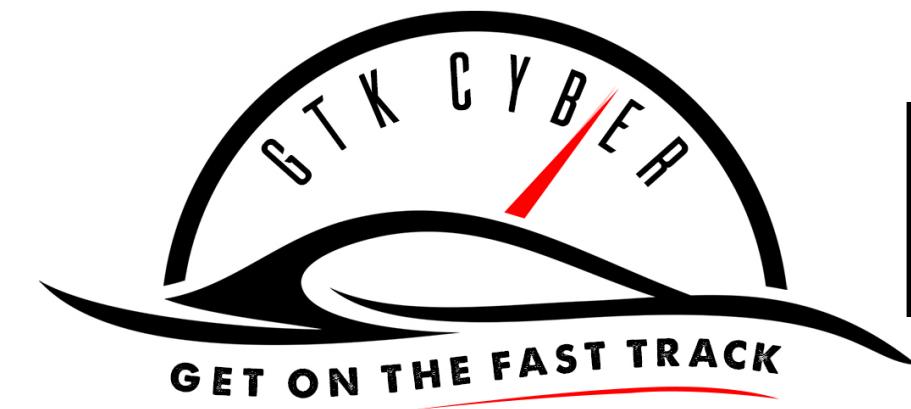




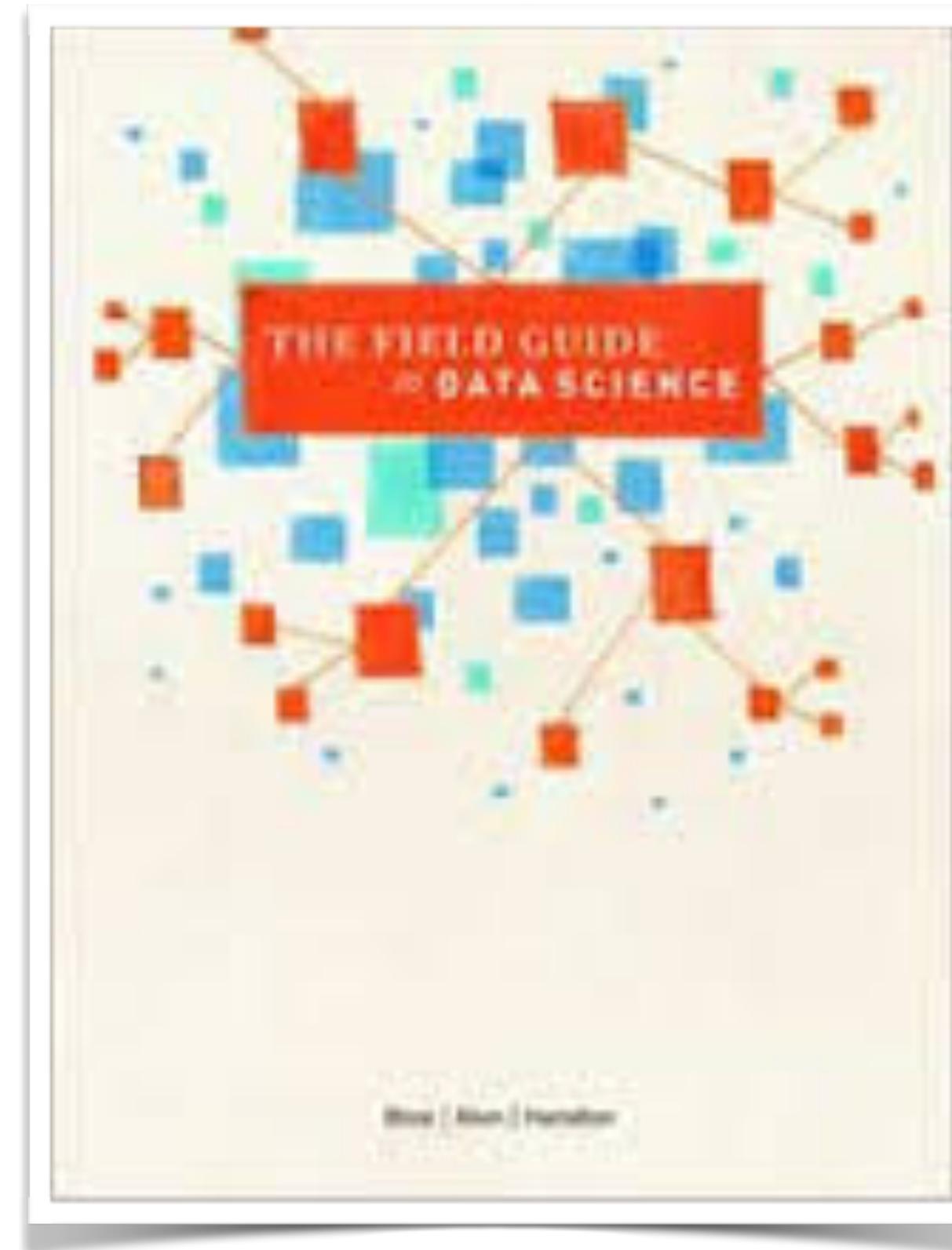
Recommended Reading



<http://datadrivensecurity.info>

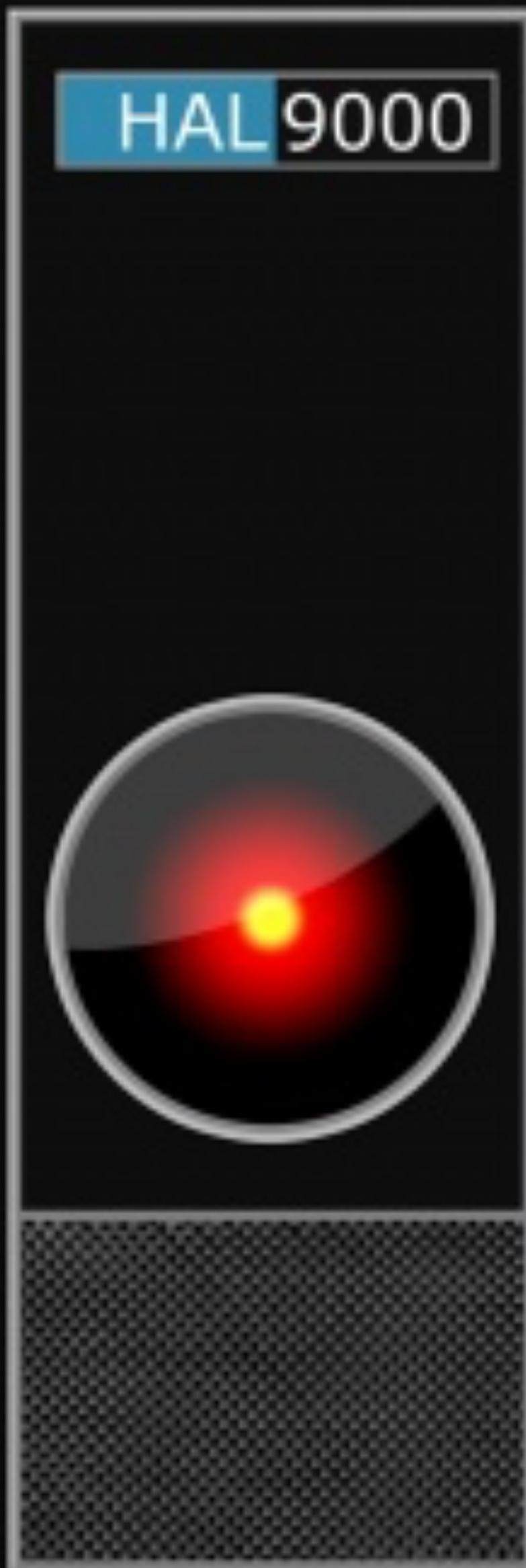


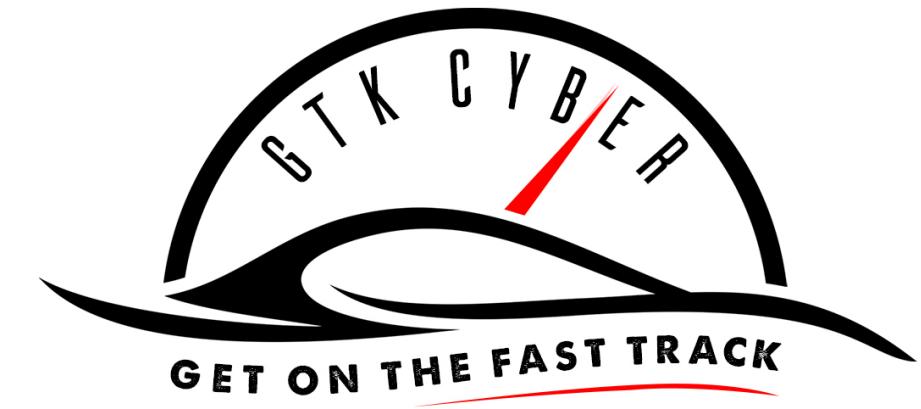
Recommended Reading



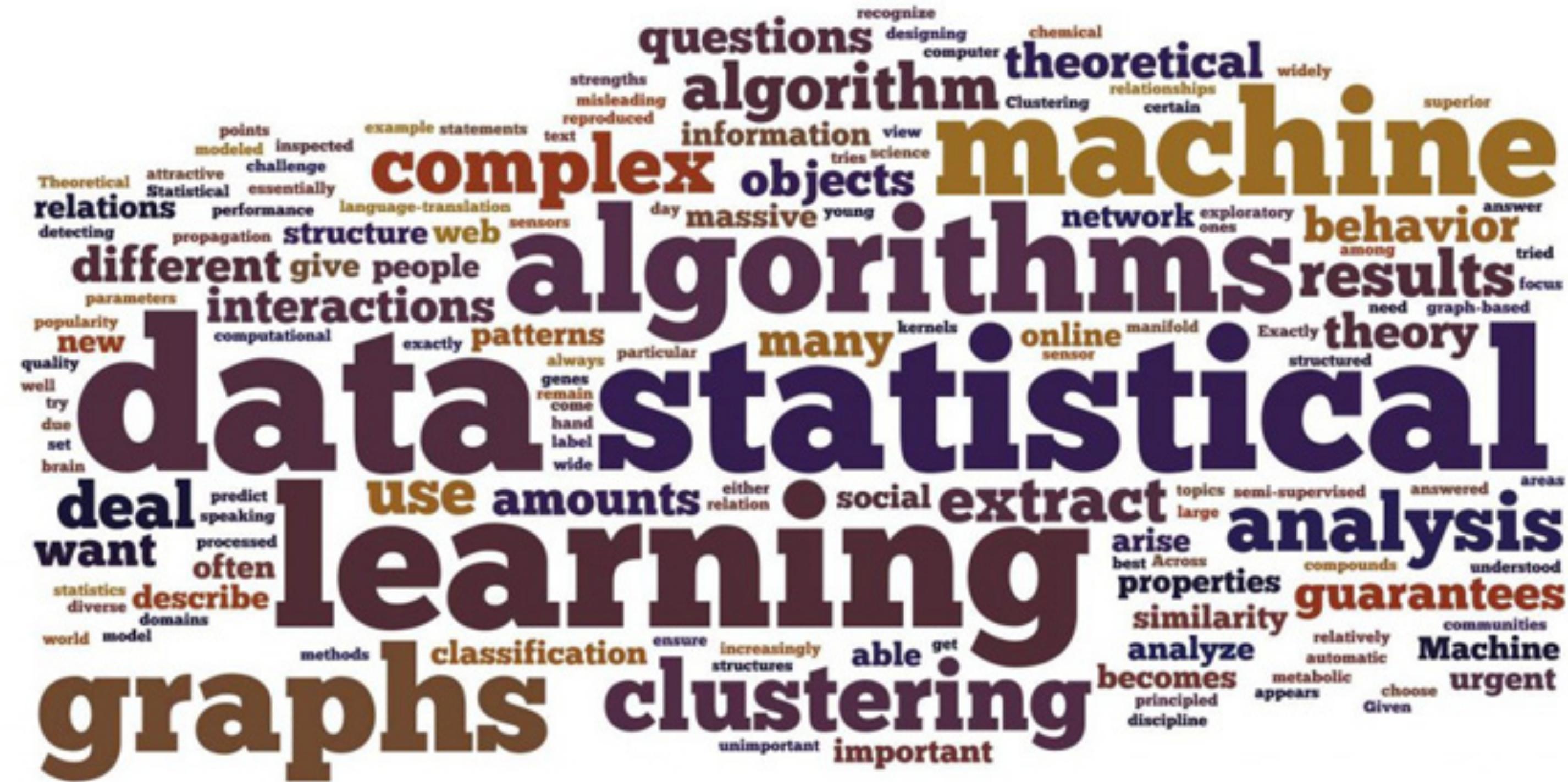
<http://www.boozallen.com/media/file/The-Field-Guide-to-Data-Science.pdf>

Machine Learning



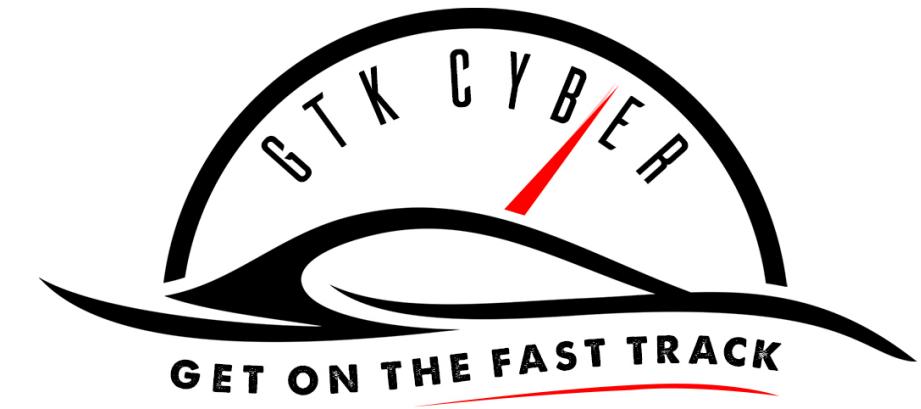


What is Machine Learning (ML) Artificial Intelligence (AI)



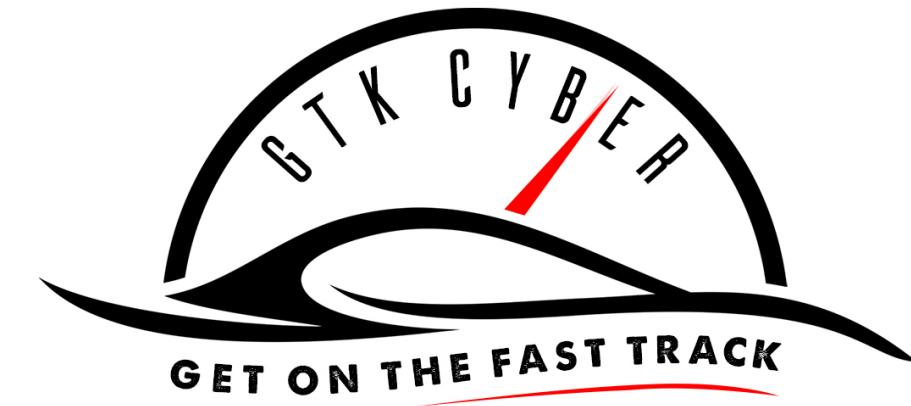
“I have no idea. The latest buzzword? ”

—The guy in the back of the class



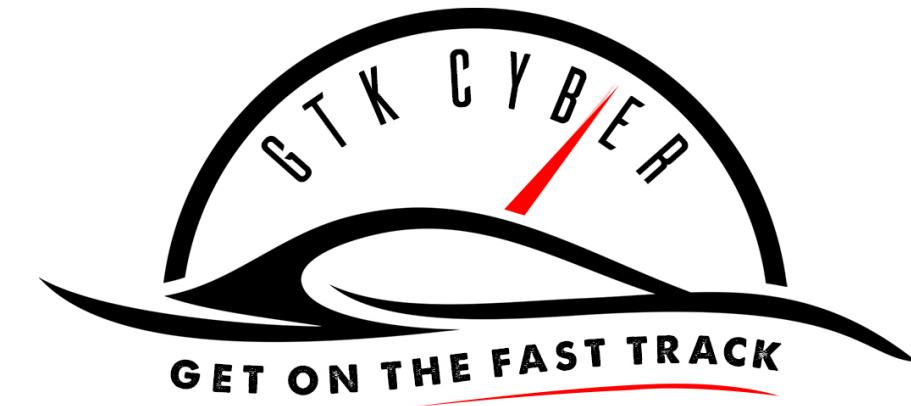
“Machine learning is the science of getting computers to **act without being explicitly programmed.** ”

– <https://www.coursera.org/course/ml>



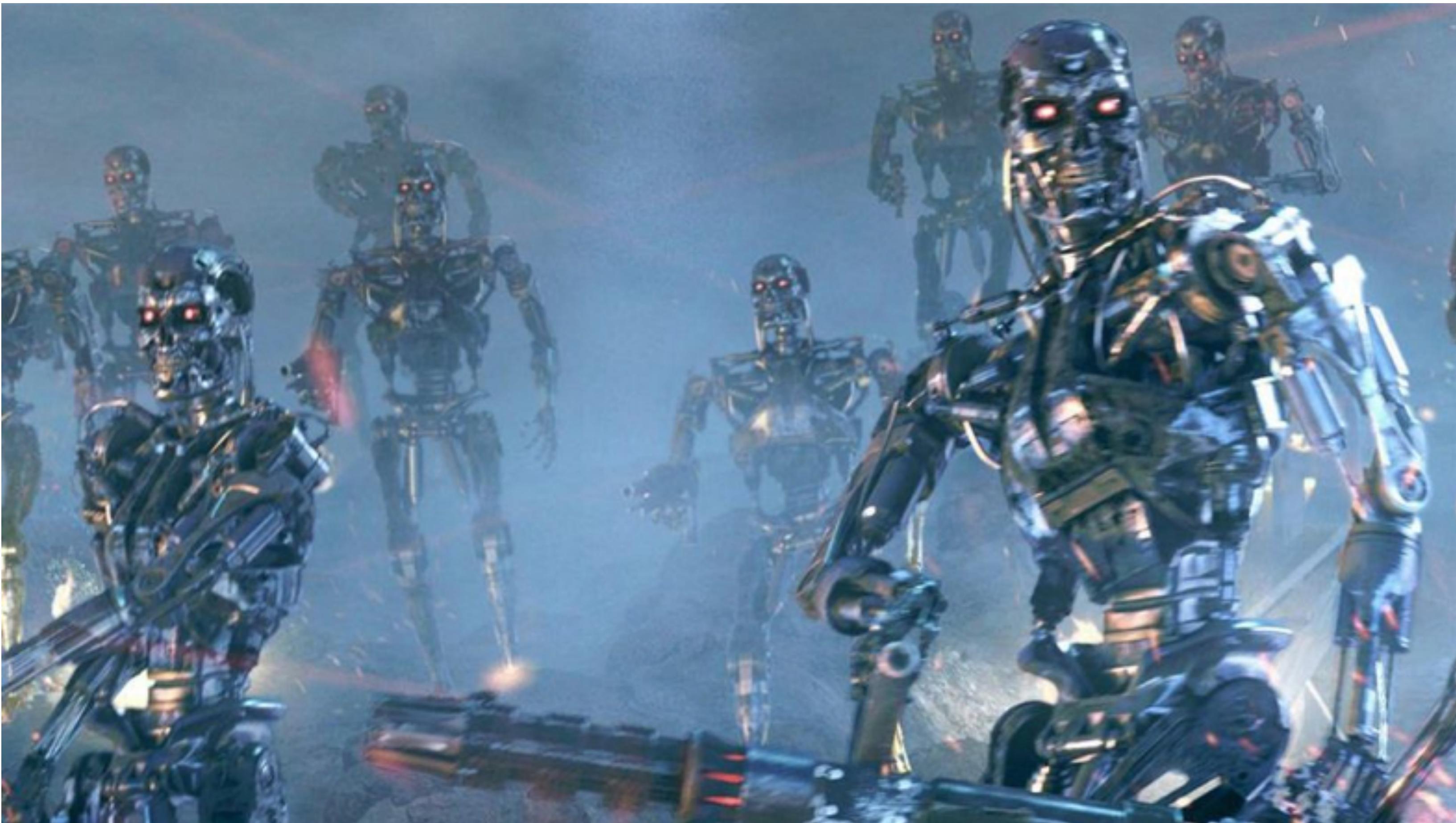
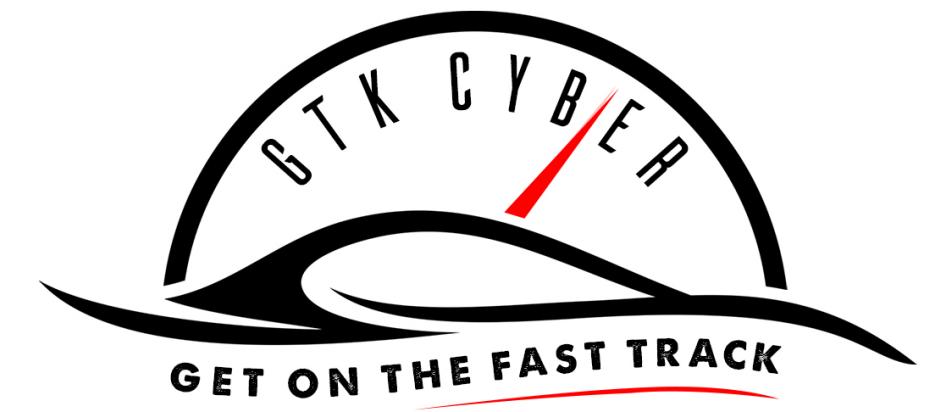
“A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.”

–Tom Mitchell, Carnegie Mellon University



“Machine learning explores the construction and study of algorithms that can learn from and **make predictions on data**. Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions, **rather than following strictly static program instructions**.”

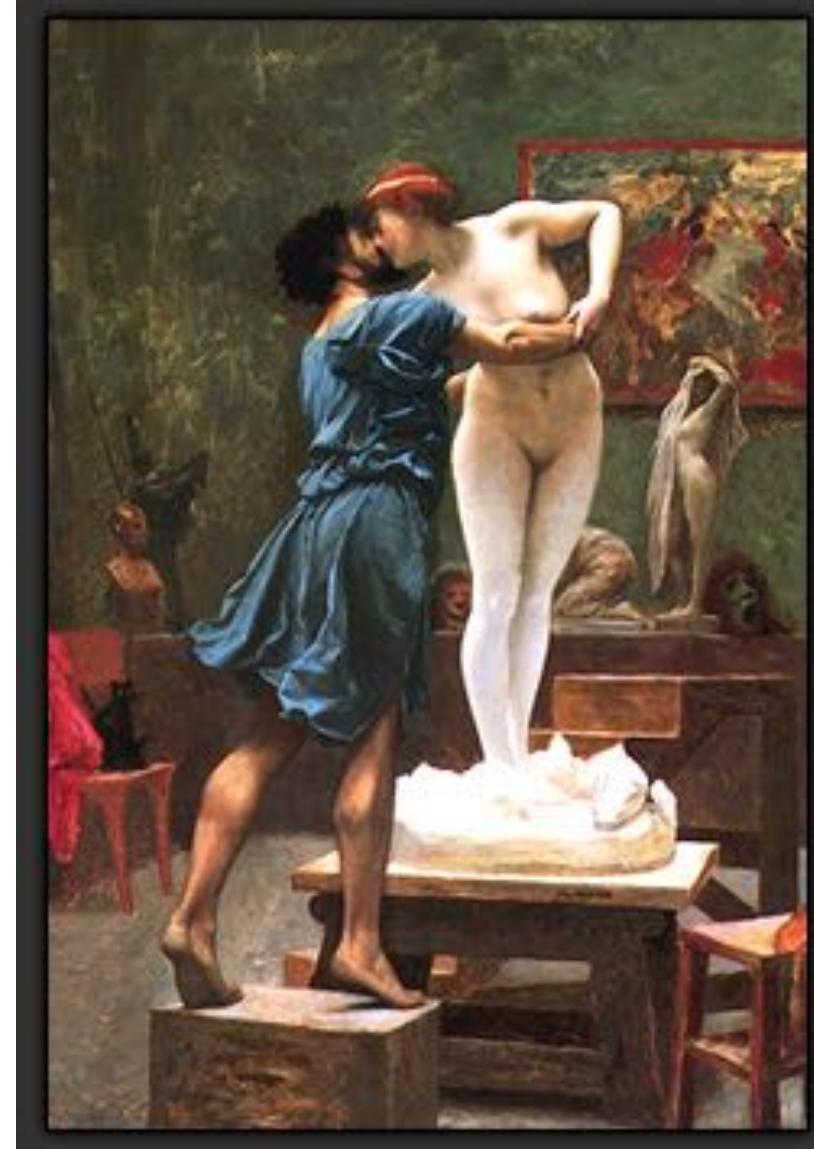
—https://en.wikipedia.org/wiki/Machine_learning



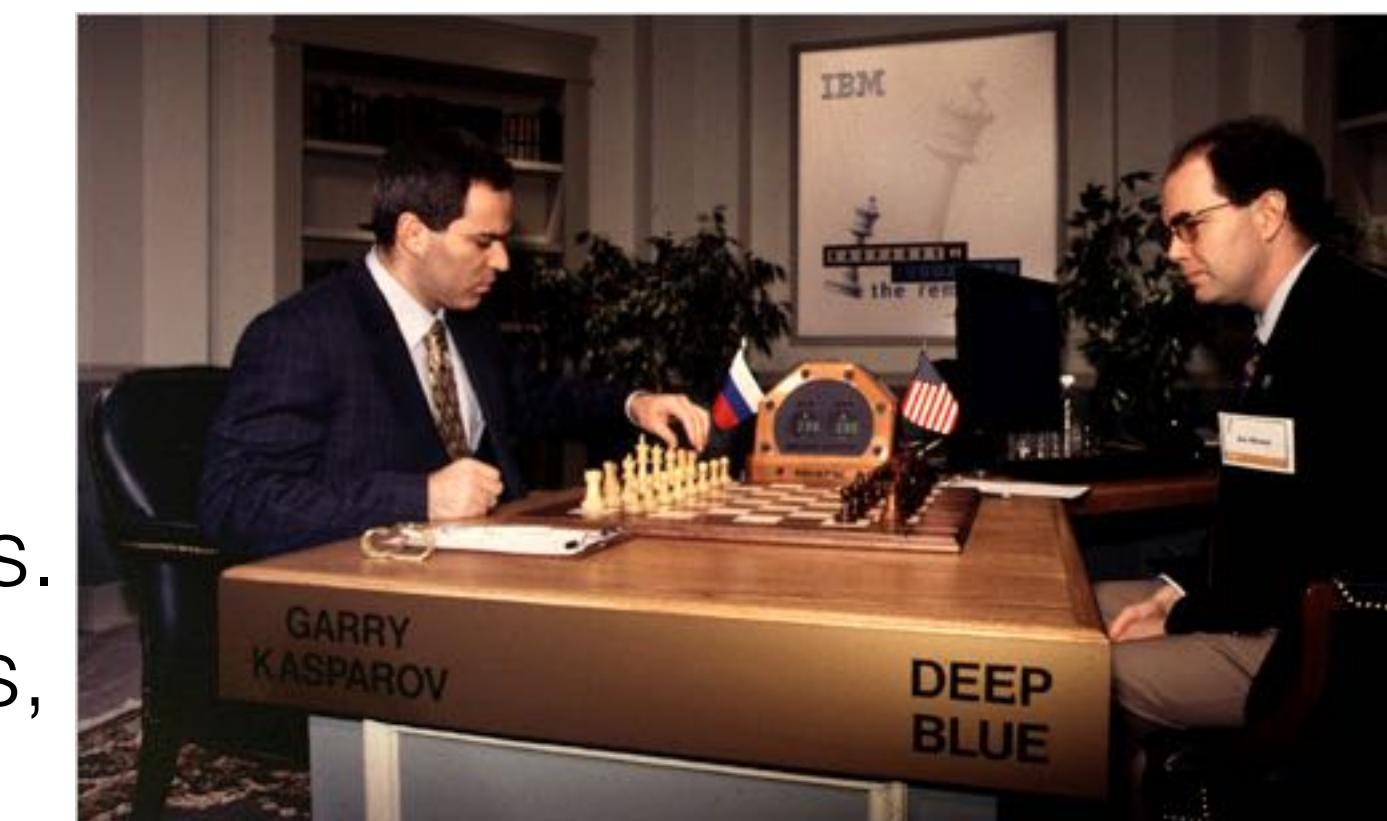
What it is not



History of AI and ML



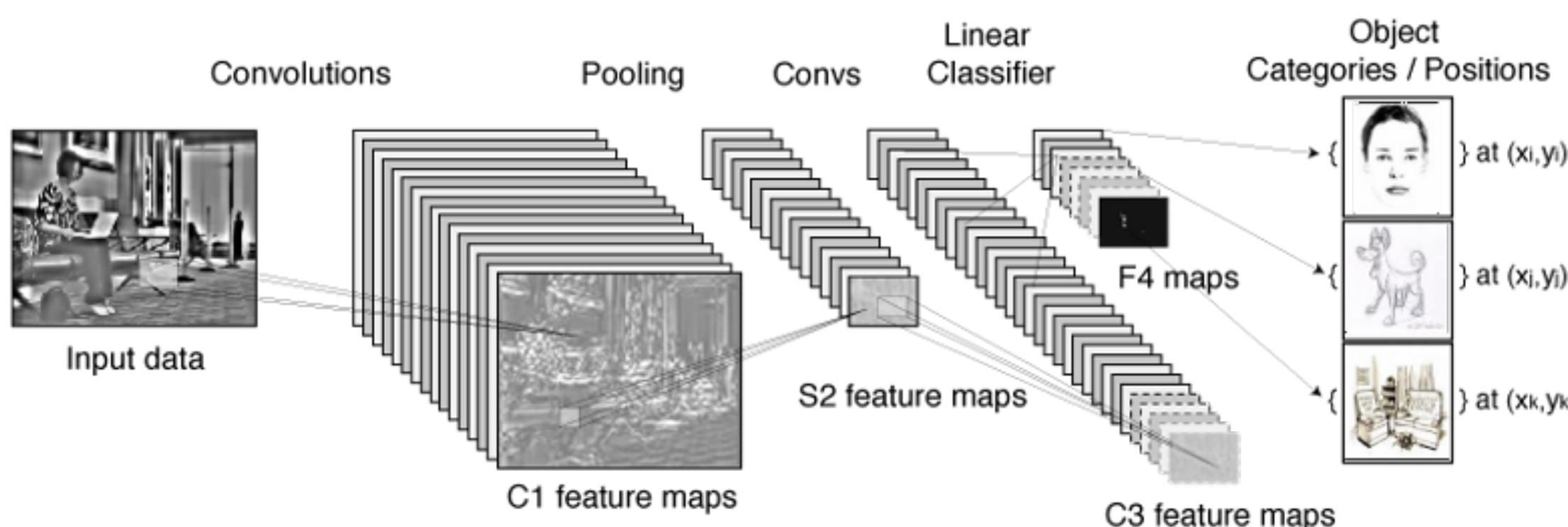
- Mythical figure Pygmalion: Breath of life in a statue (ancient Greece)
- Turing Test 1950s by Alan Turing: Test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human.
- Early AI: solve problems that are difficult for humans, but straightforward for computers (e.g. formal mathematical rules) IBM Deep Blue chess-playing system defeated world champion Garry Kasparov 1997.
- Real Challenge: Intuitive, easy problems for humans. They are hard to describe formally (e.g recognizing faces, words).

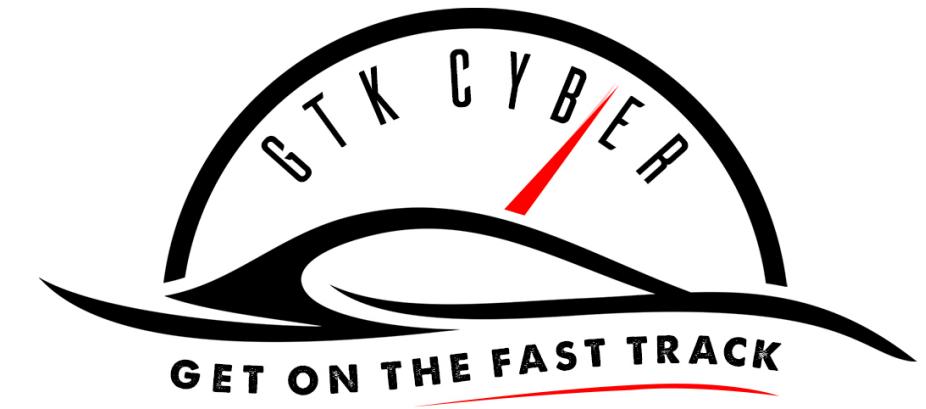




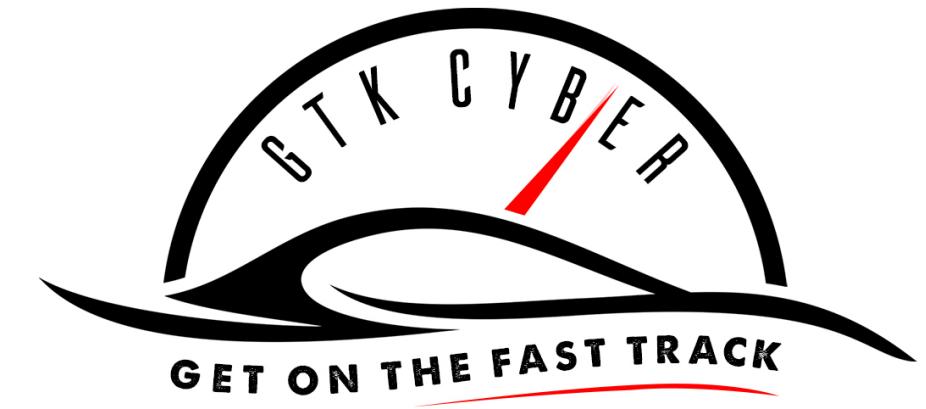
Path to Artificial Intelligence?

- Hard-coding knowledge failed (e.g. Cyc 1989).
- **Gathering knowledge from experience** eliminates the need to formally specify all knowledge the computer needs.
- AI needs to acquire own knowledge by **extracting pattern from raw data.**
- Success depends heavily on **representation of data, aka features.**
- How to choose features?
- Is Deep Learning the answer?

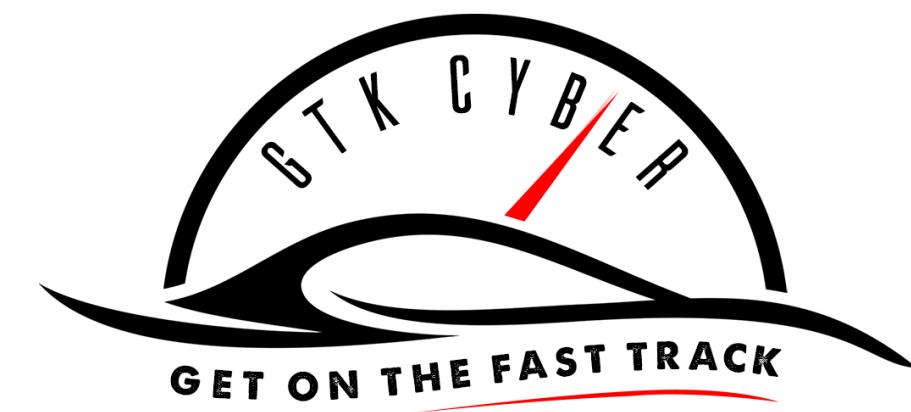




Automating Decisions

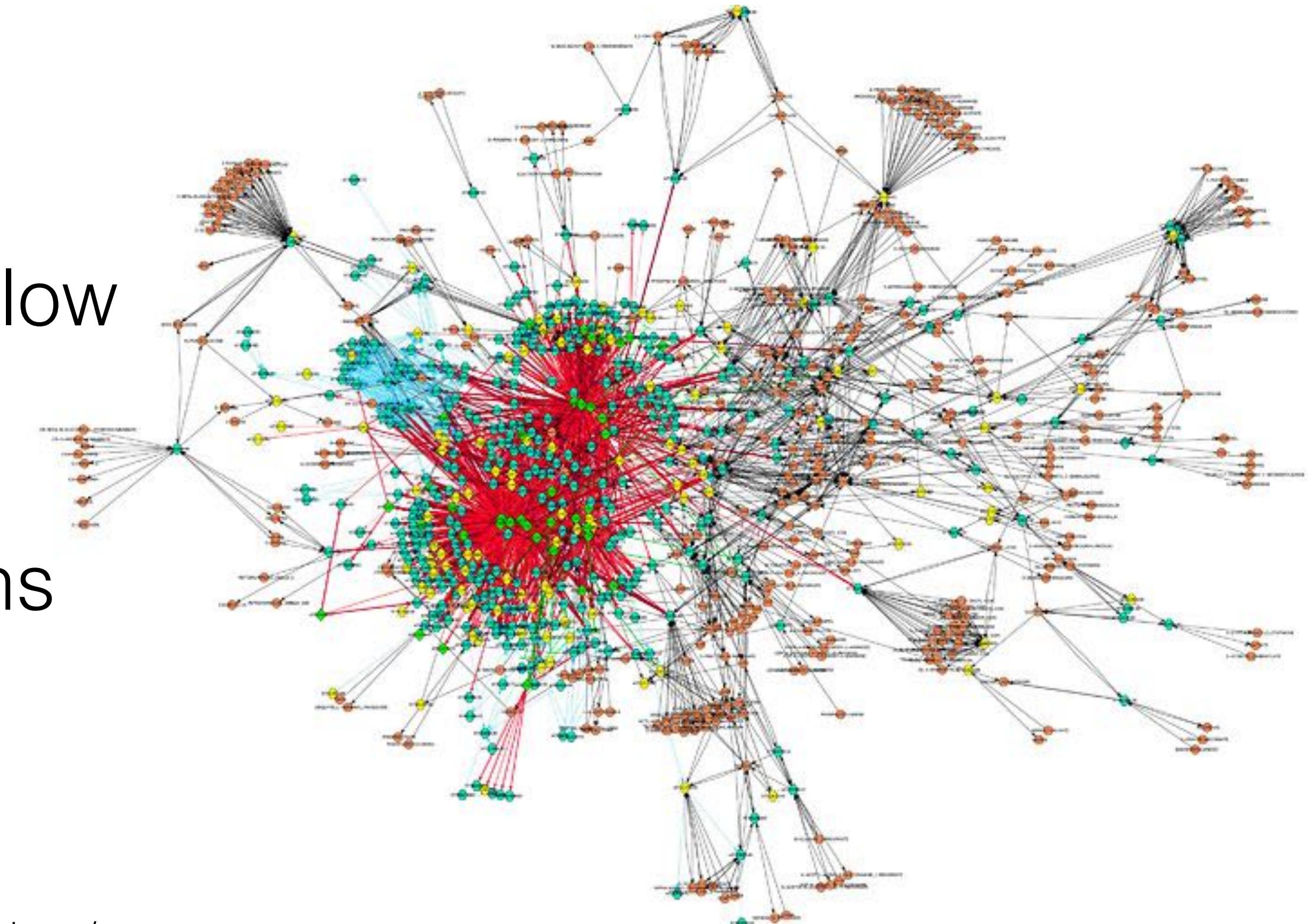


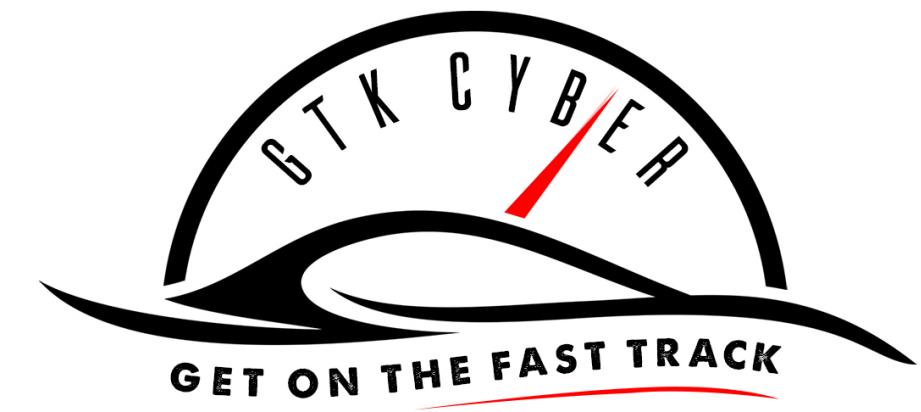
Applications to Cyber



Network-Based Intrusion Detection

- Derive Features from Network Traffic
Captures “pcap” at packet level or NetFlow level (tools: tshark, tcpdump, bro...)
- Example Features based on header information: 2s-windowing of connections
> duration, protocol, src and dst bytes, service.
- Get data sets: <http://www.netresec.com/?page=PcapFiles>, <https://maccdc.org/>, <http://www.westpoint.edu/crc/SitePages/DataSets.aspx> <http://www.unb.ca/cic/research/datasets/>,

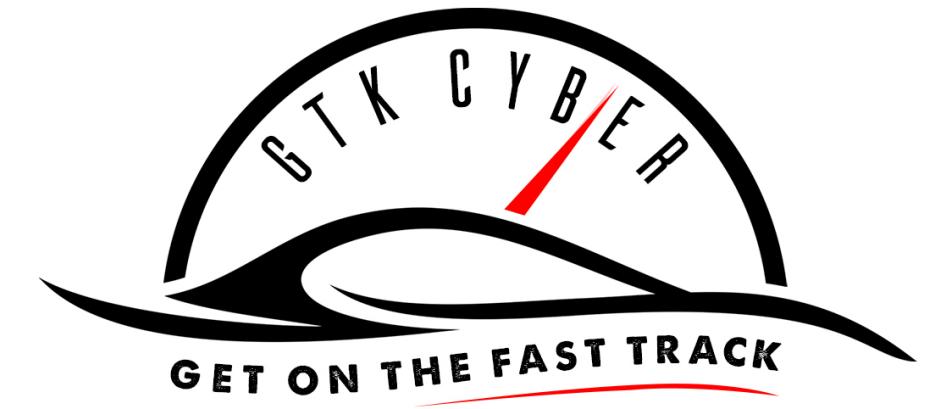




Malware Detection/Classification

- Derive Features from Binary Content and metadata manifest (function calls, string obtained from IDA Disassembler)
- Example Features: opcode count (n-grams), segment count, asm pixel intensity, n-gramming of bytes, function name.
- Featureless Deep Learning with word2vec embedding
- Get open source malware samples: Vx Heaven, Virus Share, Maltrieve, Open Malware



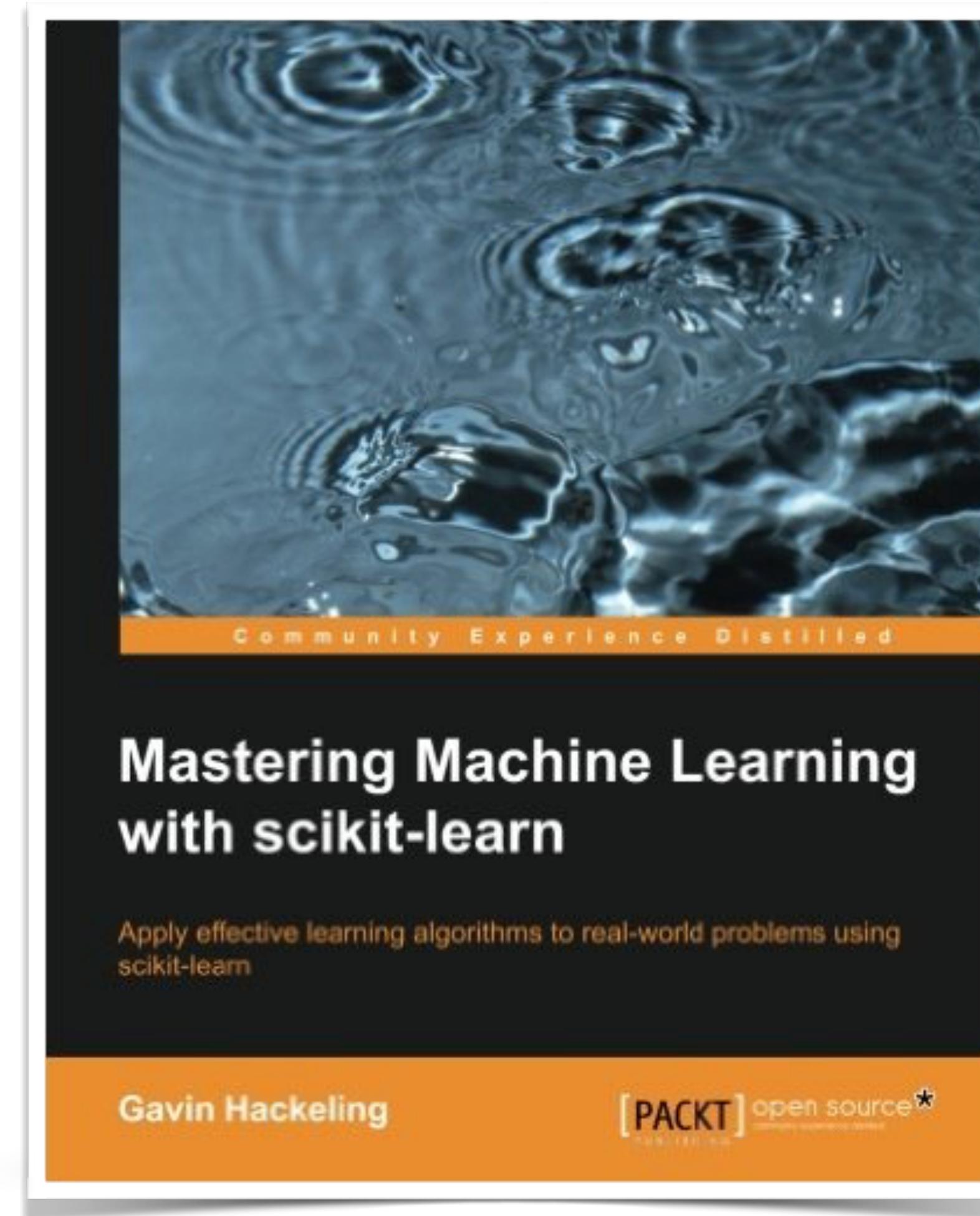


More Applications

- Domain Generation Algorithm (DGA) Detection
- **Malicious URL Detection**
- Network Traffic: Beaconing Detection
- Detection of new classes of malware
- General Network Traffic Anomaly Detection
- Log Analysis - Anomaly Detection
- Phishing Detection
- Identifying SQL Injection
- Identifying XSS cross-site scripting
- DOS/DDOS Detection
- Authentication

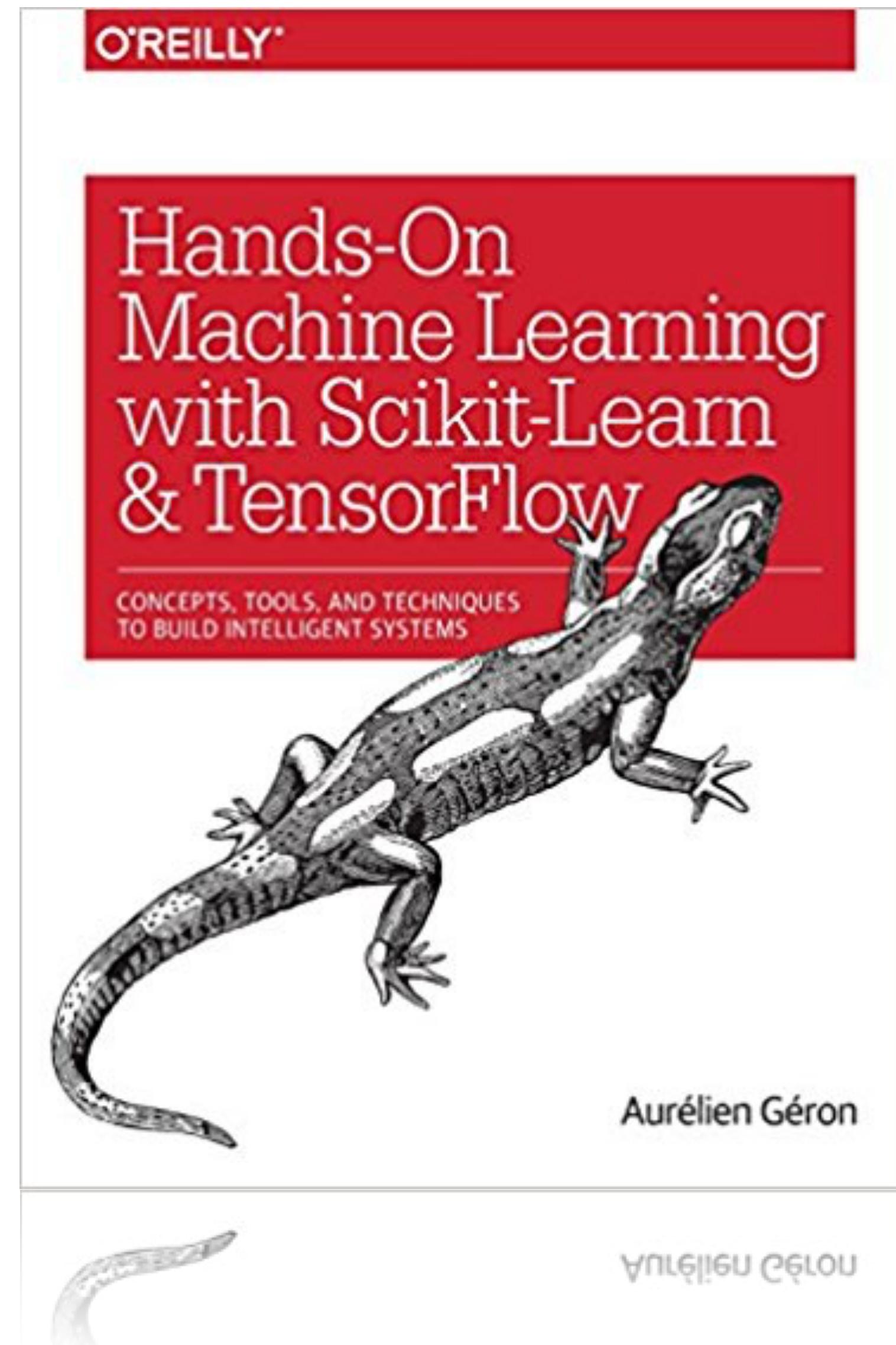


Recommended Reading





Recommended Reading

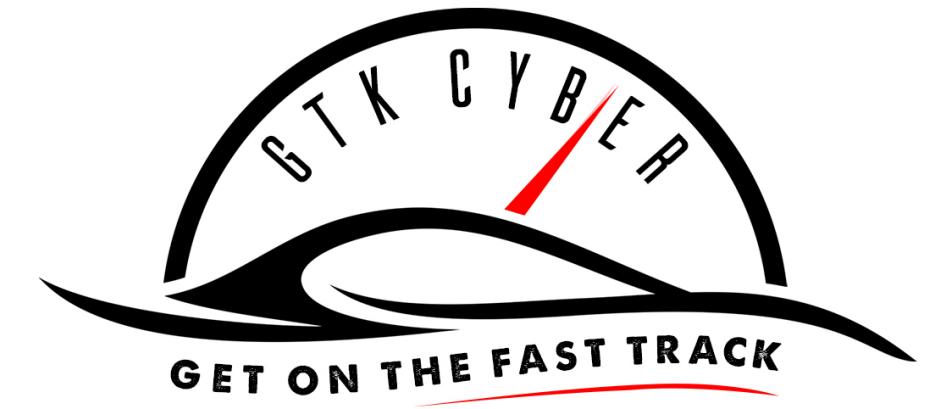


Exploratory Data Analysis

Part 1: One Dimensional Data



What is it?



Vectorized Data Structures let you
perform operations on your data
all at once



```
for x in range(0, len( data )):  
    data[ x ] = data[ x ] + 1
```

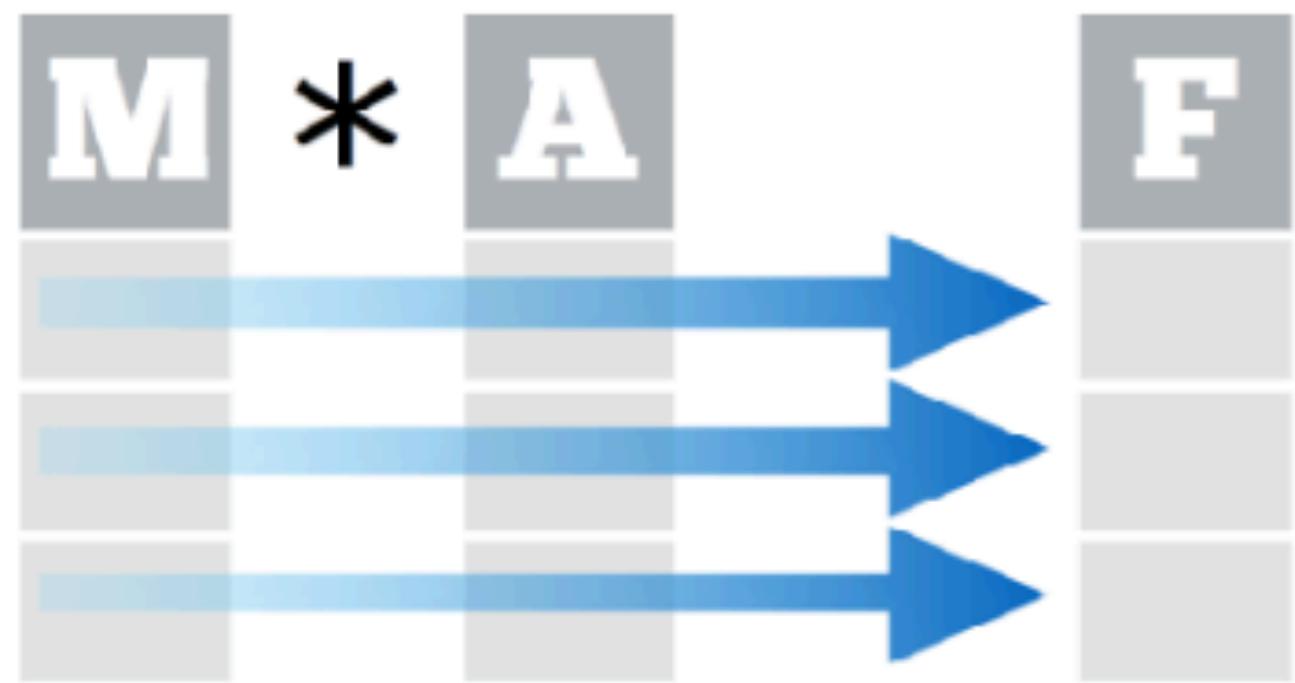


```
for x in range(0, len( data )):  
    data[ x ] = data[ x ] + 1
```

odds = evens + 1



$$F = M * A$$



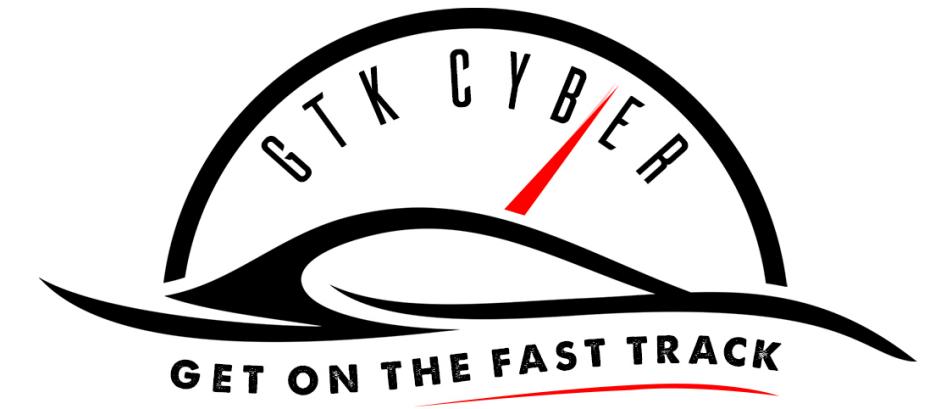
Each **variable** is saved
in its own **column**



&



Each **observation** is
saved in its own **row**



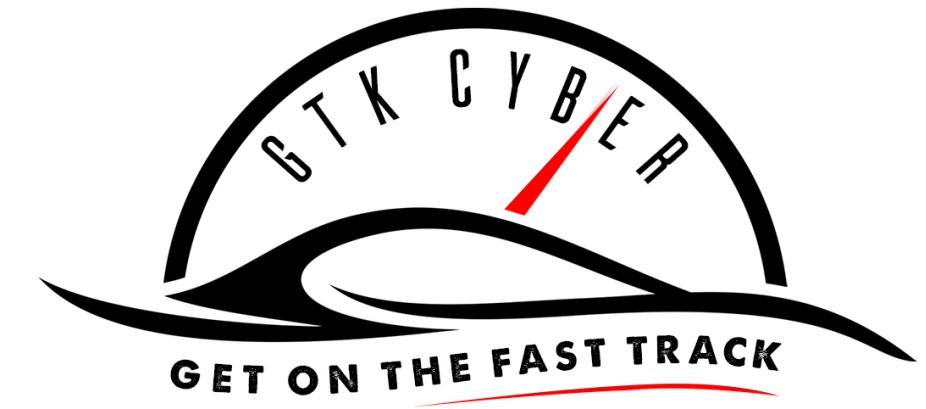
No loops



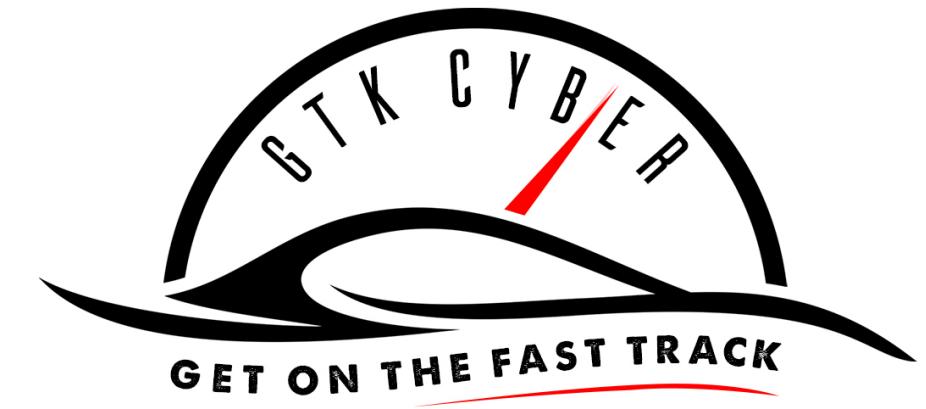




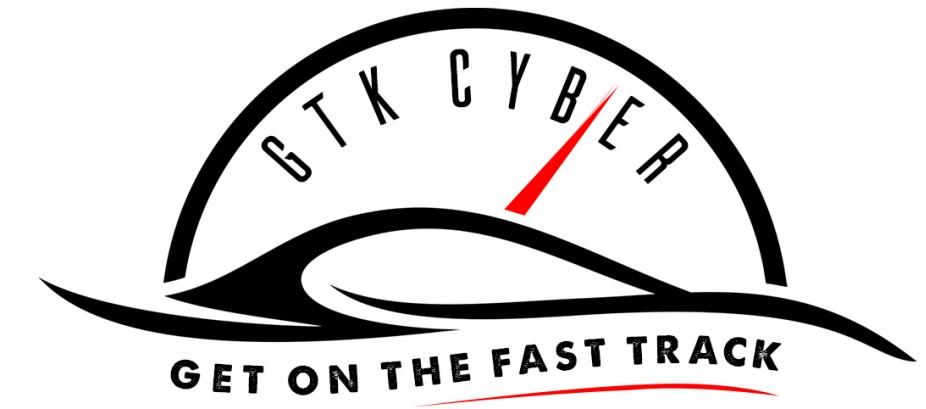
Dimensions	Name	Description
1	Series	Indexed 1 dimensional data structure
1	Timeseries	Series using timestamps as an index
2	DataFrame	A two dimensional table
3	Panel	A three dimensional mutable data structure



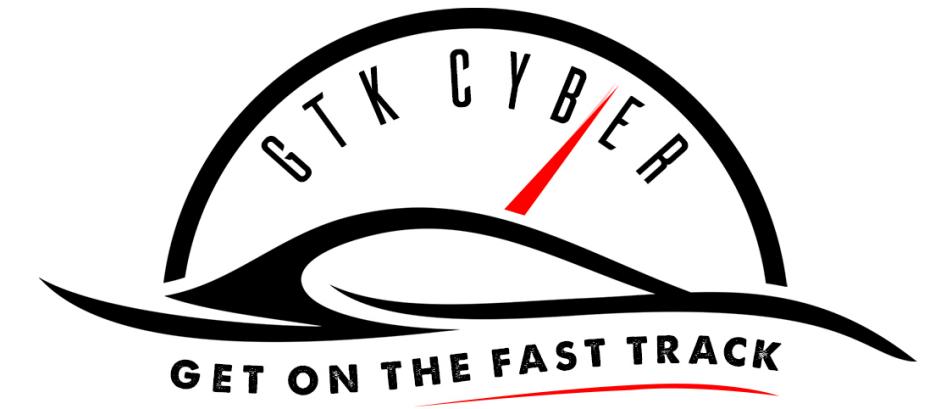
The Series Object



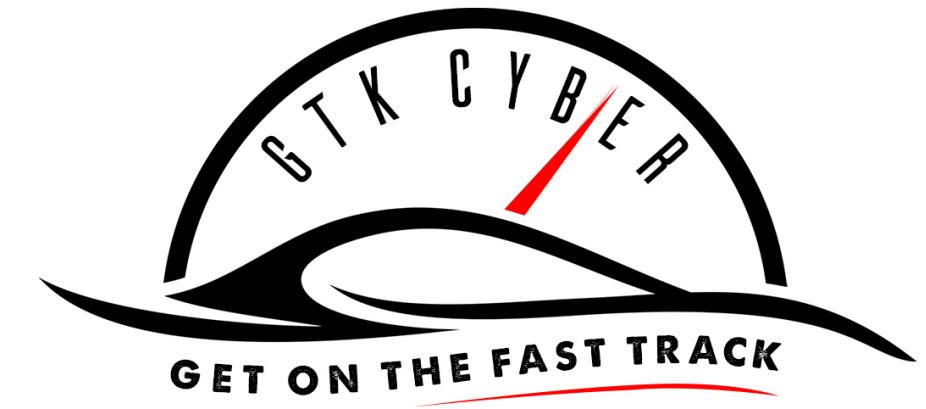
A **Series** is like a list



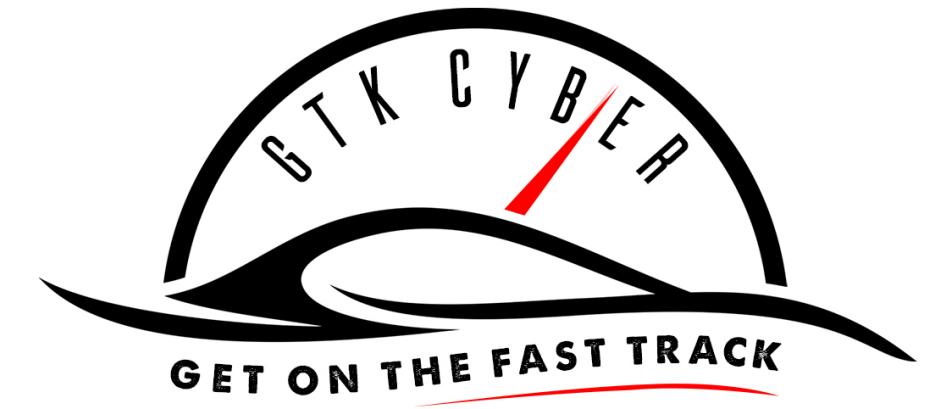
A **Series** is like a list or a
dictionary



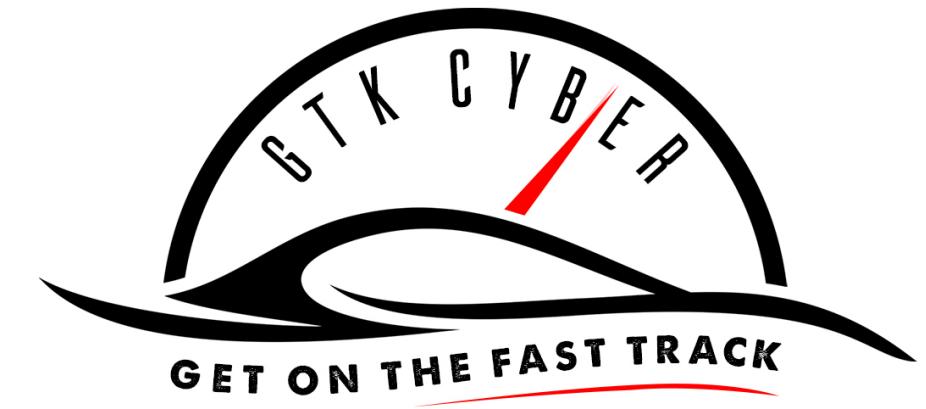
A **Series** is like a list or a
dictionary **but BETTER!!**



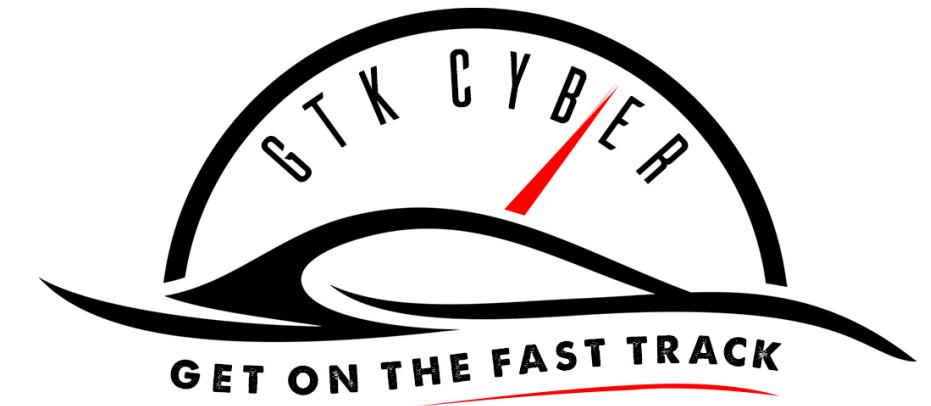
The **Series** is the **primary building block** for all the other data structures we will discuss



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



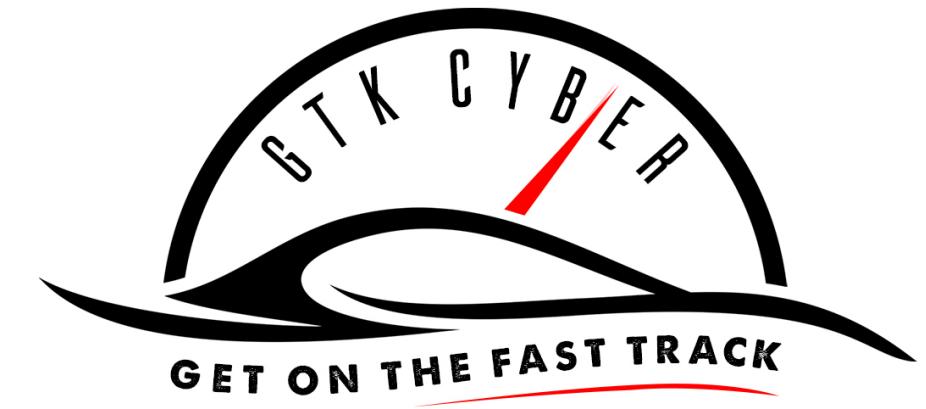
```
myData = pd.Series( <data>, index=<index> )
```



Creating a Series

```
series1 = pd.Series( [ 'a' , 'b' , 'c' , 'd' , 'e' ] )
```

```
series2 = pd.Series( { 'firstName' : 'Charles' ,  
'lastName' : 'Givre' } )
```



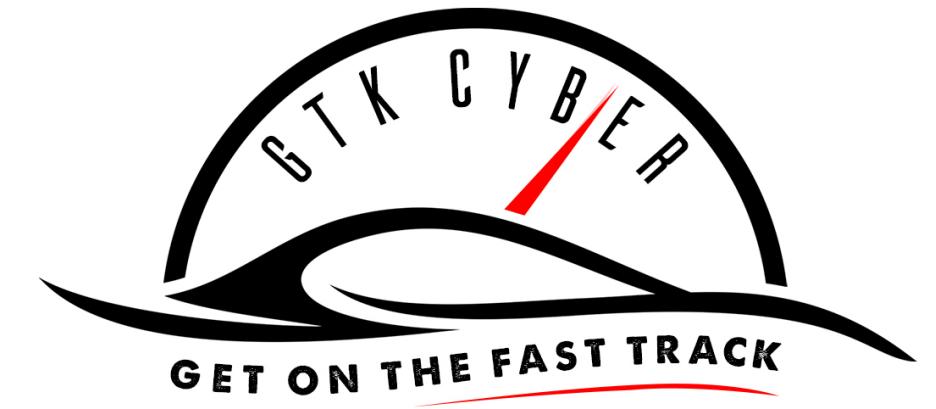
Accessing a Series

```
series1 = pd.Series( [ 'a' , 'b' , 'c' , 'd' , 'e' ] )
```

```
print( series1 )
```

0	a
1	b
2	c
3	d
4	e

dtype: object

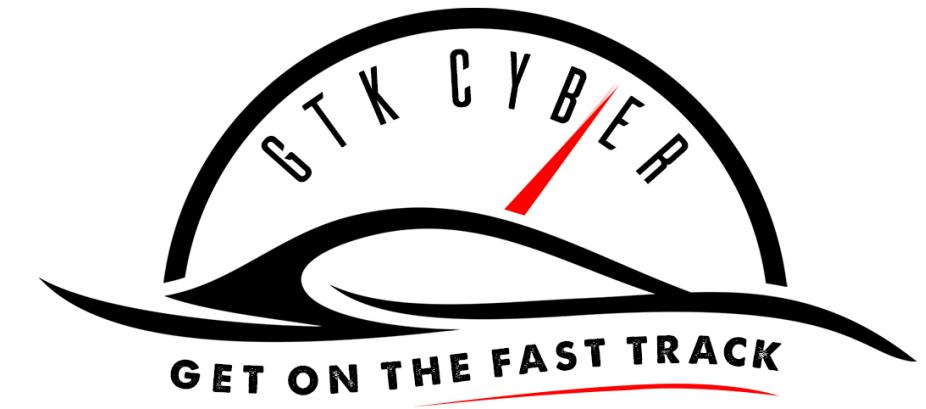


Accessing a Series

```
series1 = pd.Series( [ 'a' , 'b' , 'c' , 'd' , 'e' ] )
```

```
print( series1[3] )
```

d



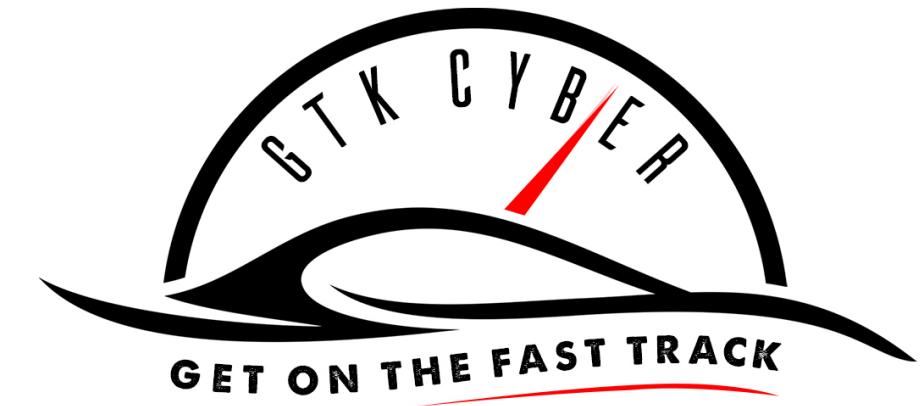
Accessing a Series

```
series1 = pd.Series( [ 'a' , 'b' , 'c' , 'd' , 'e' ] )
```

```
print( series1[1:3] )
```

1	b
2	c

dtype: object



Accessing a Series

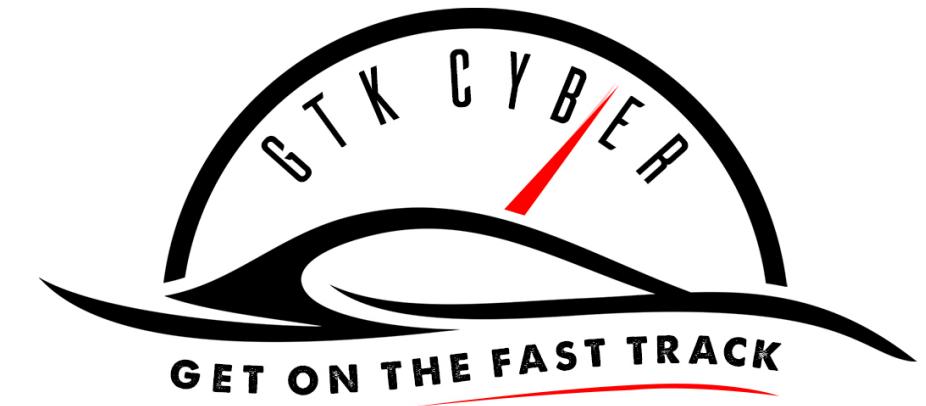
```
series1 = pd.Series( [ 'a' , 'b' , 'c' , 'd' , 'e' ] )
```

```
print( series1[1:3] )
```

1	b
2	c

dtype: object

```
series2 = series1[1:3]  
series2.reset_index( drop=True )
```

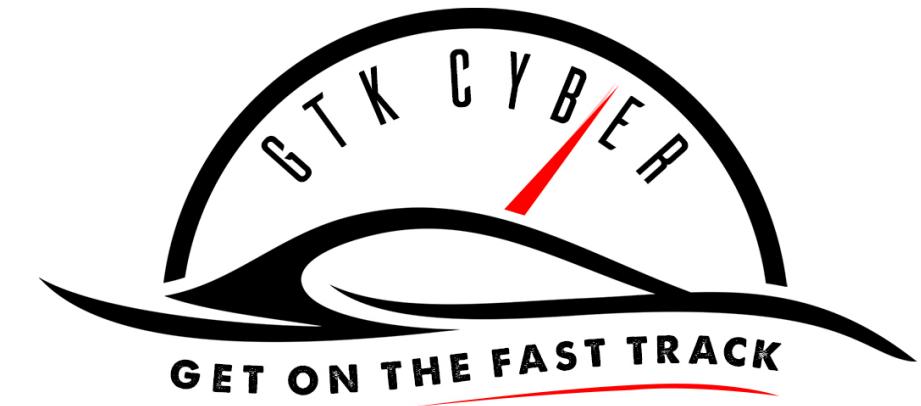


Accessing a Series

```
record = pd.Series(  
    { 'firstname': 'Charles',  
      'lastname': 'Givre',  
      'middle': 'classified' } )
```

```
record[ 'firstname' ]
```

Charles

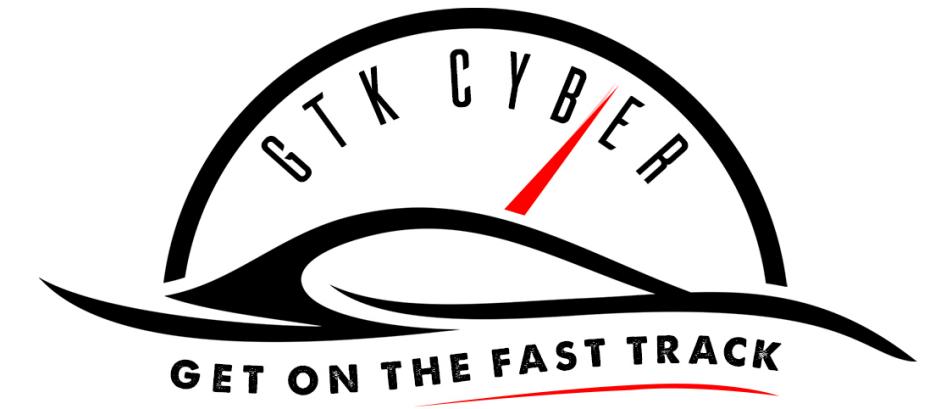


Accessing a Series

```
record = pd.Series(  
    {'firstname': 'Charles',  
     'lastname': 'Givre',  
     'middle': 'classified' } )
```

```
record[ 'firstname', 'lastname' ]
```

```
firstname      Charles  
lastname       Givre  
dtype: object
```



Accessing a Series

```
record = pd.Series(  
    {'firstname': 'Charles',  
     'lastname': 'Givre',  
     'middle': 'classified' } )
```

```
record[0]
```

Charles



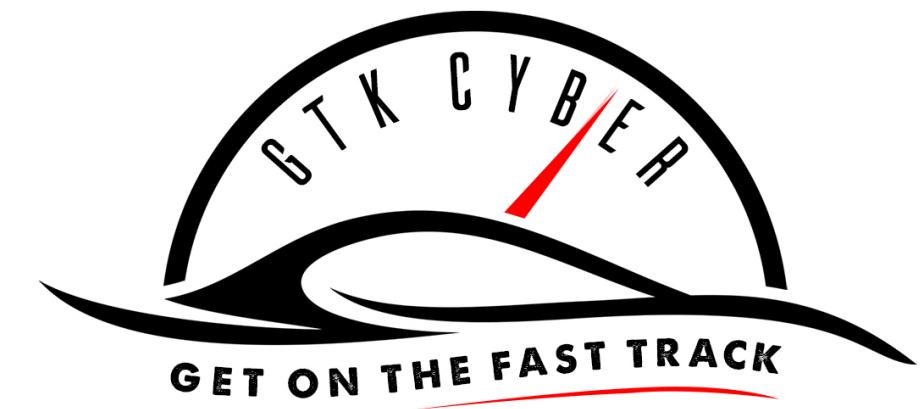
Accessing a Series

```
randomNumbers = pd.Series(  
    np.random.random_integers(1, 100, 50) )
```

```
randomNumbers.head()
```

```
0      48  
1      34  
2      84  
3      85  
4      58
```

```
dtype: int64
```



Accessing a Series

```
randomNumbers = pd.Series(  
    np.random.random_integers(1, 100, 50) )
```

```
randomNumbers.tail( 7 )
```

```
43      66  
44      66  
45      43  
46      55  
47      99  
48      82  
49      19
```

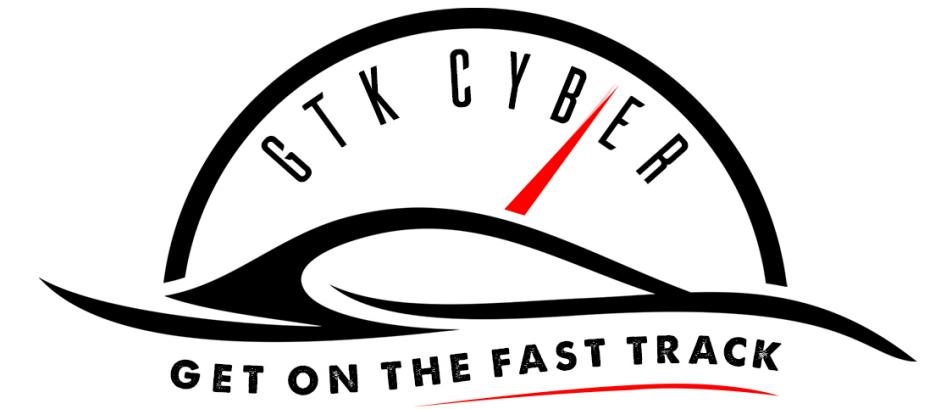
```
dtype: int64
```



Filtering Data in a Series

```
randomNumbers [ <boolean condition> ]  
randomNumbers [ randomNumbers < 10 ]
```

```
12      5  
21      2  
24      1  
27      1  
29      1  
dtype: int64
```



```
record.str.contains('Cha')
```

```
firstname      True
```

```
lastname      False
```

```
middle      False
```

```
dtype: bool
```

```
record[ record.str.contains('Cha') ]
```

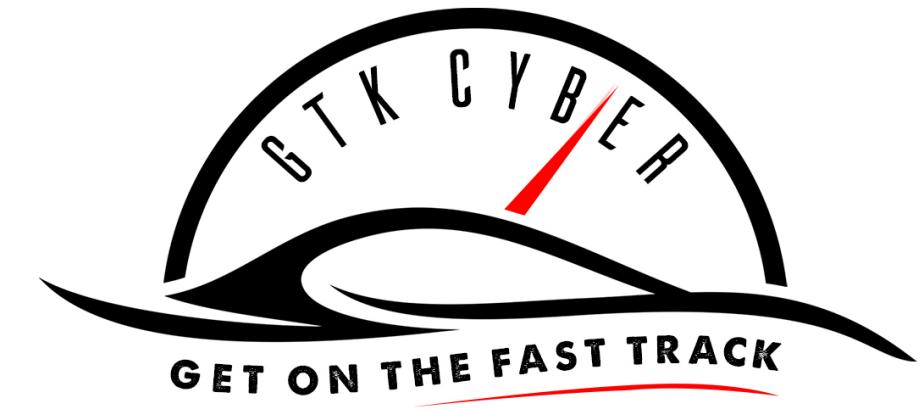
```
firstname      Charles
```

```
dtype: object
```



String Functions

Function	Explanation
Series.str.contains(<pattern>)	Returns true/false if text matches a pattern
Series.str.count(<pattern>)	Returns number of occurrences of a pattern in a string
Series.str.extract(<pattern>)	Returns matching groups from a string
Series.str.find(<string>)	Returns index of first occurrences of a substring (Note: not regex)
Series.str.findall(<pattern>)	Returns all occurrences of a regex
Series.str.len()	Returns the length of text
Series.str.replace(<pat>, <replace>)	Replaces matches with a replacement string



Manipulating Data

Pandas Data Structures allow you
to **perform operations**
on your **entire data set at once.**

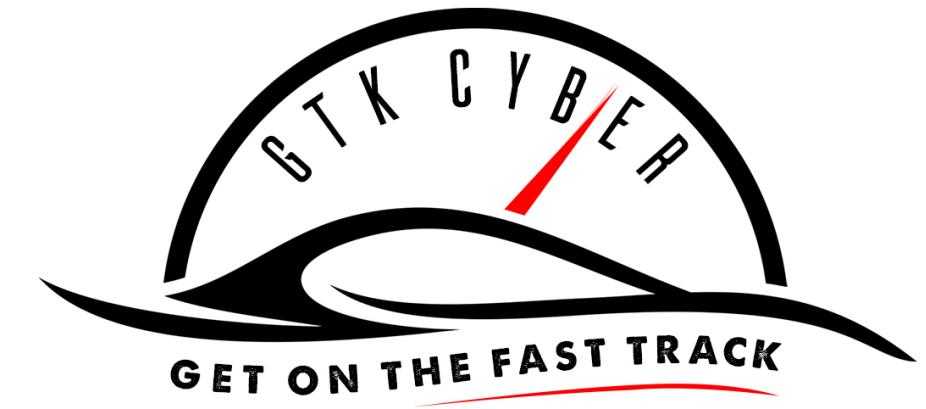


```
for x in range(0, len( data )):  
    data[ x ] = data[ x ] + 1
```

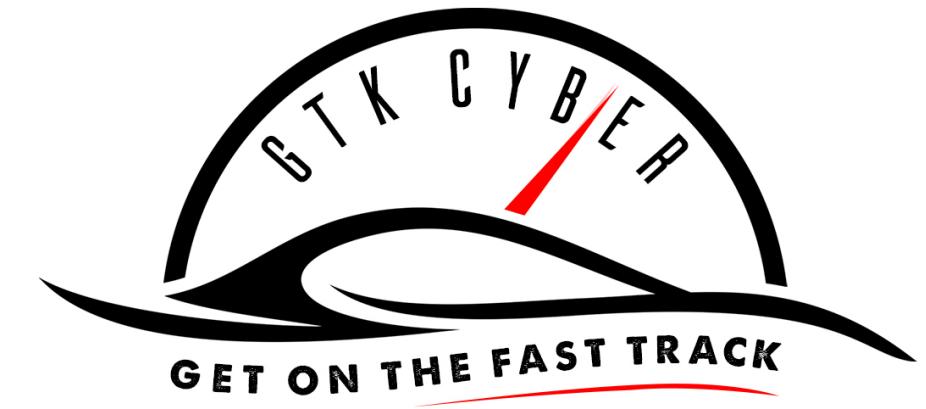


```
for x in range(0, len( data )):  
    data[ x ] = data[ x ] + 1
```

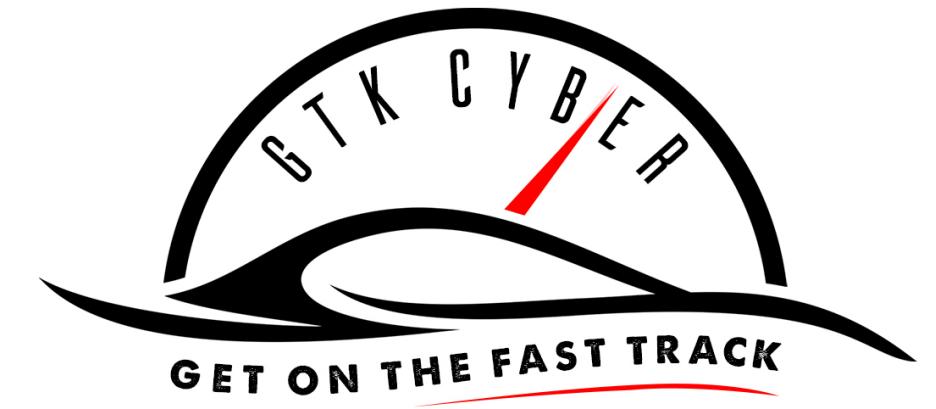
odds = evens + 1



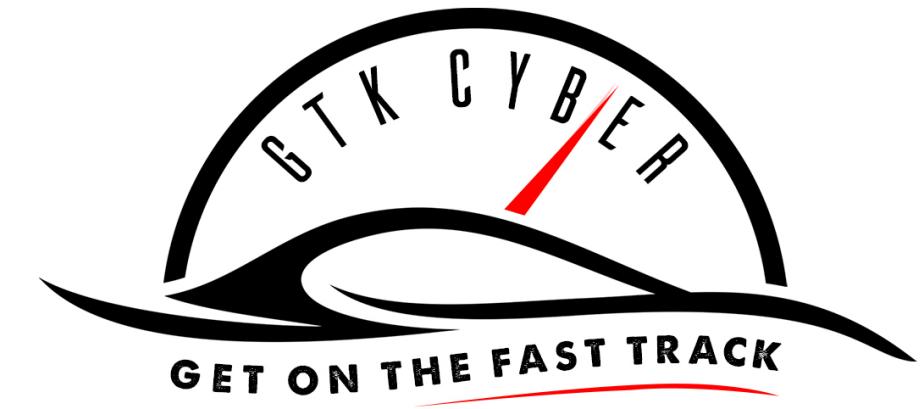
```
combinedSeries = series1.add( series2 )
```



```
def addTwo( n ):  
    return n + 2  
  
odds.apply( addTwo )
```

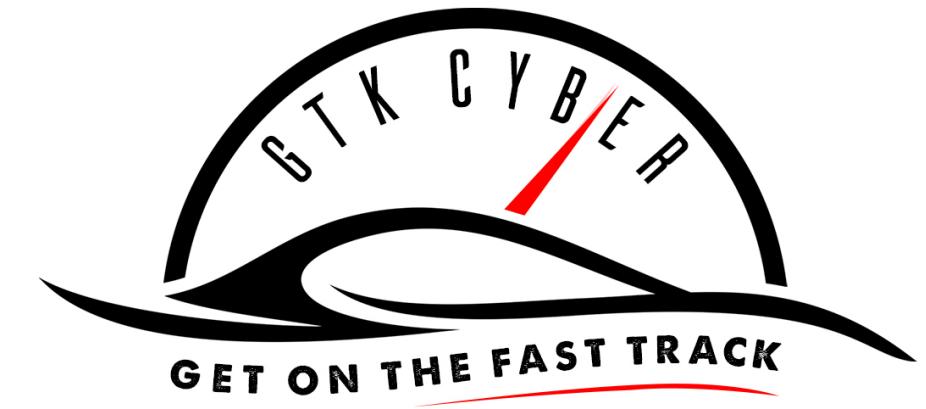


```
odds.apply( lambda x: x + 1 )
```

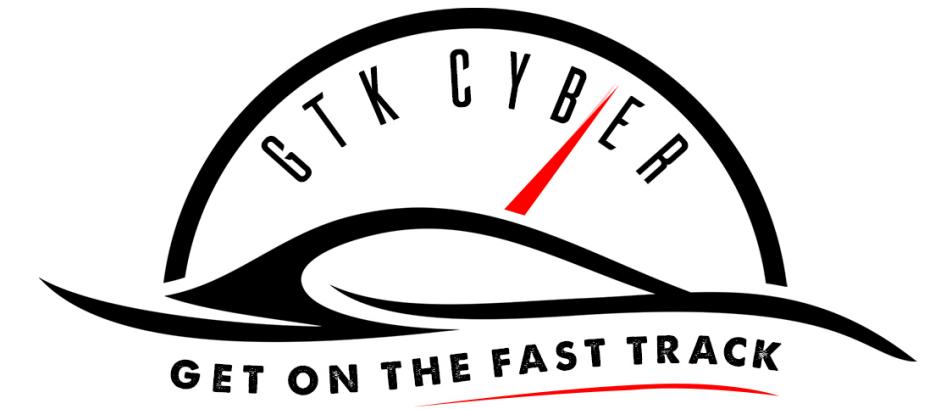


`Series.dropna()`

`Series.fillna(value="<something>")`

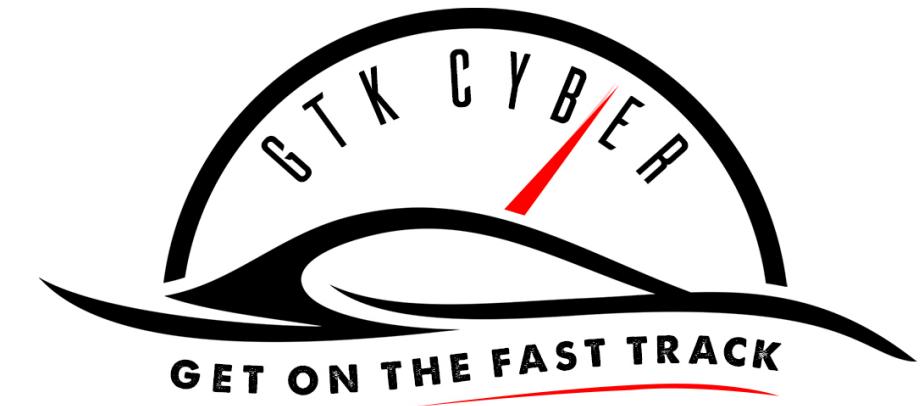


Questions?



In Class Exercise

Please take 20 minutes and complete
Worksheet 1: Working with One Dimensional Data

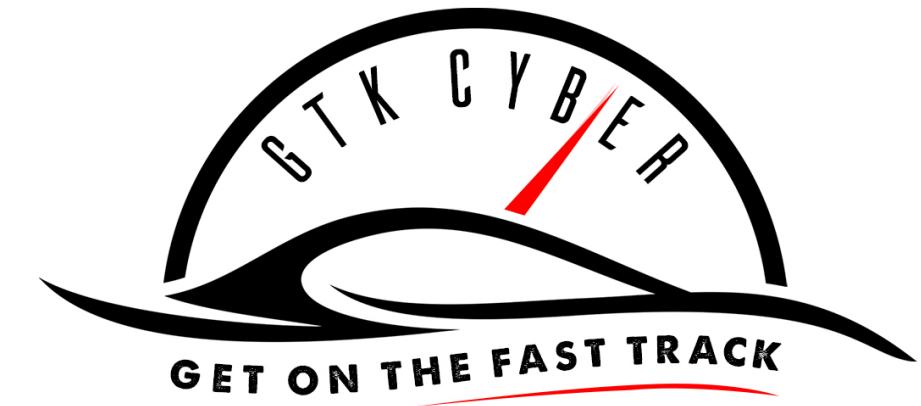


Exercise 1

```
email_series = pd.Series(emails)
filtered_emails =
email_series[email_series.str.contains('.edu')]
```

```
2      mdixon2@cmu.edu
11     jjonesb@arizona.edu
15     eberryf@brandeis.edu
```

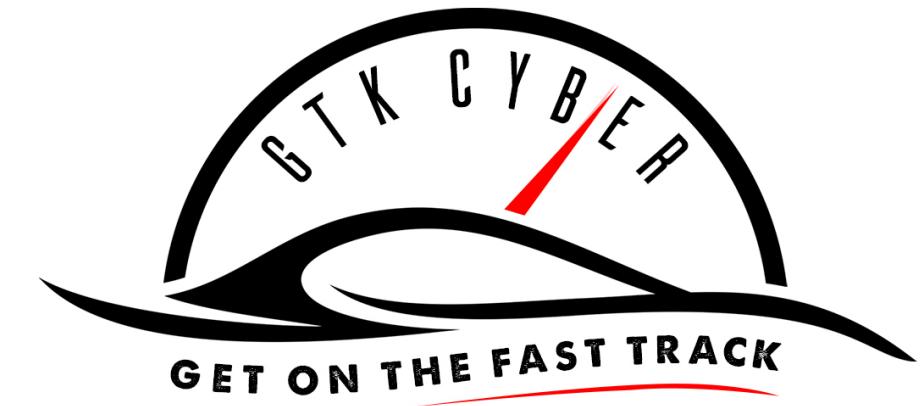
```
accounts = filtered_emails.str.split('@').str[0]
2      mdixon2
11     jjonesb
15     eberryf
dtype: object
```



Exercise 2

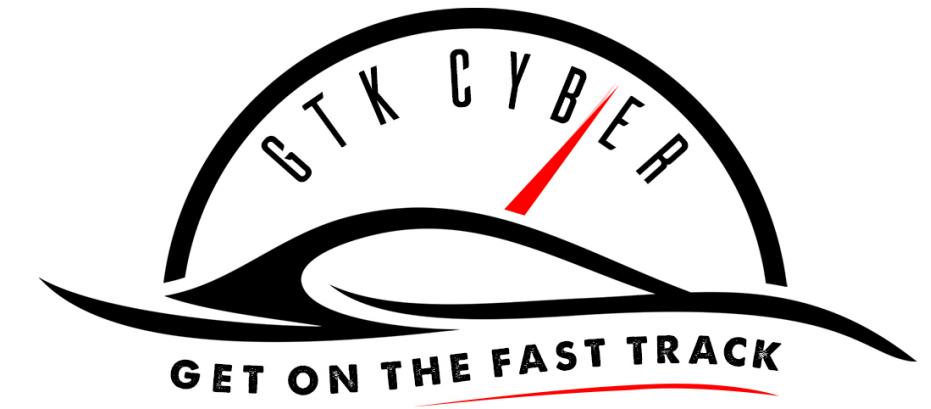
```
pounds = pd.Series( weights )
```

```
kilos = pounds.apply( poundsToKilograms )
```



Exercise 3

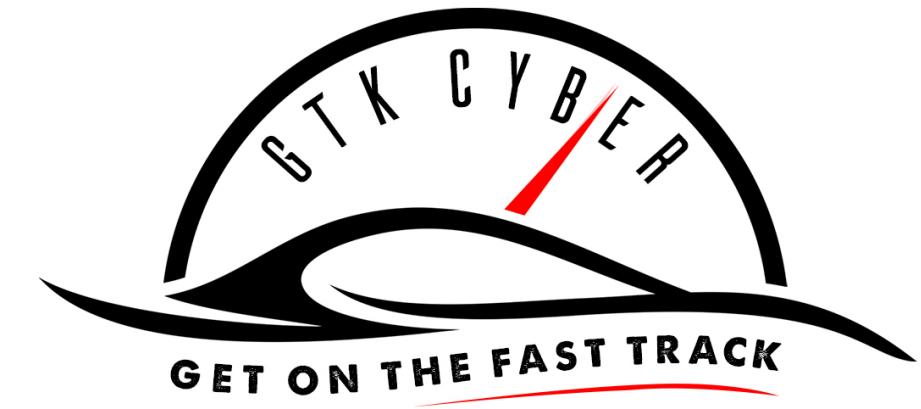
```
IPData = pd.Series( hosts )
privateIPs = IPData[
    IPData.apply(
        lambda x : ip_address(x).is_private
    )
]
```



The Data Frame

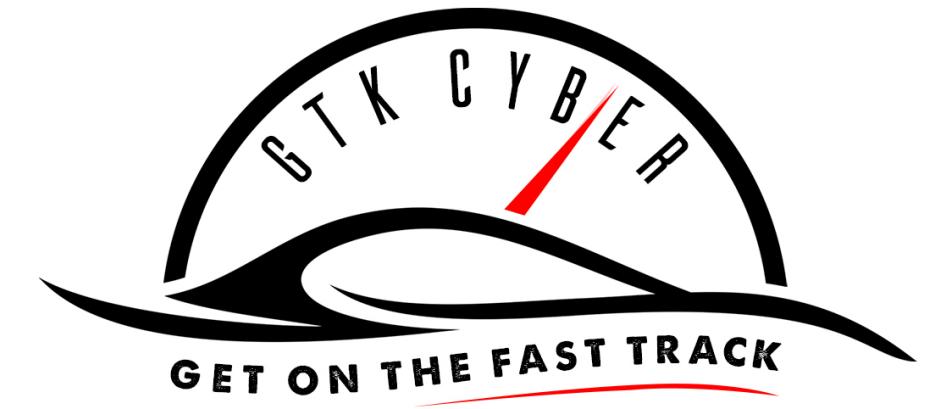


```
data = pd.DataFrame( <data>, <index>,
                     <column_names> )
```



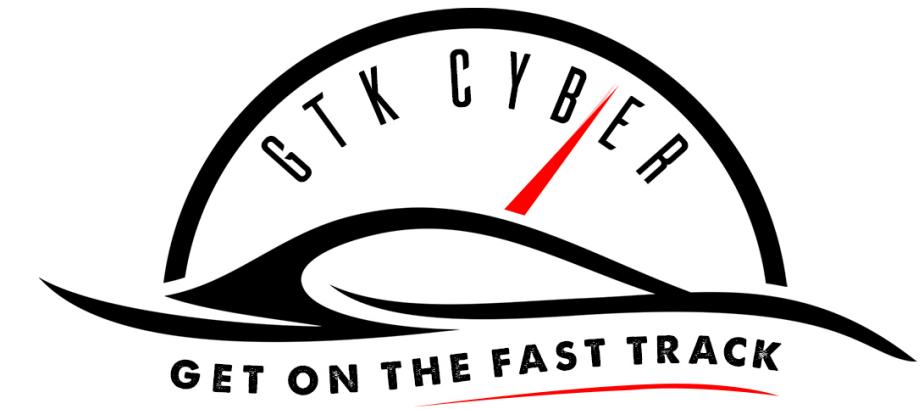
CSV

```
data = pd.read_csv( <file> )
```

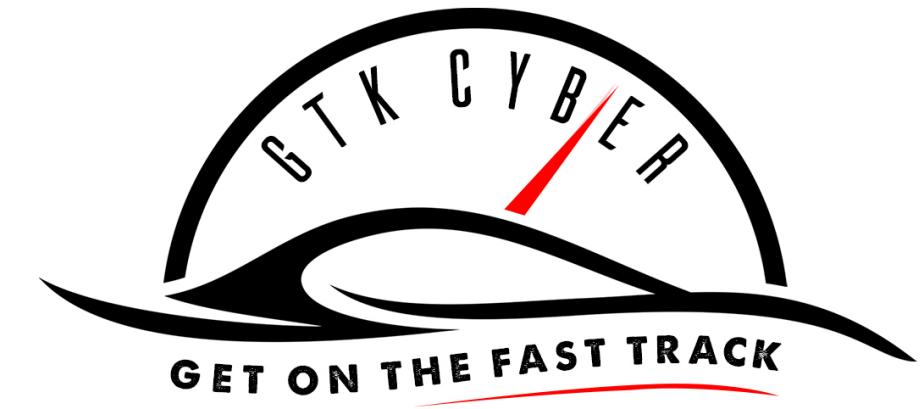


Excel

```
data = pd.read_excel( <file>, sheetname=<sheetname> )
```



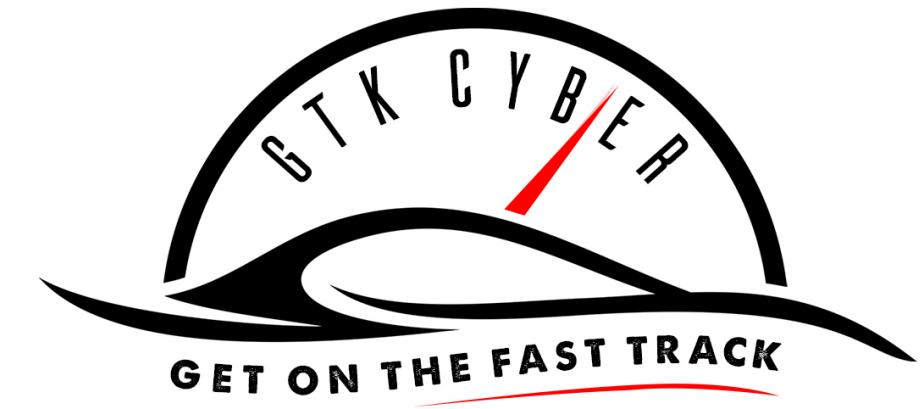
```
[  
  {  
    "changed": "2015-04-0300:00:00",  
    "created": "2014-04-10 00:00:00",  
    "dnssec": "False",  
    "expires": "2016-04-10 00:00:00",  
    "isMalicious": 0,  
    "url": "nuteczki.com"  
  },  
  ...  
]
```



JSON

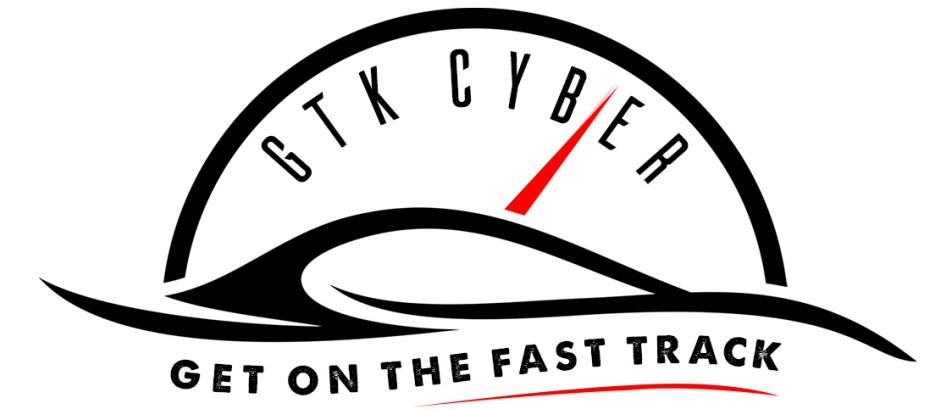
```
data = pd.read_json( <file> )  
or
```

```
data = pd.read_json( <url> )
```



From a Database

```
data = pd.read_sql( <query> , <connection> )
```

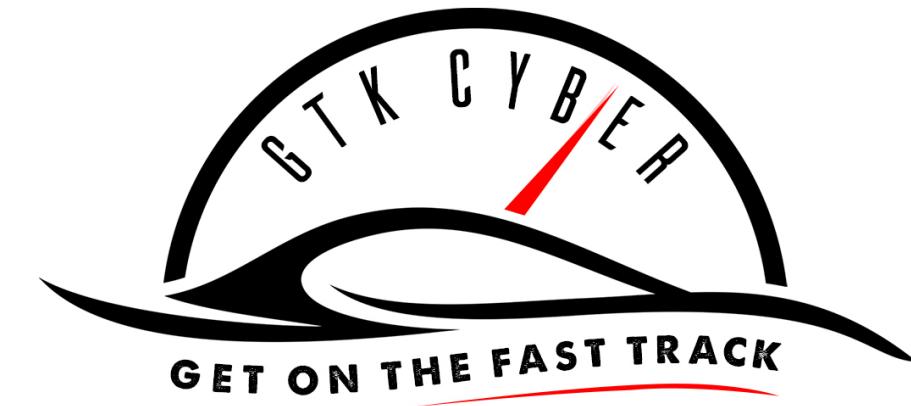


HTML

```
data = pd.read_html( <source> )
```

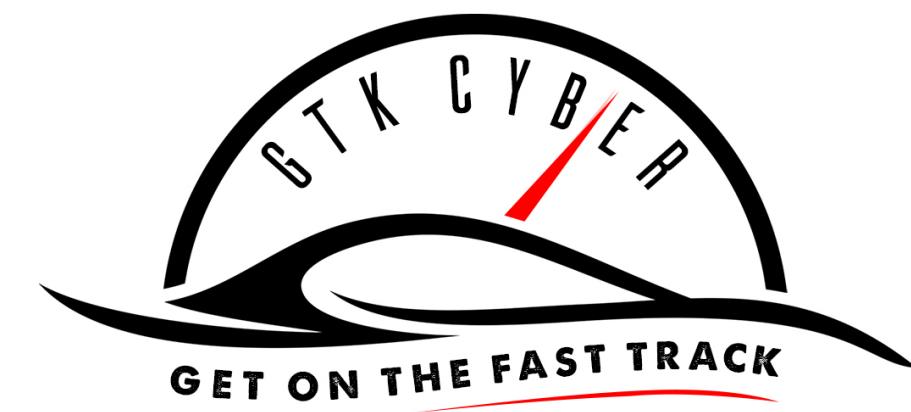


Log Files...



070823 21:00:32
070823 21:00:48
070823 21:00:56
070917 16:29:01
070917 16:29:12

	1	Connect	root@localhost on test1
	1	Query	show tables
	1	Query	select * from category
	21	Query	select * from location
	21	Query	select * from location where id = 1 LIMIT 1



070823 21:00:32	1	Connect	root@localhost on test1
070823 21:00:48	1	Query	show tables
070823 21:00:56	1	Query	select * from category
070917 16:29:01	21	Query	select * from location
070917 16:29:12	21	Query	select * from location where id = 1 LIMIT 1

```
logdf = pd.read_table('..../data/mysql.log', names=['raw'])
```

	raw
0	070823 21:00:32 1 Connect root@localhost on test1
1	070823 21:00:48 1 Query show tables
2	070823 21:00:56 1 Query select * from category
3	070917 16:29:01 21 Query select * from location
4	070917 16:29:12 21 Query select * from location where id = 1 LIMIT 1



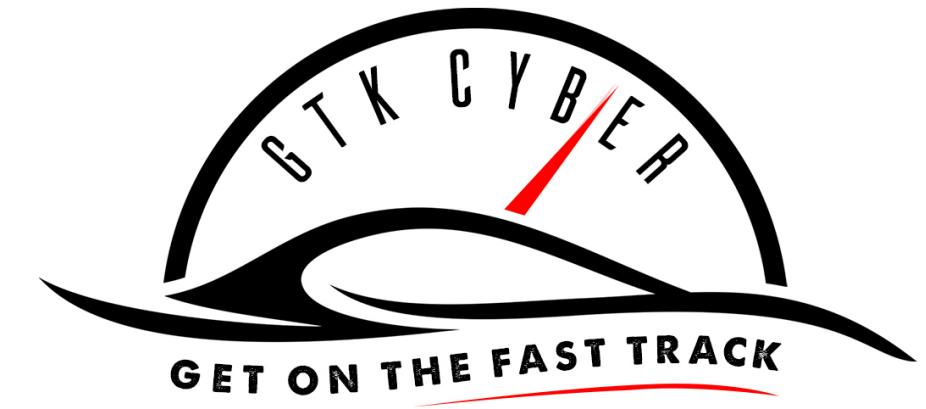
```
logdf = pd.read_table('../data/mysql.log', names=['raw'])
logdf['raw'].str.extract('(\d{6}\s\d{2}:\d{2}:\d{2})\s+(\d+)\s(\S+)
\S(.+)', expand=False)
```

	0	1	2	3
0	070823 21:00:32	1	Connect	root@localhost on test1
1	070823 21:00:48	1	Query	show tables
2	070823 21:00:56	1	Query	select * from category
3	070917 16:29:01	21	Query	select * from location
4	070917 16:29:12	21	Query	select * from location where id = 1 LIMIT 1



```
logdf = pd.read_table('..../data/mysql.log', names=[ 'raw' ])
logdf[ 'raw' ].str.extract('(?P<date>\d{6}\s\d{2}:\d{2}:\d{2})\s+(?P<PID>\d+)\s(?P<Action>\S+)\s(?P<Query>.+)', expand=False)
```

	Date	PID	Action	Query
0	070823 21:00:32	1	Connect	root@localhost on test1
1	070823 21:00:48	1	Query	show tables
2	070823 21:00:56	1	Query	select * from category
3	070917 16:29:01	21	Query	select * from location
4	070917 16:29:12	21	Query	select * from location where id = 1 LIMIT 1



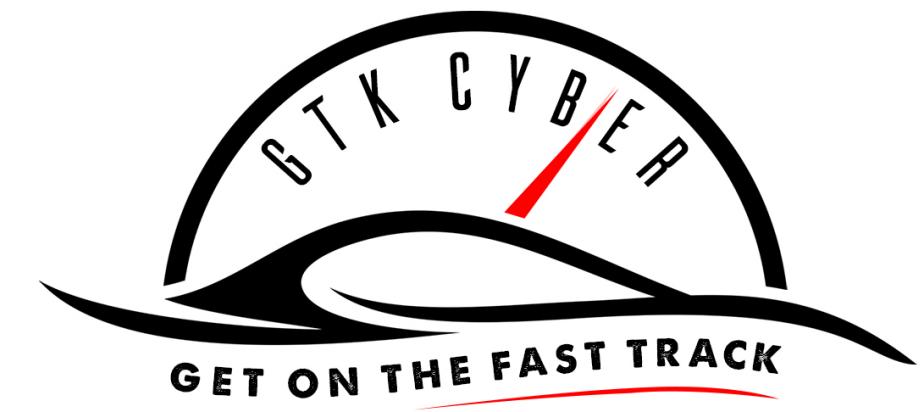
Web Server Logs





Web Server Logs

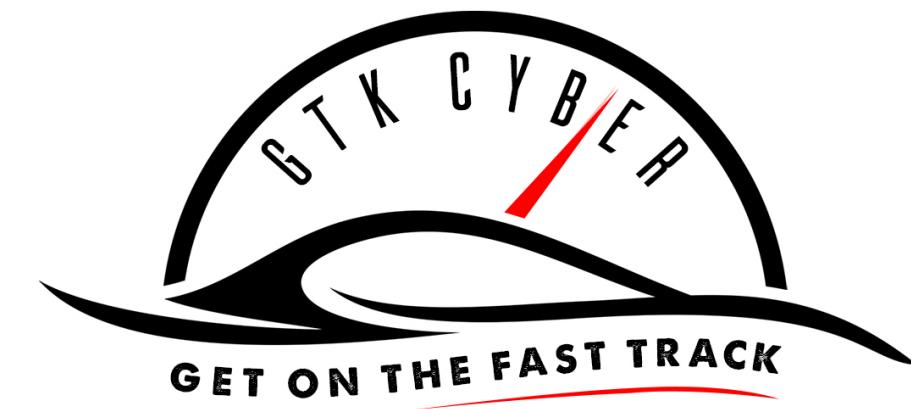
```
195.154.46.135 - - [25/Oct/2015:04:11:25 +0100] "GET /linux/
doing-pxe-without-dhcp-control HTTP/1.1" 200 24323 "http://
howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0)
Gecko/20100101 Firefox/35.0"
```



Web Server Logs

```
195.154.46.135 - - [25/Oct/2015:04:11:25 +0100] "GET /linux/
doing-pxe-without-dhcp-control HTTP/1.1" 200 24323 "http://
howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0)
Gecko/20100101 Firefox/35.0"
```

```
import apache_log_parser
line_parser = apache_log_parser.make_parser("%h %l %u %t
 \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"")
```

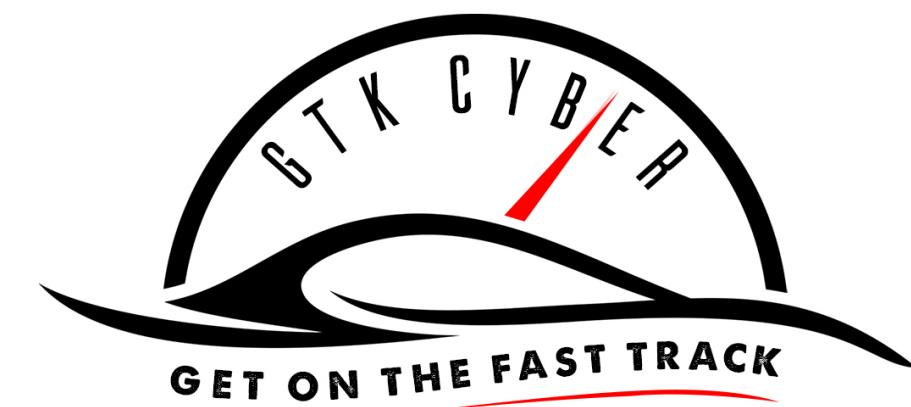


Web Server Logs

```
import apache_log_parser
line_parser = apache_log_parser.make_parser("%h %l %u %t
 \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"")

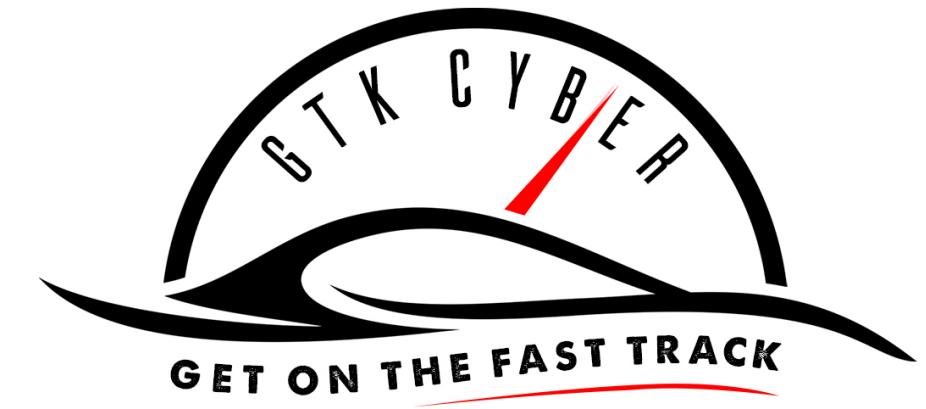
server_log = open("../data/hackers-access.httpd", "r")
parsed_server_data = []
for line in server_log:
    data = {}
    data = line_parser(line)
    parsed_server_data.append( data )

server_df = pd.DataFrame( parsed_server_data )
```

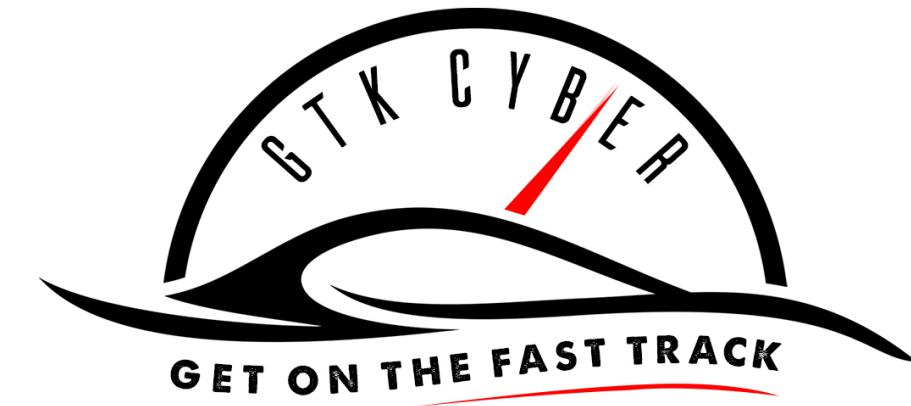


Web Server Logs

requester	request_first_line	request_header_referer	request_header_user_agent	request_http_ver	request_method	request_url
	GET /linux/doing-pxe-without-dhcp-control HTTP/...	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.1	GET	/linux/doing-pxe-v...
	GET /join_form HTTP/1.0	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.0	GET	/join_form
	POST /join_form HTTP/1.1	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.1	POST	/join_form
	GET /join_form HTTP/1.0	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.0	GET	/join_form
	POST /join_form HTTP/1.1	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.1	POST	/join_form
	GET /acl_users/credentials_cookie_auth/require...	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.1	GET	/acl_users/creden...



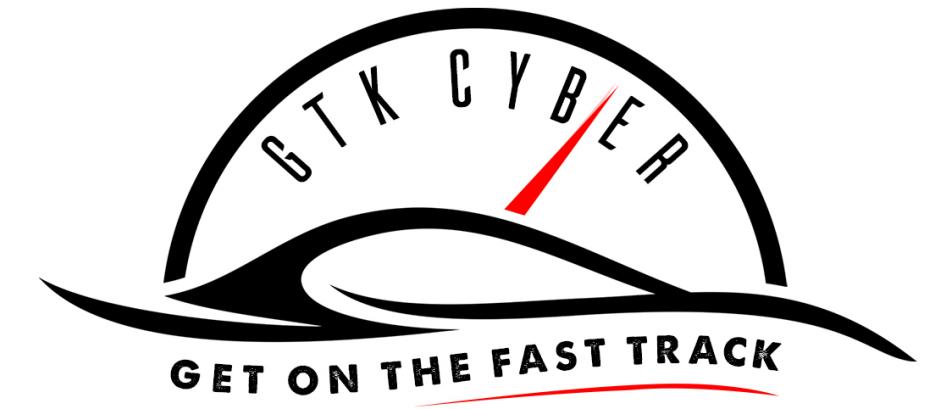
Manipulating a DataFrame



```
df = data[ 'column1' ]
```

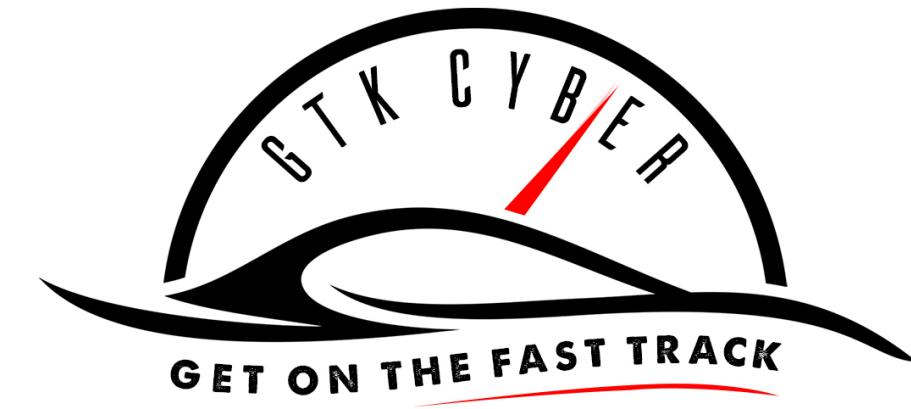
Returns a series

```
df[ 'ip' ].value_counts().head()
```



```
df = data[ ['column1', 'column2', 'column3'] ]
```

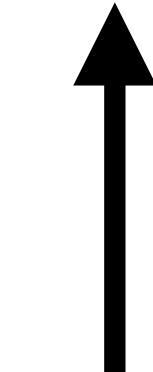
Returns a DataFrame



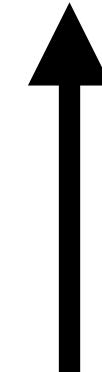
Filtering a DataFrame

`df[<boolean condition>]`

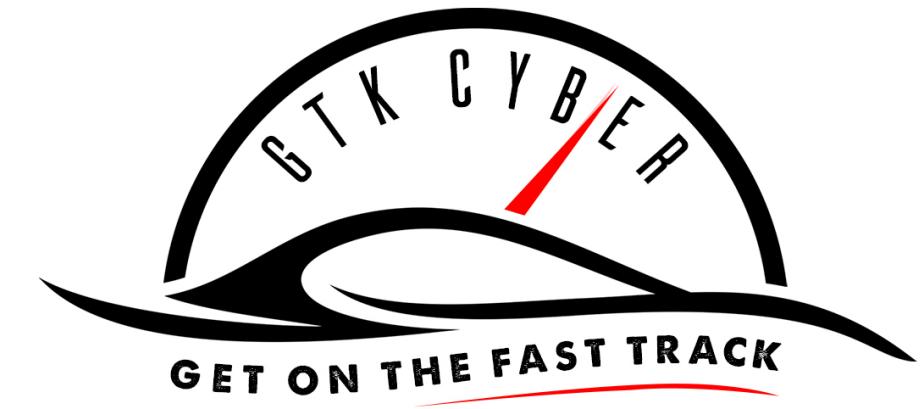
`df[['col1', 'col2']][col3 > 5]`



Columns



Filter

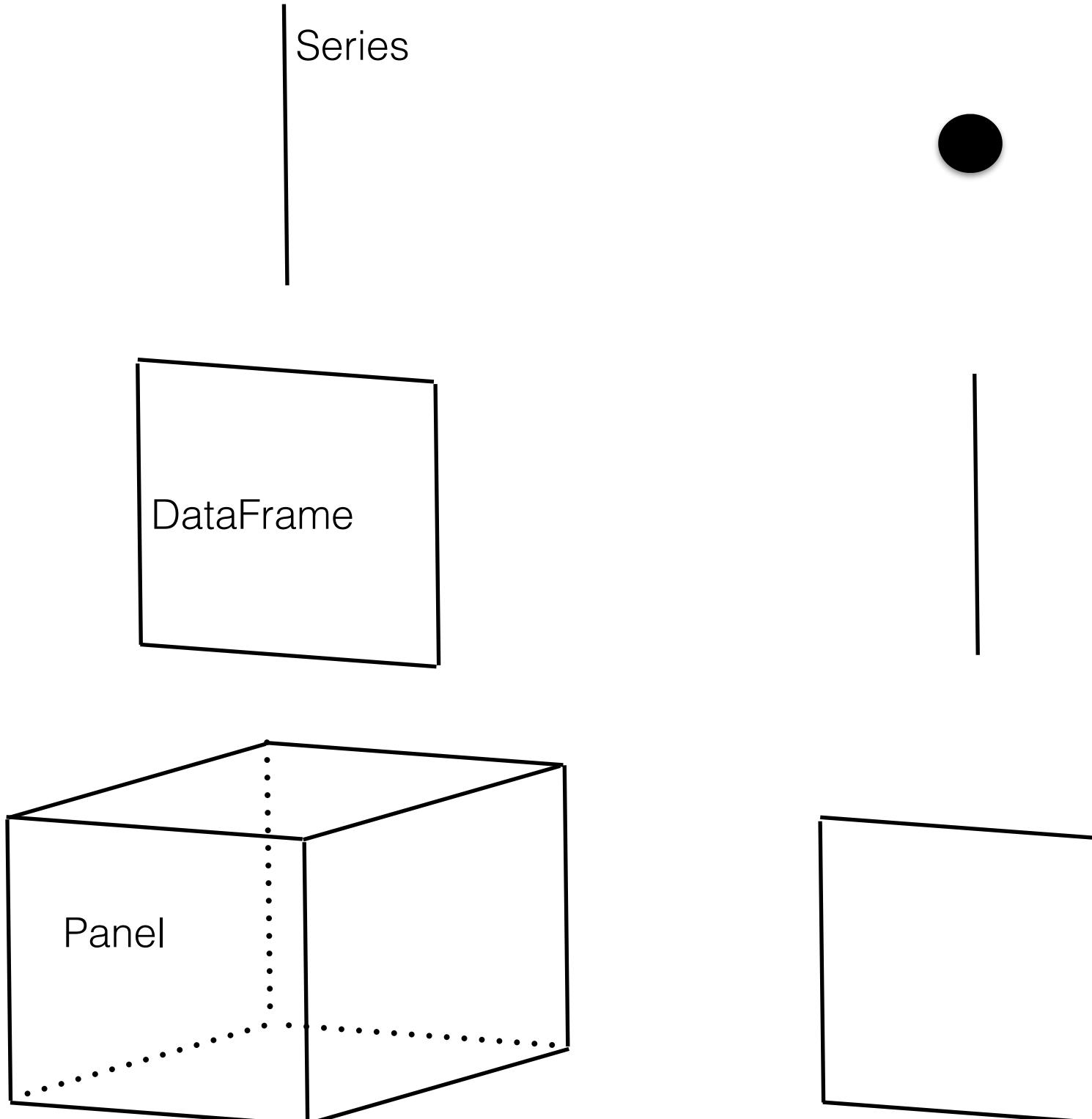


```
data.loc[<index>]  
data.loc[<list of indexes>]  
data.sample(<n>)
```



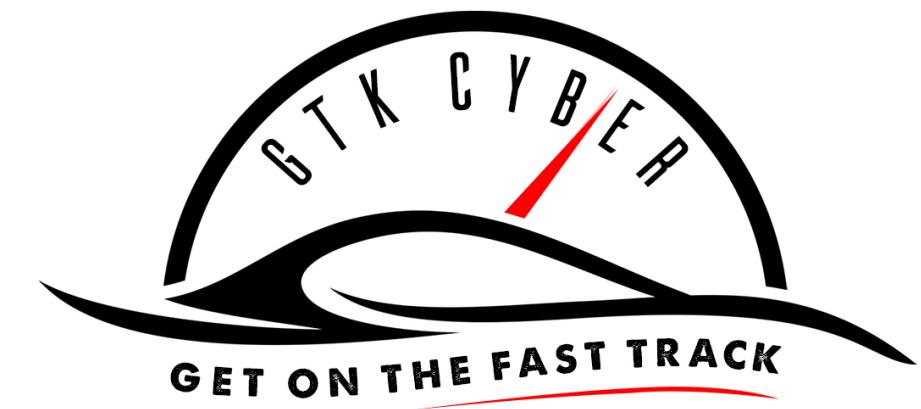
data.apply(<function>)

IF
Call Apply On...



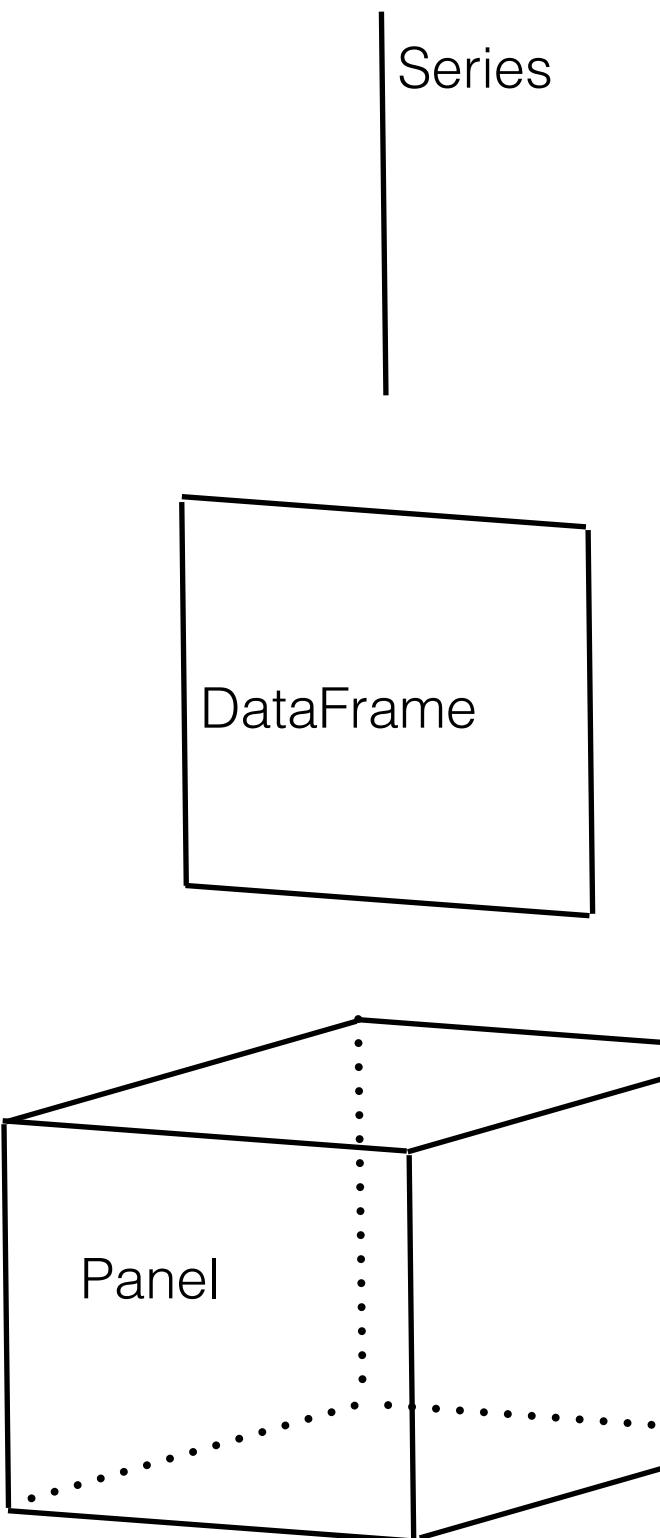
THEN
Function
Receives



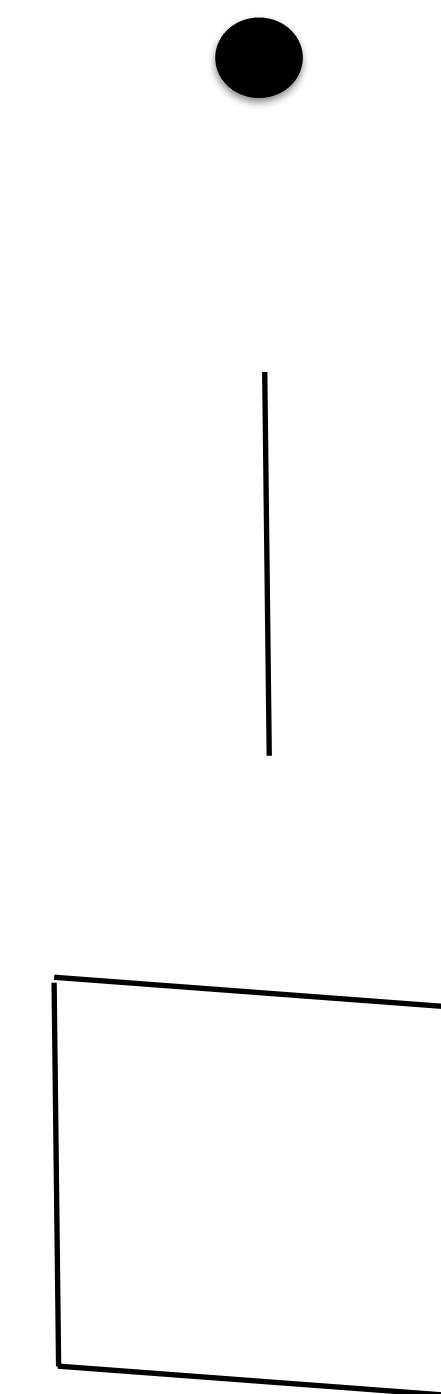


data.apply(<function>)

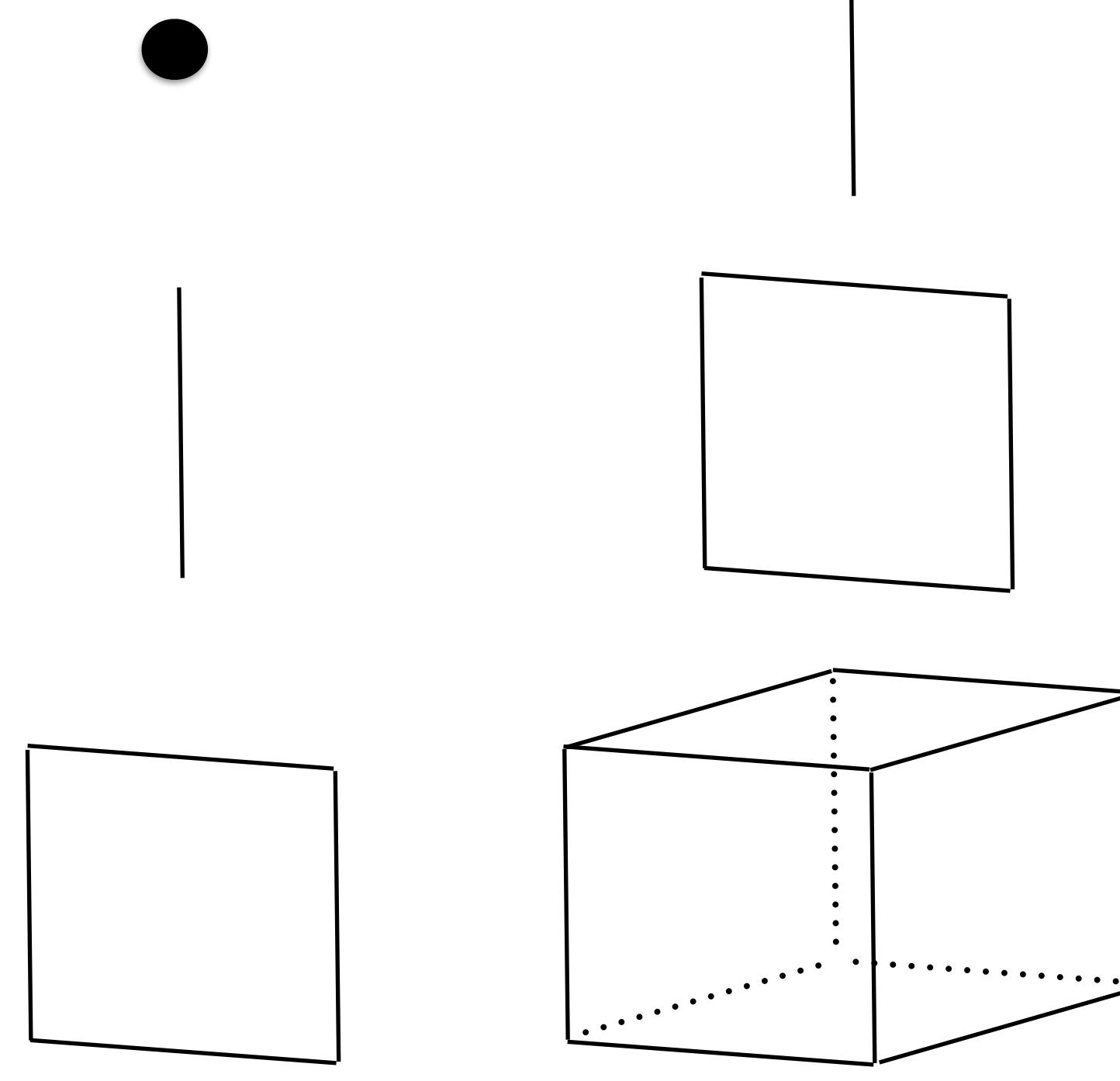
IF
Call Apply On...



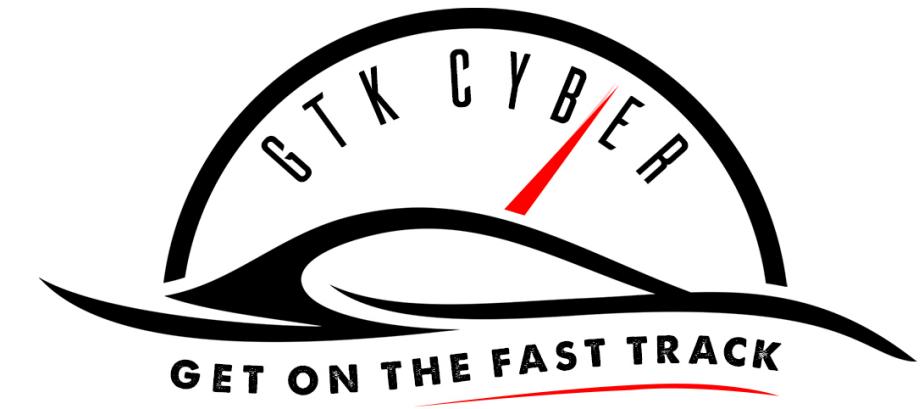
THEN
Function
Receives



IF
Function
Returns



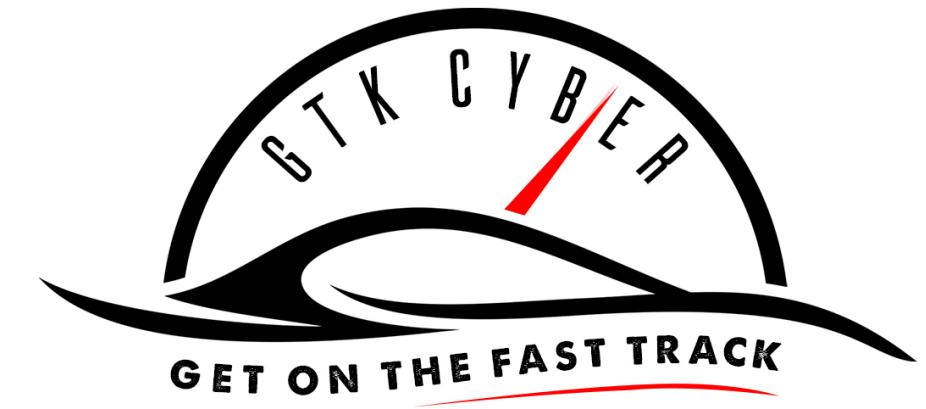
THEN
Apply
Returns...



1	2	3
4	5	6
7	8	9

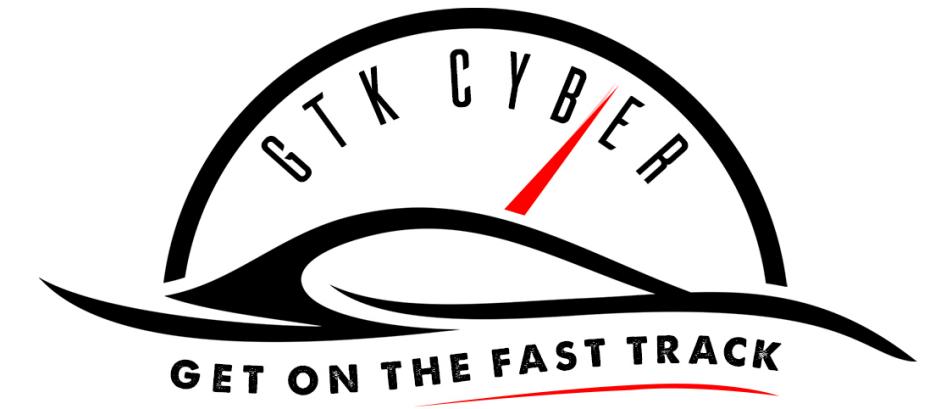
data.T

1	4	7
2	5	8
3	6	9

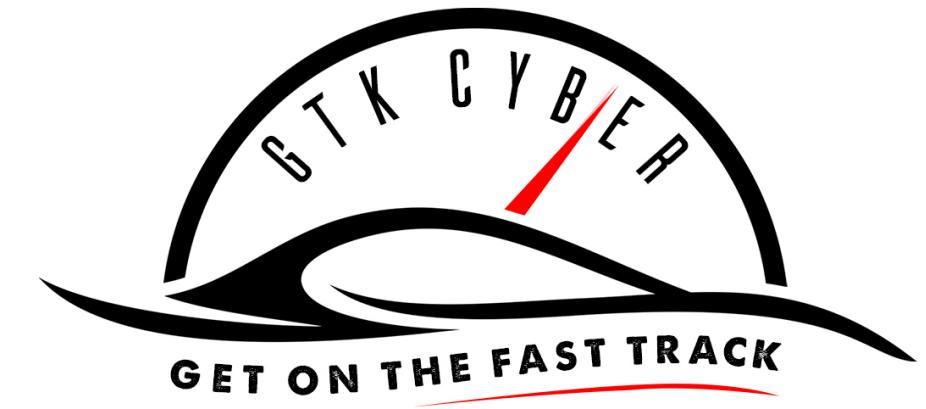


#Gets you the sum of columns
data.sum(axis=0)

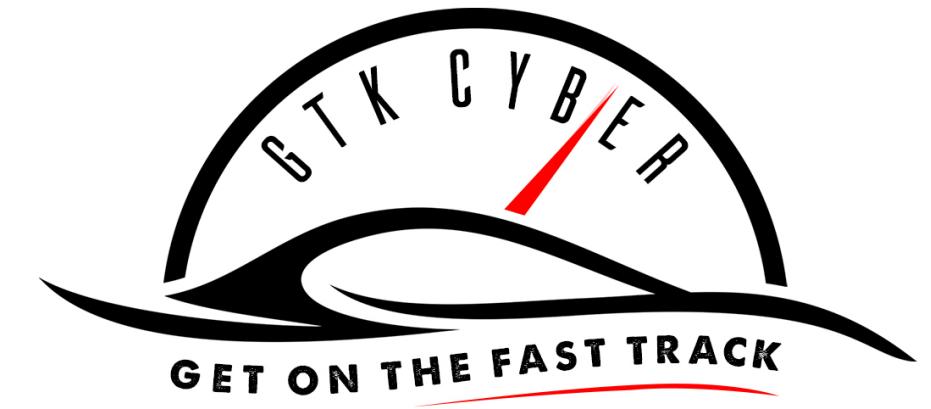
#Gets you the sum of the rows
data.sum(**axis=1**)



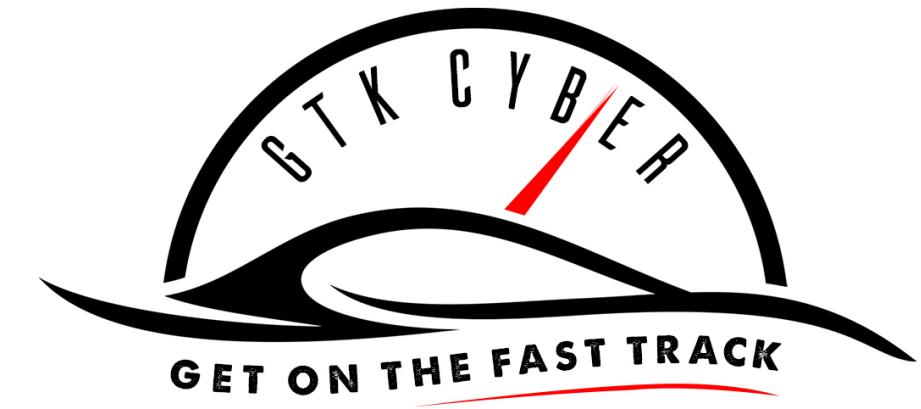
```
DataFrame.drop(labels,  
axis=0,  
level=None,  
inplace=False,  
errors='raise')¶
```



Merging Data Sets



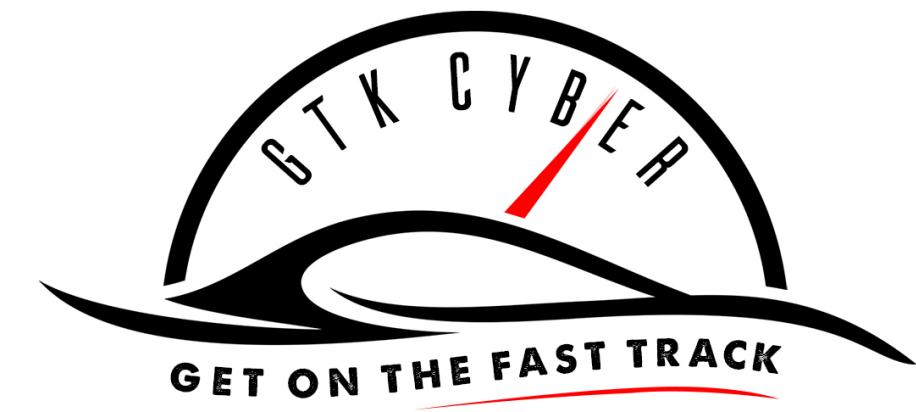
Union



Union

Series 1

Index	Value
0	6
1	4
2	2
3	3



Union

Series 1

Index	Value
0	6
1	4
2	2
3	3

Series 2

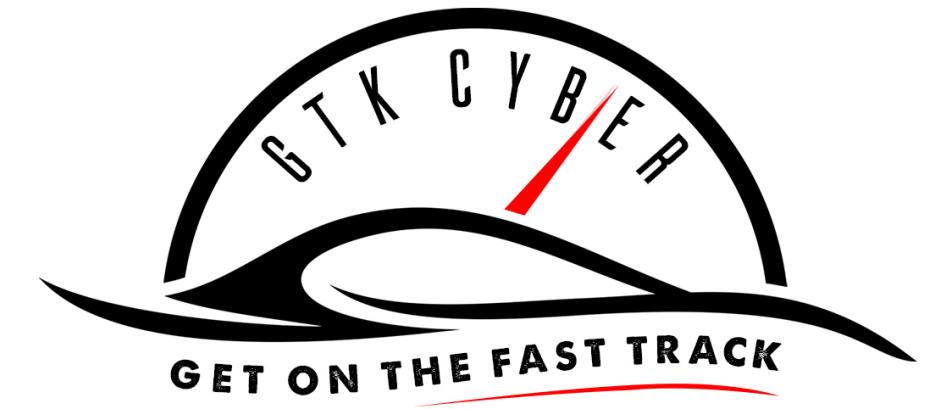
Index	Value
0	7
1	5
2	3
3	4



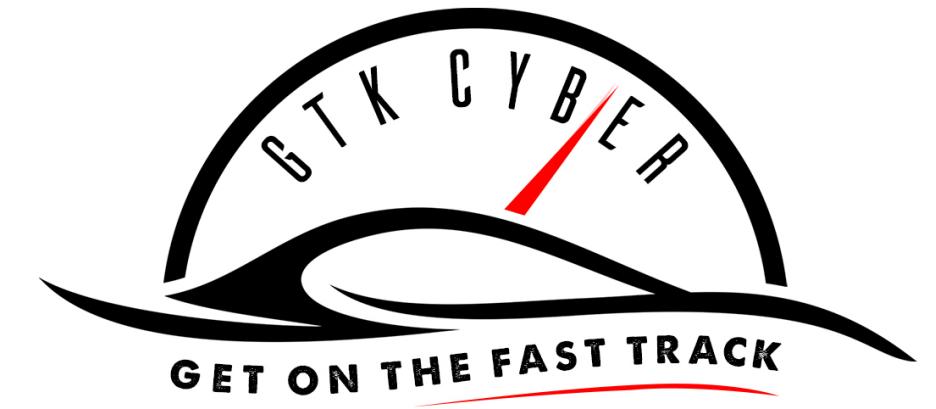
Union

combinedSeries

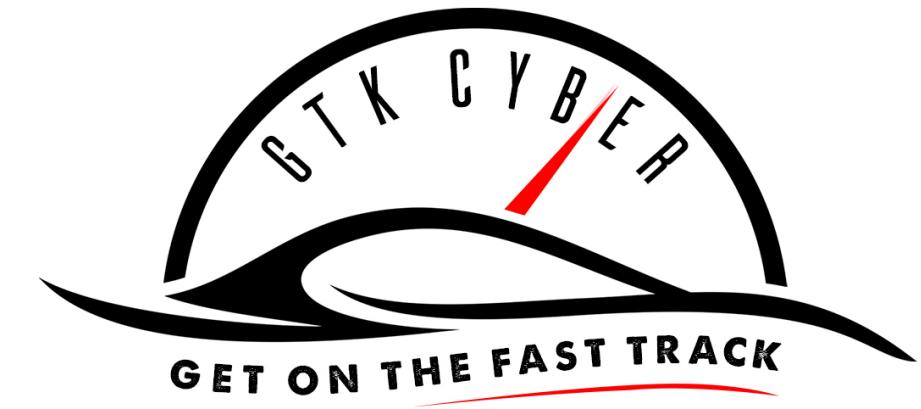
Index	Value
0	6
1	4
2	2
3	3
4	7
5	5
6	3
7	4



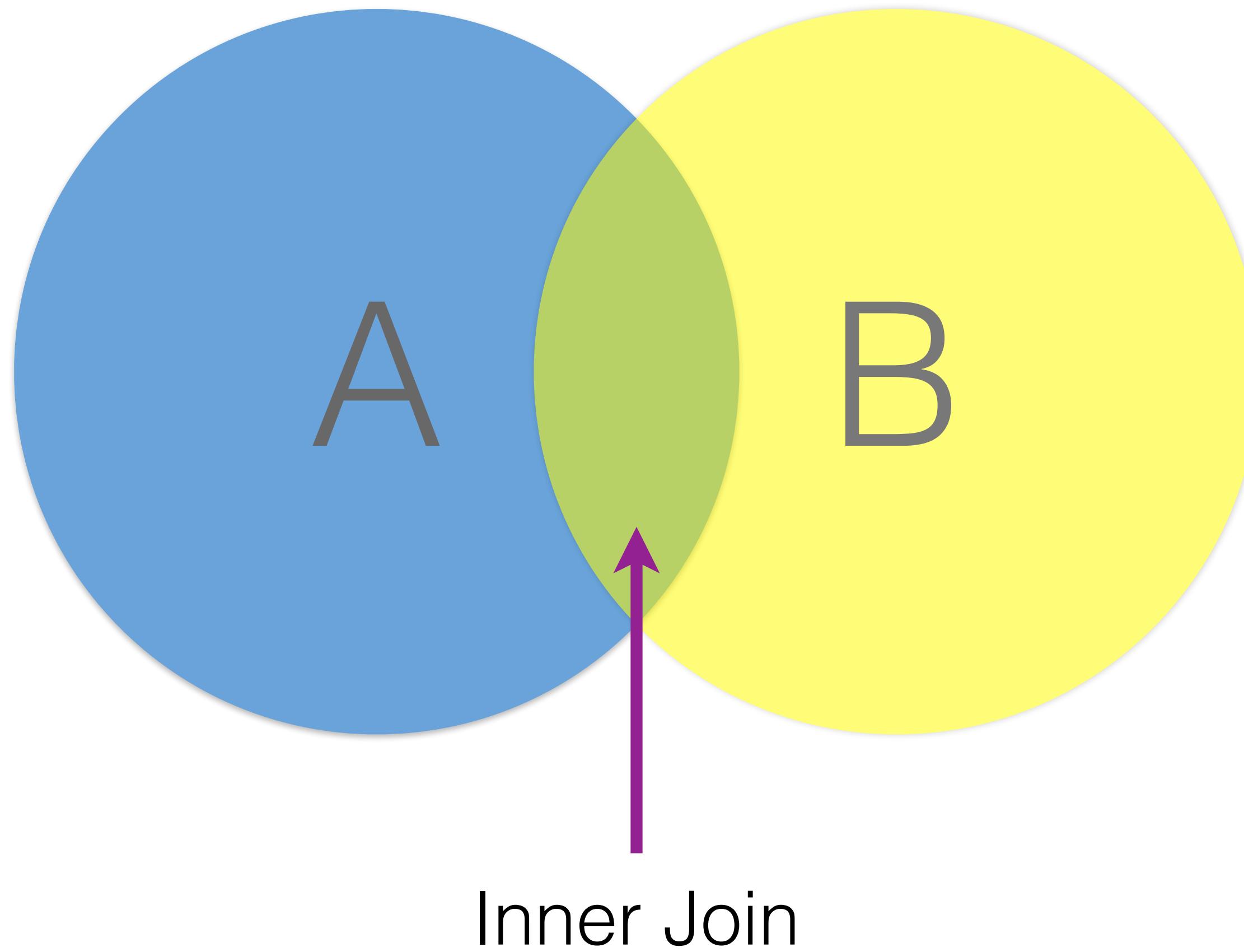
```
combinedSeries = pd.concat(  
[series1, series2],  
ignore_index=True )
```

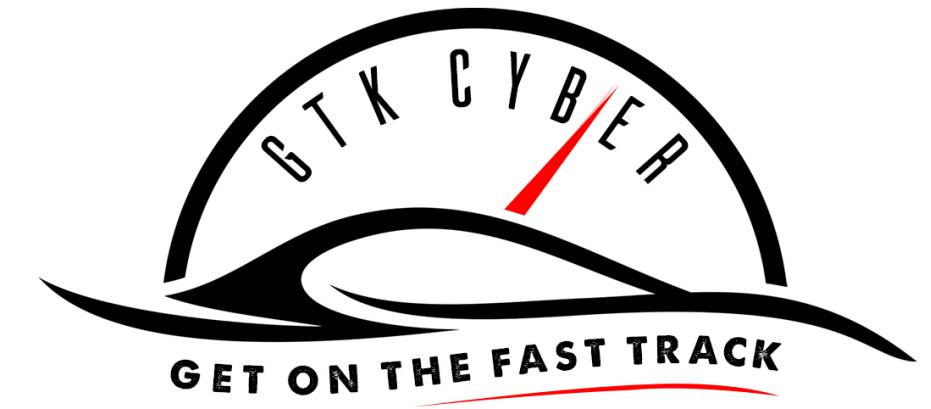


Joins

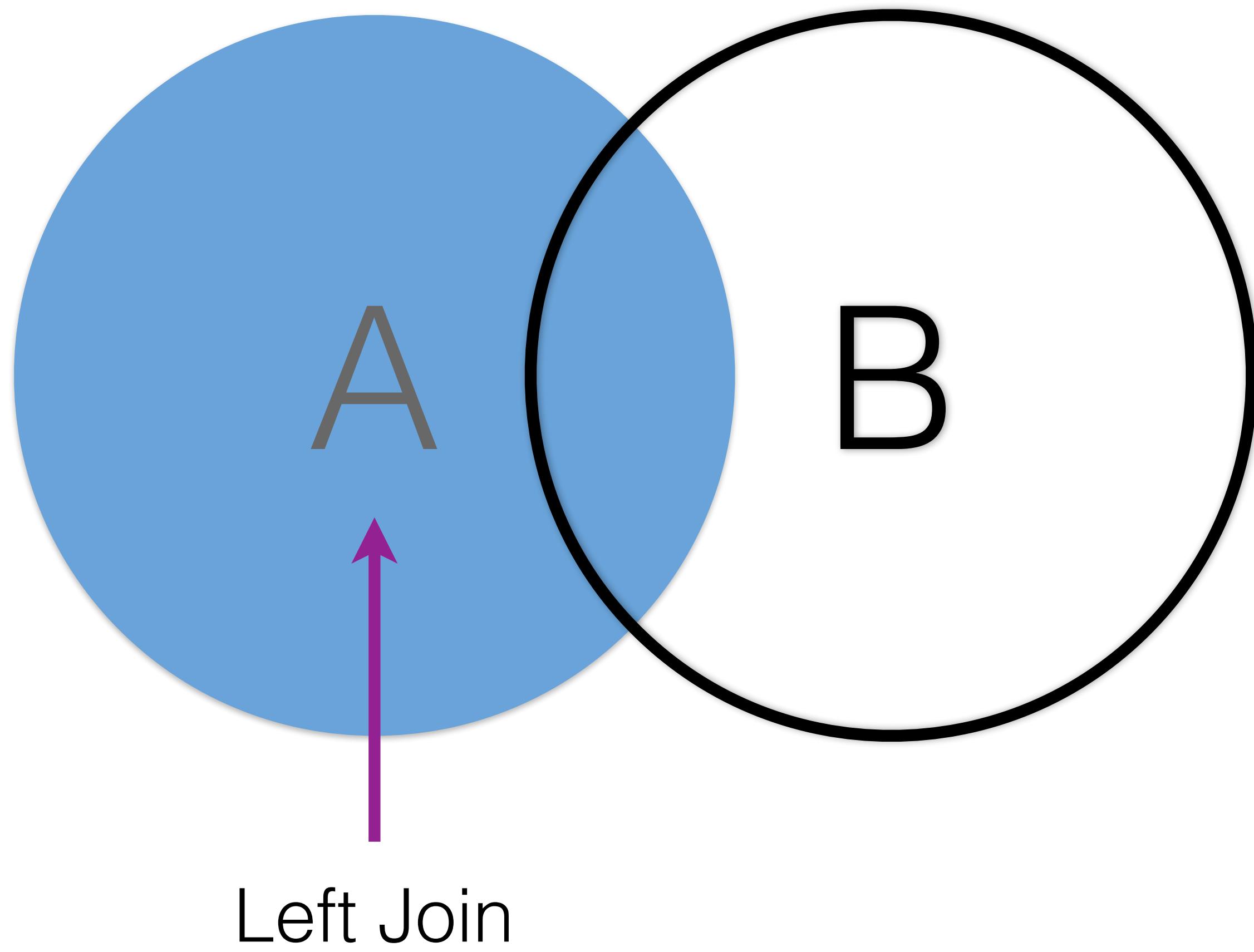


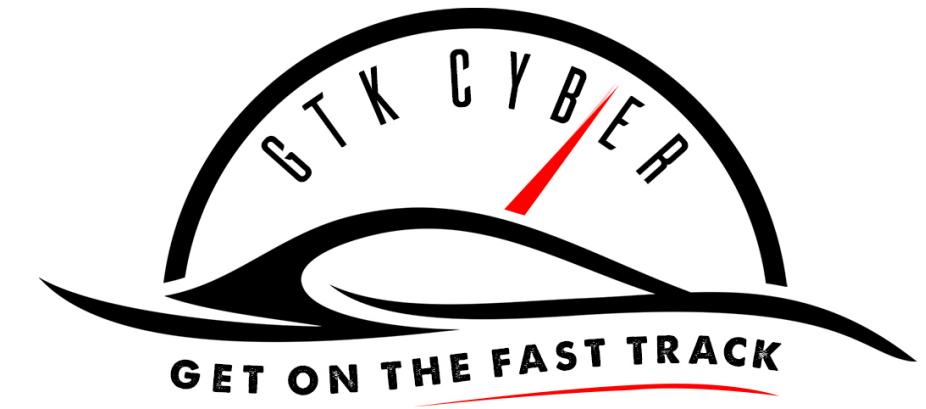
Inner Join (Intersection)



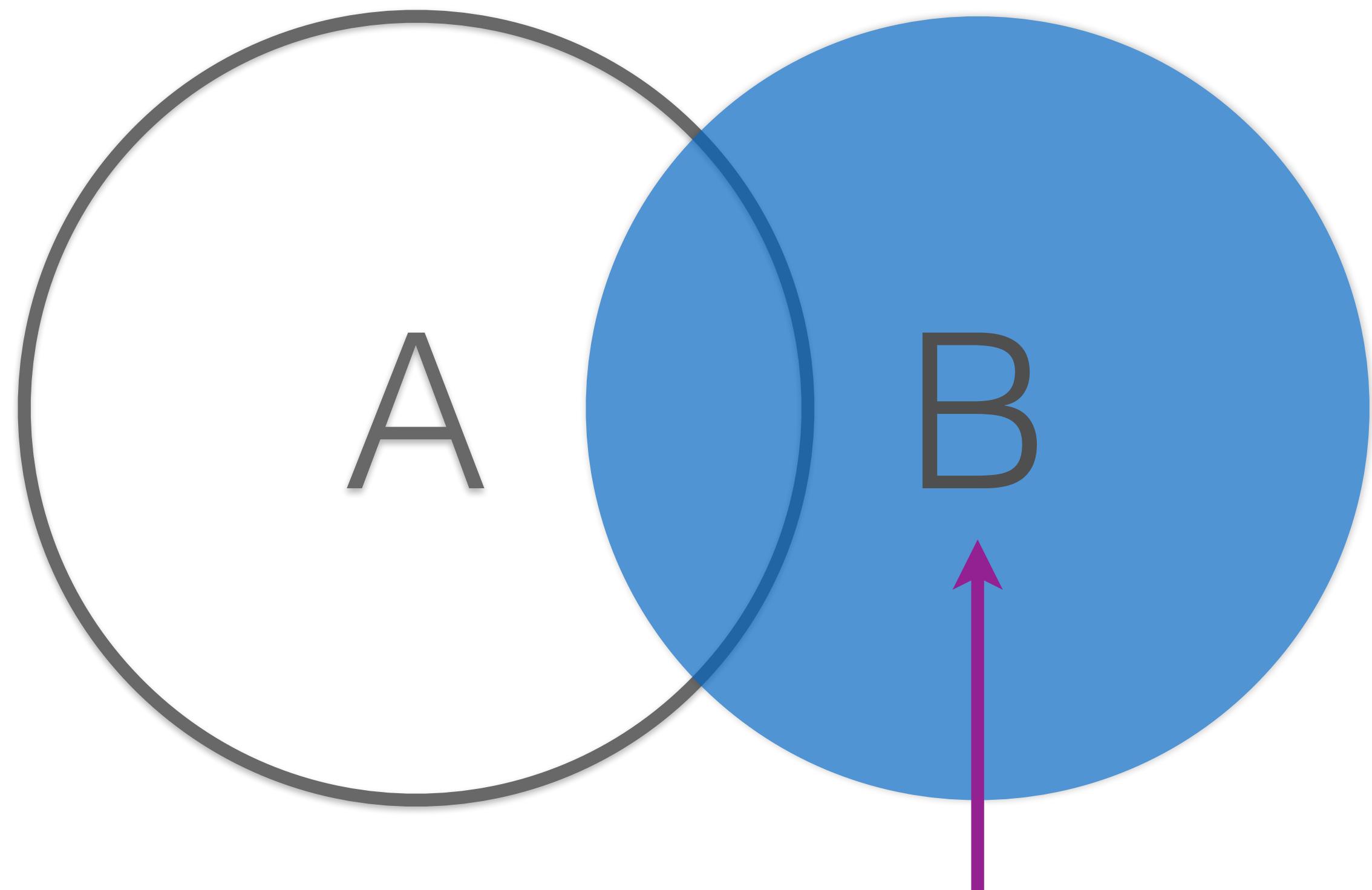


Left Join

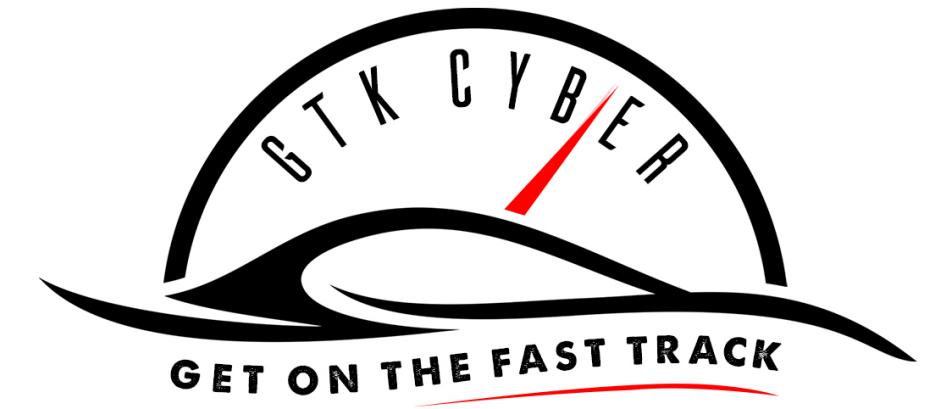




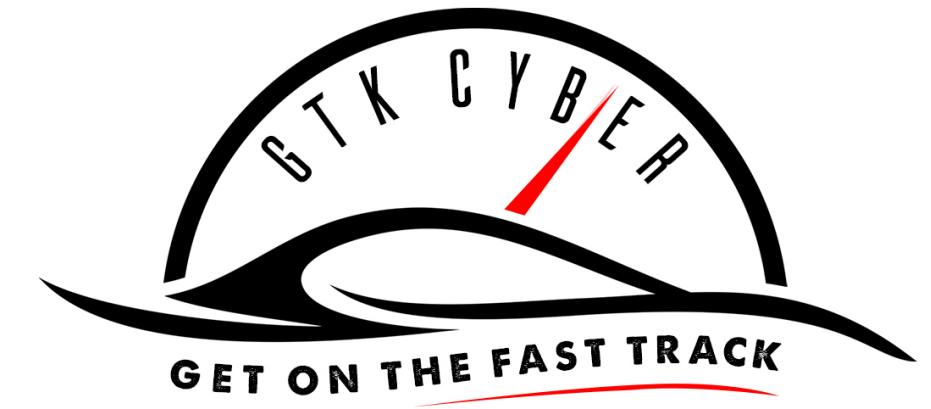
Right Join



Right Join



```
pd.merge( leftData, rightData )
```

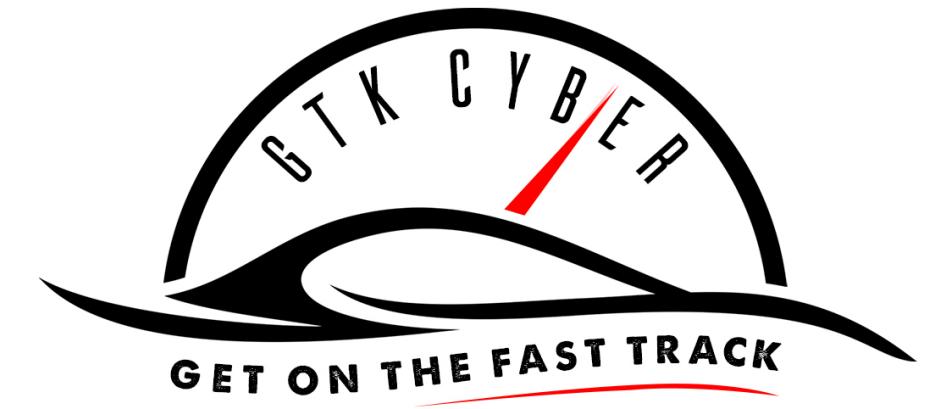


```
pd.merge( leftData, rightData, how="<join type>" )
```

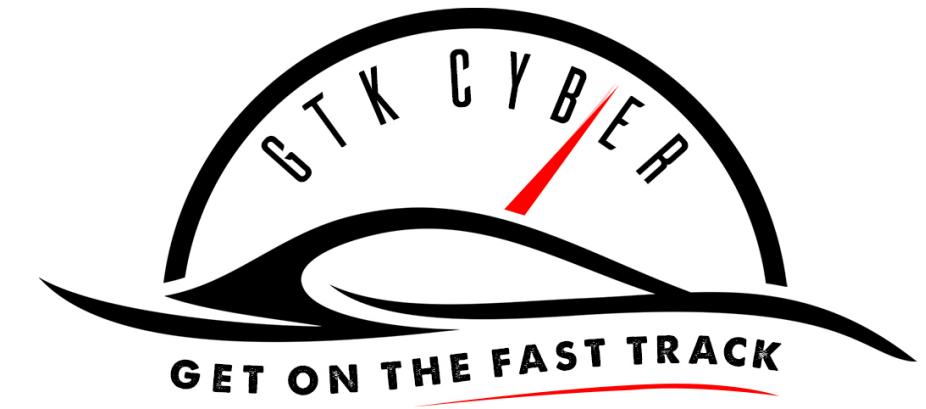


`pd.merge(leftData, rightData, how="<join type>")`

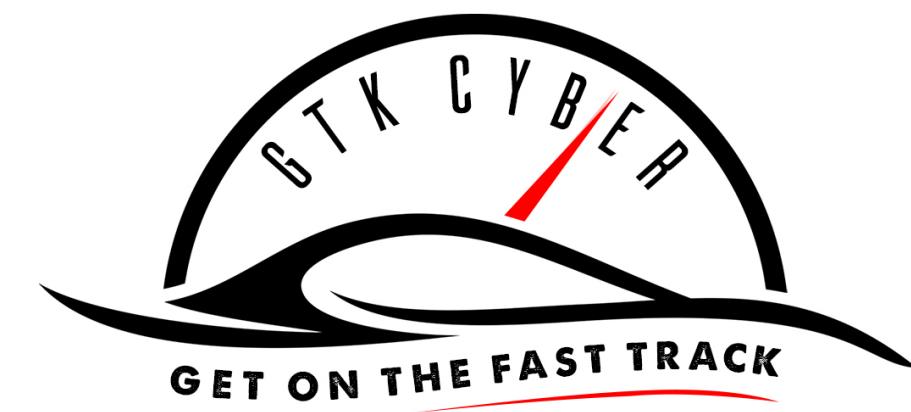
Option	SQL Equivalent	Description
inner	INNER JOIN	Intersection
left	LEFT OUTER JOIN	Returns items in Set A, but not in Set B
right	RIGHT OUTER JOIN	Returns items in Set B, but not in Set A
outer	FULL OUTER JOIN	Returns the union of both sets



```
pd.merge( leftData, rightData,  
how="<join type>",<br>  
on=<field list> )
```



Grouping and Aggregating Data



```
df_grouped = df.groupby(  
['Protocol', 'Source', 'Destination'])  
print(df_grouped.size())
```

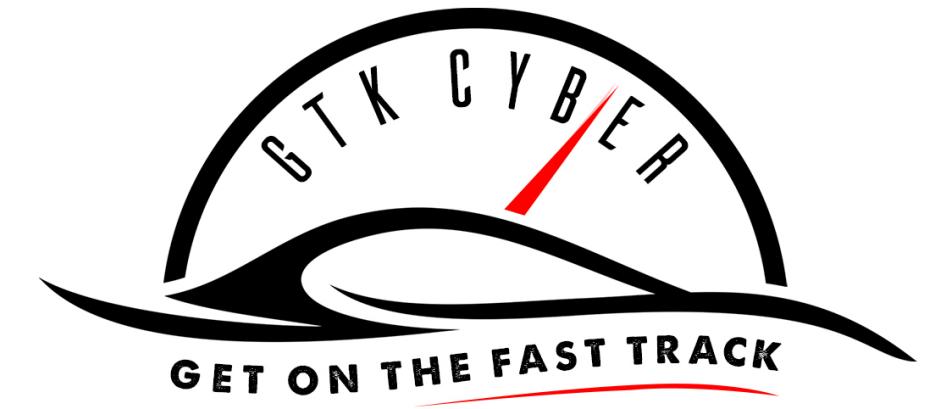
Protocol	Source	Destination	
ARP	MoxaTech_27:8c:37	Westermo_1a:61:83	610
	Pegatron_3a:0d:e8	Ruggedco_64:85:c2	871
	Ruggedco_64:85:c2	Pegatron_3a:0d:e8	871
	SiemensN_27:64:11	Siemens-_89:59:82	428
	Westermo_1a:61:83	MoxaTech_27:8c:37	610
DNS	192.168.88.1	192.168.88.61	9938
	192.168.88.61	192.168.88.1	9937
	192.168.89.2	8.8.8.8	2194
ICMP	192.168.89.1	192.168.89.2	2194
LOOP	CiscoInc_95:1d:8b	CiscoInc_95:1d:8b	1992
S7COMM	10.10.10.10	10.10.10.20	19913
	10.10.10.20	10.10.10.10	19914
TCP	10.10.10.10	10.10.10.20	23409
	10.10.10.20	10.10.10.10	59739

Name: Protocol, dtype: int64



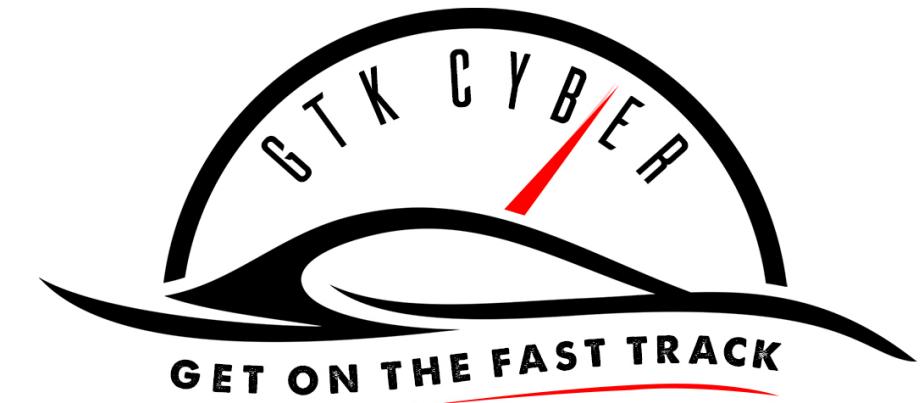
```
Stats_Packets = df_grouped['Length'].agg( { 'No  
Packets': len, 'Volume': sum, \  
    'SD': lambda x: np.std(x, ddof=1) if  
len(x)>2 else 0} )
```

Protocol	Source	Destination	Volume	SD	No Packets
ARP	MoxaTech_27:8c:37	Westermo_1a:61:83	45140	0.000000	610
	Pegatron_3a:0d:e8	Ruggedco_64:85:c2	52260	0.000000	871
	Ruggedco_64:85:c2	Pegatron_3a:0d:e8	52260	0.000000	871
	SiemensN_27:64:11	Siemens-_89:59:82	25680	0.000000	428
	Westermo_1a:61:83	MoxaTech_27:8c:37	36600	0.000000	610
DNS	192.168.88.1	192.168.88.61	725474	0.000000	9938
	192.168.88.61	192.168.88.1	725401	0.000000	9937
	192.168.89.2	8.8.8.8	152391	1.442727	2194
ICMP	192.168.89.1	192.168.89.2	213823	1.442727	2194
LOOP	CiscoInc_95:1d:8b	CiscoInc_95:1d:8b	119520	0.000000	1992
S7COMM	10.10.10.10	10.10.10.20	2070952	0.000000	19913
	10.10.10.20	10.10.10.10	3046842	0.000000	19914
TCP	10.10.10.10	10.10.10.20	2280712	15.683145	23409
	10.10.10.20	10.10.10.10	5436249	43.840987	59739



In Class Exercise:

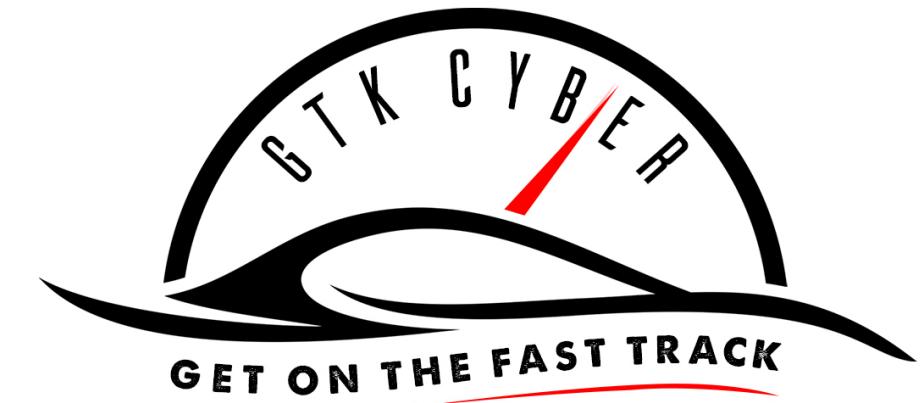
Please complete Worksheet 2: Exploring Two Dimensional Data



Exercise 1

```
#Data1
{"first_name": {"0": "Robert", "1": "Steve", "2": "Anne", "3": "Alice"},  
"last_name": {"0": "Hernandez", "1": "Smith", "2": "Raps", "3": "Muller"},  
"birthday": {"0": "5\\3\\67", "1": "8\\4\\84", "2": "9\\13\\  
91", "3": "4\\15\\75"}  
}
```

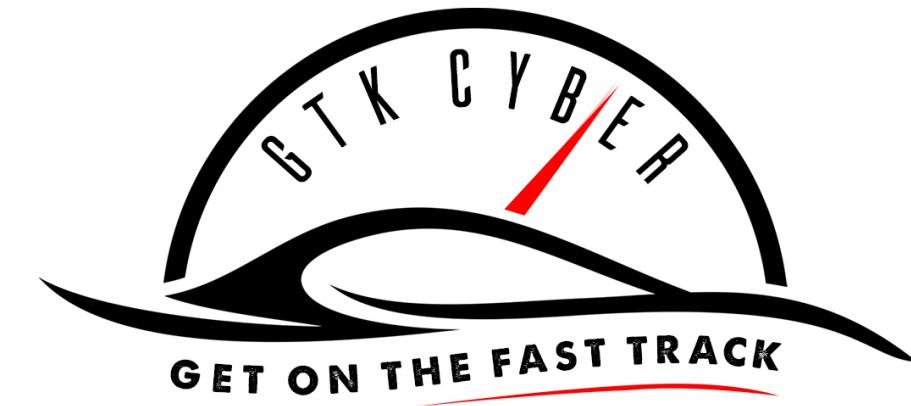
```
df1 = pd.read_json('..../data/data1.json')
```



Exercise 1

```
#Data2
[{"0": {"first_name": "Robert", "last_name": "Hernandez", "birthday": "5\\\"3\\
\\67"},  
"1": {"first_name": "Steve", "last_name": "Smith", "birthday": "8\\\"4\\\"
84"},  
"2": {"first_name": "Anne", "last_name": "Raps", "birthday": "9\\\"13\\\"91"},  
"3": {"first_name": "Alice", "last_name": "Muller", "birthday": "4\\\"15\\\"
75"}}
```

```
df2 = pd.read_json('..../data/data2.json', orient='index')
```



Exercise 1

```
#Data3
[
  {"first_name": "Robert", "last_name": "Hernandez", "birthday": "5\\3\\
67"},  

  {"first_name": "Steve", "last_name": "Smith", "birthday": "8\\4\\84"},  

  {"first_name": "Anne", "last_name": "Raps", "birthday": "9\\13\\91"},  

  {"first_name": "Alice", "last_name": "Muller", "birthday": "4\\15\\75"} ]  
  
df3 = pd.read_json('..../data/data3.json',
orient='columns')
```



Exercise 1

```
#Data4
{"columns": ["first_name", "last_name", "birthday"],
"index": [0,1,2,3],
"data": [ ["Robert", "Hernandez", "5\\3\\67"], ["Steve", "Smith", "8\\4\\84"], ["Anne", "Raps", "9\\13\\91"], ["Alice", "Muller", "4\\15\\75"] ]}
```

```
df4 = pd.read_json('..../data/data4.json', orient='split')
```



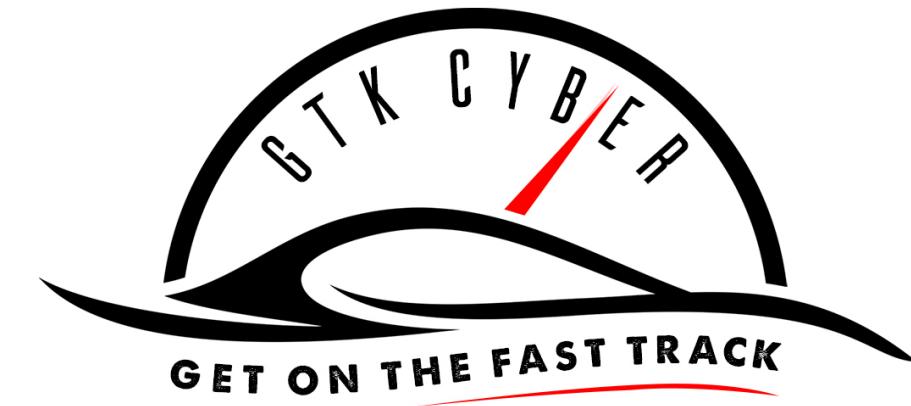
Exercise 2

#Write the functions

```
def get_os(x):  
    user_agent = parse(x)  
    return user_agent.os.family
```

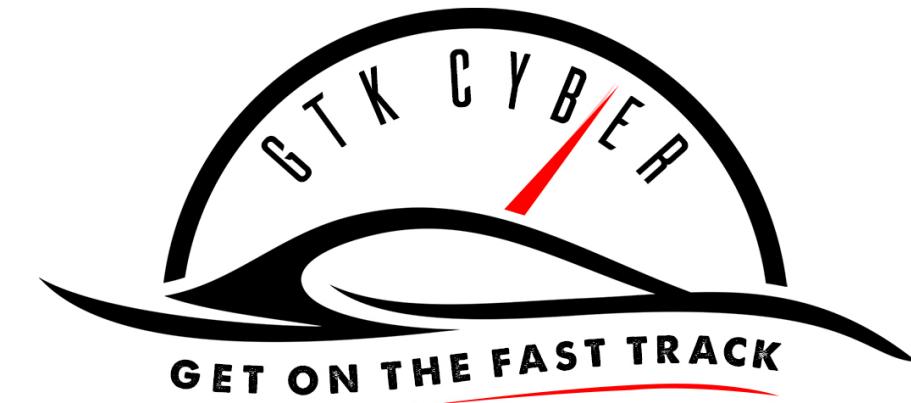
```
def get_browser(x):
```

```
    user_agent = parse(x)  
    return user_agent.browser.family
```



Exercise 2

```
#Apply the functions to the dataframe
server_df['os'] =
server_df['request_header_user_agent'].apply( get_os )
server_df['browser'] =
server_df['request_header_user_agent'].apply( get_browser )
```



Exercise 2

```
#Apply the functions to the dataframe
server_df['os'] =
server_df['request_header_user_agent'].apply( get_os )
server_df['browser'] =
server_df['request_header_user_agent'].apply( get_browser )

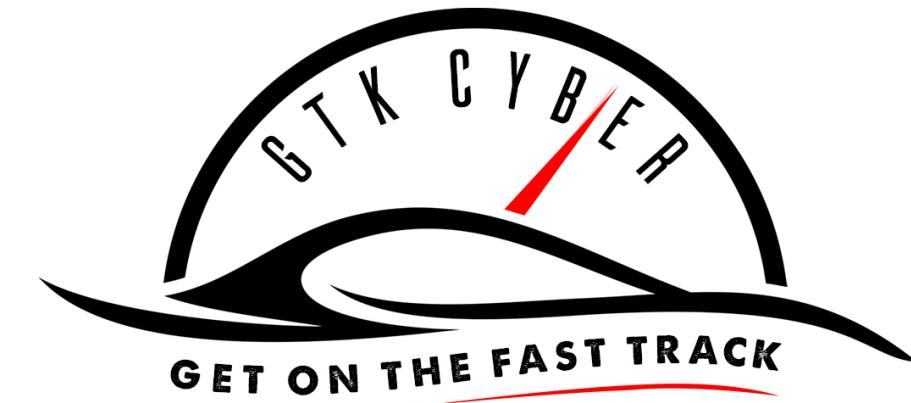
#Get the top 10 values
server_df['os'].value_counts().head(10)
```



Exercise 2

```
#Get the top 10 values
server_df['os'].value_counts().head(10)
```

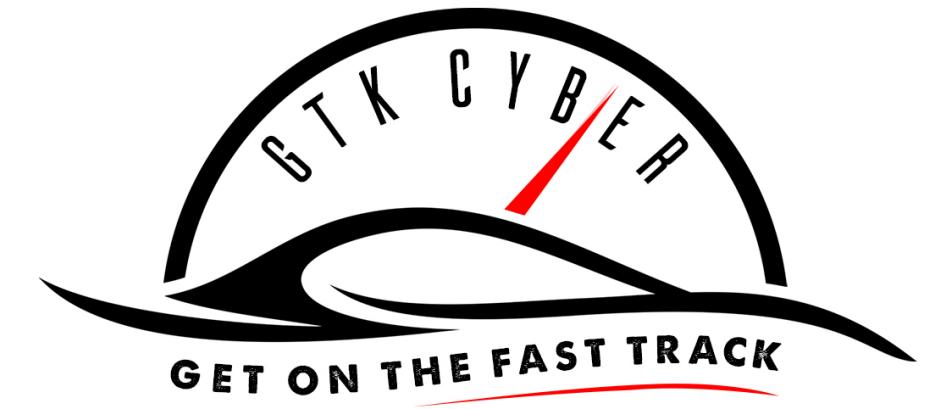
Windows 7	2041
Windows Vista	500
Windows XP	423
Windows 8.1	221
Linux	125
Mac OS X	80
Chrome OS	60
Ubuntu	6



Exercise 3

```
bots = pd.read_csv('..../data/dailybots.csv')
gov_bots = bots[['botfam', 'hosts']][bots['industry'] ==
"Government/Politics"]

gov_bots.groupby('botfam', as_index=False).sum()
```



Questions?