# HFTC

September 19, 2018

## 20 September 2018 ## Machine Learning and Big Data ## Druce Vertes

## 1 Agenda

- Some observations on big data and machine learning
- What is big data? How is machine learning different from statistics?
- A roadmap for applying machine learning - code examples, resources
- Some fun examples

Thesis: Nothing new under the sun, it's all just fashion, new words for what we always did.

And a dose of title inflation - Excel monkey -> Business Intelligence Analyst - SQL report jockey -> Data Scientist - Forecaster -> Machine learning scientist, AI researcher

More things change, the more they remain the same

Antithesis: It's the Singularity!

Small quantitative changes add up, and eventually reach a tipping point where you see a large qualitative change, like water turning to ice.

*"It is said that there are no sudden changes in nature, and the common view has it that when we speak of a growth or a destruction, we always imagine a gradual growth or disappearance. Yet we have seen cases in which the alteration of existence involves not only a transition from one proportion to another, but also a transition, by a sudden leap, into a...qualitatively different thing; an interruption of a gradual process, differing qualitatively from the preceding, the former state"*

*G. W. F. Hegel (1770-1831)*

German philosopher, first to write about the hype cycle and the tipping point

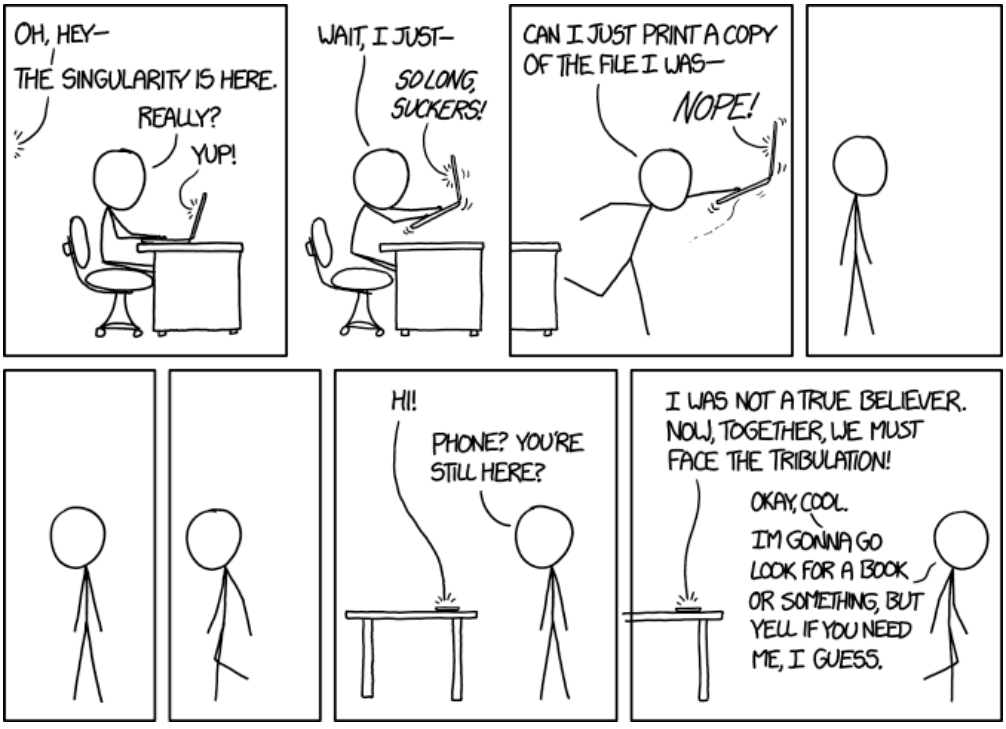
Many ML algorithms go back decades, we had most of the pieces but recently hit takeoff velocity:

- Better hardware (especially GPUs)
- Better algorithms (efficient backpropagation, new neural network architectures)
- Much more data from digital everything

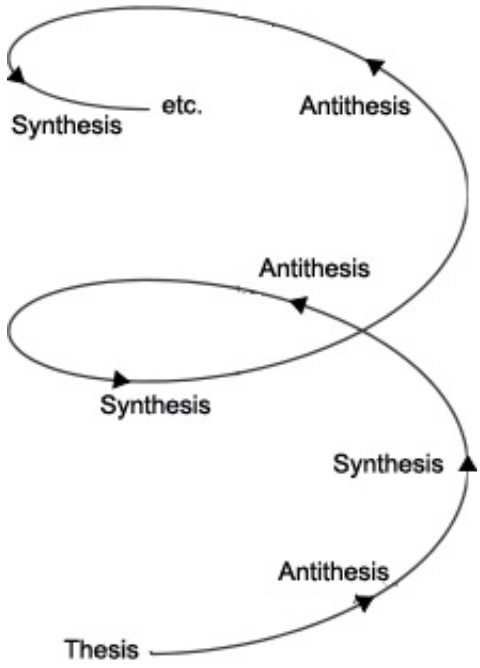
Virtuous cycle - AlexNet (2012), a first sign of spring, very successful image classification deep NN - -> More research - -> More deployed apps, speech recognition, image recognition, machine translation, self-driving cars, robots, - -> More data, more investment in hardware and algos - -> Huge advances in many applications - -> Huge growth in investment in research, better hardware, perpetuating the cycle - -> General AI and beyond? (Or a new AI winter?)

Synthesis -


Narayanan2.png



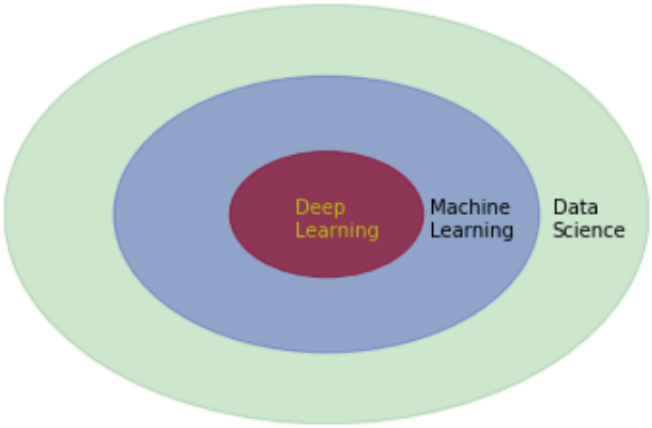

Singularity



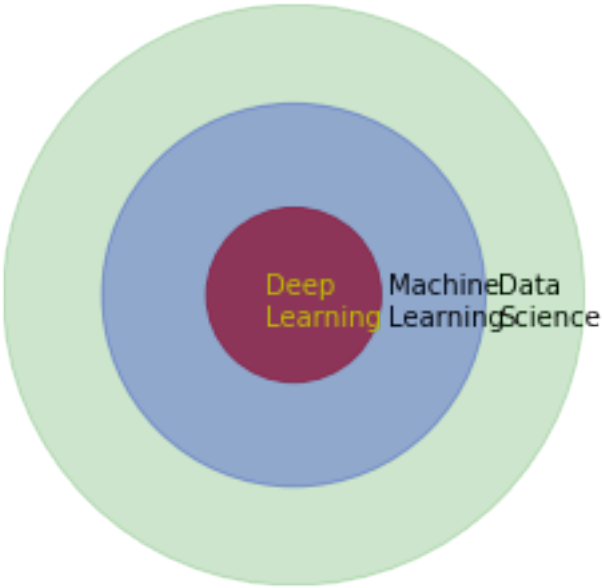

heg2.jpg

## 2 An Evolutionary Process

```
In [96]: import matplotlib.pyplot as plt

         circle3 = plt.Circle((0.5, 0.5), 0.5, color='g', alpha=0.2)
         circle2 = plt.Circle((0.5, 0.5), 0.33, color='blue', alpha=0.3)
         circle1 = plt.Circle((0.5, 0.5), 0.15, color='r', alpha=0.8)

         fig = plt.figure()
         ax = fig.add_subplot(111, aspect='equal')

         # (or if you have an existing figure)
         # fig = plt.gcf()
         # ax = fig.gca()

         ax.add_artist(circle1)
         ax.add_artist(circle2)
         ax.add_artist(circle3)
         ax.annotate('Deep\nLearning', xy=(0.45, 0.45), color = 'y')
         ax.annotate('Machine\nLearning', xy=(0.66, 0.45))
         ax.annotate('Data\nScience', xy=(0.85, 0.45))

         plt.axis('off')

         plt.show()
```




subsets.png




## 3 What is Machine Learning?

- Data science
- Anything involving data: collecting, scrubbing, structuring, feature engineering, visualization
- Machine learning
- Algorithms that learn from data without explicit programming
- Deep learning

fencing.jpg

- Many machine learning algos (neural net layers) trained end to end on a complex task
- Human like performance in a hard or uncertain domain: computer vision, natural language, games, self-driving cars
- Can magically outperform humans, but (for now) in a narrow domain, stable, non-adversarial context
- Strong general AI
- Human-like performance in broad, highly variable, adversarial context
- Still a pipe dream:
  - We still segregate robots on factory floor for safety
  - Algorithms often easy to defeat in messy real world situations like self-driving cars

Log scale in terms complexity of how much data you need: - Machine learning: simple algos can do something useful with hundreds or thousands of data points - Deep learning: typically millions of data points - General AI ??? billions ???

# 4 Machine Learning = Statistics For Street Fighting

## 4.1 Statistics

## 4.2 Machine Learning

# 5 When in doubt, use brute force - Ken Thompson

# 6

```
In [ ]: Just think of yourself as Batman
```


streetfight.jpg

xx

Batman

# 7 I do nothing that a man of unlimited funds, superb physical endurance, and maximum scientific knowledge could not do. - Batman

# 8 It's sometimes difficult to think clearly when you're strapped to a printing press. - also Batman

# 9 Statistics:

## 9.1 Descriptive + inferential statistics:

## 9.2 - What can we say about data that's true?

## 9.3 - What can we infer about future data?

# 10 Machine Learning:

## 10.1 Predict as accurately as possible, unconstrained by data and computing resources

# 11 Statistical prediction and inference - simple regression

## 11.1 - You have some data

## 11.2 - You fit a line, get a slope and an intercept

## 11.3 - Get some analysis of variation - ANOVA

In [1]:
```
import plotly
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
from plotly.graph_objs import *
from plotly.figure_factory import create_table
init_notebook_mode(connected=True)

import numpy as np
import pandas as pd
import scipy

import statsmodels
import statsmodels.api as sm
from statsmodels.formula.api import ols

# create a data set, sin wave plus random noise
nobs = 4000
x = np.linspace(0, 6*np.pi, num=nobs)
y = -np.cos(x) + x*0.05 + np.random.normal(0, 0.25, nobs)
z = np.sin(x) +  x*0.05 + np.random.normal(0, 0.25, nobs)

df = pd.DataFrame({'x' : x,'y': y,'z': z})

# chart it
```

```
        def mychart(*args):

            # pass some 2d n x 1 arrays, x, y, z

            # 1st array is independent vars
            # reshape to 1 dimensional array
            x = args[0].reshape(-1)

            # following are dependent vars plotted on y axis
            data = []
            for i in range(1, len(args)):
                data.append(Scatter(x=x,
                                    y=args[i].reshape(-1),
                                    mode = 'markers',
                                    marker = dict(size = 2)
                                   ))

            layout = Layout(
                autosize=False,
                width=800,
                height=600,
                yaxis=dict(
                    autorange=True))

            fig = Figure(data=data, layout=layout)

            return iplot(fig) # , image='png' to save notebook w/static image
```

```
In [2]: mychart(x,y)
```

```
In [3]: #table = create_table(df)
        #iplot(table, filename='mydata')

        # Very Important
        ### These are  *in-sample* statistics on the training data
        ### Hence the disclaimer

        formula = 'y ~ x'
        model = ols(formula, df).fit()
        ypred = model.predict(df)
        model.summary()
```

```
Out[3]: <class 'statsmodels.iolib.summary.Summary'>
        """
                                    OLS Regression Results
        ==============================================================================
```

```
Dep. Variable:                      y   R-squared:                       0.115
Model:                            OLS   Adj. R-squared:                  0.115
Method:                 Least Squares   F-statistic:                     520.1
Date:                Wed, 19 Sep 2018   Prob (F-statistic):          2.44e-108
Time:                        08:57:26   Log-Likelihood:                -4523.5
No. Observations:                4000   AIC:                             9051.
Df Residuals:                    3998   BIC:                             9064.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0043      0.024     -0.180      0.857      -0.051       0.042
x              0.0497      0.002     22.807      0.000       0.045       0.054
==============================================================================
Omnibus:                     3223.807   Durbin-Watson:                   0.216
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              238.012
Skew:                          -0.009   Prob(JB):                     2.07e-52
Kurtosis:                       1.805   Cond. No.                         21.9
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
"""
```

```
In [4]: mychart(x,y,np.array(ypred))
```

# 12 Potential pitfalls of the OLS model

## 12.1 For traditional statistical inference, you want

- A valid, complete *a priori* linear model
- Normally distributed errors

## 12.2 Things go awry if your data does not match assumptions of OLS

- If your model is not correctly specified, your regression line may not be as predictive as it should be.
- If you don't have normally distributed errors, your p-values and inference about future distributions are wrong because your errors are not random.
- (Even if future is like the past (stationarity))

# 13 Violations of assumptions of OLS: Anscombe's Quartet

#### 13.0.1 These all have the same slope, intercept, R-squared

#### 13.0.2 You can take this further and construct crazy stuff

#### 13.0.3 The Datasaurus Dirty Dozen

## 13.1 You need to know what you are doing.

- Visualize your raw data and errors
- Sanity-check your model
- Use appropriate tests to avoid violations of assumptions of OLS:
- Nonlinearity
- Heteroskedascisity
- Autocorrelation
- Multicollinearity
- Other non-normality of errors

If you don't have a good linear model, you need to find a better model, transform the data, add variables. If your data violates the assumptions of OLS, you need to understand why and fix it. You can't just throw data at statistical models without knowing what you're doing.

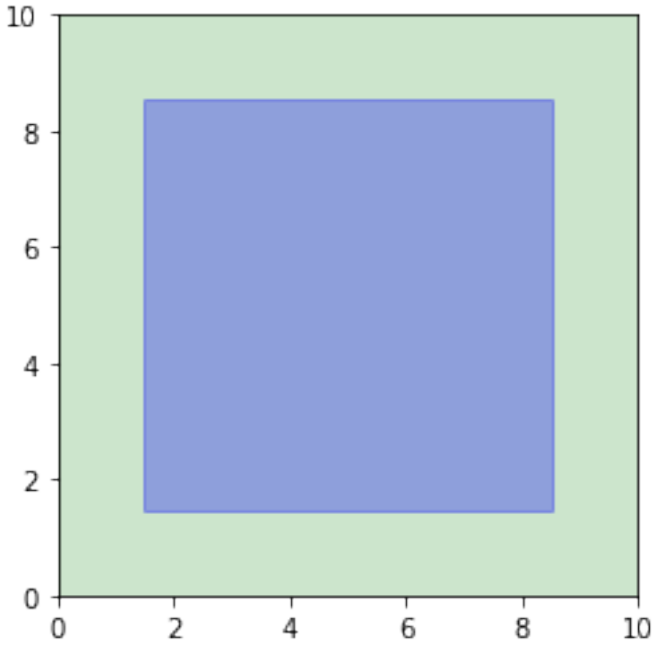
# 14 "With four parameters you can fit an elephant to a curve, with five you can make him wiggle his trunk" - John von Neumann

# 15 OLS is not very good when you throw a lot of predictors at it, and don't have a LOT of data

- Quick demo

```
In [95]: import matplotlib.pyplot as plt

         # set up the figure
         fig = plt.figure()
         ax = fig.add_subplot(111)
         ax.set_xlim(0,11)
         ax.set_ylim(0,11)

         # draw lines
         xmin = 0
         xmax = 10
         y = 5
         height = 1

         plt.hlines(y, xmin, xmax)
         plt.hlines(y, 2.5, 7.5, color='r')

         plt.vlines(xmin, y - height / 2., y + height / 2.)
```

```
plt.vlines(xmax, y - height / 2., y + height / 2.)
plt.vlines(2.5, y - height / 2., y + height / 2., color='r')
plt.vlines(7.5, y - height / 2., y + height / 2., color='r')

# add numbers
plt.text(xmin - 0.1, y, '0', horizontalalignment='right')
plt.text(xmax + 0.1, y, '100', horizontalalignment='left')
plt.axis('off')
plt.show()

fig = plt.figure()
ax = fig.add_subplot(111, aspect='equal')
plt.axis('on')
#plt.axes().set_aspect('equal', 'datalim')

offset = np.sqrt(50)/2

ax.set_xlim(0,10)
ax.set_ylim(0,10)

# (or if you have an existing figure)
# fig = plt.gcf()
# ax = fig.gca()

square1 = plt.Rectangle((0, 0), width=10, height=10, color='g', alpha=0.2)
square2 = plt.Rectangle((5-offset, 5-offset), width=offset*2, height=offset*2, color='b

ax.add_artist(square1)
ax.add_artist(square2)


plt.show()
```


```
In [7]: sz = 0.5 ** (1/3)
        start = (1-sz)/2
        end = start + sz
```

```
        print(start)
        print(end)
        data = [
            Mesh3d(
                x = [start, start, end, end, start, start, end, end],
                y = [start, end, end, start, start, end, end, start],
                z = [start, start, start, start, end, end, end, end],
                colorscale = [[0, 'rgb(255, 0, 255)'],
                              [0.5, 'rgb(0, 255, 0)'],
                                [1, 'rgb(0, 0, 255)']],
                intensity = [0, 0.142857142857143, 0.285714285714286,
                             0.428571428571429, 0.571428571428571,
                             0.714285714285714, 0.857142857142857, 1],
                i = [7, 0, 0, 0, 4, 4, 6, 6, 4, 0, 3, 2],
                j = [3, 4, 1, 2, 5, 6, 5, 2, 0, 1, 6, 3],
                k = [0, 7, 2, 3, 6, 7, 1, 1, 5, 5, 7, 6],
                name='y',
                showscale=True
            )
        ]
        layout = Layout(
            xaxis=plotly.graph_objs.layout.XAxis(
                title='x',
            ),
            yaxis=plotly.graph_objs.layout.YAxis(
                title='y',
                range=[0, 1]
            )
        )
        fig = Figure(data=data, layout=layout)
        iplot(fig, filename='3d-mesh-cube-python')
```

```
0.1031497370079501
0.8968502629920498
```

- When you increase the number of dimensions, most of the data are outliers
- Even if you increase the amount of data! Doesn't help
- Looked at another way, if 10% of your data on a dimension is an outlier:
- after 7 dimensions, most of your data is an outlier on some dimension (1-0.9^7)

```
In [9]: x1 = [n for n in range(1,50)]
        y1 = [(0.9 ** n) for n in range(1,50)]
        pd.DataFrame({'x' : x1, 'y' : y1})
```

Out[9]:

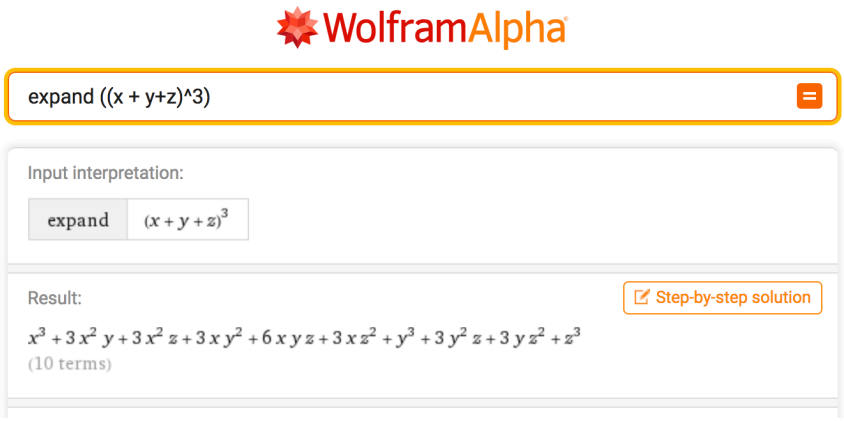
| | x | y |
|---|---|---|
| 0 | 1 | 0.900000 |
| 1 | 2 | 0.810000 |
| 2 | 3 | 0.729000 |

```
3    4  0.656100
4    5  0.590490
5    6  0.531441
6    7  0.478297
7    8  0.430467
8    9  0.387420
9   10  0.348678
10  11  0.313811
11  12  0.282430
12  13  0.254187
13  14  0.228768
14  15  0.205891
15  16  0.185302
16  17  0.166772
17  18  0.150095
18  19  0.135085
19  20  0.121577
20  21  0.109419
21  22  0.098477
22  23  0.088629
23  24  0.079766
24  25  0.071790
25  26  0.064611
26  27  0.058150
27  28  0.052335
28  29  0.047101
29  30  0.042391
30  31  0.038152
31  32  0.034337
32  33  0.030903
33  34  0.027813
34  35  0.025032
35  36  0.022528
36  37  0.020276
37  38  0.018248
38  39  0.016423
39  40  0.014781
40  41  0.013303
41  42  0.011973
42  43  0.010775
43  44  0.009698
44  45  0.008728
45  46  0.007855
46  47  0.007070
47  48  0.006363
48  49  0.005726
```

In [10]:
```
mychart(np.array(x1), np.array(y1))
```

- Curse of dimensionality
- Exponential growth is a powerful thing
- Underlies growth in hedge funds ... underlies encryption
- More dimensions you add, the more likely that you have an outlier along some dimension
- Outliers have a lot of leverage when you are minimizing squared error

- Suppose you want nonlinearity, 3 inflection points in each direction




- (I want to say C(deg+n, deg) but doesn't look right, left as an exercise)

## 15.1 Machine Learning: how you predict when you are no longer constrained by

- Data size - effectively unbounded data
- Computational complexity - effectively unbounded computation capacity

### 15.1.1 Train

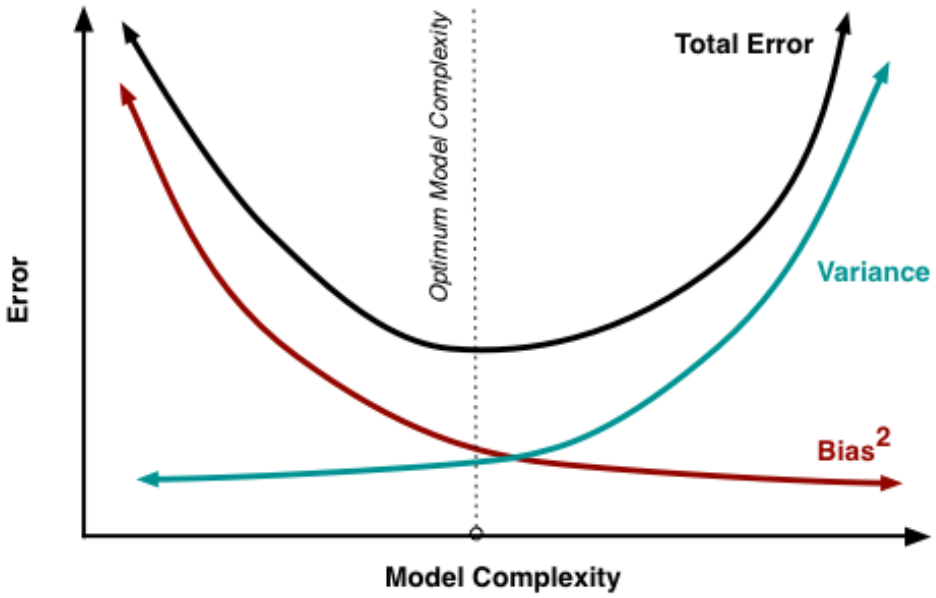
- Draw a training sample
- Train a machine learning models on training set (as many as you want)
- Try different algorithms
- Try different degree of complexity
  - Different degrees of nonlinearity
  - Different degrees of smoothing/regularization (will discuss further)

### 15.1.2 Cross-validate

- Draw a *cross-validation set*
- Take your trained models, use them to predict in cross-validation
- Choose the model that performs best in *cross-validation*
- Now, the act of picking a model contaminates the cross-validation set. Your best model is probably pretty good, but it's also probably one that was a little more lucky in cross-validation.

### 15.1.3 Evaluate

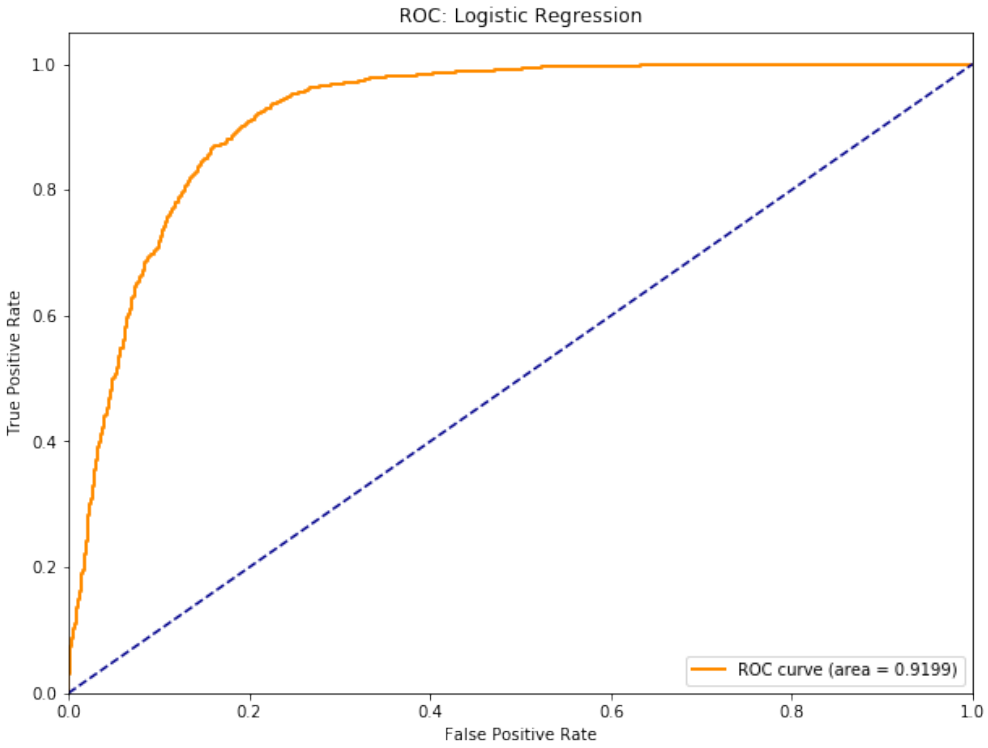
- Draw a *test set*




biasvariance.png

- Evaluate the performance of the model in the test set. Since it is effectively out-of-sample data, it's a reasonable estimate of performance to expect in out-of-sample data.

example? https://github.com/druce/Machine-learning-for-financial-market-prediction/blob/master/Replicate%20Paper%20by%20Rapach%2C%20Strauss%2C%20Tu%2C%20Zhou.ipynb https://github.com/druce/Machine-learning-for-financial-market-prediction/blob/master/Replicate%20Paper 4-Way%20Classification.ipynb


ROC: Logistic Regression
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.9199)

ROC.png

# 16 WAKE THE F**** UP!!!

## 16.1 3 key points

## 16.2 1. The Bias/Variance Tradeoff

# 17 2. The ROC curve

# 18 3. *The firewall principle:* Never make any decision to modify your model using the test set, and never use the training or cross-validation data sets to evaluate its out-of-sample performance.

# 19 1 key takeaway -

# 20 ML uses out-of-sample prediction error to identify optimal bias-variance tradeoff

# 21 If you know what this means and how to do it, you are a machine learning engineer

# 22 Regularization is very important in tuning your model and finding the right bias/variance tradeoff

- An example is L2 regularization in regression
- Instead of minimizing your mean-squared-error, minimize MSE + the sum of the squares of all your model coefficients
- This will tend to shrink your coefficients (and also equalize them). If a coefficient is large, it might be because it's very significant, or it might be because it's an outlier.
- Regularization increases *bias*, when you use the correct amount of it, you get worse fit in-sample, but better fit out-of-sample, because you are not chasing outliers. It makes your model more robust.
- With decision trees, you might use fewer or shallower trees, with ensemble models, you might use fewer models. The point is to make your model simpler, chase fewer outliers, be more robust, generalize better out-of-sample.
- The 2 key things in machine learning are
- Always pick model that performs best in xval (adjusted for complexity if possible)
- Worse is better: A lot of innovation in finding ways to make models resistant to outliers (more biased=worse) without impacting out-of-sample performance

# 23 Machine Learning paradigm vs Statistics

| Statistics | Machine learning |
|---|---|
| Small data | Big data (need a lot of data for complex models - exponential with #variables/kinks) |
| Optimize model in-sample statistics | Optimize out-of-sample statistics |
| Assume linear (or some a priori functional form) | Algorithm finds model |

| Statistics | Machine learning |
|---|---|
| Choose pre-dictors and form (par-si-mo-niously so you don't chase outliers) | Algorithm chooses from many pre-dictors and mod-els (as greed-ily as pos-sible while be-ing robust) |
| Optimize as much as possible | Worse is usu-ally better |
| Can't over-fit a par-si-mo-nious model with lim-ited data | Use reg-ular-iza-tion to tune and find opti-mal bal-ance be-tween bias and variance |

| Statistics | Machine learning |
|---|---|
| Inference description, prediction, attribution | Focus on prediction - attribution often opaque |
| You need to know what you are doing | You need to know what you are doing |

## 24 In statistics, you get not very good predictions and you know why they are not very good (simple model (bias), violations of assumptions of OLS)

## 25 In machine learning, you get good predictions but you don't know why they are very good ( complex models, opaque attribution)

- People talk about machine learning and complex models they go, ooh, you're data mining, overfitting: In fact, very focused on being robust, not chasing outliers, avoiding overfitting
- You can overfit with linear models
- Power poses - p-hacking - small sample sizes
- Ted talk - NY magazine
- With adequate data and good methodology you avoid overfitting

## 26 ML sometimes needs big data but what do we mean by big data?

### 26.1 Computational perspective

- Big data is data that doesn't fit in a box: a single computer's memory
- Google - canonical example
- Query gets multiplexed to dozens (100s?) of servers in a cluster, which each contain a shard of the index

- Results, including excerpts (!) get collected and sorted by relevance
- SERP gets built and returned in a fraction of a second
- By employing PageRank and this at the time novel cluster architecture which caches the entire Internet and its full-text index, Google achieved a quantum leap in web scale Internet services
- Big data is where you need to use clusters

### 26.2 You've come a long way, baby: current biggest AWS instance

These days big data has to be *REALLY REALLY BIG* before you *need* a Google cluster/Hadoop cluster approach.

Sometimes it might make sense to use the big data scale-out cluster approach as opposed to scale-up huge instances.

Big data can refer to this type of stack, even if the data is not big data in a Google / Facebook web-scale sense or in a computational sense.

### 26.3 Data science perspective

Big Data: - Sufficient data that you aren't constrained in terms of model complexity - Sufficient data that p-values are irrelevant because you can make them arbitrarily small by increasing sample size.

See also https://datascience.berkeley.edu/what-is-big-data/ - A lot of definitions boiling down to, size of data that was unreasonable in the PC era but tractable in cloud environment. Big enough for machine learning. A lot of people say big data when they mean machine learning, modern predictive analytics.

Suppose you can sample unlimited data and have unbounded computational capacity walk through an example

you sample a large set of training data try all the models sample a new data set of cross-validation and tune the models - train on training data - choose parameters based on cross-validation data finally - evaluate your model on test data

- Supervised
- Unsupervised
- Reinforcement learning

## 27 Supervised Learning

You have labeled data: a sample of ground truth with features and labels. You estimate a model that predicts the labels using the features. Alternative terminology: predictor variables and target variables. You predict the values of the target using the predictors.

- *Regression.* The target variable is numeric.
- Example: you want to predict the crop yield based on remote sensing data.
- Algorithms: linear regression, polynomial regression, generalized linear models.
- *Classification.* The target variable is discrete or categorical.

- Example: you want to detect the crop type that was planted using remote sensing data. Or Silicon Valley's "Not Hot Dog" application.
- Algorithms: Naïve Bayes, logistic regression, discriminant analysis, decision trees, random forests, support vector machines, neural networks of many variations: feed-forward NNs, convolutional NNs, recurrent NNs.

## 27.1 Unsupervised learning

You have a sample with unlabeled information. No single variable is the specific target of prediction. You want to learn interesting features of the data:

- Clustering. Which of these things are similar?
- Example: group consumers into relevant psychographics.
- Algorithms – k-means, hierarchical clustering.
- Anomaly detection. Which of these things are different?
- Example: credit card fraud detection.
- Algorithms: k-nearest-neighbor.
- Dimensionality reduction. How can you summarize the data in a high-dimensional data set using a lower-dimensional dataset which captures as much of the useful information as possible (possibly for further modeling with supervised or unsupervised algorithms)?
- Example: image compression.
- Algorithms: principal component analysis (PCA), neural network autoencoders.
- Representation: take a large body of text or movies or songs and create dense vectors describing them.
- Example: Recommendation engines, natural language processing (NLP)
- Algorithms: Word2vec, GlOVe

## 27.2 Reinforcement learning

You might think unlabeled/labeled pretty much covers all the bases.

Reinforcement learning can be viewed as 'meta' supervised learning.

You are presented with a game or real-world task that responds sequentially or continuously to your inputs, and you learn to optimize behavior in the form of a policy function to maximize an objective through trial and error.

- The algorithm goes out and explore the problem space and create its own high-level representation of the problem.
- Then it creates a policy to solve the problem
- Create a Markov decision matrix that maps the environment to actions - policy function.
- Label the matrix with the outcomes that followed.
- Then it optimize the matrix by gradient descent.

RL resembles supervised learning but at a higher level. - 'Imperfect' map from action to outcome - You get a big negative reinforcement when you screw up, run a stop sign, run over a pedestrian. - You get a positive reinforcement when you navigate to your destination safely. - From those high level labels, you have to model the game by finding high-level abstractions to describe situations you encounter, map them to actions using a policy function, and optimize the policy function based on the outcomes.

It's sufficiently important that nowadays people put it in its own category. (Also this is how you might implement a trading robot). https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-0-q-learning-with-tables-and-neural-networks-d195264329d0 https://www.youtube.com/watch?v=79pmNdyxEGo

## 28 More amazing examples -

Vision - Image classification - Style transfer - Google Quick, Draw! - Generating photorealistic images from (essentially) descriptions - Generating scenes using GANs

Audio - Google Duplex conversational agent - Spotify Discover Weekly

Natural language

- Embedding models More on embeddings
- Issues of bias Part deux
- Fund name generator

Reinforcement learning - Self-driving cars in the browser - What your self-driving car sees - DOTA 2 video Blog post - AlphaGo

Other great stuff - Mybridge - Product Hunt

Problems

Narrowness and brittleness of deep learning is an Achilles heel - doesn't work well in hostile environments - wave a stop sign at a self-driving car and it will stop, small problems may make it miss a stop sign, I don't think it will work in NY because people will just walk in front of cars. For most things you still need controlled environment. Hopefully we won't get disasters, pushback and decades of AI winter. - need a lot fo data

Big data: - computationally large: bigger than one machine - statistical: large enough for arbitrarily complex models without p-value problems - a stack: Spark/Hadoop etc.

Machine learning: - a menagerie of brute-force-ish methods for prediction - separation of training from tuning and testing

Deep learning: - neural networks are a particular algorithm of machine learning - deep learning is when you chain many neural networks and train the whole thing end-to-end - amazing emergent behavior like machine translation

HFs - better prediction using alt data - quant funds , HFT - crazy models like numerai

## 29 Learning resources

This deck is based on a couple of blog posts I did - https://alphaarchitect.com/2017/09/27/machine-learning-investors-primer/ - https://alphaarchitect.com/2018/06/05/machine-learning-financial-market-prediction-time-series-prediction-sklearn-keras/ - https://github.com/druce/Machine-learning-for-financial-market-prediction

Get the Anaconda distribution and dive in via a course or tutorials - https://conda.io/docs/user-guide/install/download.html

Python resources

Online books - http://www.diveintopython.net/ - http://learnpythonthehardway.org/ - http://code.google.com/edu/languages/google-python-class/introduction.html

Tutorials - http://www.learnpython.org/ - https://code.google.com/p/crunchy/

Other good reads - http://mirnazim.org/writings/python-ecosystem-introduction/ - http://wordaligned.org/articles/essential-python-reading-list#tocpython-tutorial - http://jessenoller.com/good-to-great-python-reads/ - https://bitbucket.org/gregmalcolm/python_koans/wiki/Home

Practice - https://codewars.com - solve simple problems in browser, run test cases, then see how other people solved it, often in a better way

MOOCs - https://www.datacamp.com/courses/intro-to-python-for-data-science - https://www.coursera.org/learn/machine-learning - https://see.stanford.edu/Course/CS229 - https://www.coursera.org/learn/neural-networks - https://lagunita.stanford.edu/courses/HumanitiesandScience/StatLearning/Winter2015/about (probably too easy for you) - https://www.coursera.org/learn/probabilistic-graphical-models - https://www.coursetalk.com/ - https://www.class-central.com/

Frameworks - http://scikit-learn.org/stable/ - https://keras.io/ - https://www.tensorflow.org/ - https://pytorch.org/

Textbooks

- https://www.amazon.com/Deep-Learning-Python-Francois-Chollet/dp/1617294438/
- https://www.amazon.com/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1491962291/
- https://www.amazon.com/Deep-Learning-Adaptive-Computation-Machine/dp/0262035618/ (theory textbook)
- https://web.stanford.edu/~hastie/Papers/ESLII.pdf
- https://www.amazon.com/Advances-Financial-Machine-Learning-Marcos/dp/1119482089

Blogs/roadmaps - http://yerevann.com/a-guide-to-deep-learning/ - http://ofir.io/How-to-Start-Learning-Deep-Learning/ - http://colah.github.io/ - http://karpathy.github.io/ - https://machinelearningmastery.com/blog/ - https://towardsdatascience.com/

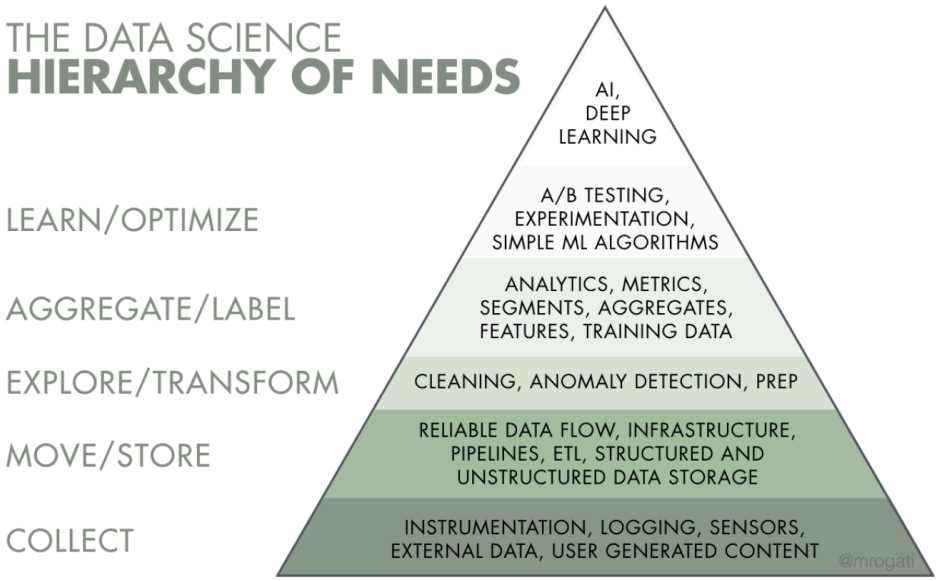
Data Science Hierarchy of Needs

Data engineering is 80% of the battle.

Alternative Data

- Everyone is selling your data ... Cell phone locations, free email apps, checkins, expense trackers, loyalty cards? just a way for our HF overlords to get richer

- It's 80%/20% data engineering / data science - this stuff is very resource-intensive
- Data is expense - $40m was a data budget I heard for big quant HFs like WorldQuant, DE Shaw, Two Sigma, RenTech
- If you think you have a great idea, Jim Simons probably already got there 10 years ago - jk but real edges fleeting, exclusive data / quant edge hard to find/ephemeral/rapidly evolving
- Bifurcation between the real quants like RenTech, Two Sigma which are basically tech firms, and fundamentalists, the latter will probably mostly use alt data research / data from 3rd parties, maybe have a couple of data scientist types internally for projects

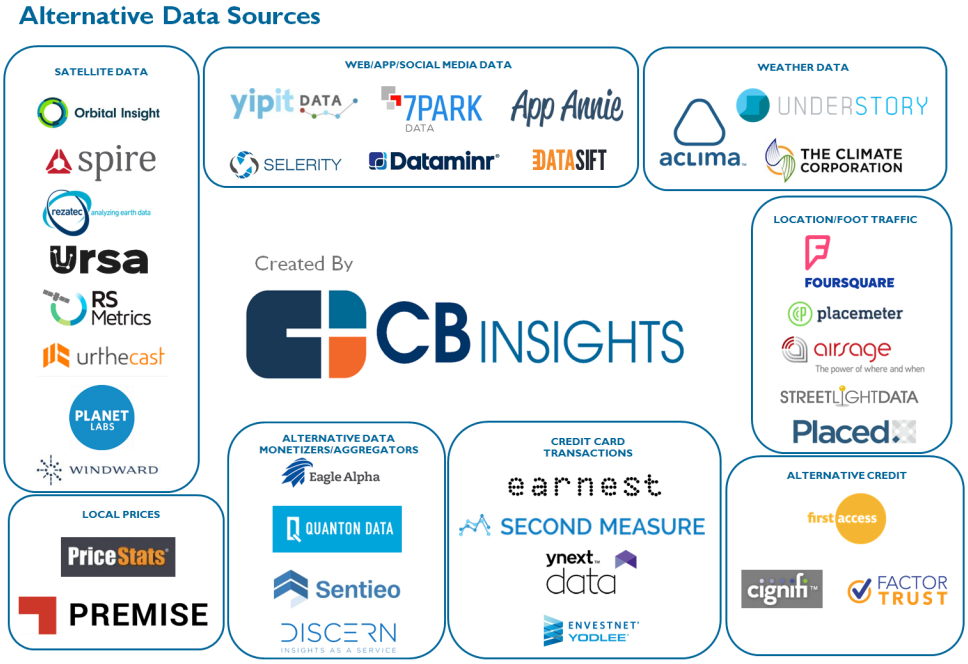


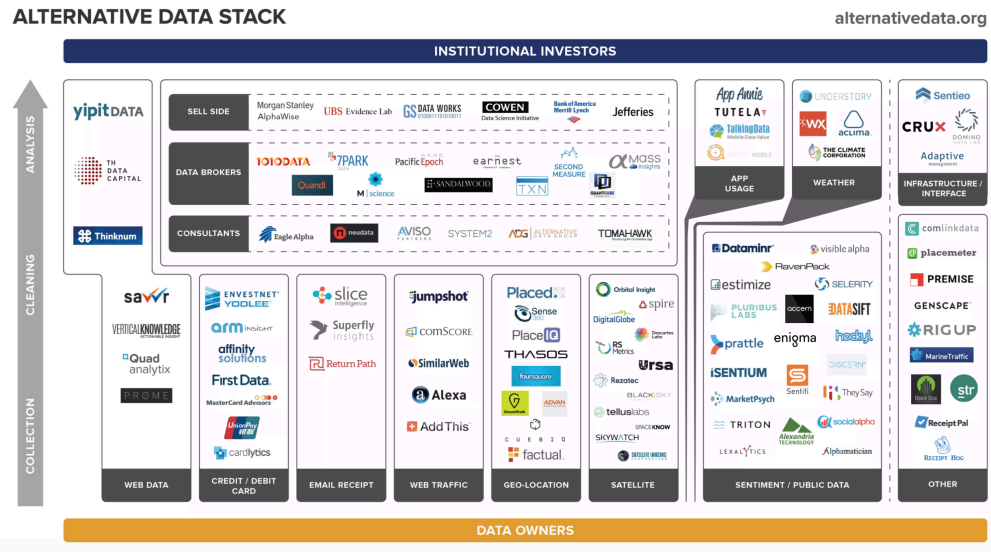

You need a solid foundation for your data before being effective with AI and machine learning.

image

- Might be transformative, if all economic activity is instrumented and feeds directly into nowcasts / markets
- Might be worst of 1984/Brave New World, everyone carrying a telescreen in their pocket which is as addictive as soma.

Alternative Data Landscape - http://alternativedata.org
http://mattturck.com/the-new-gold-rush-wall-street-wants-your-data/






image

Matt Turck - http://mattturck.com/wp-content/uploads/2017/05/Matt-Turck-FirstMark-2017-Big-Data-Landscape.png

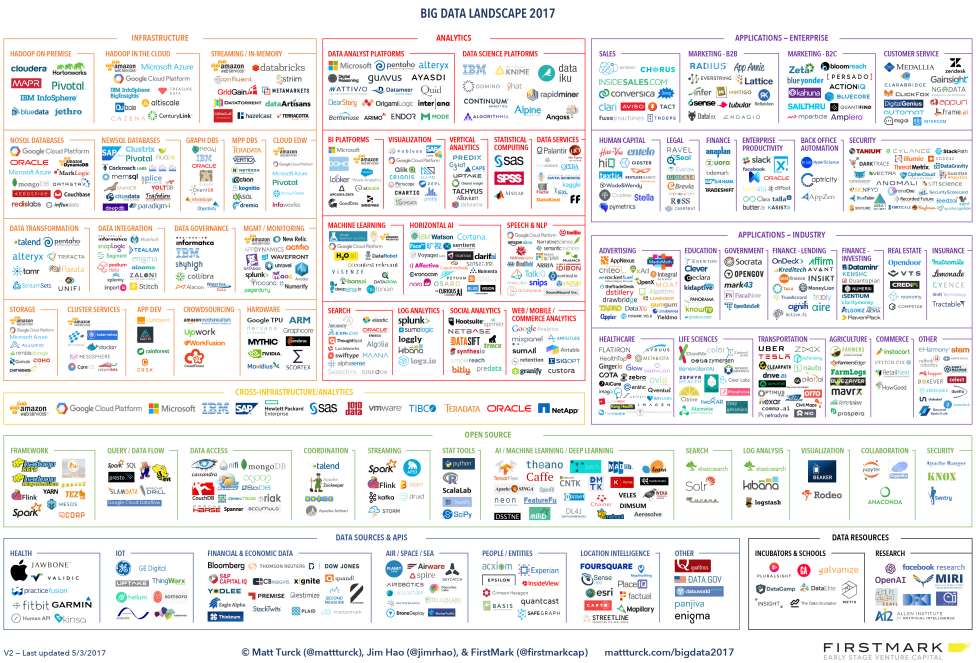
**30** ***The forecaster is a gentle man,***

**31** ***With neither sword nor pistol.***

**32** ***He walks along most daintily***

**33** ***Because his balls are crystal.***

- Follow [@streeteye](https://twitter.com/StreetEYE) on Twitter
- Blog http://blog.streeteye.com/blog
- https://www.linkedin.com/in/drucevertes/


BIG DATA LANDSCAPE 2017
INFRASTRUCTURE
ANALYTICS
APPLICATIONS – ENTERPRISE
APPLICATIONS – INDUSTRY
CROSS-INFRASTRUCTURE/ANALYTICS
OPEN SOURCE
DATA SOURCES & APIS
DATA RESOURCES
© Matt Turck (@mattturck), Jim Hao (@jimrhao), & FirstMark (@firstmarkcap) mattturck.com/bigdata2017
V2 – Last updated 5/3/2017
FIRSTMARK
EARLY STAGE VENTURE CAPITAL

image