How Machine Learning Can Help Improve Your Game Design

Tiago Tex Pine

@texpine

2018

Who am I

- Former Producer at Glu and other companies, mostly with the free-to-play model.
- Former Economy Designer at Gameloft and other companies, worked on several mobile free-to-play titles.
- Now Data Analyst at Bethesda.
  - **Disclaimer**: no data in this presentation is related to any Bethesda game. ☺
  - Demonstrations of data are always picked from external sources.
  - Applicability of each model is from my past experiences and lessons.

: What is Machine Learning.

- "Field of study that gives computers the ability to learn without being explicity programmed." - Arthur Samuel
- How does a baby learns?

*source: Pexels

: What is Machine Learning.

- "Field of study that gives computers the ability to learn without being explicity programmed." - Arthur Samuel
- How does a baby learns?

Observes

*source: Pexels

: What is Machine Learning.

- "Field of study that gives computers the ability to learn without being explicity programmed." - Arthur Samuel
- How does a baby learns?

Observes

Experiments

*source: Pexels

: What is Machine Learning.

- "Field of study that gives computers the ability to learn without being explicity programmed." - Arthur Samuel
- How does a baby learns?

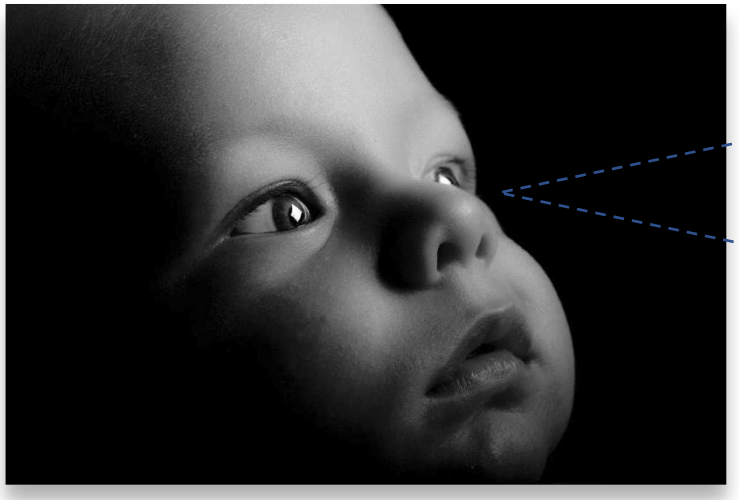
Observes

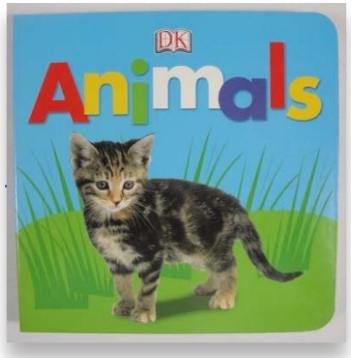
Does Experiments

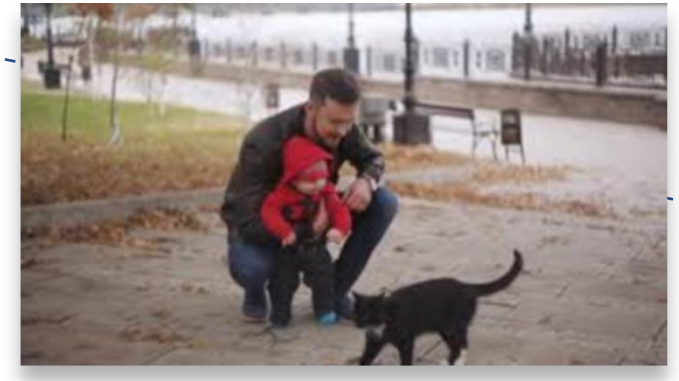
Gets Feedback

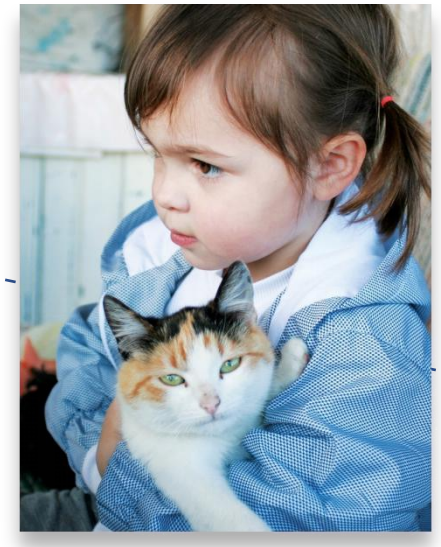
*source: Pexels

- Think about how many times a child needs to see images of a cat and cats in real life so he finally learns what the concept of a "cat" is.



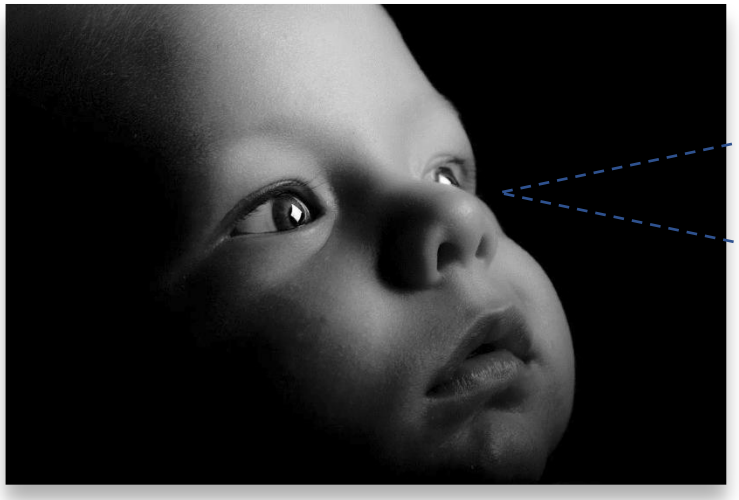




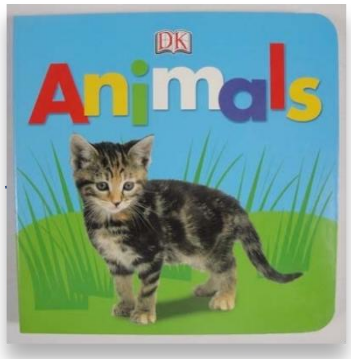


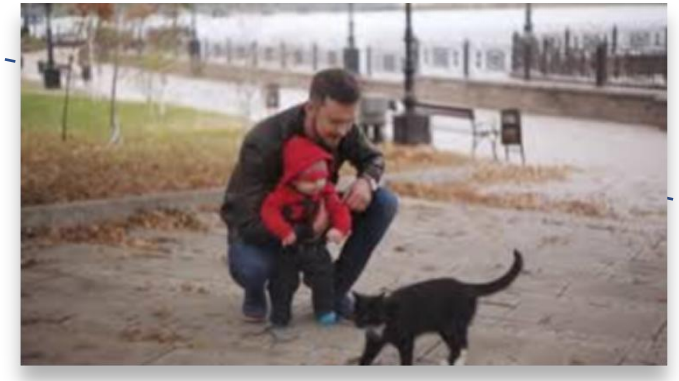
*source: Pexels

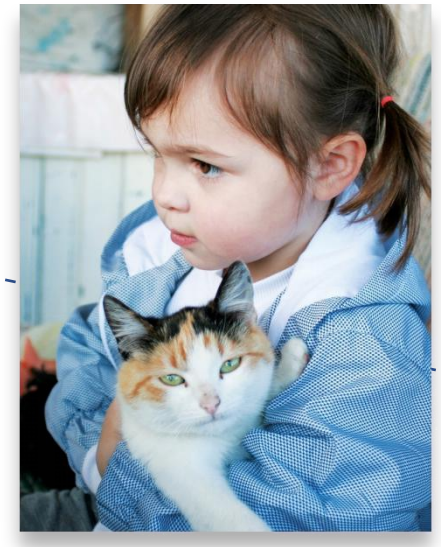
- Think about how many times a child needs to see images of a cat and cats in real life so he finally learns what the concept of a "cat" is.



Data!







In only 2 years, literally thousands if not millions of moments their minds take "snapshots" of what a cat is.

- Think about how many times a child needs to see images of a cat and cats in real life so he finally learns what the concept of a "cat" is.

Experiments

Gets Feedback

Learns

- Machine Learning are mathematical and statistical algorithms that learn like babies:
  - Absorb LOTs of data.
  - Do Experiments (iterations) to improve tuning.
  - Get Feedback from humans (or other algorithms).
  - Improves.

Scen.	Total Flows	Botnet Flows	Normal Flows	C&C Flows	Background Flows
1	2,824,636	39,933(1.41%)	30,387(1.07%)	1,026(0.03%)	2,753,290(97.47%)
2	1,808,122	18,839(1.04%)	9,120(0.5%)	2,102(0.11%)	1,778,061(98.33%)
3	4,710,638	26,759(0.56%)	116,887(2.48%)	63(0.001%)	4,566,929(96.94%)
4	1,121,076	1,719(0.15%)	25,268(2.25%)	49(0.004%)	1,094,040(97.58%)
5	129,832	695(0.53%)	4,679(3.6%)	206(1.15%)	124,252(95.7%)
6	558,919	4,431(0.79%)	7,494(1.34%)	199(0.03%)	546,795(97.83%)
7	114,077	37(0.03%)	1,677(1.47%)	26(0.02%)	112,337(98.47%)
8	2,954,230	5,052(0.17%)	72,822(2.46%)	1,074(2.4%)	2,875,282(97.32%)
9	2,753,884	179,880(6.5%)	43,340(1.57%)	5,099(0.18%)	2,525,565(91.7%)
10	1,309,791	106,315(8.11%)	15,847(1.2%)	37(0.002%)	1,187,592(90.67%)
11	107,251	8,161(7.6%)	2,718(2.53%)	3(0.002%)	96,369(89.85%)
12	325,471	2,143(0.65%)	7,628(2.34%)	25(0.007%)	315,675(96.99%)
13	1,925,149	38,791(2.01%)	31,939(1.65%)	1,202(0.06%)	1,853,217(96.26%)

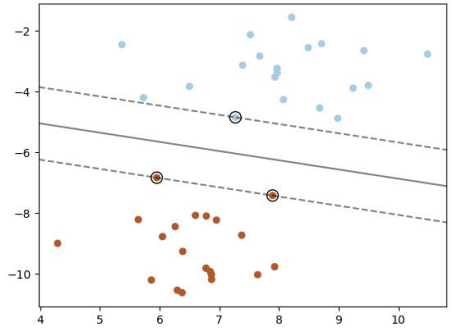
Data

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$
$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, \ldots, n$$

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$
$$\text{subject to } y^T \alpha = 0$$
$$0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

$$\text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho\right)$$

Iterates on Data



Gets Feedback (Tuning of *C*)

Improves Tuning.

So...

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$

$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, \ldots, n$$

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$

$$\text{subject to } y^T \alpha = 0$$
$$0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

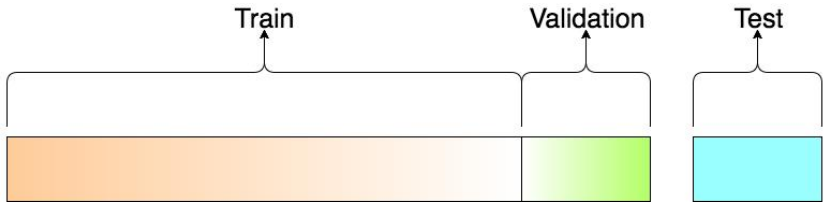
$$\operatorname{sgn}(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho)$$

=

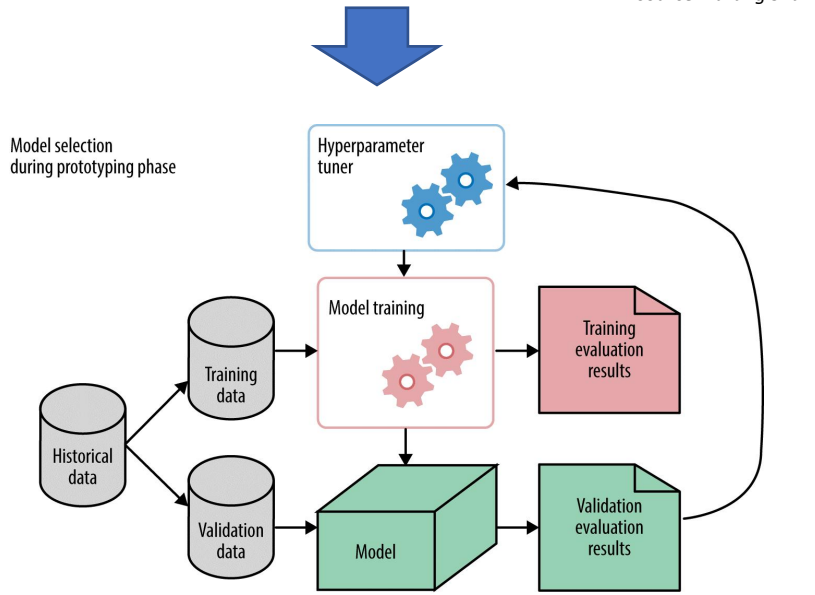
*source: Pexels

: Req 1 - You need data.

- Machine Learning = Statistical Learning.
- Needs data of enough volume to be statistically robust for the algorithms.
- Hopefully enough to split your data in 3 subsets (70%/15%/15%) and still have > 3,000 data points (table rows) in the smaller ones.
  - Those are sets to, respectively, train, test and validate the models.
- Anything *below 50 users is unusable* by any algorithm.
  - So no, you can't use ML in data from playtests and control groups.



*source: Tarang Shah



*source: Alice Zheng

: Req 2 - You need *quality* data.

- Your models will only be as good as the source of your data. Garbage in, garbage out.
- Design your trackers thoughtfully. Prioritize them.
  - Implementing analytics can be an "infinite" task...
- **Test them.**
  - Iterate with programmers to test and analyse precise conditions in which each tracker os being fired.
  - Sometimes programmers will make assumptions that may invalidate the validity of an entire event until the next app update.
- Use quality database services to receive and store your data.
  - Consider scalable cloud services with features designed to receive data streaming from thousands of clients.

: Req 3 - You need to engineer your data.

- Your data must be wrangled so that ML algorithms can understand: a big table with one data point per row, and one column per feature the ML model should use to learn. We call this columns "**features**" of the data.

  (In fact, 80% of all time of machine learning pratitioners is usually just cleaning and wrangling data to feed their baby models!)

*source: Pexels

Tracking Event of Purchase in shop

Timestamp	Player ID	Player Level	Item purchased	Item Level	Amount
2018-05-01 12:53:53	0f0f0872-d7c8-4fdd	1	Tires	2	1
2018-05-01 12:53:54	1b5715da-2e97-43	2	Nitro	3	5
2018-05-01 12:58:01	22edcfcc-db5b-437	2	Gas	1	10
2018-05-01 12:58:21	439159ba-4161-4d	1	Suspension	2	1
2018-05-01 12:58:22	c7bcb7ae-616e-480	5	Sedan Car	2	1
2018-05-01 13:01:11	80c56126-fff6-4f9a-	4	Sports Car	5	1
2018-05-01 13:01:15	ddb27f9f-de44-442	7	Nitro	5	5
2018-05-01 13:01:16	8af79539-d2f0-40c	2	Tires	1	1
2018-05-01 13:01:51	6ff03879-5377-480	3	Mechanic Upgrade	4	1

Tracking Event of inventory

Timestamp	Player ID	Player Level	Item owned	Item Level	Amount
2018-05-01 12:53:53	0f0f0872-d7c8-4fdd	1	Sedan Car	1	1
2018-05-01 12:53:53	0f0f0872-d7c8-4fdd	1	Engine	2	1
2018-05-01 12:53:53	0f0f0872-d7c8-4fdd	1	Gas	1	18
2018-05-01 12:53:53	0f0f0872-d7c8-4fdd	1	Suspension	2	2
2018-05-01 13:01:11	c7bcb7ae-616e-480	5	Sports Car	2	1
2018-05-01 13:01:11	c7bcb7ae-616e-480	5	Suspension	5	1
2018-05-01 13:01:11	c7bcb7ae-616e-480	5	Nitro	5	5
2018-05-01 13:01:11	c7bcb7ae-616e-480	5	Tires	1	1

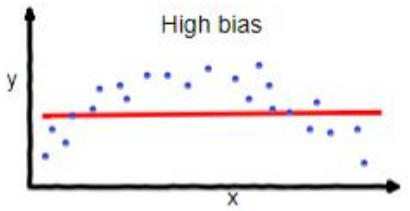
Each column is a "feature" / "dimension"

Data engineered to feed a Predictor of purchases

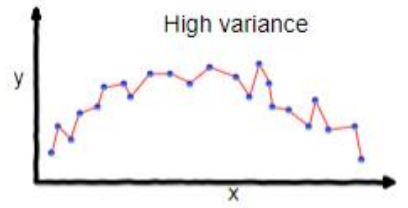
Player ID	Player Level	Purchased	Pur level	Pur amount	Inventory 1	Inv 1 level	Inv 1 amount	Inventory 2	Inv 2 level	Inv 2 amount	<etc...>
0f0f0872-d7c8-4fdd-8c	1	Tires	2	1	Sedan Car	1	1	Engine	2	1	...
c7bcb7ae-616e-4805-8	5	Sports Car	5	1	Sports Car	2	1	Suspension	5	1	...
<etc>	...	...	...	...	...	...	...	...	...	...	...

: Goal - You are looking for *generalizations*.

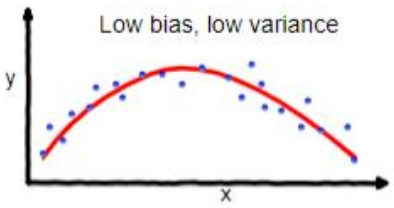
- The best ML models are the ones that are capable of generalizing. They are neither trying ot be 100% precise nor too broad.
  - Bias vs. Variance tradeoff
  - Overfitting vs. Underfitting
- When properly tuned, they provide generalizations that are more robust statistics than Averages.
  - KPIs based on Averages are common and easy to explain
  - But *dangerous* simplifications.
  - *"When Bill Gates enter in a bar..."*



underfitting



overfitting



Good balance

- Finally, understand models of statistical learning that make good generalizations *will* eventually misclassify very particular cases that not represented enough in the training data.
- Tell your manager "That's OK!"

*source: DARPA

"Machine Learning models are statistically impressive, but individually unreliable."

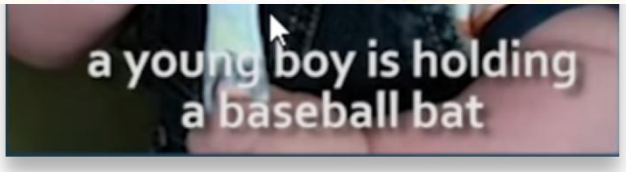
- John Launchbury, the Director of DARPA's Information Innovation Office

- Finally, models that make good generalizations will eventually misclassify or misshandle particular cases - or *outliers*.
- Tell you

idually

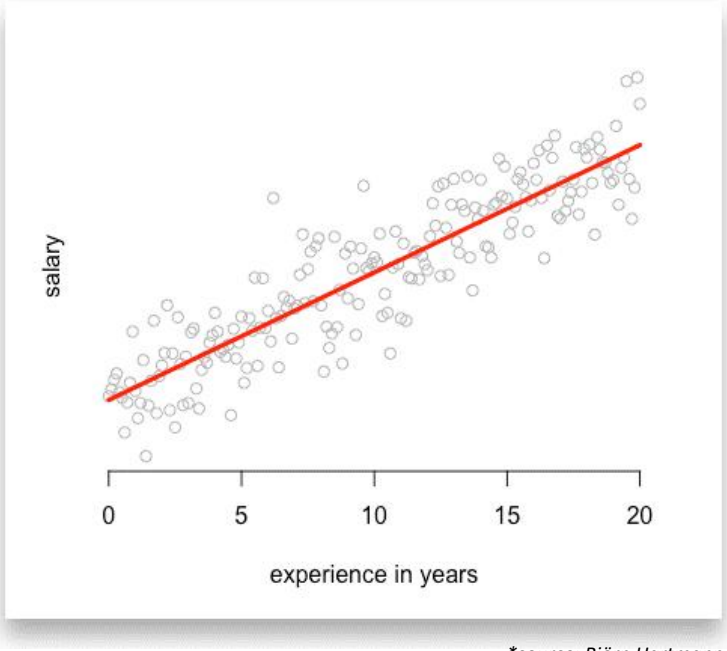
of

Office

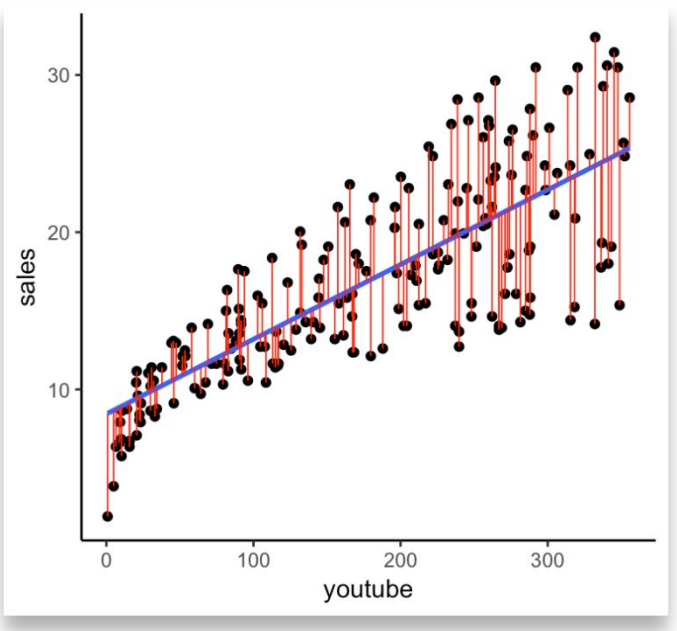


: Regression

- Algorithms that try to predict a value based on a variable + historical data. The most common ones fit a line across 2 or more dimensions.
- Examples: project salary based on years of experience, and sales based on YouTube views.

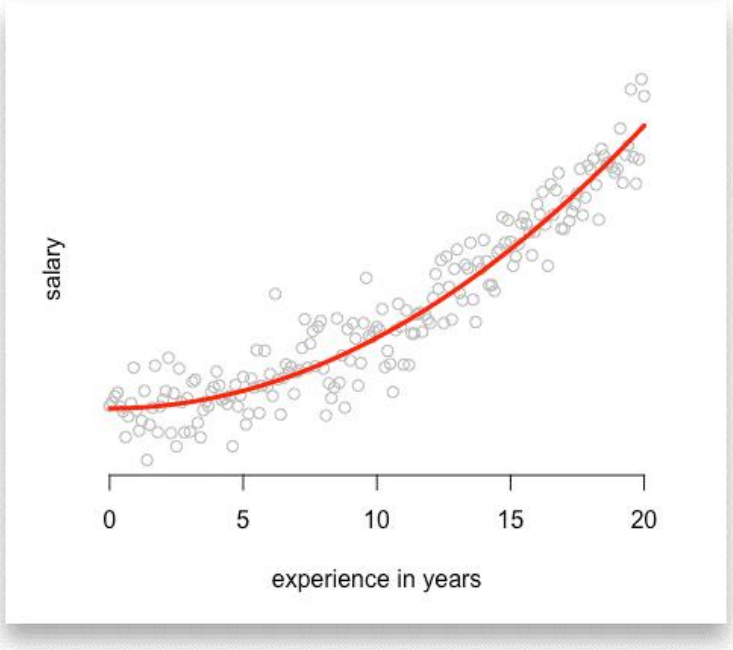


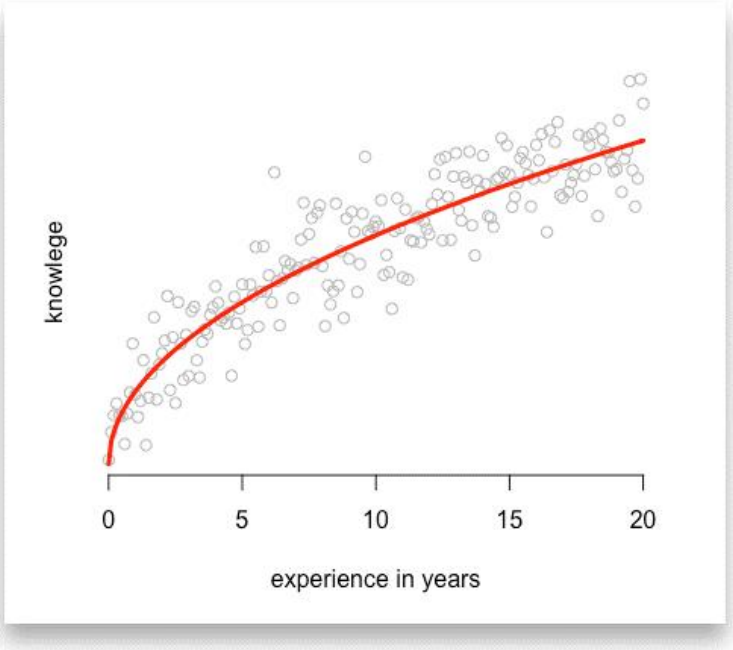
*source: Björn Hartmann

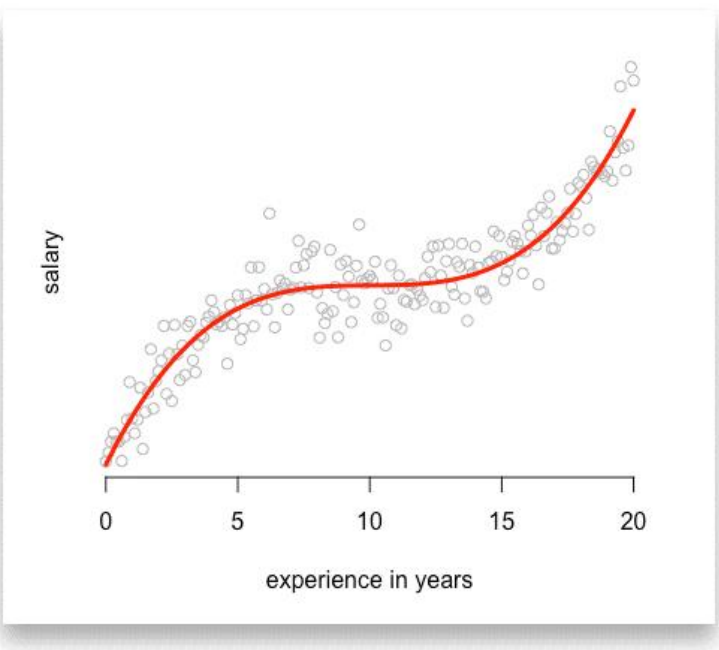


*source: Alboukadel Kassambara

- "Least Squares" regression is the simplest one, but only effective when the data is relatively linear.
- Non-linear regressions are possible if your data "looks" like it could be fit in another function, such as polynomial or logarithmical.

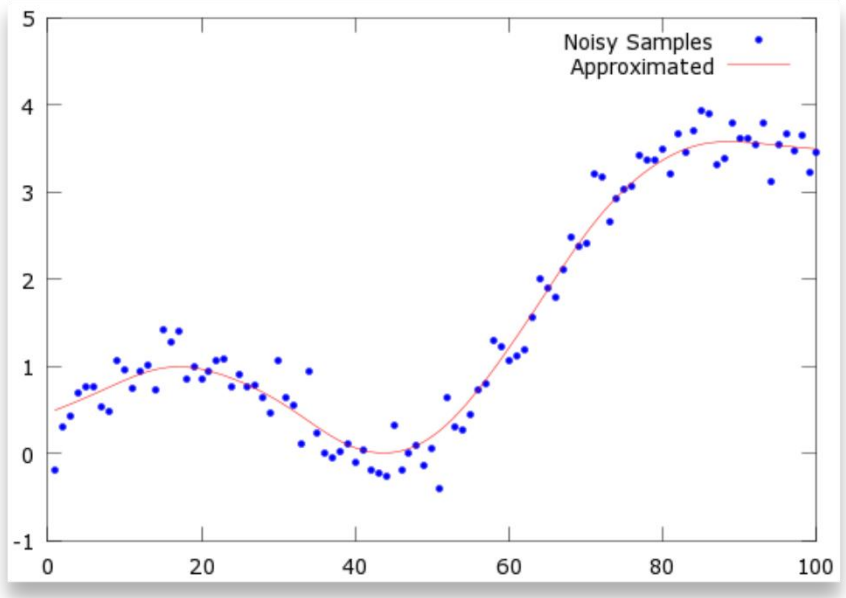




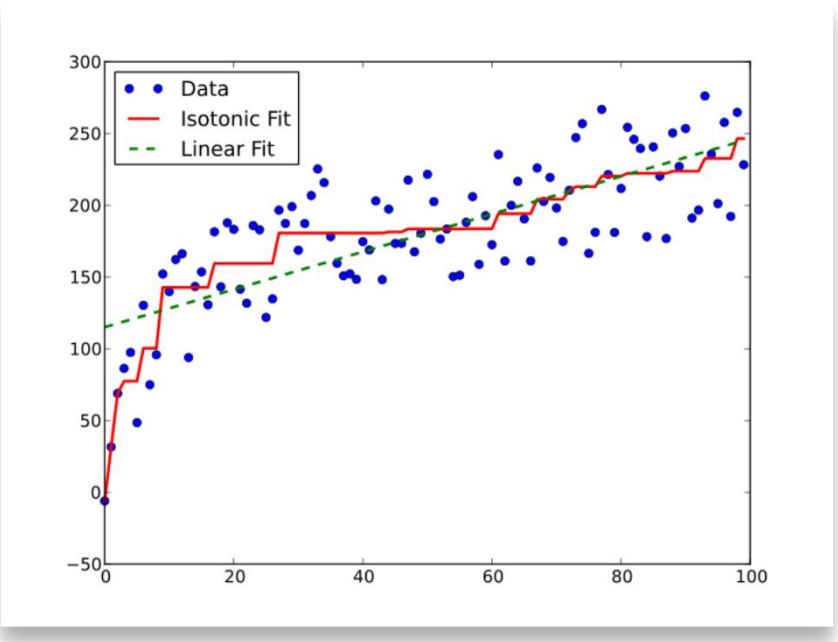


*source: Björn Hartmann

- But generally, gameplay data is much more messy and it's usually more efficient to regress on **non-parametric models**.
- The game, metagame and player habits are changing over time and "disturbing" the curves.
- More efficient models to help game design are non-parametric, such as **Isotonic** and **Gaussian Kernel** regressions.

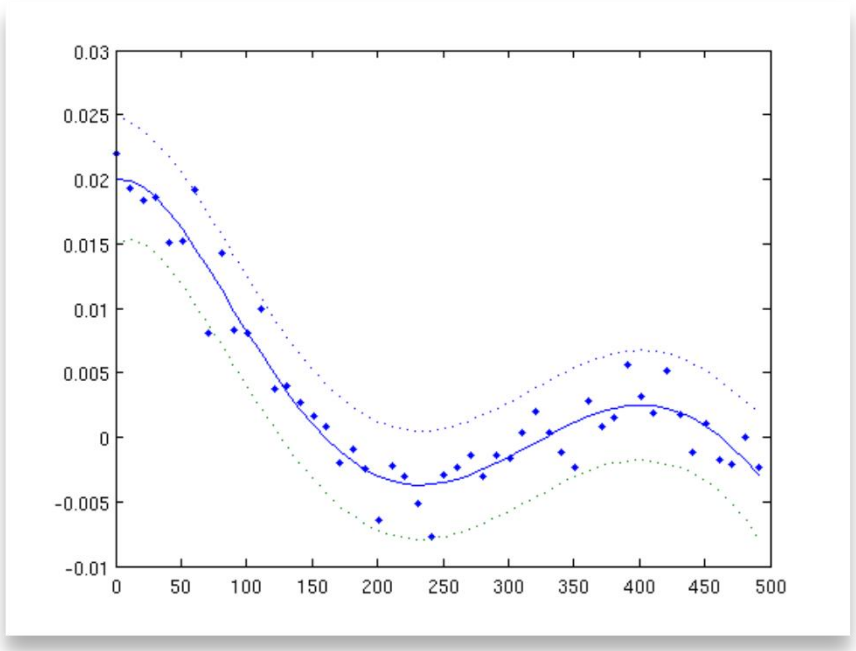


*source: Chris McCormick

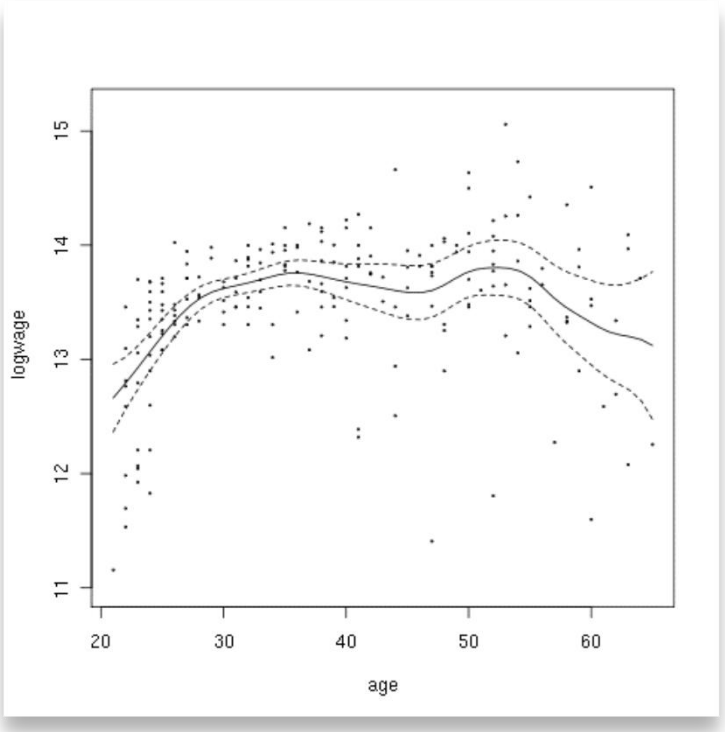


*source: Wikipedia

- It's also useful to project variability bounds, or *confidence intervals*, in the predictions. So the expected variance is well known by designers.
- Example: "We expect players that play 10 hours to have earned 2554 bucks, varying between 4125 and 2145."



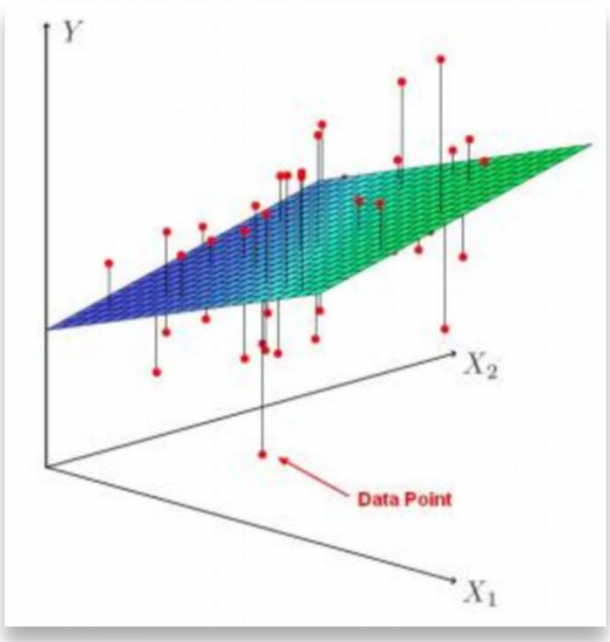
*source: MatLab



*source: Wikipedia

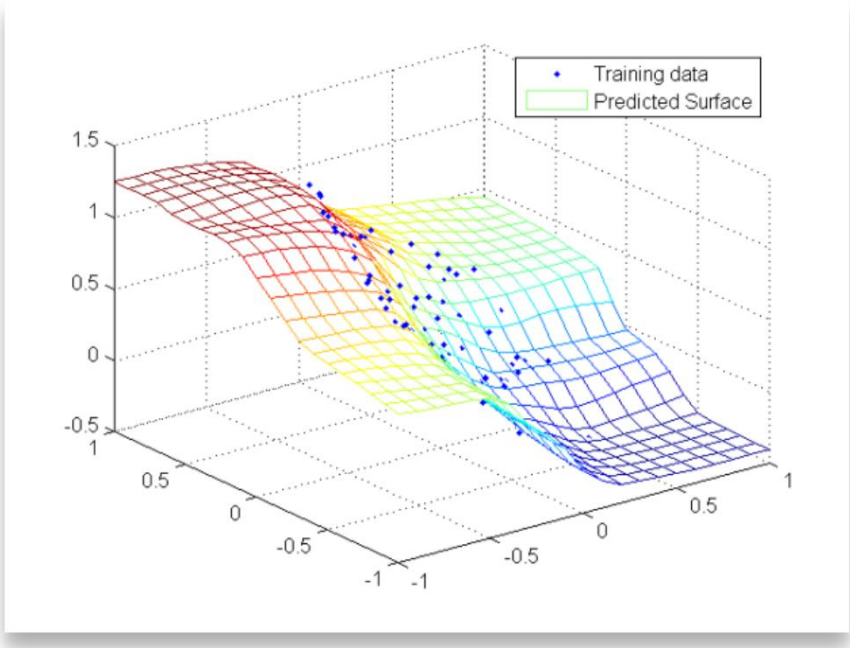
- Things get "interesting" when using multiple dimensions.
- But if too many dimensions begin to be used, the ***curse of dimensionality*** kicks in and it's probably better to use Deep Learning.

Linear:



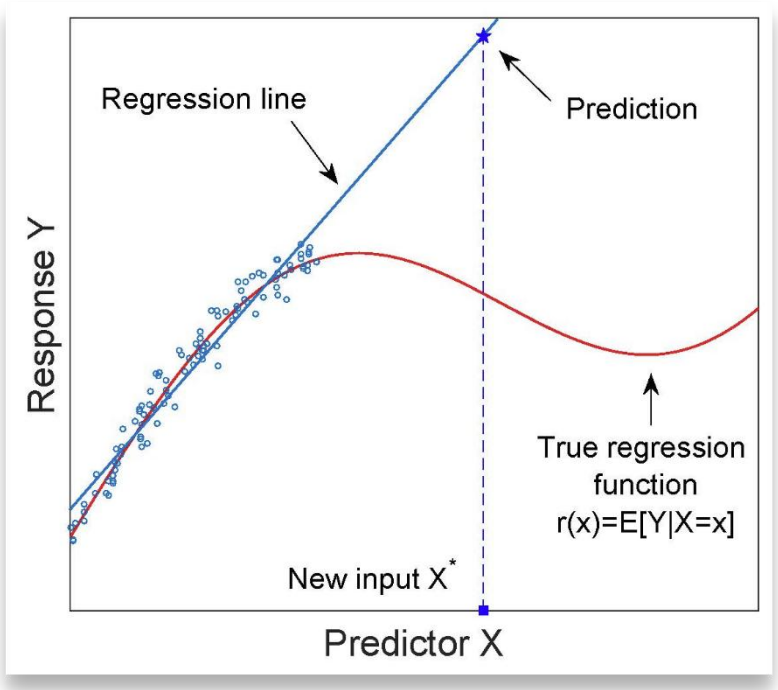
*source: MatLab

Gaussian Kernel:

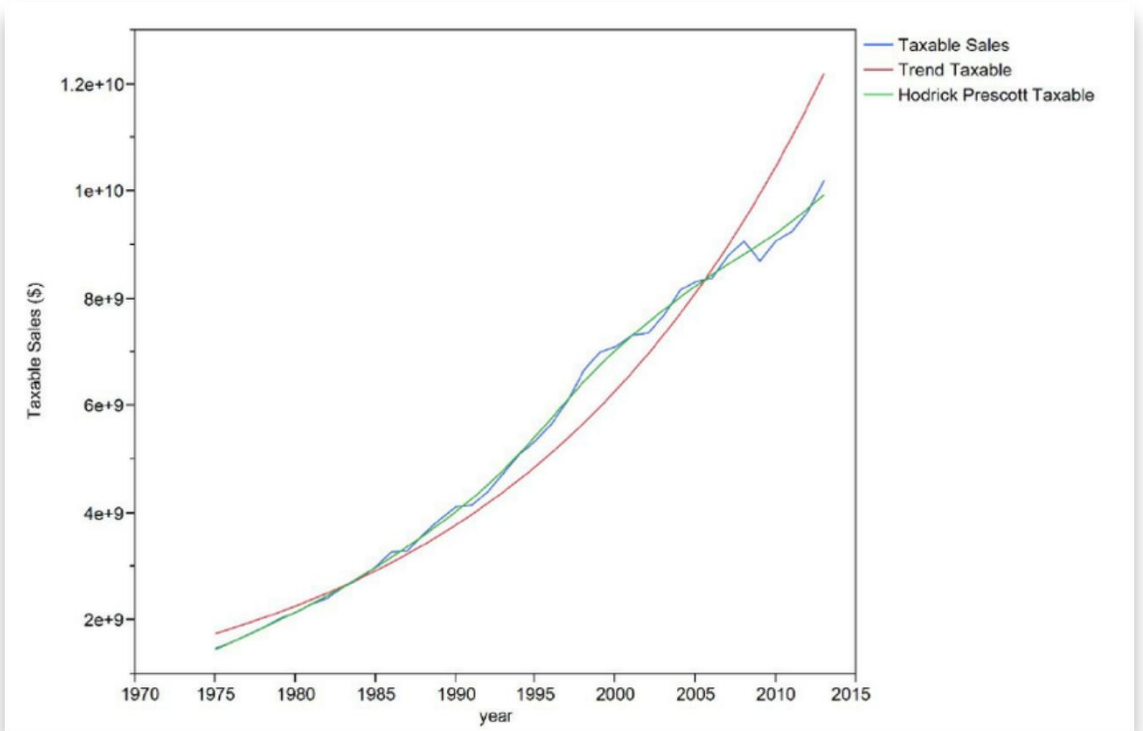


*source: MatLab

- Beware: parametric Regression models may not work well for "extrapolation into the future", because the future will hardly follow a function trained in the past.
- Regression models should be used within known boundaries of your data, or just slightly over it.



*source: Cross Validated



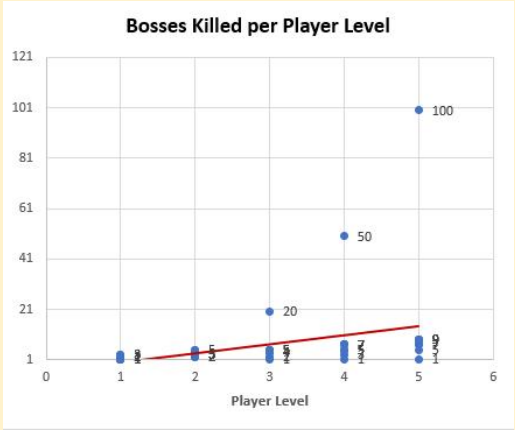
*source: Kenneth A. Kriz

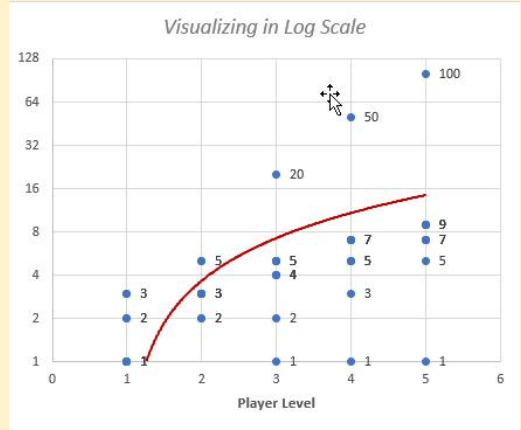
How **Regression** help Game Designers

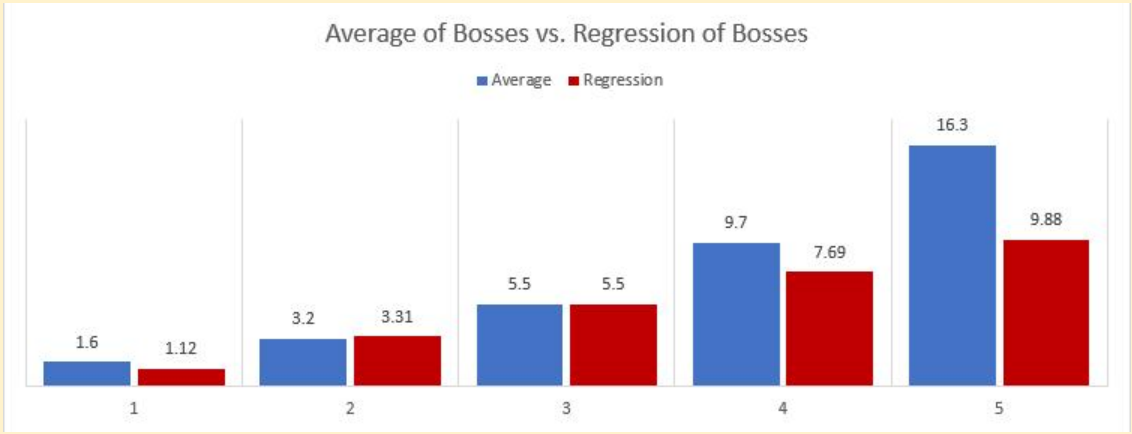
Projection of expected values across many different aspects of the player progression:

- Resource accumulation per game milestone.
- Time to reach game milestones.
- Time to accumulate high level items.
- Resource inflation over time.
- Rates of Tutorial completion.
- Participation on events / game modes.
- Combat efficiency over time and over player levels (for example, on win ratios)

Also, **Regression models could replace analysis based on averages**, as it is more robust against outliers and can use optimization techniques like Gradient Descent. Example with a *very simple* model (least squares):

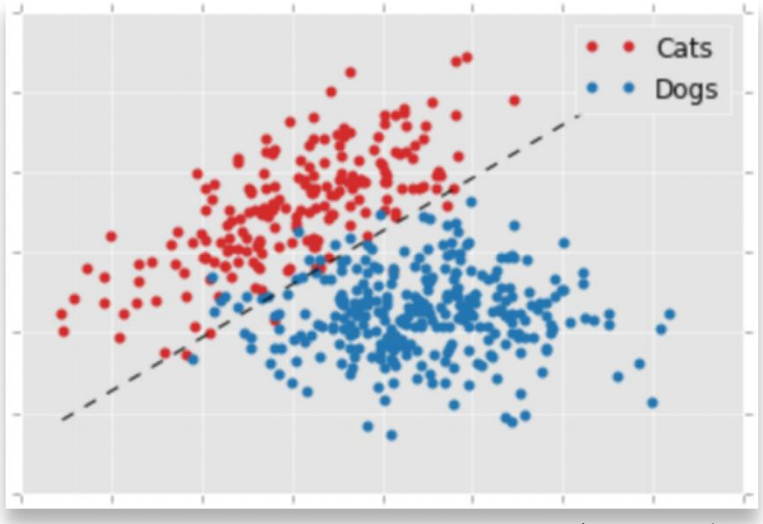




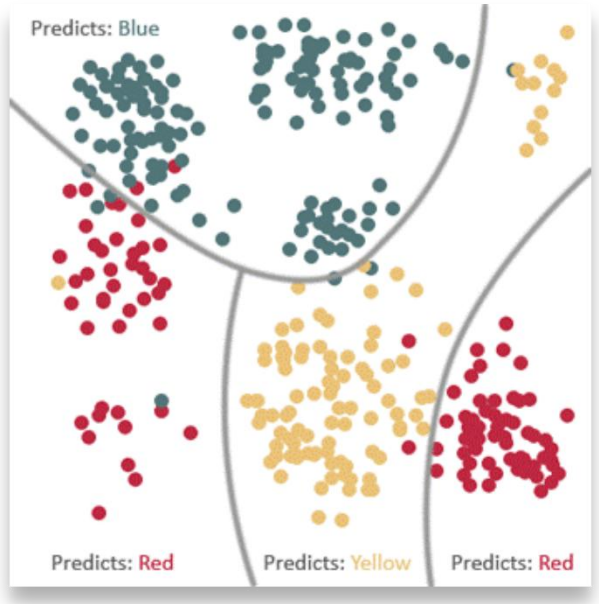


: Classification

- The goal of these models is to classify a data point into categories (or *labels*).
- The models work by discovering criteria that can somehow separate data points in a multi-dimensional space.
- Ideally, the best models are capable of creating complex boundaries to differentiate between *several classes.*



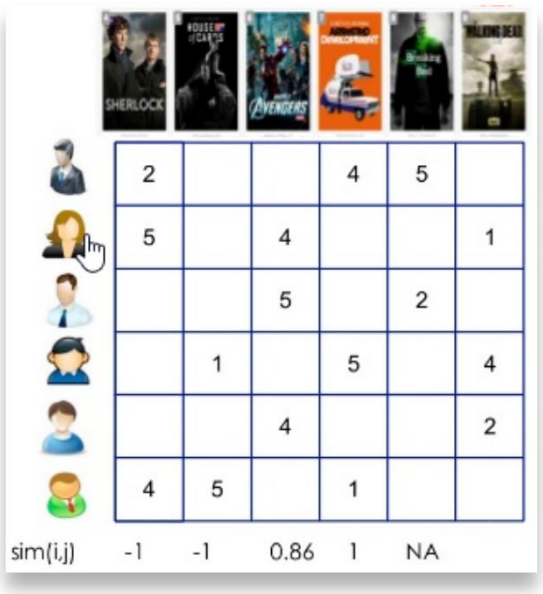
*source: Hyunjik Kim



*source: Fimarkets

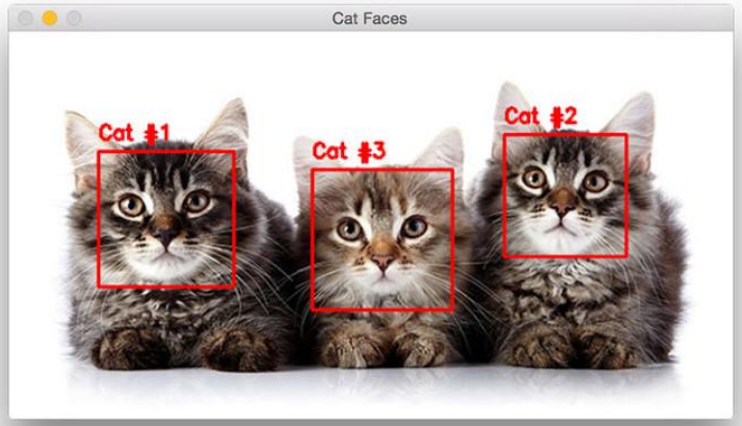
- Classification models are a very useful bunch, wildly used in the most famous applications of Machine Learning across many industries.
- A few examples:

Recommendation systems.



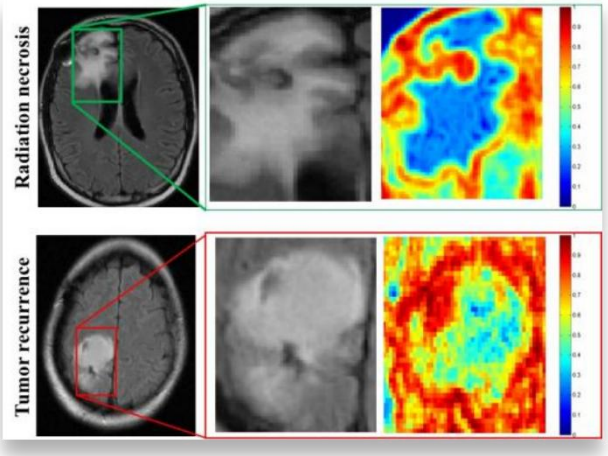
*source: Netflix

Processing and recognition of objects in photos.



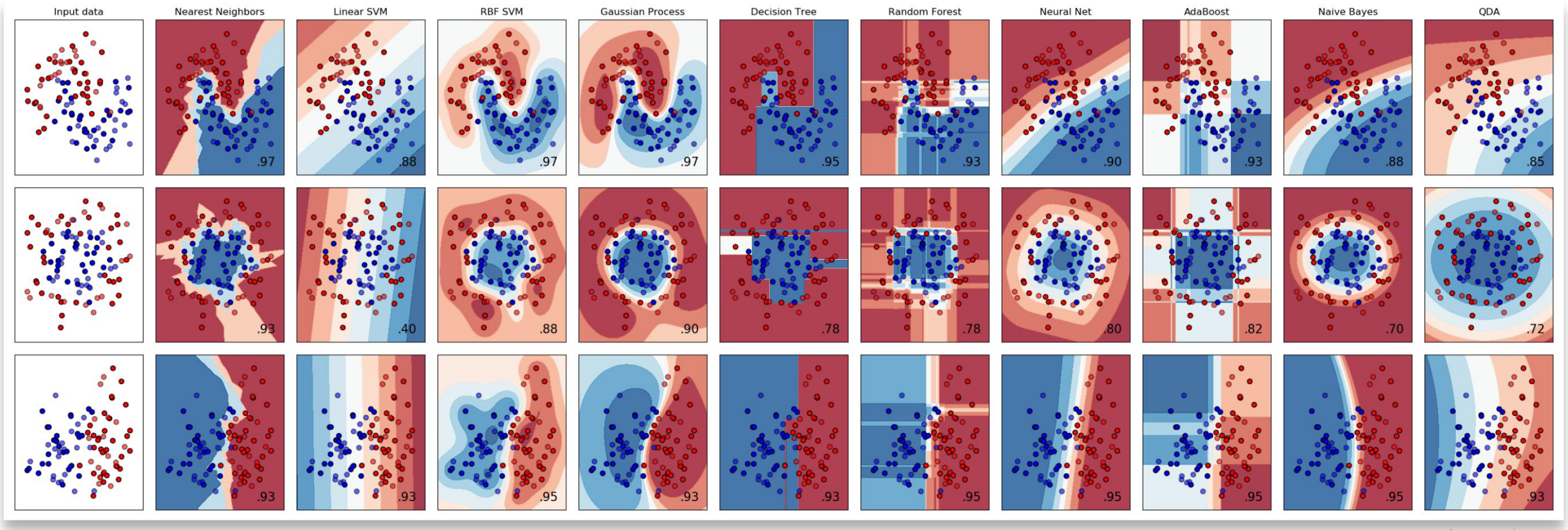
*source: Adrian Rosebrock

Image-based medical diagnosis.



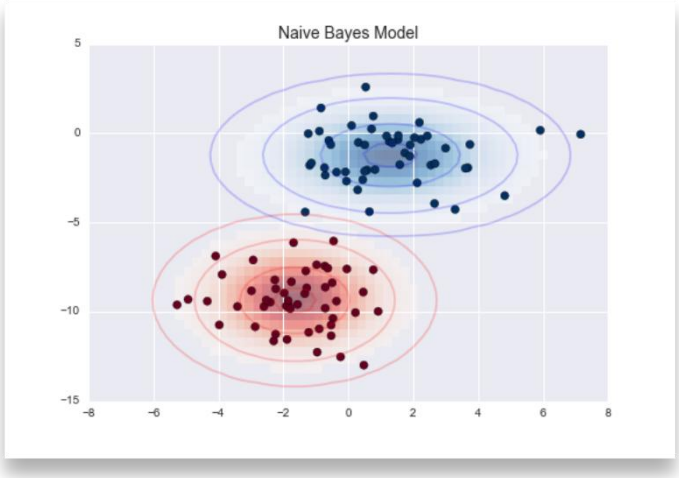
*source: Pallavi Tiwari

- Depending on the classification algorithm, boundaries between classes can be of straight lines (hyperplanes, actually), smooth curves, rules-based lines, uncertain classification in lower probability zones, and anything in-between.

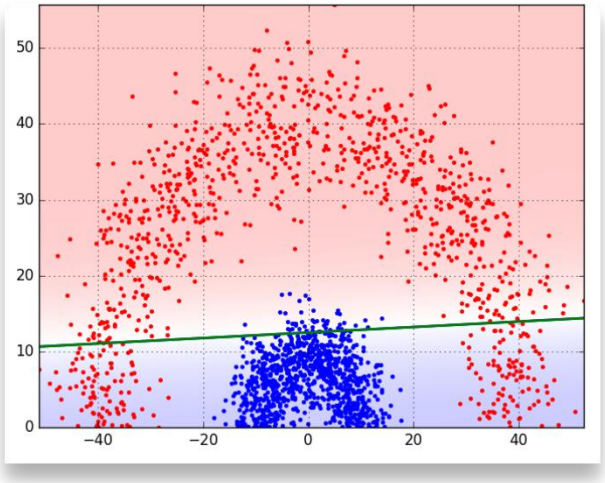


*source: Scikit-Learn

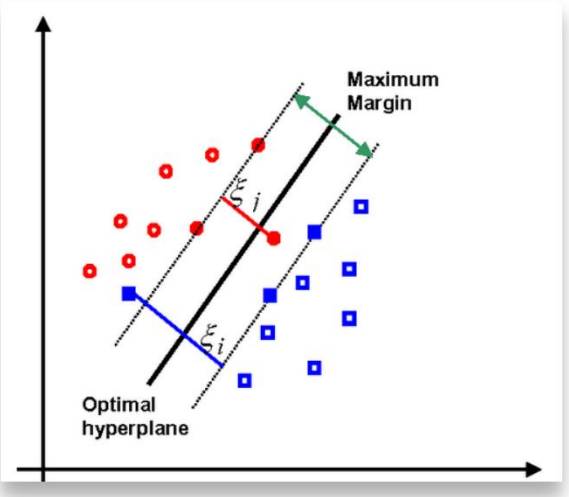
- Some of the most used Classification models used:



*source: Ganapathi Pulipaka



*source: mlxtend



*source: Karim O. Elish

Gaussian Naive Bayes

Learns the probability a data point belongs to a classification based on the Bayes theorem.

Very fast to train and experiment with. One of the models it's worth to try first. Also, may work well on relatively less training data.

Logistic Regression

Learns from data with features that is assumed to be highly correlated an linearly separable. Works best for few categories.

Very fast to train and experiment with, also another model worth to try. Can also learn "online" (when a model keeps evolving with more data)

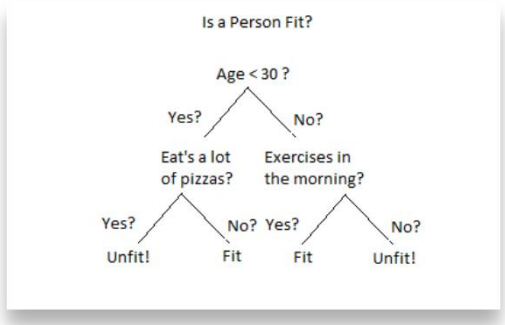
Support Vector Machines

Learns by trying to find boundaries that maximize the spatial distance from data points in the categories.

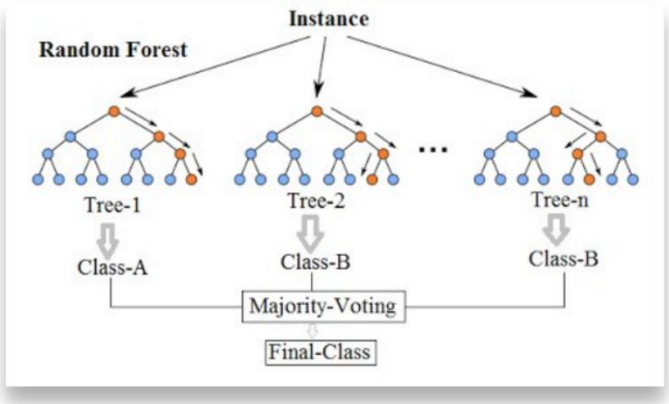
Slow to train and should only be used when you *don't* have too much data (less than 100 k data points), but can yield great results in medium-sized datasets.

- One of the best and most flexible algorithms are *Decision Tree-based models*. In particular, **Random Forests** and "**Boosted Forests**".

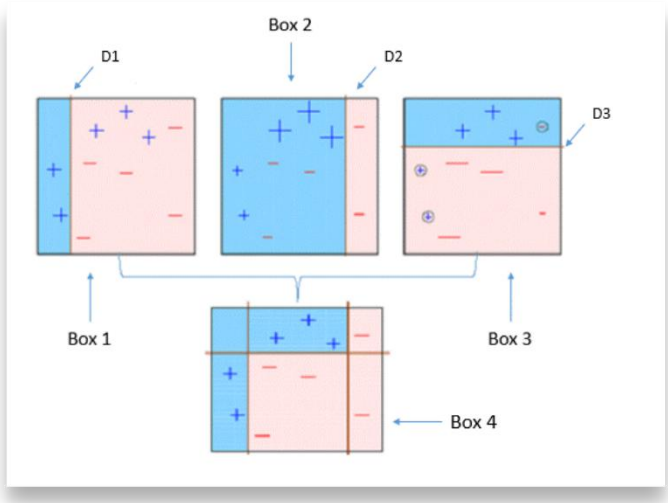
*source: Manish Pathak



*source: Mayur Kulkarni



*source: William Koehrsen



Decision Tree

Learns by finding rules in the data that stablish a if->else branching tree that ultimately classifies a data point based on its features.

Very flexible to accept any distribution of data, but very vulnerable to overfitting.

Random Forests

Trains multiple decision trees and use them in an ***ensemble*** that together "vote" to decide the classification of the data.

Excellent model, because can very well "just work" out of the box without the need of much tuning.

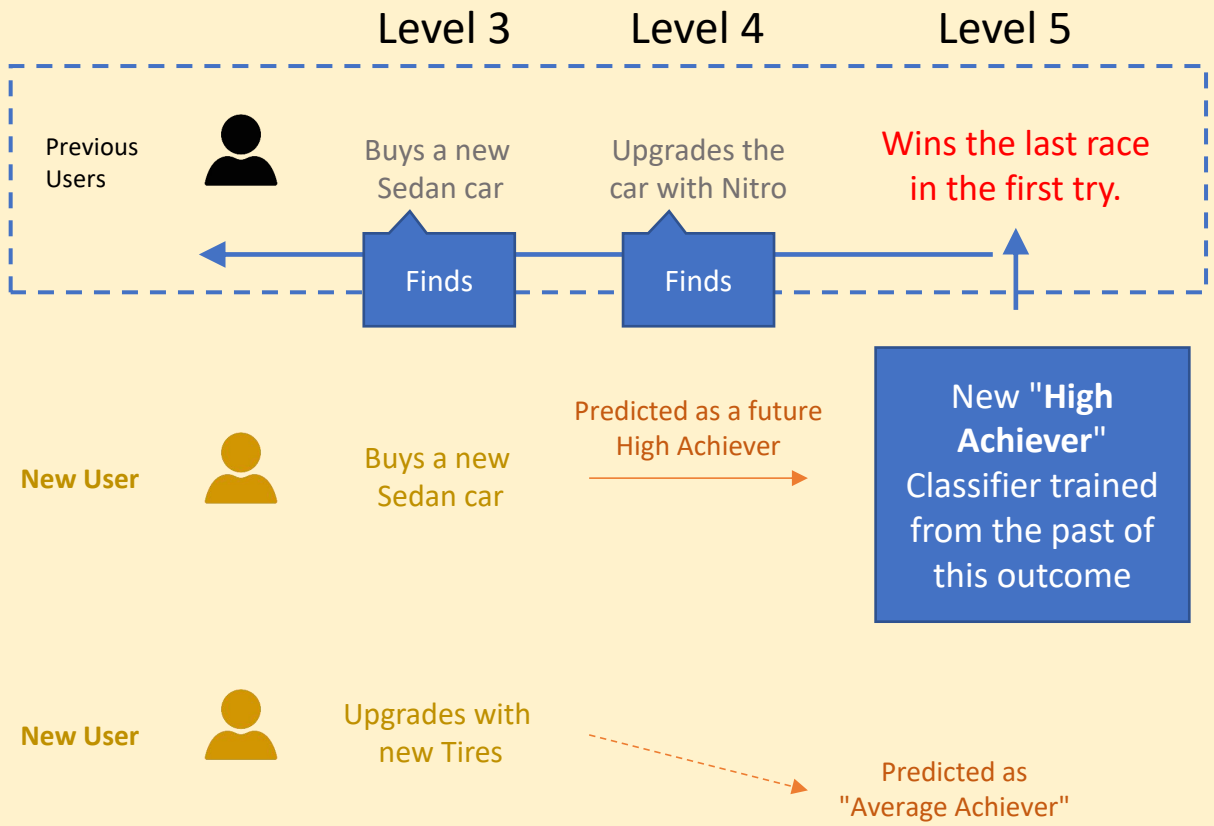
XGBoost / AdaBoost / MART

The ensemble of decision trees uses ***boosting*** techniques to combine their results based on weights derived from their accuracy.

Probably the **best non-Deep-Learning classifiers to use**. Can be hard to tune.

How **Classification** help Game Designers

Useful to predict how a player will behave in the future based on behavior of other players in the past.



Once the classifier is trained and tuned, it outputs a list of the most relevant and correlated mechanics in the game that designers can adjust.

Something like:

Feature	Importance
Sedan_1_Bought_Level_3	0.73
Nitro_2_Level_4	0.64
Sedan_1_Bought_Level_2	0.52
Gas_Consumed	0.41
Events_Completed	0.38
Nitro_1_Level_4	0.04
Tire_2_Level_4	0.03

Certain types of classifications are particularly useful:

- Players who will **churn**.
- Players who will **convert**.
- Players who will participate in an upcoming event.
- Players who will play a game mode unlocked later in the game.
- Players who will engage in social modes like Guilds.

How **Classification** help Live Ops

With the proper backend implemented, the classifier can be implemented as a real-time system to deliver custom content like gifts and in-game events.

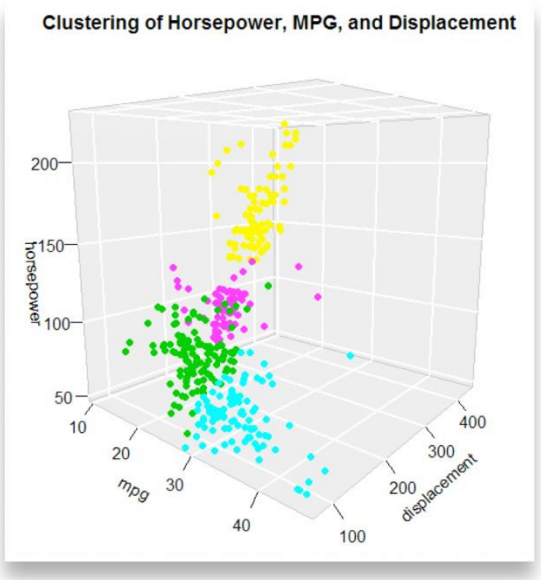
Unique Race now available

Special Offer of a new Car

Share Free Fuel for 1 Day

: Clustering

- The goal of this class of algorithms is to autonomously find clusters of data points based on its spatial proximity or feature similarity.
- Because they can "discover clusters by their own", they are categorized as *Unsupervised* Learning.
- In data science, we use clustering to gain valuable insights by seeing what groups the data points fall into. Designers can also benefit from it.



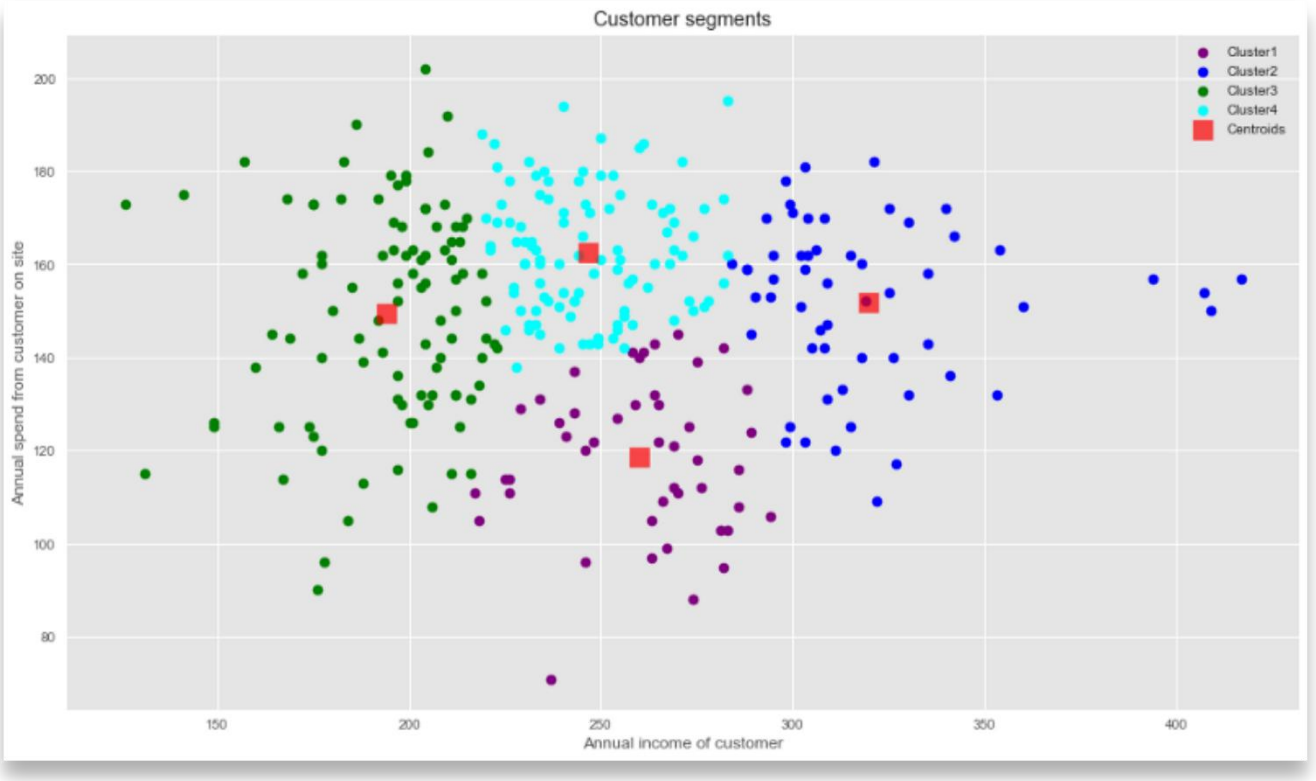
*source: Peter Jeffcock

*source: Inna Kaler

- The ability to discover clusters algorithmically eliminate the reliance on biased human judgement that may not be supported by the distribution of the data.
- Say we want to classify customers or an e-commerce site by two parameters - amount spent in the site vs. personal income (provided they gave this info).

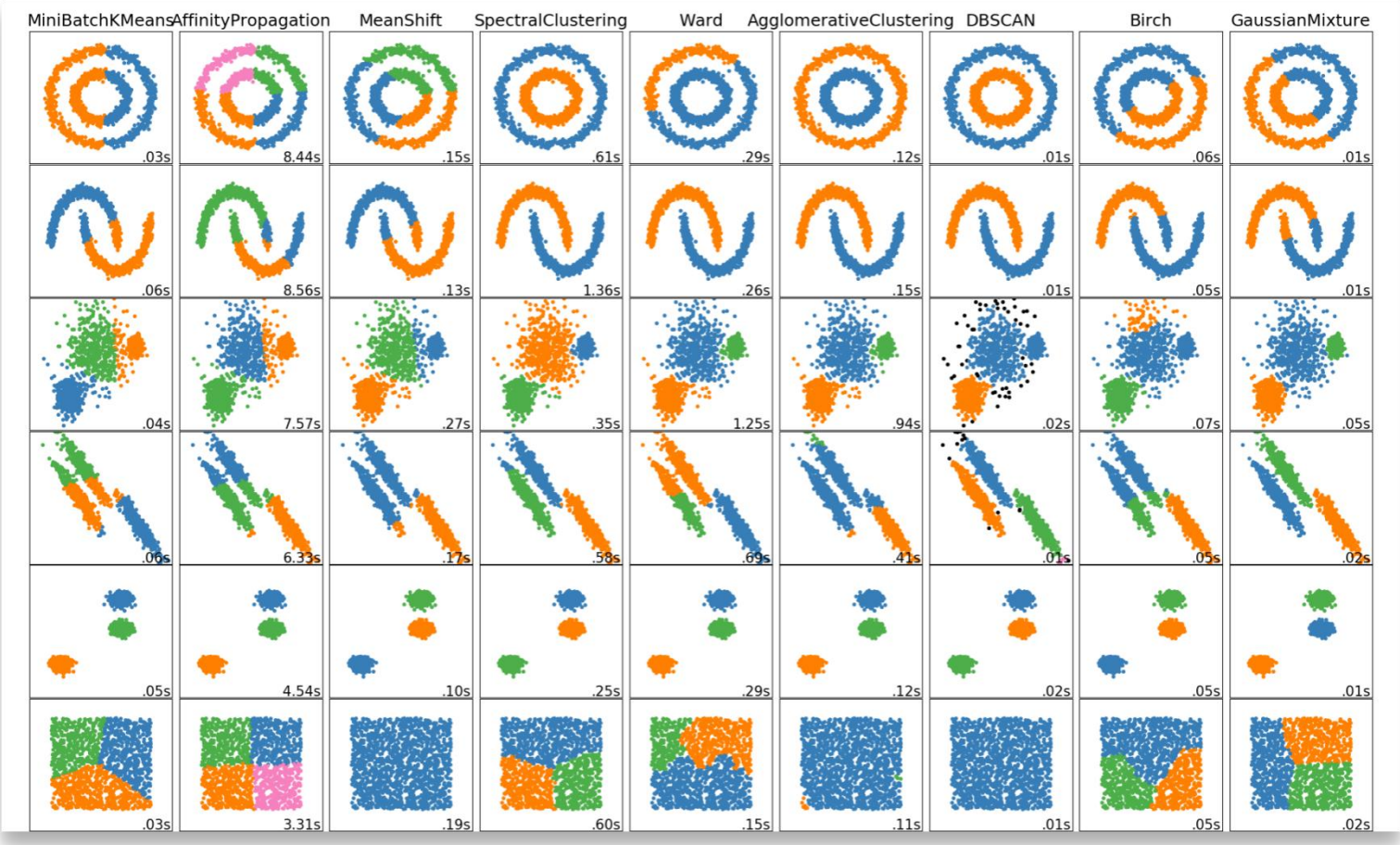
"Intuitive" Categorization		
(depends who you ask, will be different...)		
Group	**Amount Spent**	**Income**
1	Up to $10	Low
2	Up to $50	Low-Medium
3	Up to $100	Medium-High
Fans	>$100	Medium-High
Super Fans	>$200	High
Whales	>$500	High

...but...



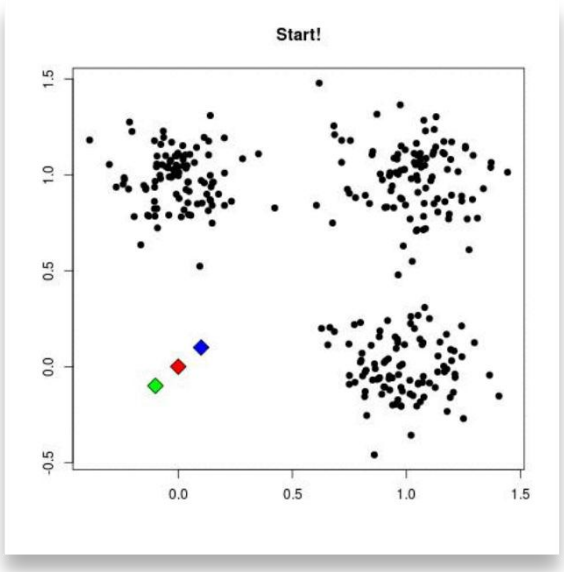
source: Sowmya Vivek

- Different algorithms employ different strategies to determine what is a "close point" in multidimensional space. The problem is harder than it seems in real-world messy datasets.



source: Scikit-Learn

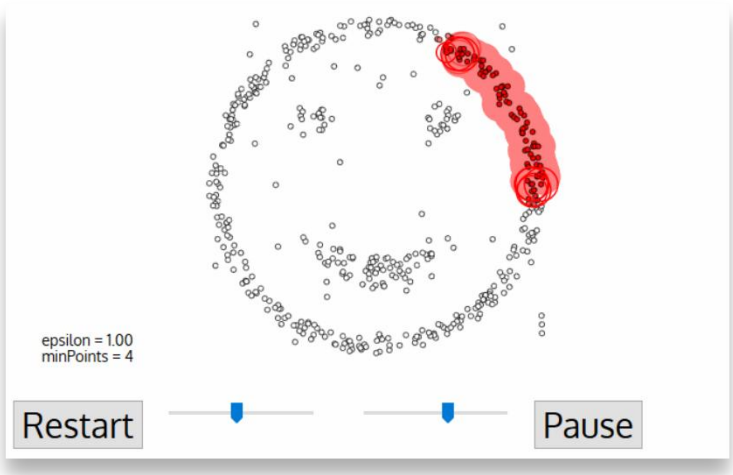
- Here's a few visualizations of how commonly used algorithms calculate clusters:



*source: Mubaris NK

k-Means

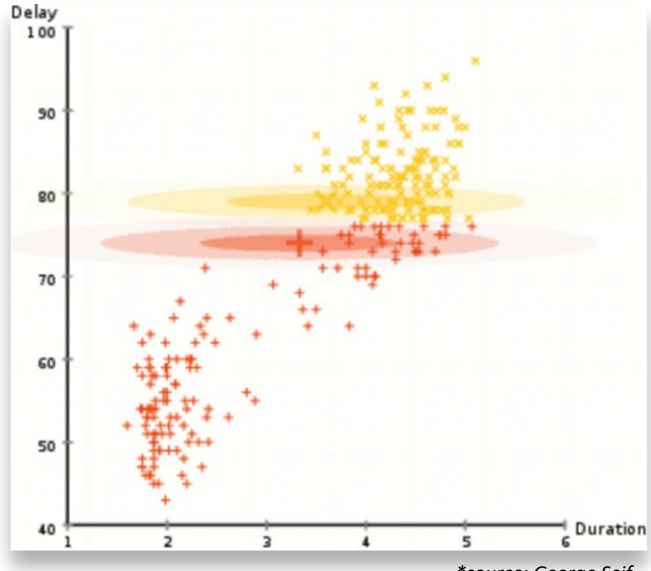
The best know and most used. Fast and scalable. Creates clusters with distance of data points, moving the centroid until convergence. Needs to be told the number of clusters - which could be too arbitrary, and data scientists often use another calculation (the "elbow rule") for this.



*source: George Seif

DBSCAN

Density-based search of clusters, works by finding points that are close to a random non-visited starting point, and continuing to include other close points. Works great but is resource-intensive and vulnerable to clusters of different densities.



*source: George Seif

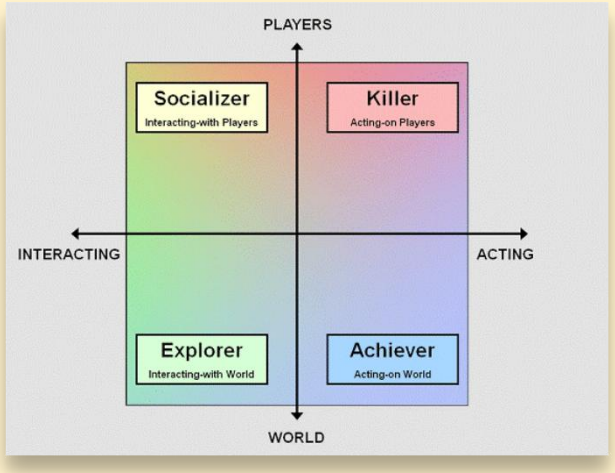
Gaussian Mixtures

Assumes clusters follow a normal (gaussian) distribution in the multi-dimensional space, and from this assumption find clusters and adapt the boundaries of such distribution. generates "soft clusters", where a data point has mixed membership (*probabilities* of belonging).

How **Clustering** help Game Designers

Study made on the MMO game ***Tera***.

Development



source: Bart Stewart

- Clustering is useful to find groups of players from their behavior in the game. Frameworks of psychology like the Bartle Types are useful for conception of features, but once you launch data science can provide you how your players really behave.

Updates

TABLE 2: INTERPRETED BEHAVIORAL CLUSTERS FOR *TERA*, LEVEL 32 BIN ONLY, K-MEANS. %P = %PLAYERS IN BIN.

Title	%P	Characteristics
Elite	5.78	Highest scores for all features except Mining and Plants which are the lowest in the game.
Stragglers	39.4	Lowest scores for all features, including deaths from monsters.
Average Joes	12.7	Better scores across all categories than Low Performers, 4th ranked overall
Dependables	18.6	Average scores across all categories, high number of friends, 3rd ranked overall, 2nd rank in monster kills
Worker I	15.9	Similar to the Average Joes, but high Mining and Plants, and loot value 3rd ranked.
Worker II	7.6	Similar to The Dependables, but highest Mining and Plants value in the game. 2nd ranked overall. Loot 2nd ranked.

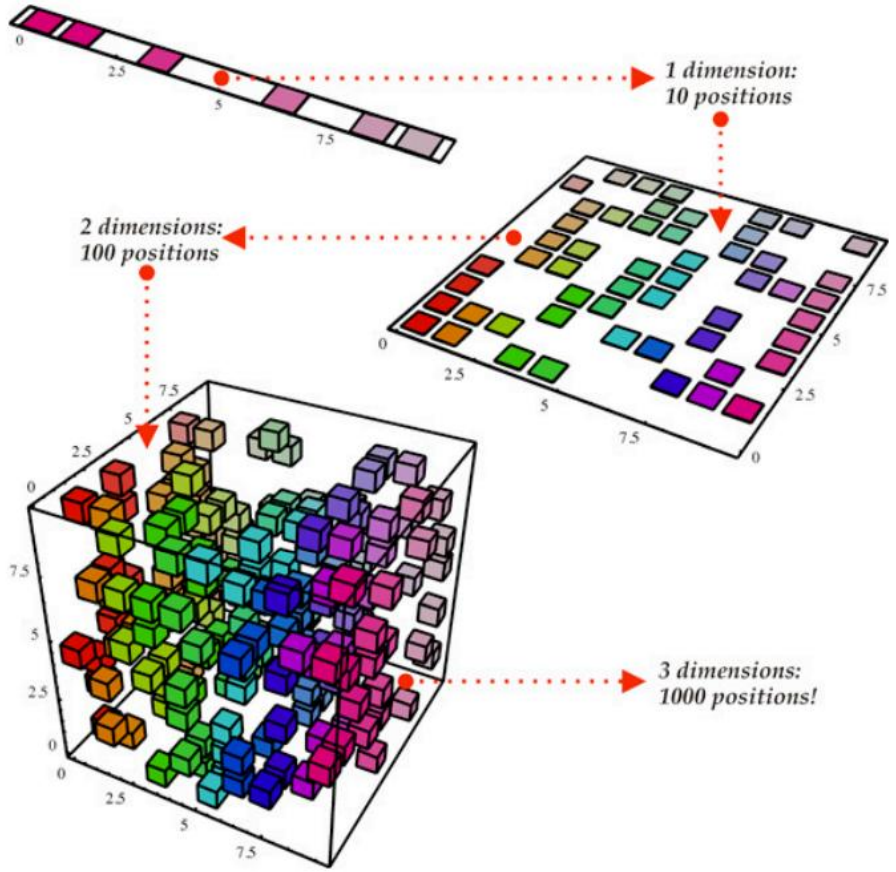
TABLE 3: INTERPRETED BEHAVIORAL CLUSTERS FOR *TERA*, LEVEL 32 BIN ONLY, SIVM. %P = %PLAYERS IN BIN

Title	%P	Characteristics
Elite	3.9	High scores overall, except for Mining/Plants and deaths from monsters. No auctions created.
Stragglers	7.6	Low scores overall, dies a lot from monsters
Planters	21.6	Middling scores, but high Plants skill
Miners	15.0	Middling scores, but high Mining skill
Auction Devils	1.1	Highest auction and achievement score. 2nd ranked loot and kills scores. 2nd ranked friends score, high mining score
Friendly Pros	50.8	Highest friends score, scores similar to Auction Devils apart from low auction score and 2nd lowest loot score

source: Anders Drachen, Rafet Sifa, Christian Bauckhage and Christian Thurau

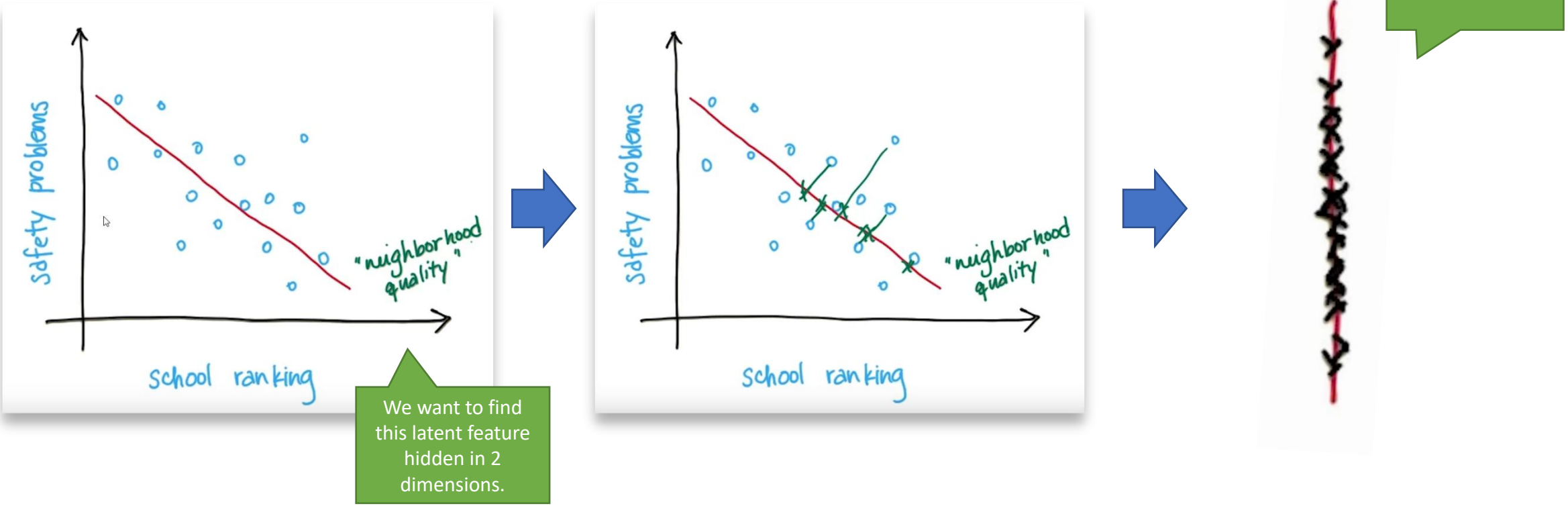
: Dimensionality Reduction

- A group of algorithms that can decompose multiple dimensions of data into fewer ones.
- The decomposed dimensions represent "parts" of data that were highly aligned with each other.
- The mathematical description of these new dimensions may seem clunky, but are interpretable by a data analyst.



*source: BigSnarf blog

- The most used method is the **Principal Component Analysis (PCA).**
- It works by finding new coordinate systems in the data that can potentially describe latent features, and then projecting 2 or more dimensions into a single one, losing information in the process.
- For example:



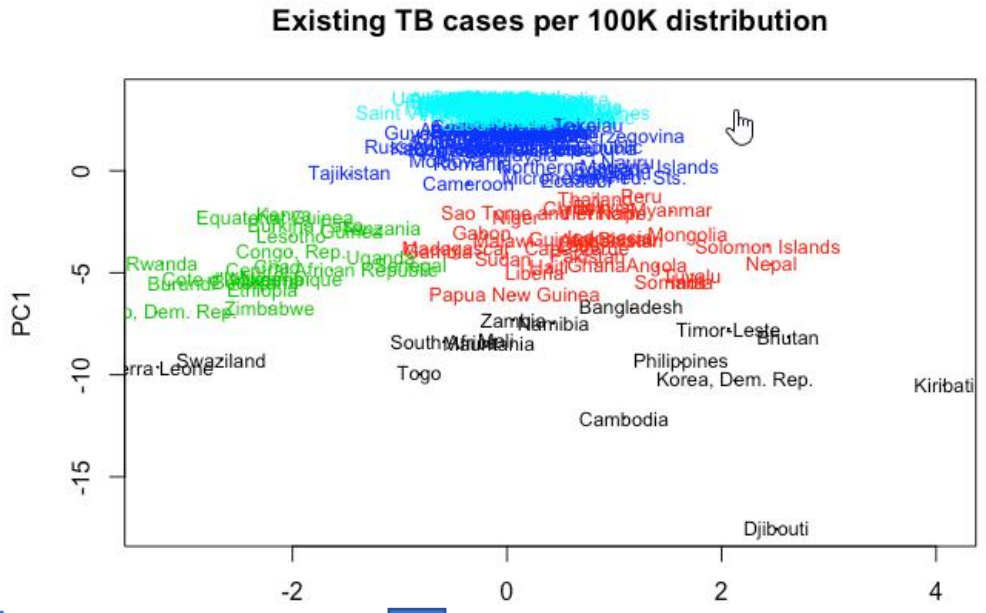
*source: Udacity

- Also great to visualize multi-dimensional spaces in ways anyone can understand.
- Example: here's an analysis made by Jose A. Dianes on cases of Tuberculosis across 2 decades and all countries:

```
##                      PC1         PC2
## Afghanistan    -3.490274  0.973495650
## Albania         2.929002  0.012141345
## Algeria         2.719073 -0.184591877
## American Samoa  3.437263  0.005609367
## Andorra         3.173621  0.033839606
## Angola         -4.695625  1.398306461
```

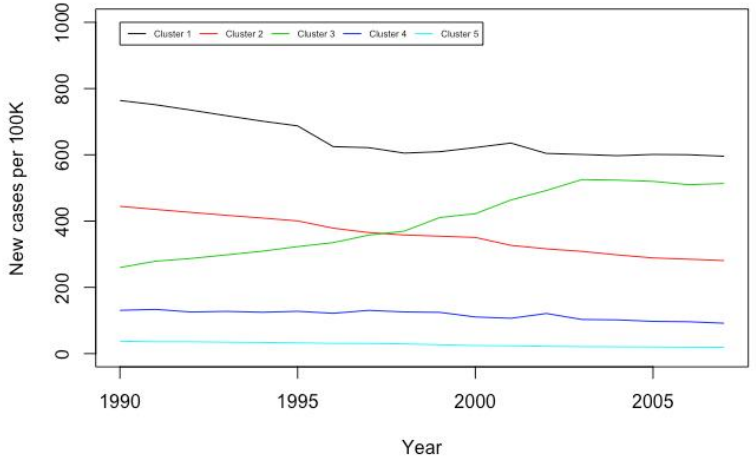
Compressing all years to 2 Principal Components

Plotting and Clustering



Analysis of what the clusters represent.

year	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002
country													
Afghanistan	436	429	422	415	407	397	397	387	374	373	346	326	304
Albania	42	40	41	42	42	43	42	44	43	42	40	34	32
Algeria	45	44	44	43	43	42	43	44	45	46	48	49	50
American Samoa	42	14	4	18	17	22	0	25	12	8	8	6	5
Andorra	39	37	35	33	32	30	28	23	24	22	20	20	21

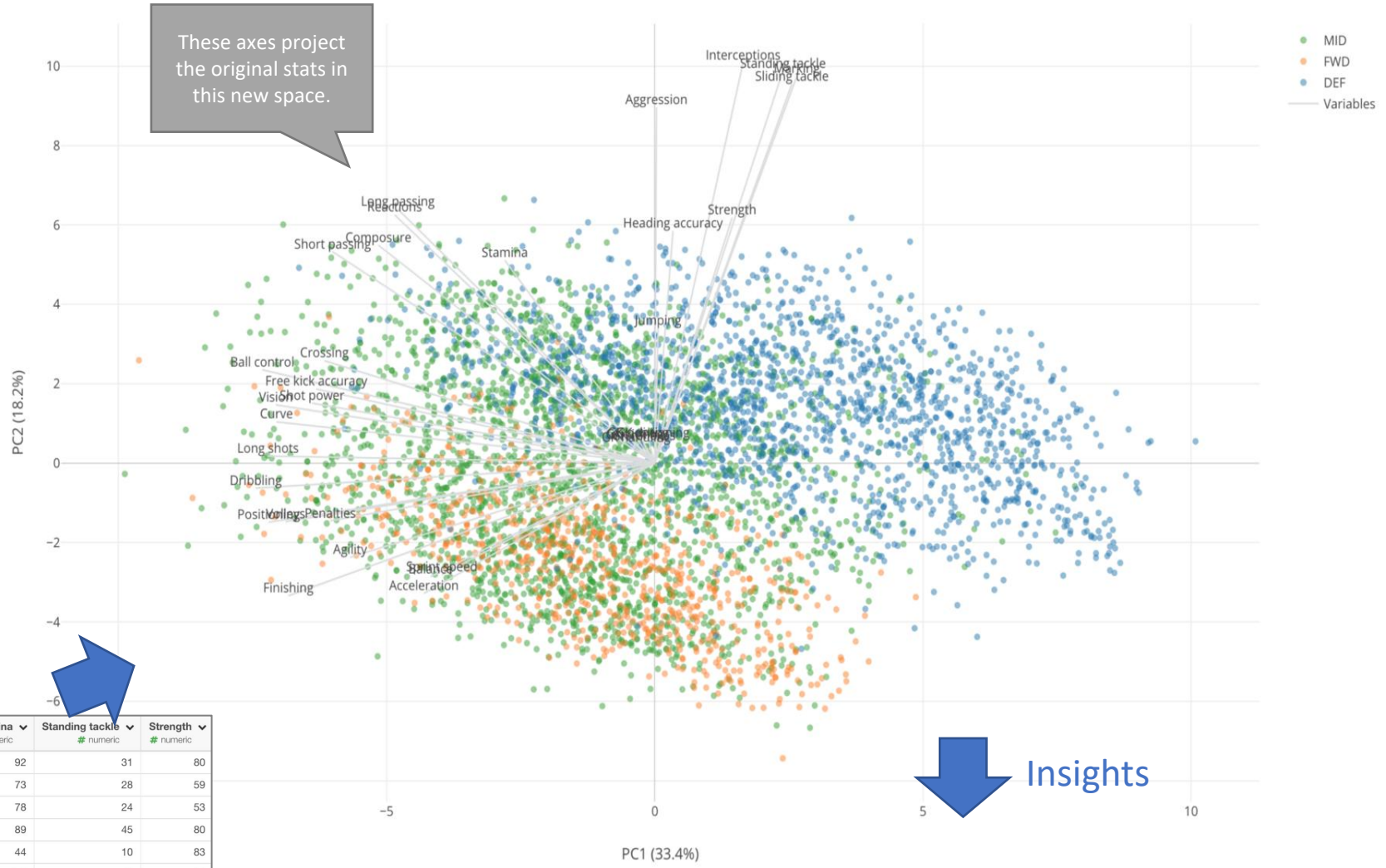


How **PCAs** help Game Designers

- Like in the example before, where the clusters captured the trend over time for several countries at once, PCAs can be used to find latent information and trends in your data
- Here's an example decomposing 34 stats of soccer players in 2018 into only 2 dimensions, made by Kan Nishida

Penalties # numeric	Positioning # numeric	Reactions # numeric	Short passing # numeric	Shot power # numeric	Sliding tackle # numeric	Sprint speed # numeric	Stamina # numeric	Standing tackle # numeric	Strength # numeric
85	95	96	83	94	23	91	92	31	80
74	93	95	88	85	26	87	73	28	59
81	90	88	81	80	33	90	78	24	53
85	92	93	83	87	38	77	89	45	80
47	12	85	55	25	11	61	44	10	83
81	91	91	83	88	19	83	79	42	84
40	12	88	50	31	13	58	40	21	64
86	85	85	86	79	22	87	79	27	65
73	79	86	90	87	69	52	77	82	74
70	92	88	75	88	18	80	72	22	85
68	52	85	78	79	91	77	84	89	81
77	84	88	90	85	40	75	87	51	73
27	13	81	32	36	16	52	38	18	70
77	86	87	81	84	35	84	85	39	72
80	79	88	92	73	73	71	82	80	58
76	86	87	86	91	52	95	76	55	80

These axes project the original stats in this new space.



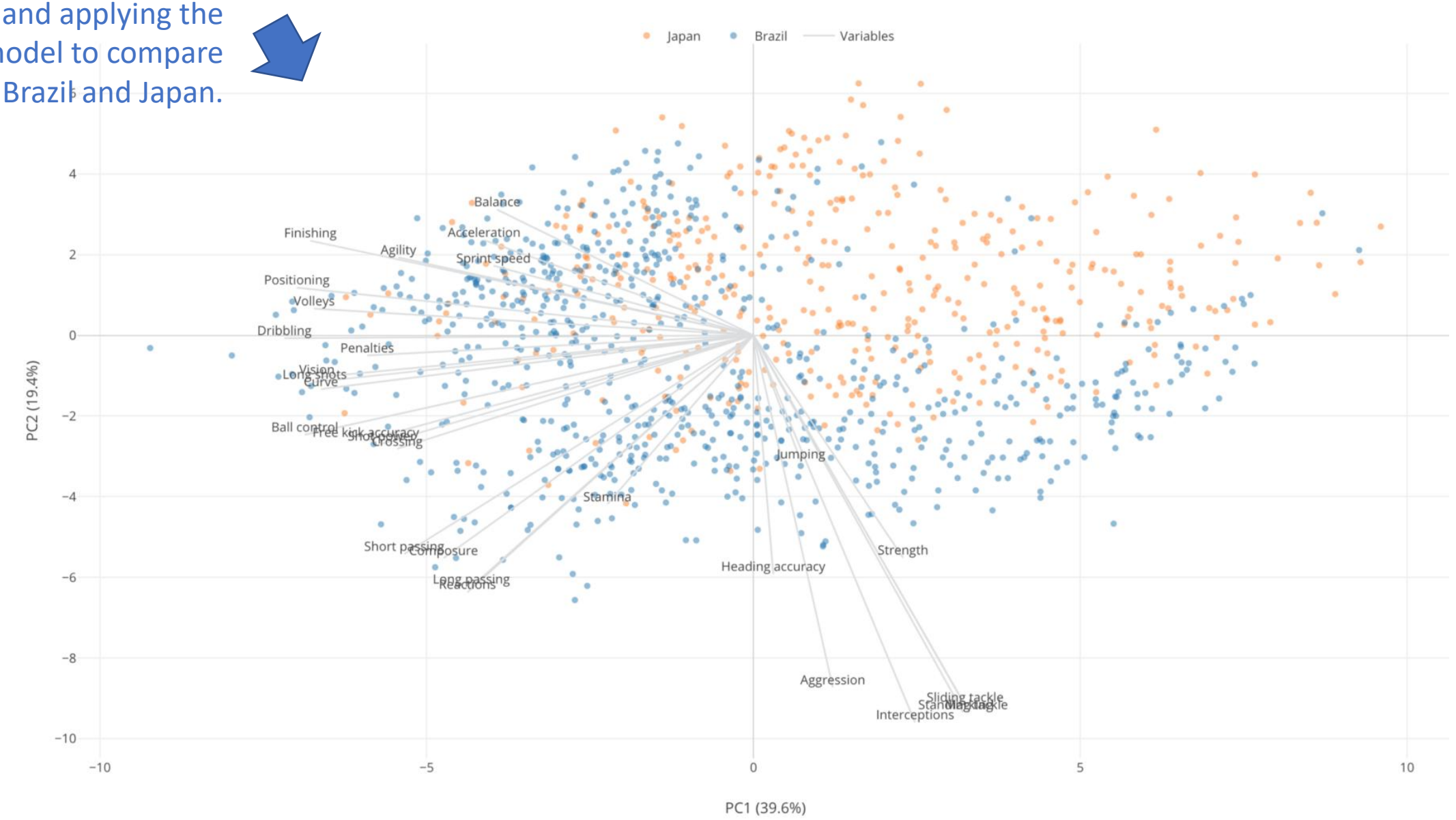
Insights

1. The dimension PC2 is strongly associated to stats that help Defense, like Marking, Interception, Strength, Aggregation.

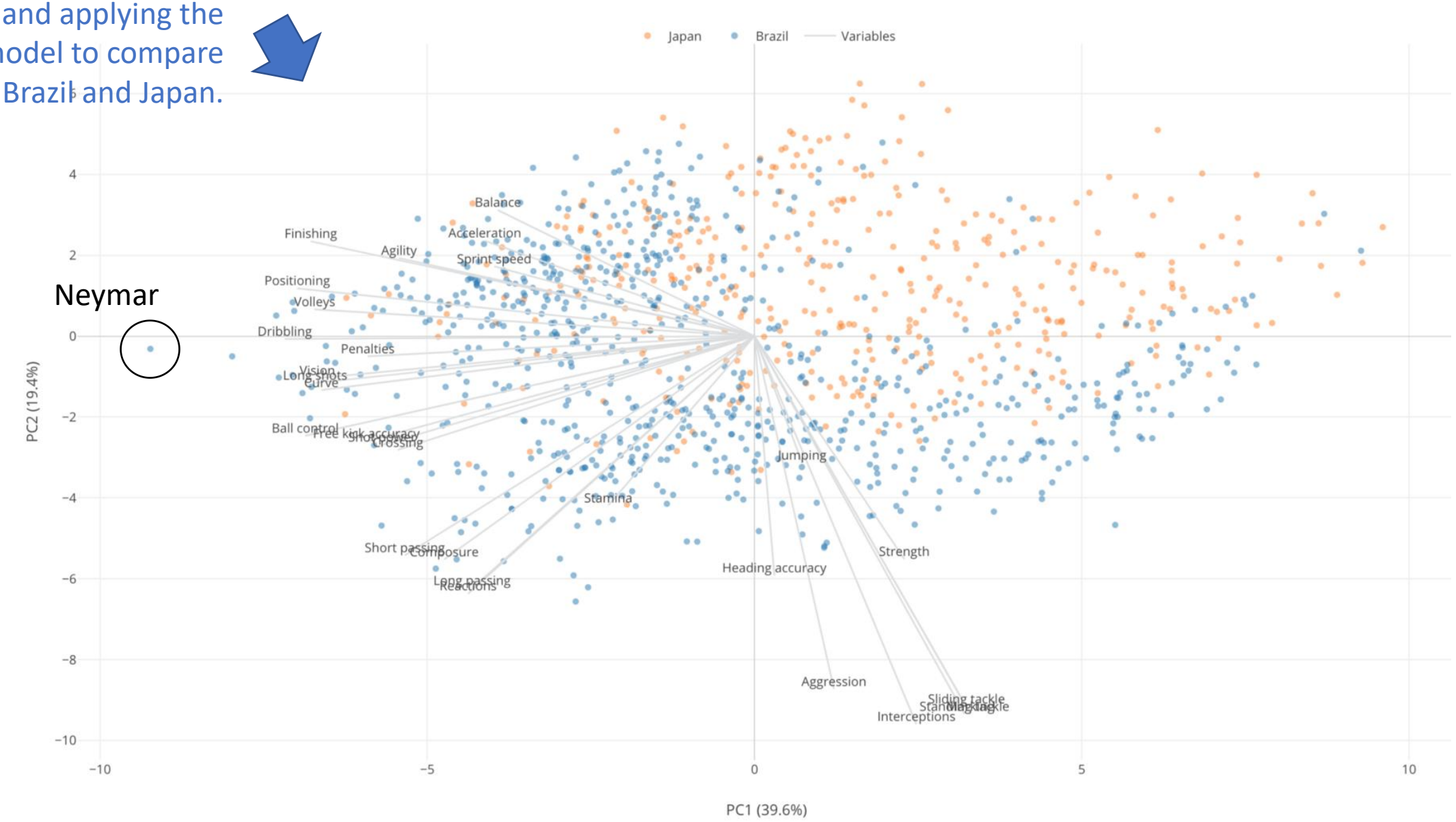
2. Mid Fields cluster more towards:
   - Short Pass
   - Stamina
   - Crossing
   - Free Kick Accuracy
   - Vision
   - Shot Power

3. Forward players share much of the same skill space in PC1 with Mid Fielders, but cluster more towards:
   - Agility
   - Acceleration
   - Spring Speed
   - Finishing.

... and applying the model to compare only Brazil and Japan.



... and applying the model to compare only Brazil and Japan.



... and applying the model to compare only Brazil and Japan.

Neymar

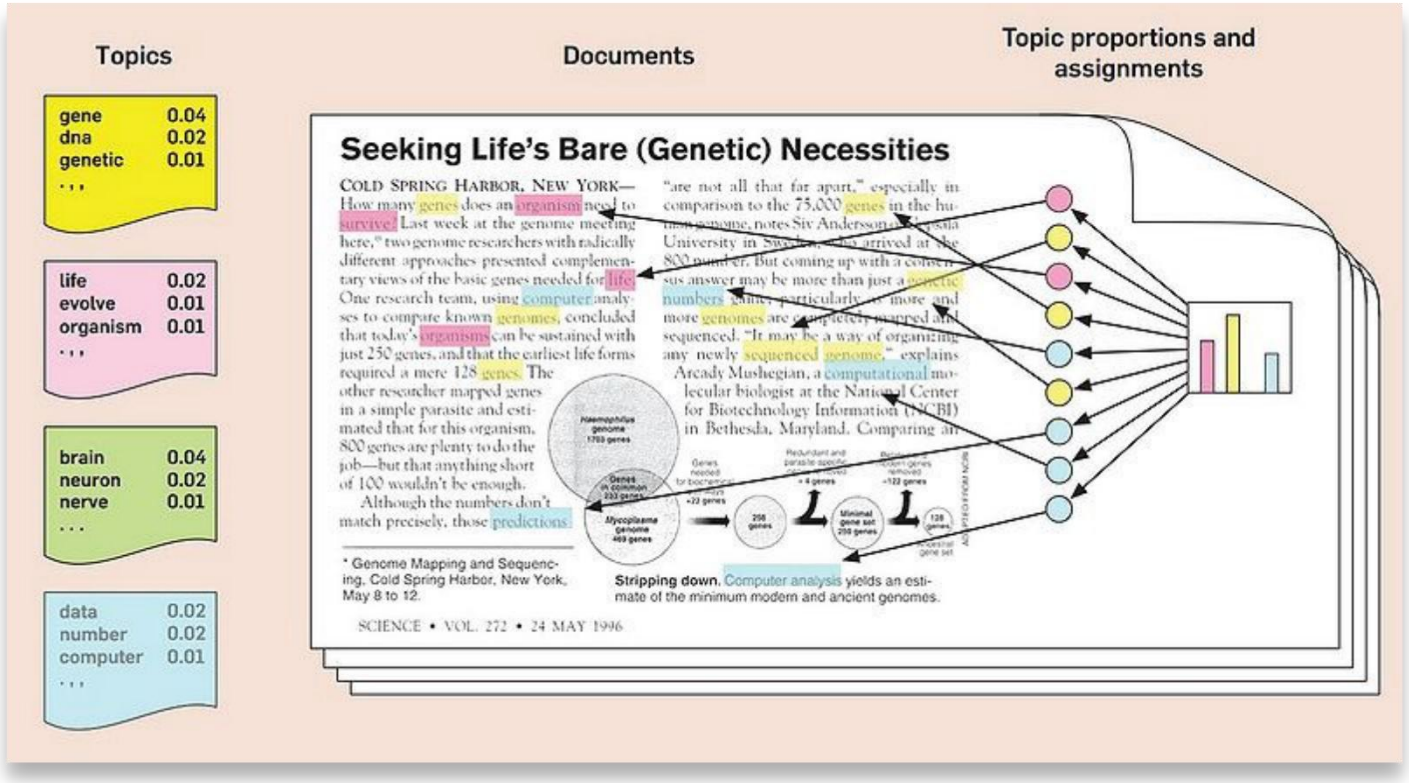
Japan Brazil Variables

Balance
Finishing
Acceleration
Agility
Sprint speed
Positioning
Volleys
Dribbling
Penalties
Vision
Long shots
Curve
Jumping
Stamina
Composure
Heading accuracy
Strength
Long passing
Reactions
Aggression
Sliding tackle
Interceptions

PC1 (39.6%)

: Latent Dirichlet Allocation

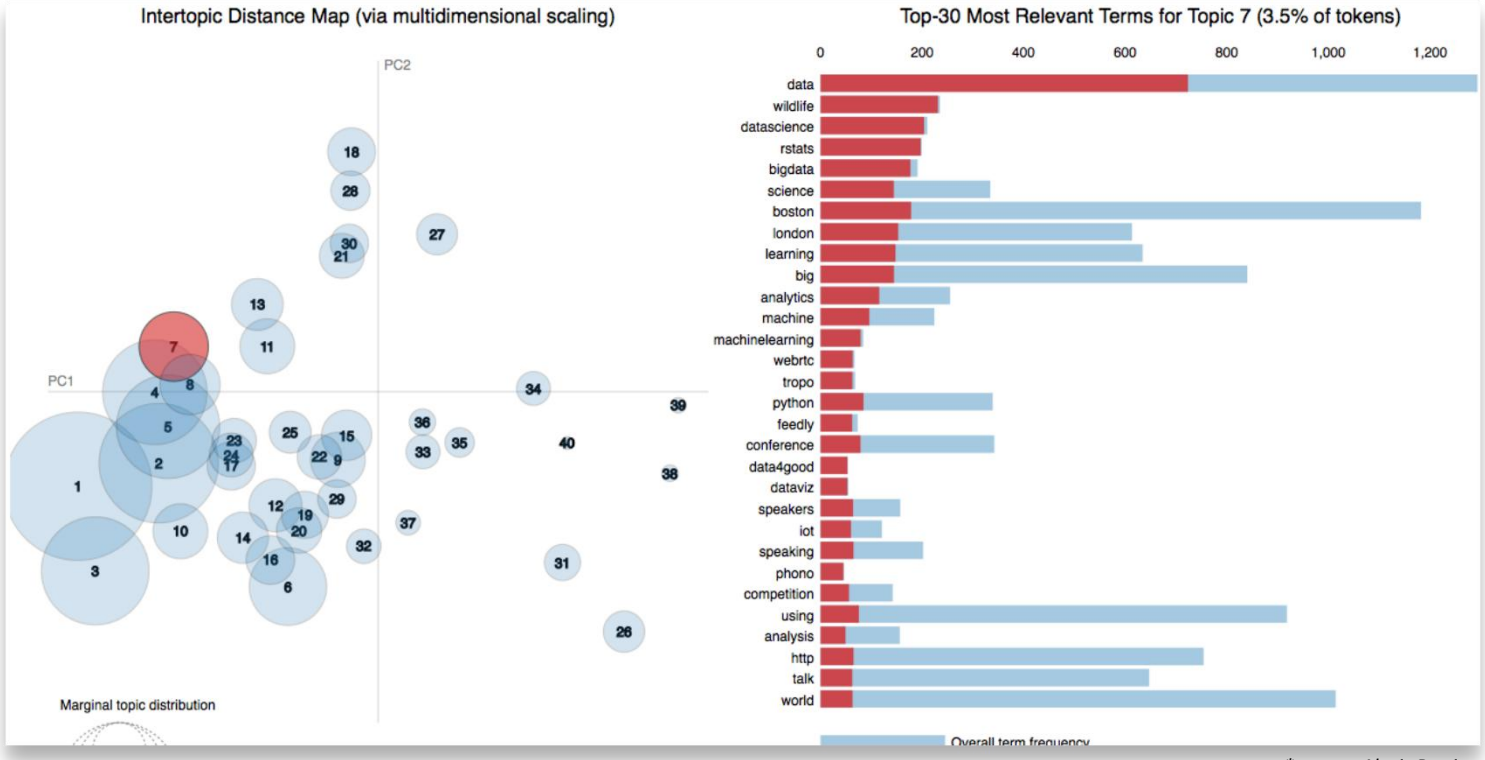
- A *semi-supervised* learning algorithm, used to predict topics and classify data points based on the repetition of a certain content across those data points.
- It's a "soft" classification method, in the sense that a single document can belong to 2 or more topic.
- Hard to train and tune, but very effective.



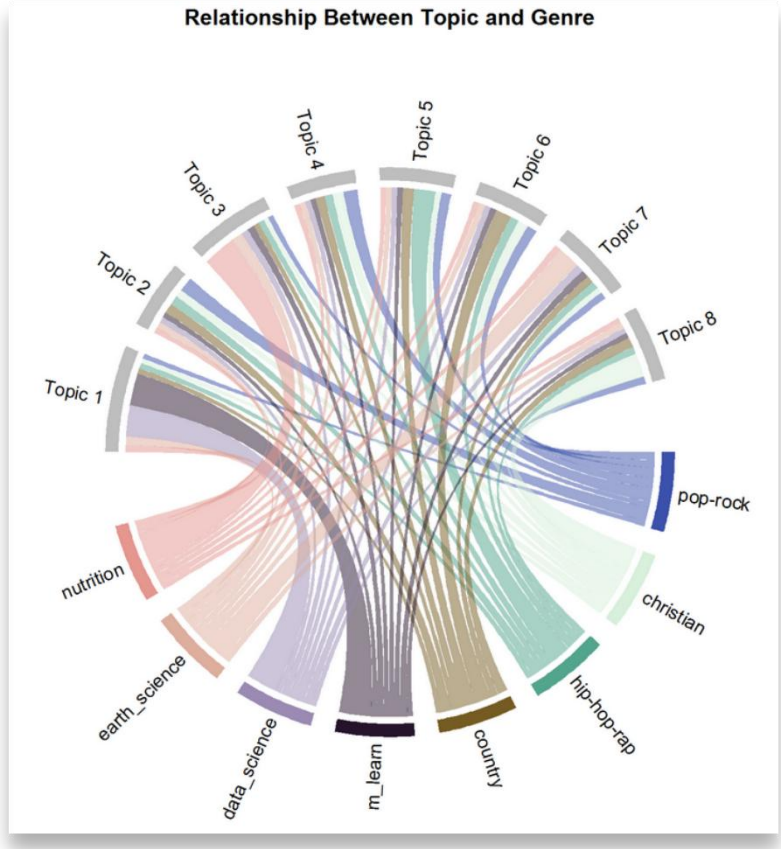
source: BigSnarf blog

- LDA Topic Modelling is used to discover broader topics of documents of any type, (include scientific data, like genetic measurements) and relationships between such topics.
- But was also used for content recommendation and analysis of overlapping communities in social networks.

LDA on Twitter messages



*source: Alexis Perrier



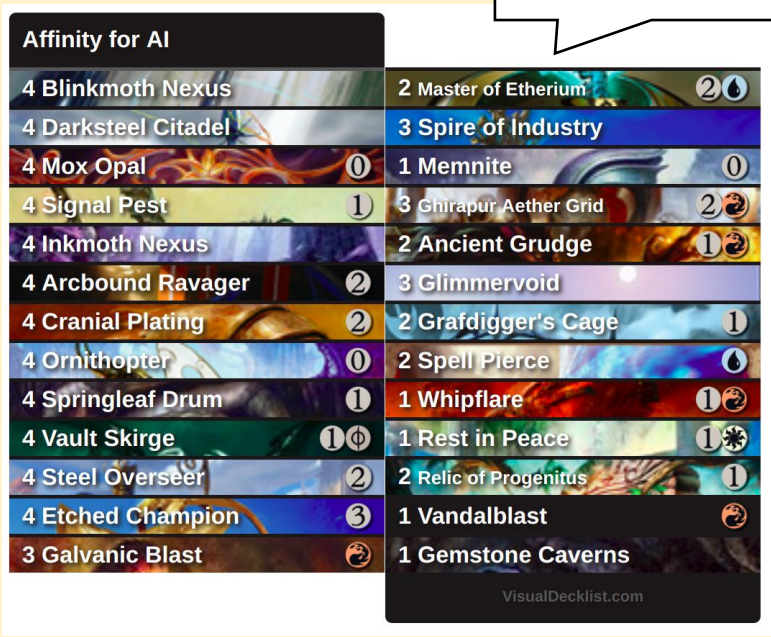
*source: Debbie Liske

How **LDA Topic Modelling** help Game Designers

- By revealing and structuring metagame patterns and strategies that players are using.
- Here's an [analysis from Hlynur Davíð Hlynsson] using LDA to discover and relate deck strategies in Magic: The Gathering.

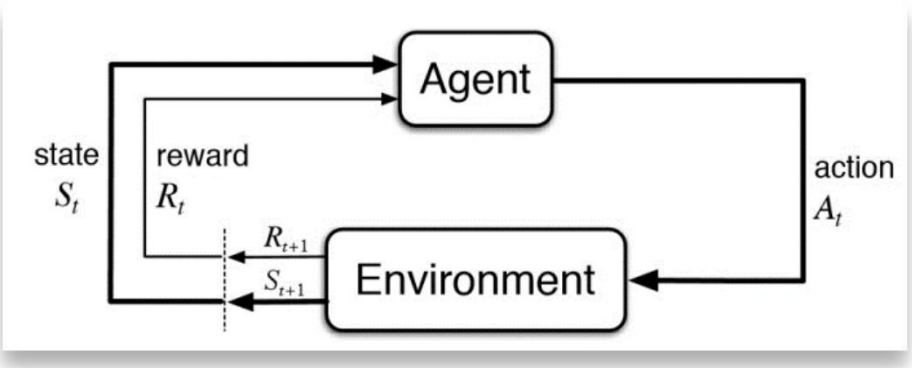
Since LDA is a generative model, we can also make this model create new decks!

The model discover Archetypes of decks and **the probability that a certain card will be present** in each deck Archetype.

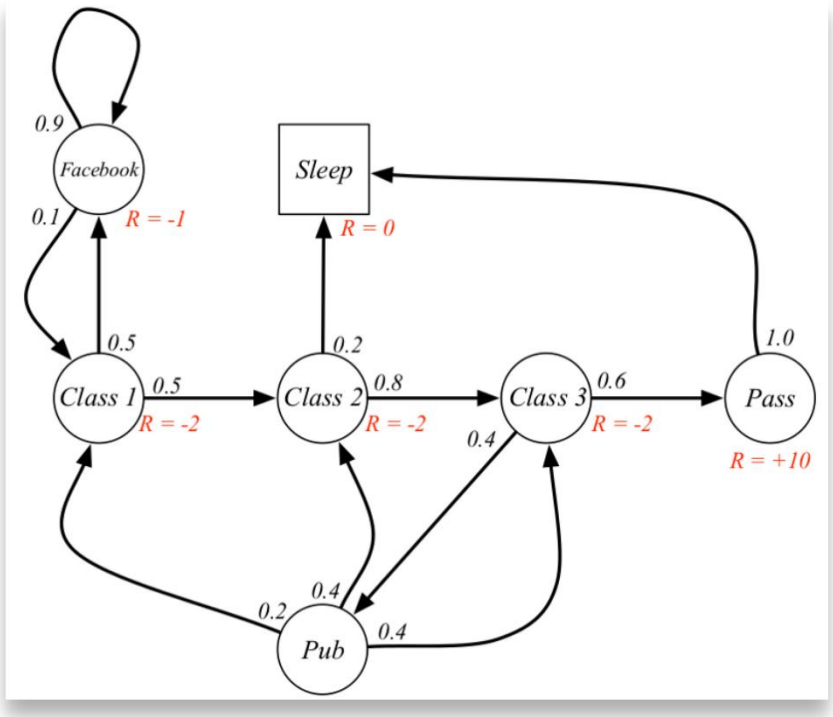


: Reinforcement Learning

- The one type of Machine Learning that doesn't require loads of data, but... doesn't do the same things either. :)
- The goal of Reinforcement models is to *learn how to execute tasks from scratch*, with minimal human intervention.
- It's a model based on Markov Decision Processes that learns on trial-and-error across thousands (millions?) of attempts, being guided by reward functions across many tries.



*source: Shweta Bhatt



*source: David Silver

- Is famously known for agents that learn to play games by themselves. (A team of *Mnih et al. 2015* trained agents in several different Atari games, frequently achieving super-human performance.)
- But is also used in other industries to manage resources in computer clusters, control traffic lights, robotics and even chemistry.

Google just gave control over data center cooling to an AI

In a first, Google is trusting a self-taught algorithm to manage part of its infrastructure.

by Will Knight August 17, 2018

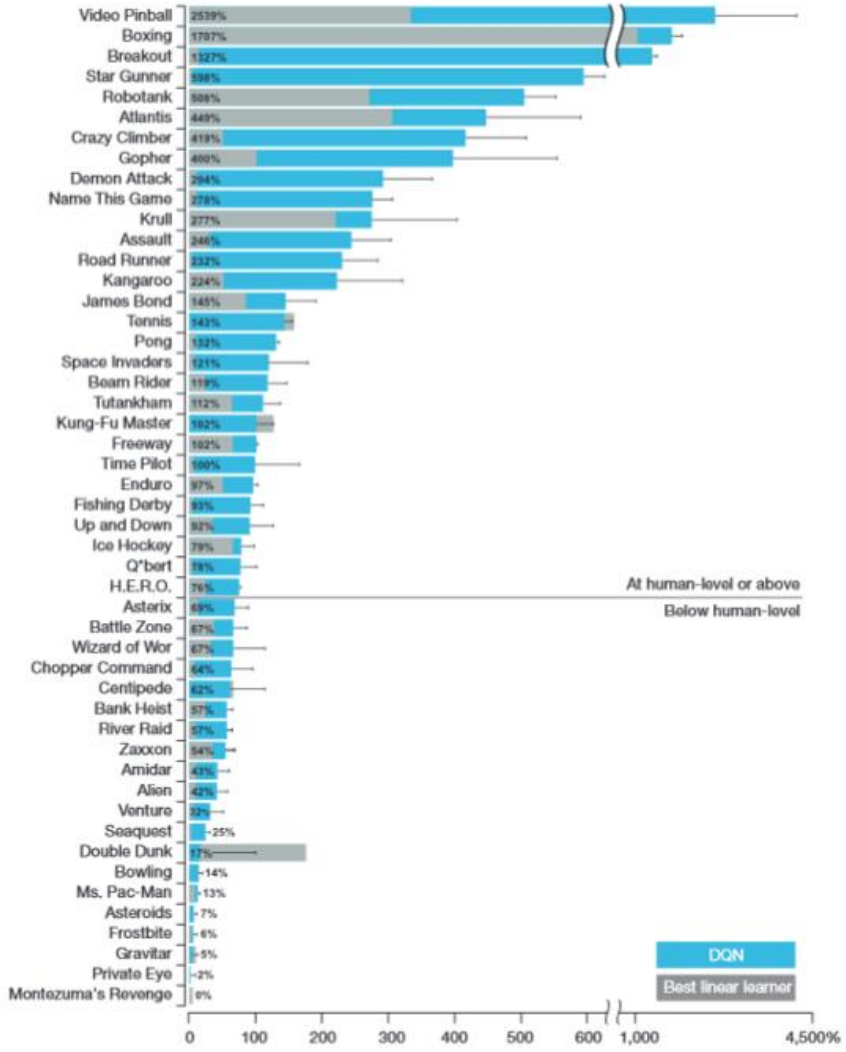


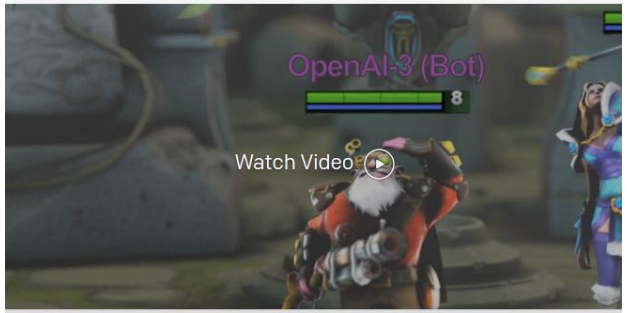
Figure 3 | Comparison of the DQN agent with the best reinforcement learning methods[15] in the literature. The performance of DQN is normalized with respect to a professional human games tester (that is, 100% level) and random play (that is, 0% level). Note that the normalized performance of DQN, expressed as a percentage, is calculated as: 100 × (DQN score − random play score)/(human score − random play score). It can be seen that DQN outperforms competing methods (also see Extended Data Table 2) in almost all the games, and performs at a level that is broadly comparable with or superior to a professional human games tester (that is, operationalized as a level of 75% or above) in the majority of games. Audio output was disabled for both human players and agents. Error bars indicate s.d. across the 30 evaluation episodes, starting with different initial conditions.

- A competition using Doom has pit together different reinforcement learning models from competitors in a bloody free-for-all!

- Current science of OpenAI labs on Deep Reinforcement Learning are finding ways around very hard AI problems around certain game mechanics: long-term planning, curiosity, team play.
- Finding an architecture that can learn to play **anything** in the near future is becoming less and less an impossible dream.

OpenAI Five

JUNE 25, 2018



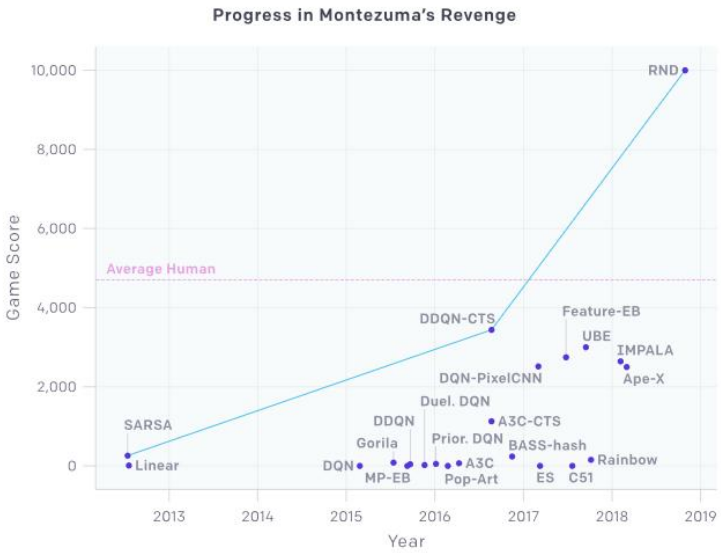
Our team of five neural networks, OpenAI Five, has started to defeat amateur human teams at Dota 2. While today we play with restrictions, we aim to beat a team of top professionals at The International in August subject only to a limited set of heroes. We may not succeed: Dota 2 is one of the most popular and complex esports games in the world, with creative and motivated professionals who train year-round to earn part of Dota's annual $40M prize pool (the largest of any esports game).

OpenAI Five plays 180 years worth of games against itself every day, learning via self-play. It trains using a scaled-up version of Proximal Policy Optimization running on 256 GPUs and 128,000 CPU cores — a larger-scale version of the system we built to play the much-simpler solo variant of the game last year. Using a separate LSTM for each hero and no human data, it learns recognizable strategies. This indicates that reinforcement learning can yield long-term planning with large but achievable scale — without fundamental advances, contrary to our own expectations upon starting the project.

Reinforcement Learning with Prediction-Based Rewards

OCTOBER 31, 2018

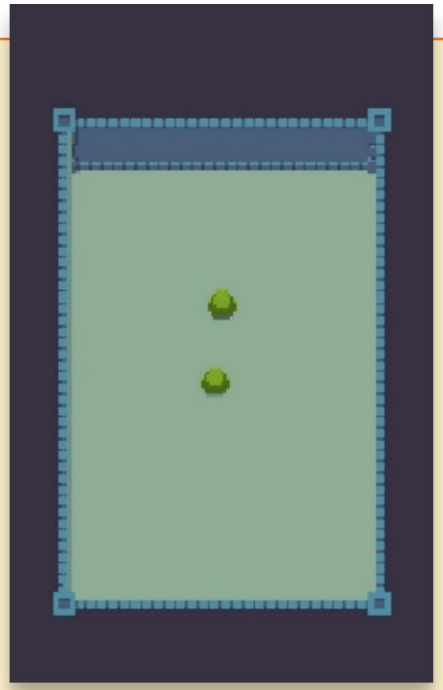
We've developed **Random Network Distillation (RND)**, a prediction-based method for encouraging reinforcement learning agents to explore their environments through curiosity, which for the first time[1] exceeds average human performance on Montezuma's Revenge. **RND achieves state-of-the-art performance, periodically finds all 24 rooms and solves the first level without using demonstrations or having access to the underlying state of the game.**



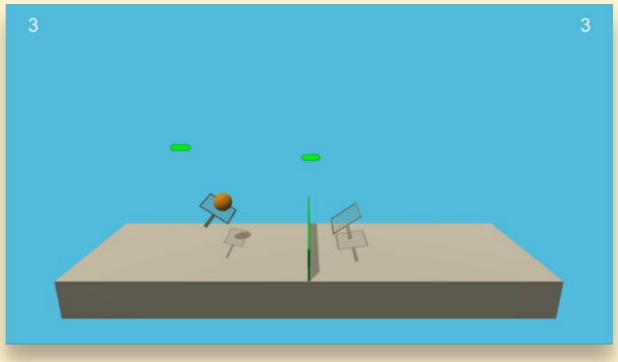
How **Reinforcement Learning** help Game Designers

Reinforcement learning models open a world of possibilities *in conjunction with* ***Deep Neural Networks*****.** Since they require very specialized engineering and science, at the moment we see few systems like that in game development, but here's some **future possibilities**:

- Enemy AI that is *trained instead of coded*.
- Testing bots that play the engine 24/7 and help QA.
- Agents that play the metagame and test long-term progression design and economy.
- Generative Adversarial Networks that *help balance the game economy and progression.*
- Narrative agents without pre-scripted dialogues, that react to the player's actions and choices more organically.
- Matchmaking for PvP games.



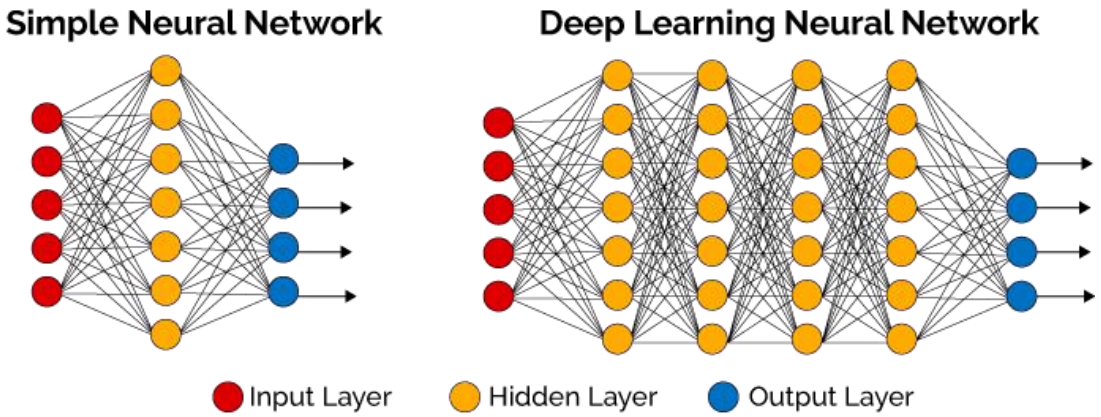
*source: Unity



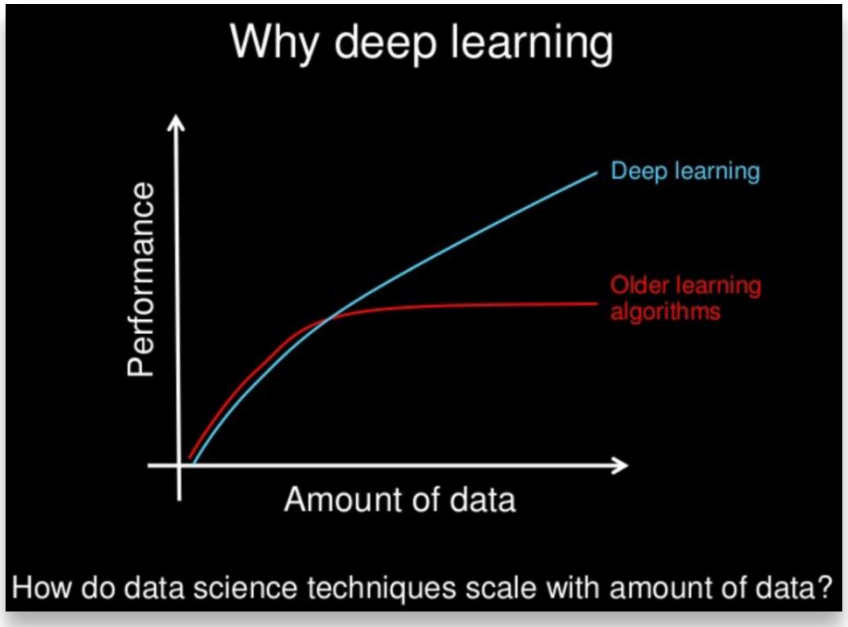
*source: Unity

: Deep Learning

*source: Favio Vázquez



- A class of ML models composed of stacked Neural Networks.
- **Very flexible and very powerful**, and *continue to improve over time* as the amount of data increases.
- **But costly:** hard to tune, time-consuming, require GPUs, take a *lot* of processing time to train.
- Also, nearly impossible to know why and how a deep learning model makes decisions.

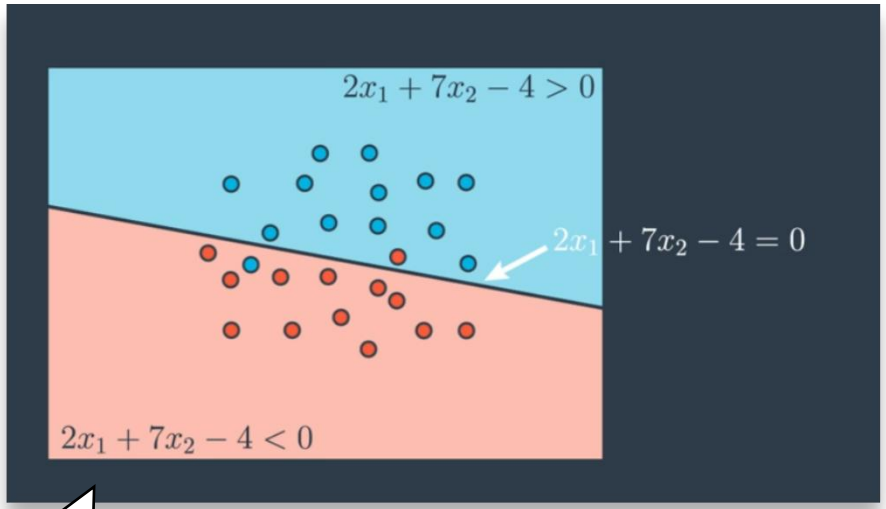


*source: Andew Ng

- First, lets understand what a Neural Network does. It's an architecture (usually used as a Classifier) that combines the separation made by various "neurons".

A "neuron" (Perceptron)

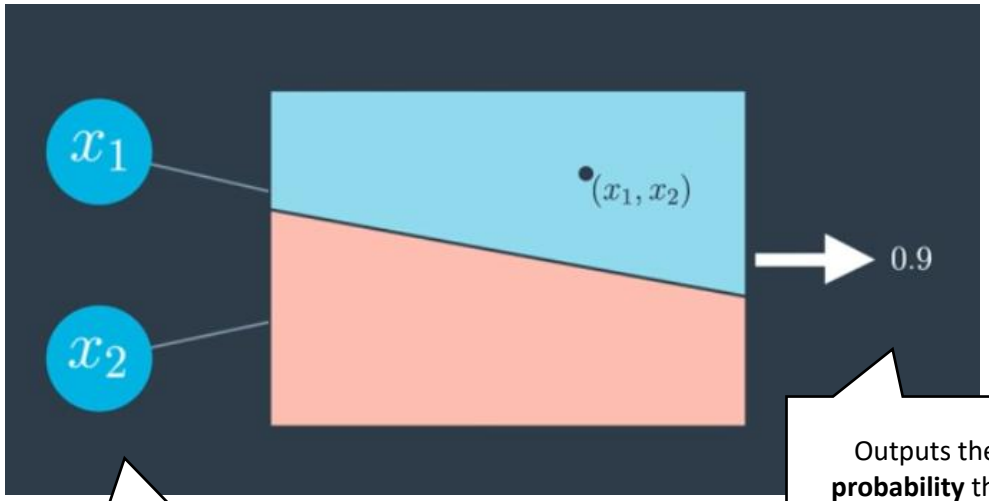
Training Data



*source: Udacity

Trained to distinguish **the blue category** with a linear function.

*source: Udacity



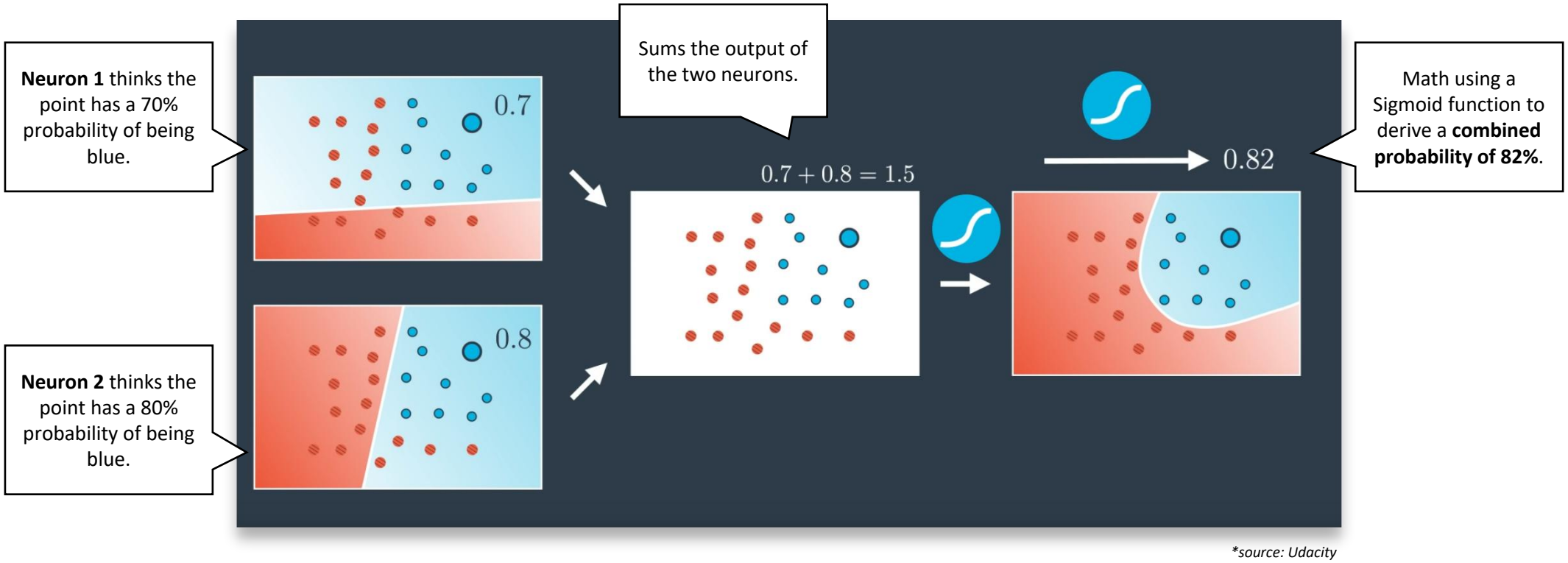
Receives coordinates of a new data point.

Outputs the **probability** that point belongs to the blue category.

- However, in the real world, very few datasets are separable by one line. So then, like in the example below, we create a network of 2 neurons to create a more complex boundary.

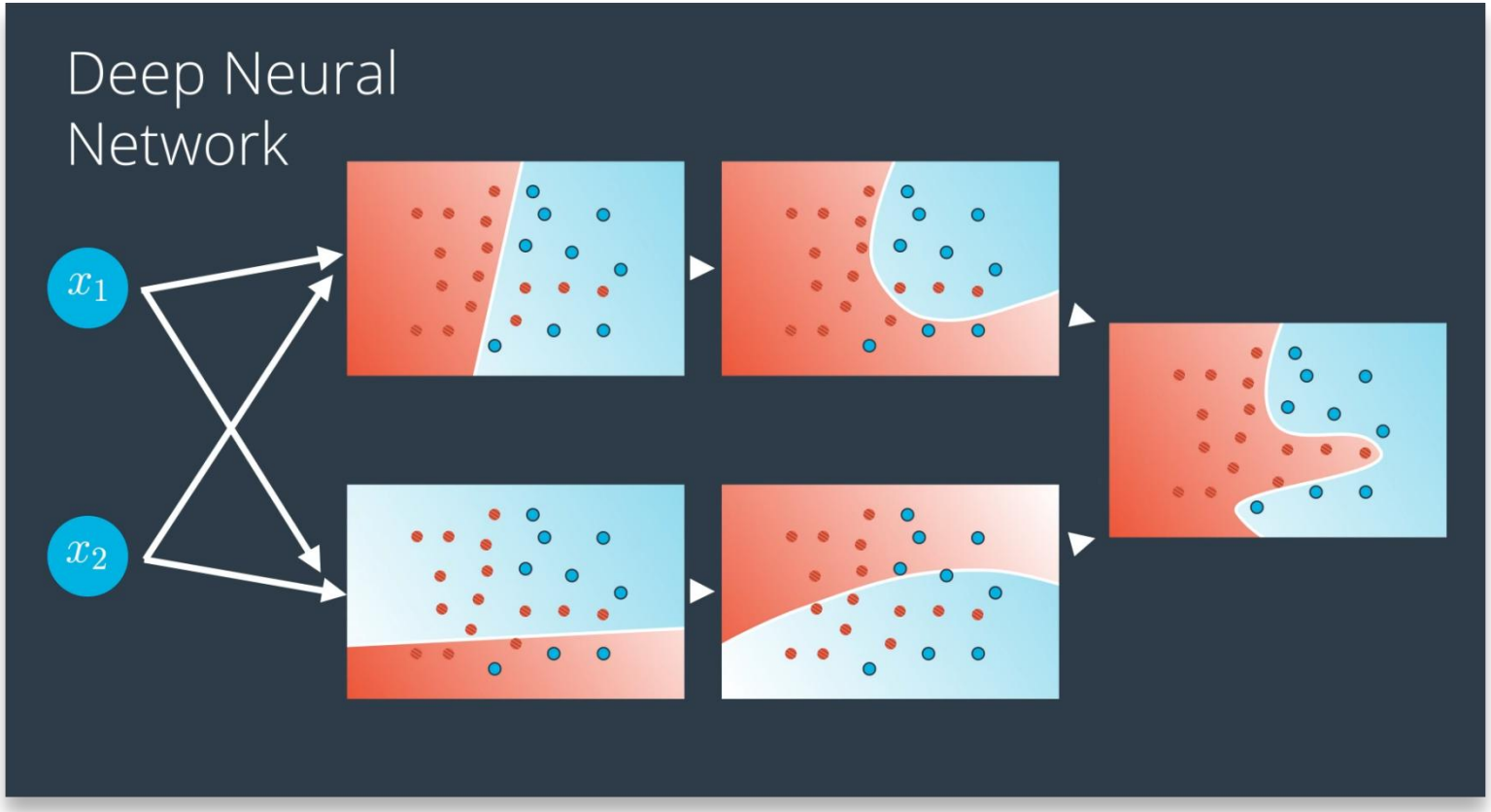
*source: Udacity

A Neural Network



*source: Udacity

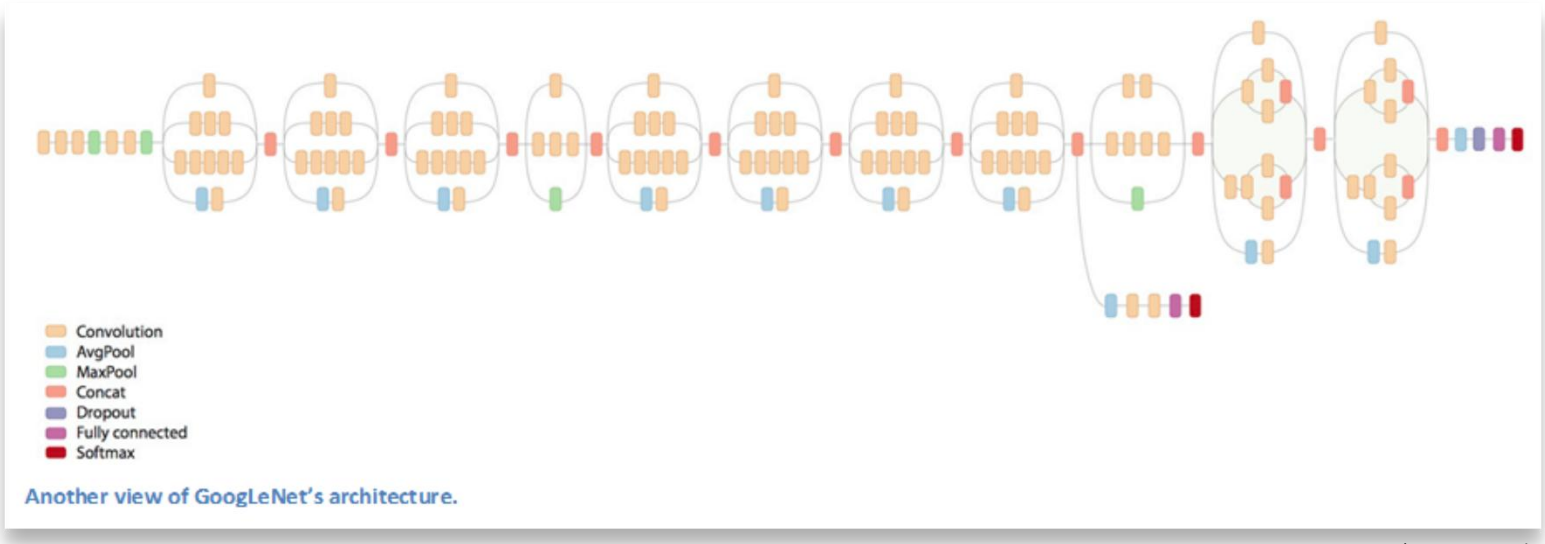
- Finally, for really complicated classifications, we can stack more layers of neurons that together create a very sophisticated classifier. The non-linear model of a layer is further expanded into more complex non-linear boundaries. **That's a deep neural network.**



*source: Udacity


WE NEED TO GO DEEPER

- The ability to stack more layers and experiment with different architectures when connecting neurons is what makes deep neural networks so flexible. And it can become really sophisticated.
- For example, Google's Inception models:



Another view of GoogLeNet's architecture.

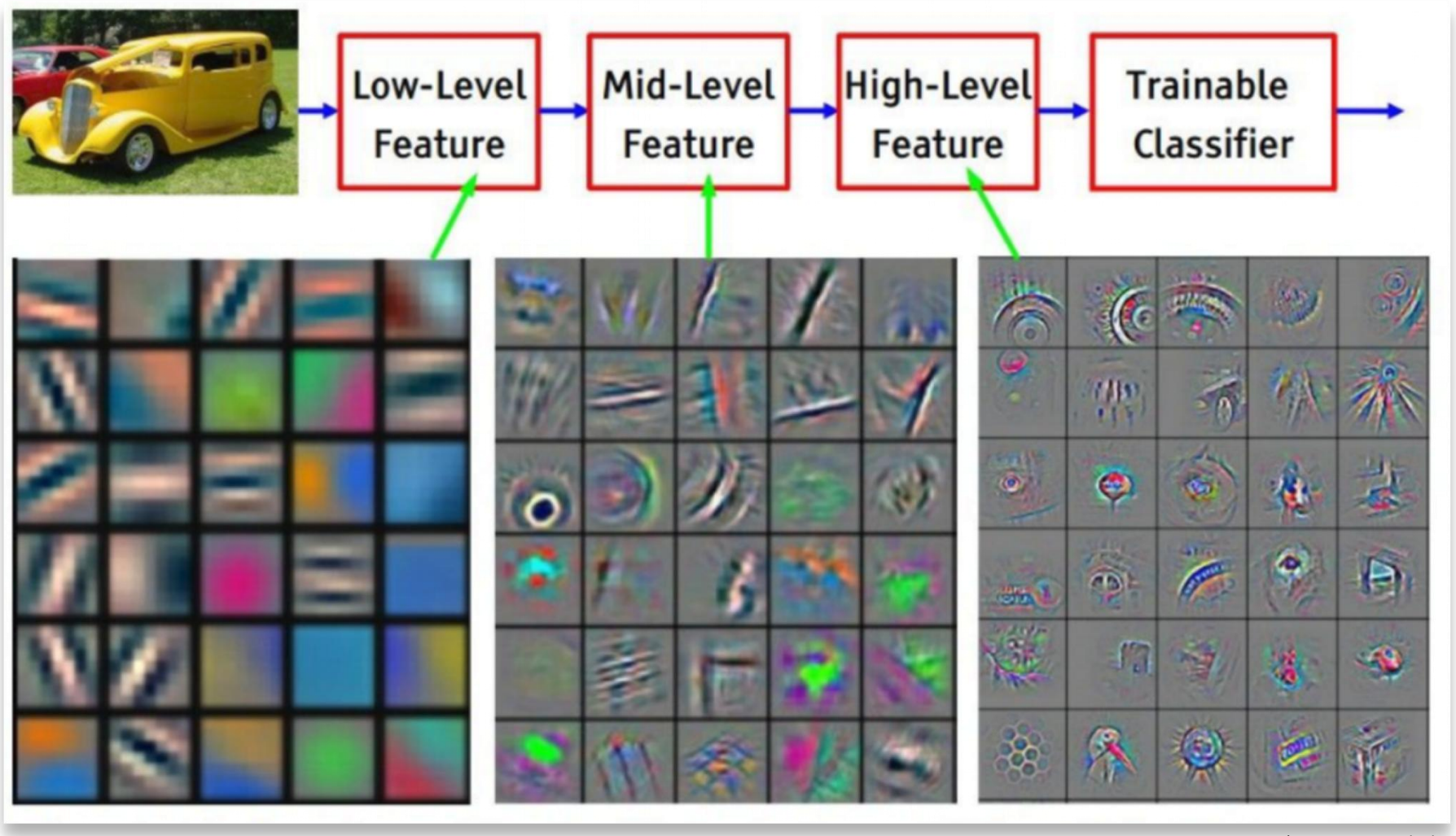
*source: Google

*source: MathWorks

```
Using Inception_V3 CNN
Prediction: Probability
bison: 93.83%
ox: 0.13%
ibex, Capra ibex: 0.05%
rhinoceros beetle: 0.05%
brown bear, bruin, Ursus arctos: 0.03%
ladybug, ladybeetle, lady beetle, ladybird, ladybird beetle: 0.03%
Leonberg: 0.03%
promontory, headland, head, foreland: 0.03%
cougar, puma, catamount, mountain lion, painter, panther, Felis concolor: 0.03%
wool, woolen, woollen: 0.03%
```

*source:William Koehrsen

- This demonstrates how stacking layers in a deep neural network results in deeper neurons learning higher and higher-level abstractions.



*source: NamHyuk Ahn

- My own experiments with Microsoft's ResNet 50

==========

I see a dog!

This guys seems to be a...
Welsh_springer_spaniel

==========

I see a dog!

This guys seems to be a...
Labrador_retriever

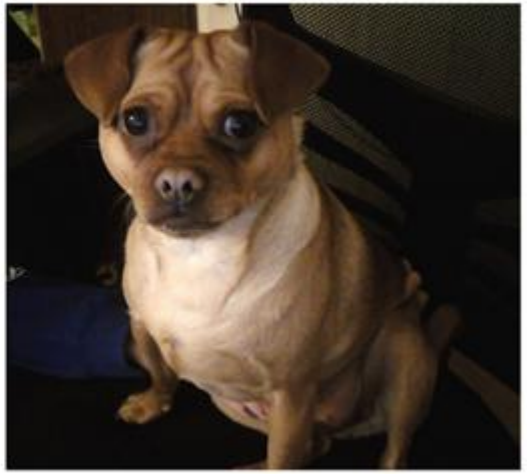
==========

I see a dog!

This guys seems to be a...
Brittany

==========

I see a dog!



This guys seems to be a...
Chihuahua

==========

I found some other object.

It looks like a...
joystick

==========

I found some other object.

It looks like a...
hourglass

*source of images: ImageNet

```
================

Hello, human!
```

```
You look quite like a...
French_bulldog
```

Really?...

How **Deep Learning** help Game Designers

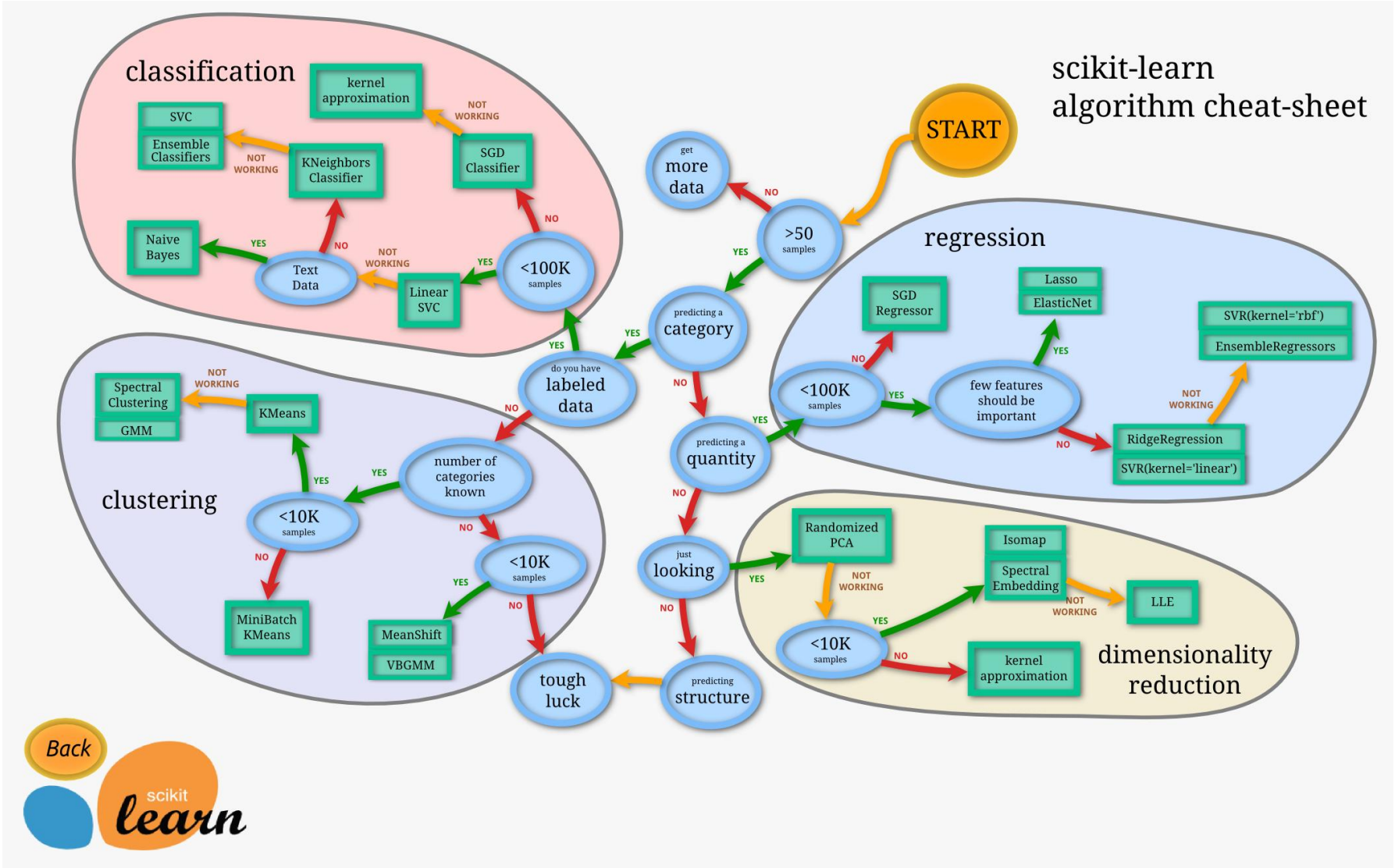
Like for everyone else: essentially, **Deep Neural Networks can do anything that previous algorithms do** but in a more robust, scalable and accurate way.

However, since they are still more costly to develop and deploy, we will have some years of transition. (Also because the field of Deep Learning still needs to mature more in "models ready to go" for specific business needs, like what is happening on image recognition.)

Moreover, Deep Learning models are also potentially much better at content generation and prediction tasks. Some functions that are hard to develop but will eventually be faster to develop and train, and help game designers immensely:

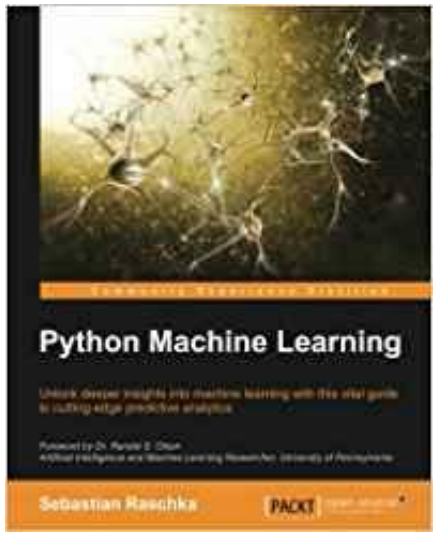
- Predict the next moves and decisions of each user.
- Predict resource inflation.
- **Procedural content creation, like level design.**
- Balance game economy.
- Train NPCs that react more naturally.

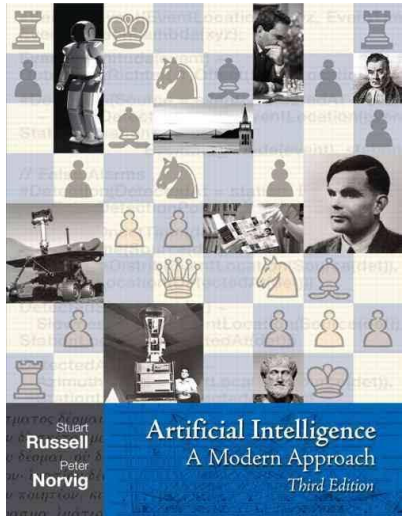
: Deciding what to use

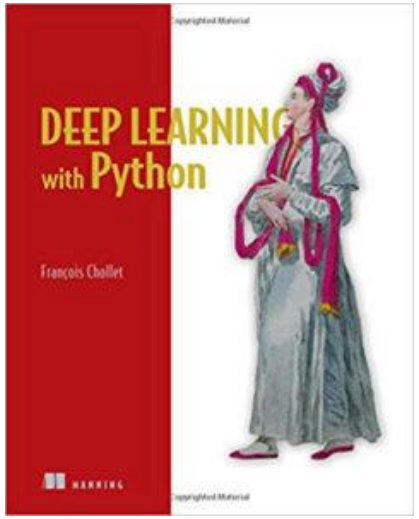


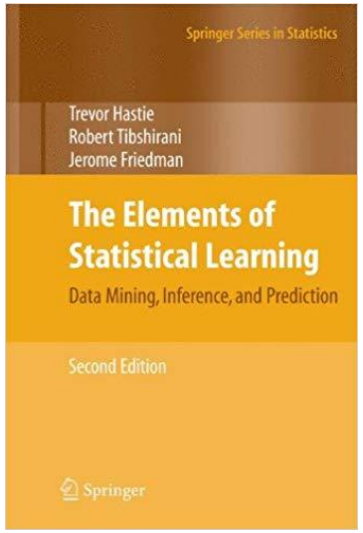
Start here!

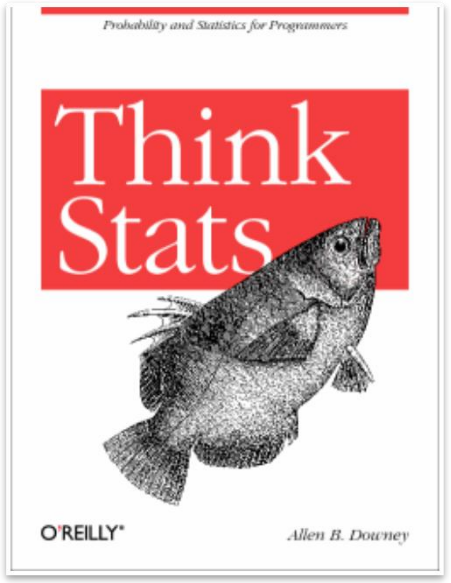
http://www.r2d3.us/

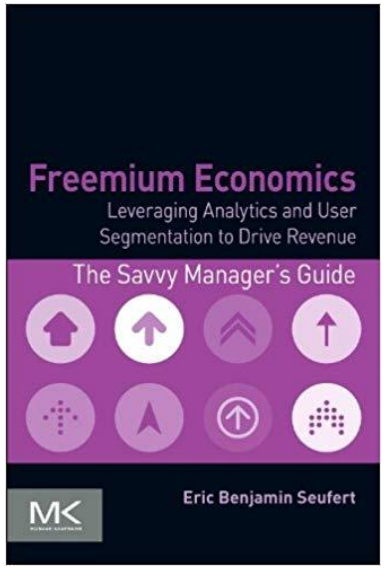


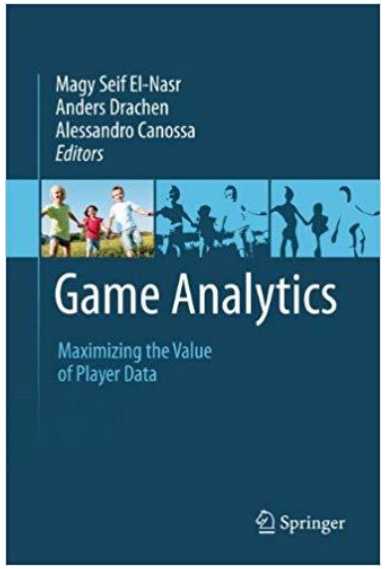












Thank you!

Questions?

@texpine

: Appendix


- Icon Game by Orin zuu from the Noun Project


- Icon Lighthouse by Nikita Kozin from the Noun Project


- Icon User by Gagana from the Noun Project


- Icon Car on Sale from all-free-download.com / BSGStudio


- Icon Flag from all-free-download.com / Vector Graphics


- Icon Flag from all-free-download.com / BSGStudio
-