2-27-2019

Kaggle and Click-Through Rate Prediction

Todd W. Neller
Gettysburg College

Kaggle and Click-Through Rate Prediction

Abstract

Neller presented a look at Kaggle.com, an online Data Science and Machine Learning learning community, as a place to seek rapid, experiential peer education for most any Data Science topic. Using the specific challenge of Click-Through Rate Prediction (CTRP), he focused on lessons learned from relevant Kaggle competitions on how to perform CTRP.

Keywords

Data science, machine learning, Kaggle, click-through rate prediction

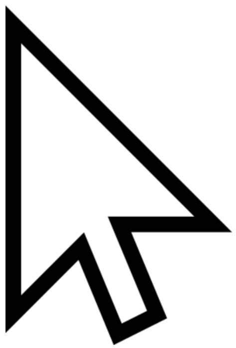
Disciplines

Computer Sciences | Databases and Information Systems

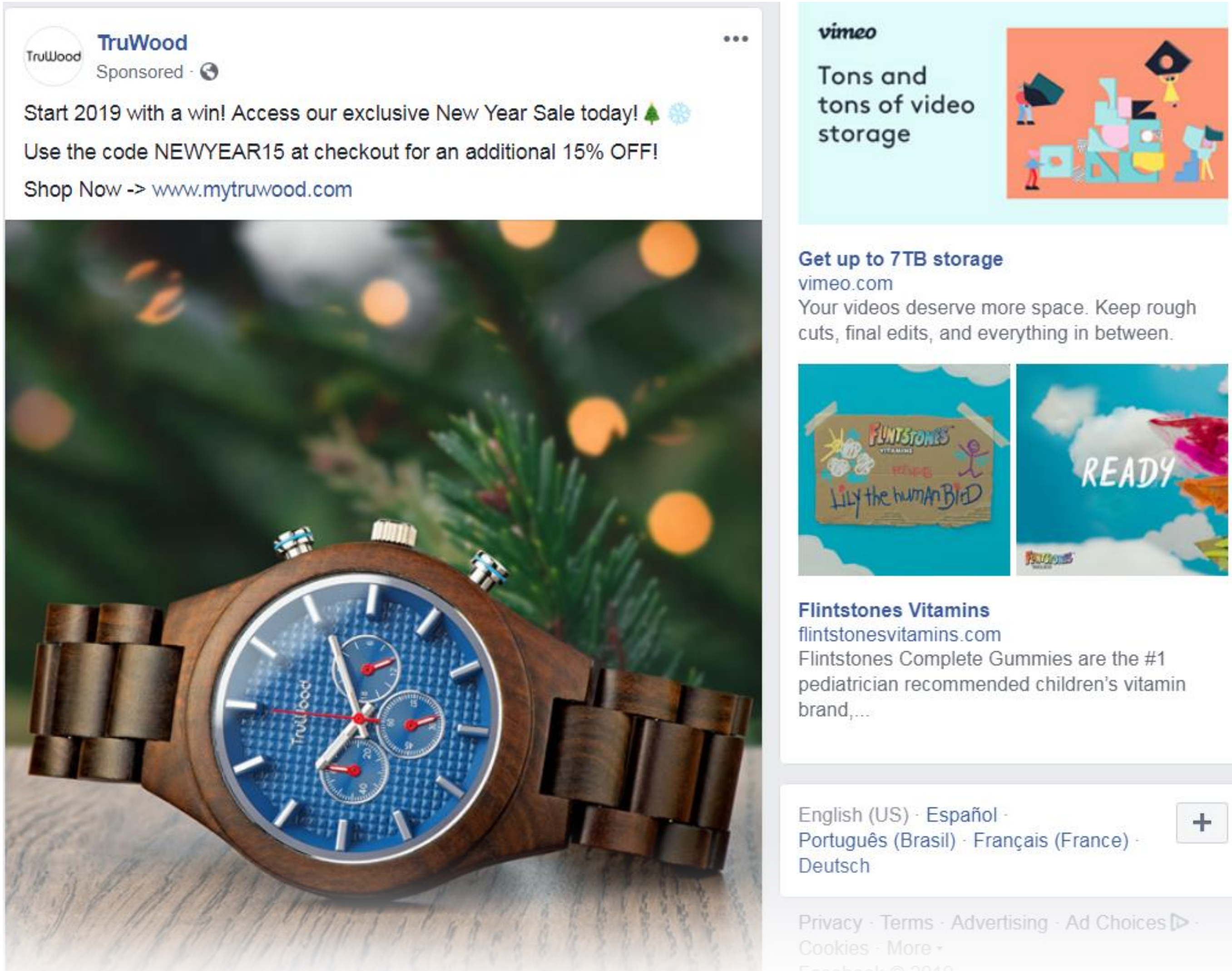
Comments

This presentation was given at the Gettysburg College Computer Science Colloquium on February 27th, 2019.

Kaggle and Click-Through Rate Prediction



Dr. Todd W. Neller
Professor of Computer Science
Gettysburg College


TruWood
TruWood
Sponsored ·
Start 2019 with a win! Access our exclusive New Year Sale today! 🌲 ❄️
Use the code NEWYEAR15 at checkout for an additional 15% OFF!
Shop Now -> www.mytruwood.com
vimeo
Tons and tons of video storage
Get up to 7TB storage
vimeo.com
Your videos deserve more space. Keep rough cuts, final edits, and everything in between.
FLINTSTONES
Lily the human Bird
READY
Flintstones Vitamins
flintstonesvitamins.com
Flintstones Complete Gummies are the #1 pediatrician recommended children's vitamin brand,...
English (US) · Español · Português (Brasil) · Français (France) · Deutsch
+
Privacy · Terms · Advertising · Ad Choices · Cookies · More ·
Facebook © 2019

Click-Through Rate (CTR) Prediction

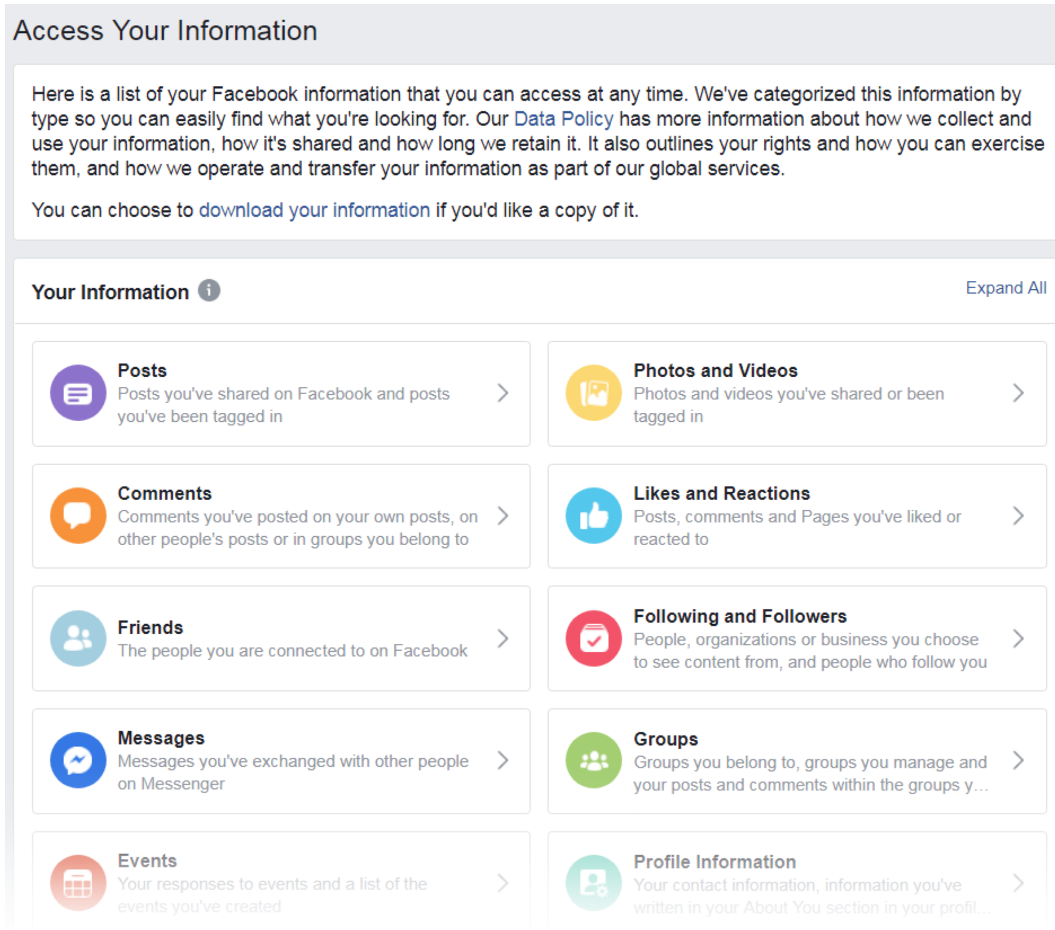
$$\text{CTR} = \frac{\text{Number of click-throughs}}{\text{Number of impressions}} \times 100(\%)$$

- Number of impressions = number of times an advertisement was served/offered
- <u>Given:</u> much data on past link offerings and whether or not users clicked on those links
- <u>Predict:</u> the probability that a current user will click on a given link

Example Data on Past Link Offerings

- User data:
  - User ID from site login, cookie
  - User IP address, IP address location
- Link context data:
  - Site ID, page ID, prior page(s)
  - Time, date
- Link data:
  - Link ID, keywords
  - Position offered on page

Example: Facebook Information



Why is CTR Prediction Important?

- Advertising Industry View:
  - Much of online advertising is billed using a **pay-per-click model**.

Better CTR Prediction

Better Ad Selection

Greater Click-Through Rate

Greater Ad Revenue

New Idea?

https://www.slideshare.net/savvakos/how-you-can-earn-attention-in-the-digital-world-80695468

Benefits Beyond Advertising

- Herbert Simon, 1971:
  - "In an information-rich world, the wealth of information means a dearth of something else: a scarcity of whatever it is that information consumes. What information consumes is rather obvious: the **attention** of its recipients."
- Better CTR prediction → more *relevance* → better use of scarce time

Outline

- Click-Through Rate Predition (CTRP) Introduction
- Kaggle
  - Learning community offerings incentives
  - CTRP Competitions
- Feature Engineering
  - Numbers, Categories, and Missing Values
- Favored regression techniques for CTRP
  - Logistic Regression
  - Gradient Boosted Decision Trees (e.g. xgBoost)
  - Field-aware Factorization Machines (FFMs)
- Future Recommendations

What is Kaggle.com?

- Data Science and Machine Learning Community featuring
  - Competitions → $$$, peer learning, experience, portfolio
  - Datasets
  - Kernels
  - Discussions
  - Tutorials ("Courses")
  - Etc.
- Status incentives

15 Active Competitions

Two Sigma: Using News to Predict Stock Movements
Use news analytics to predict stock price performance
Featured · Kernels Competition · 5 months to go · news agencies, time series, finance, money
$100,000
2,897 teams

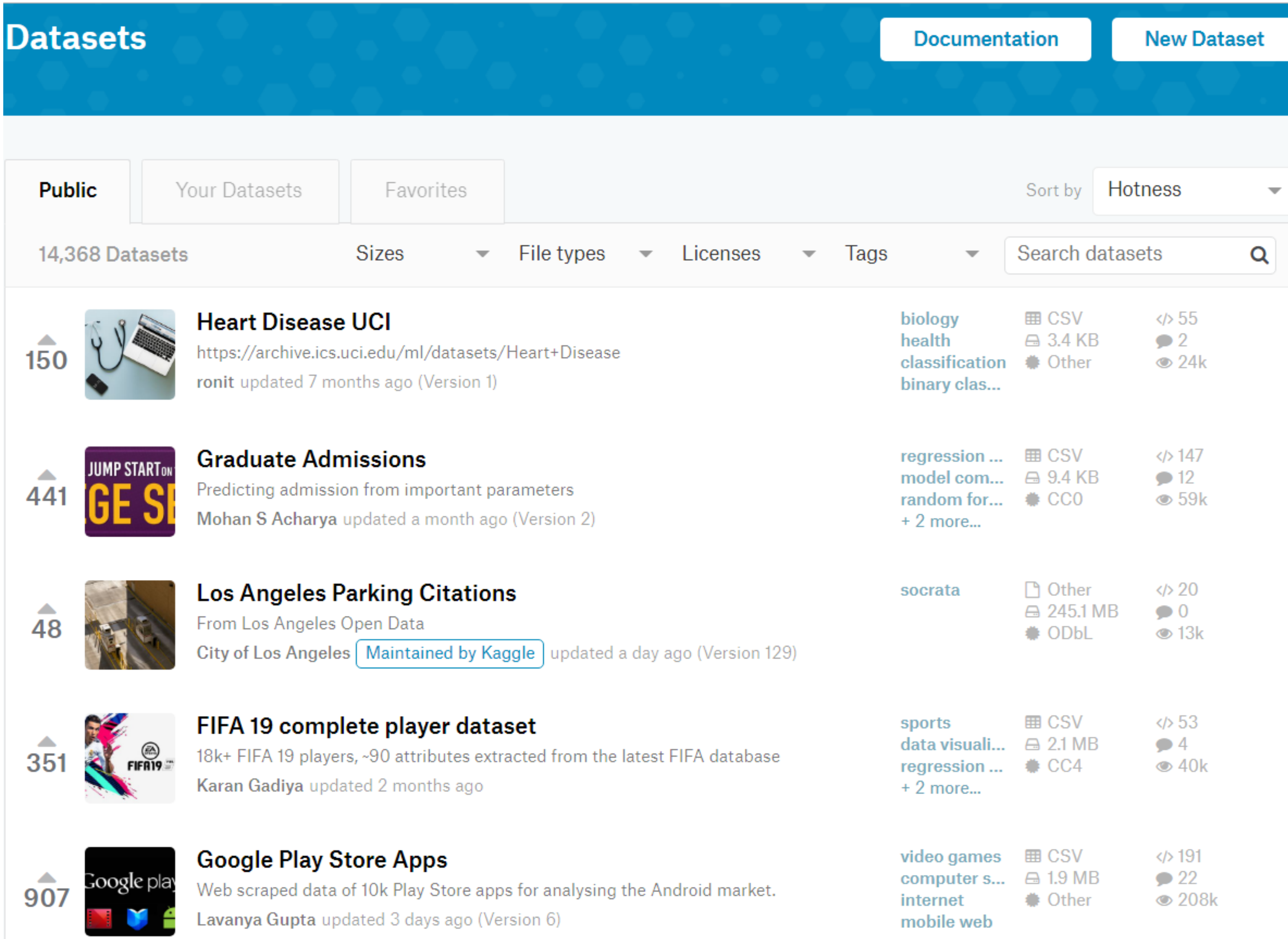
LANL Earthquake Prediction
Can you predict upcoming laboratory earthquakes?
Research · 4 months to go · earth sciences, physics, signal processing
$50,000
938 teams

Elo Merchant Category Recommendation
Help understand customer loyalty
Featured · 20 days to go · tabular data, banking, regression
$50,000
3,487 teams

Google Analytics Customer Revenue Prediction
Predict how much GStore customers will spend
Featured · 9 days to go · regression, tabular data
$45,000
1,104 teams

Gendered Pronoun Resolution
Pair pronouns to their correct entities
Research · 3 months to go · nlp, text data
$25,000
30 teams

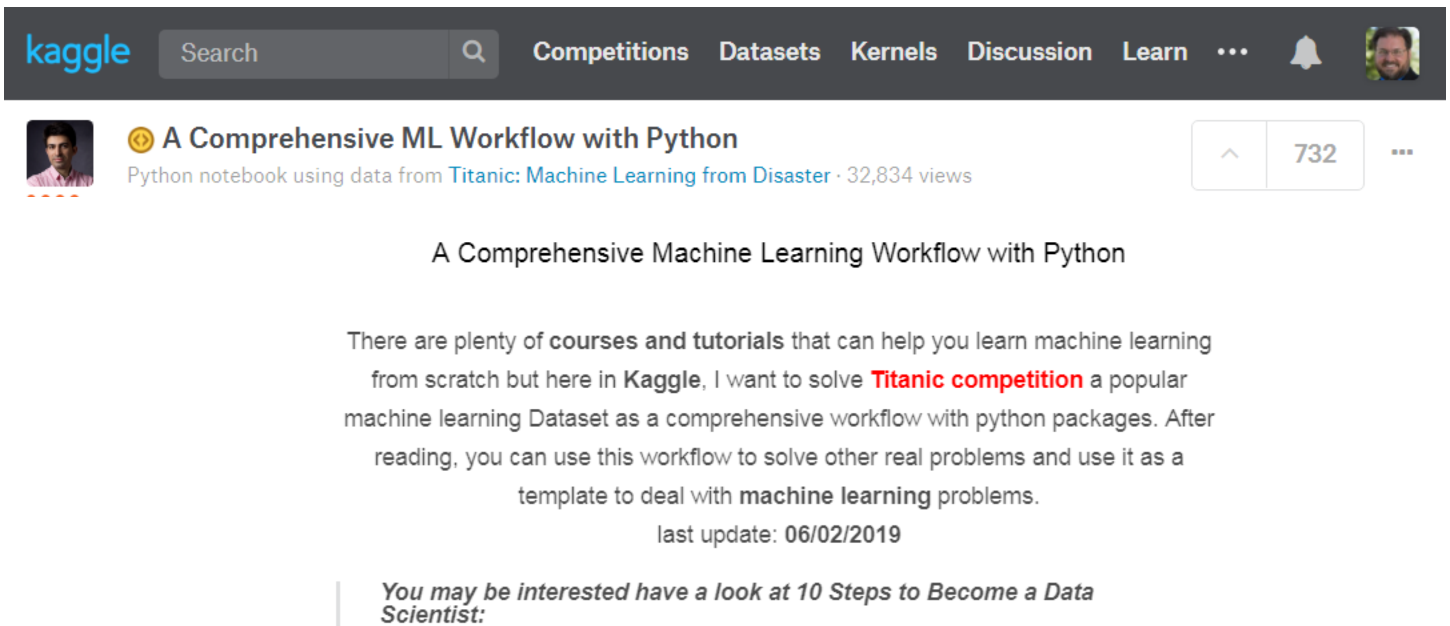
PetFinder.my Adoption Prediction
How cute is that doggy in the shelter?
$25,000
972 teams



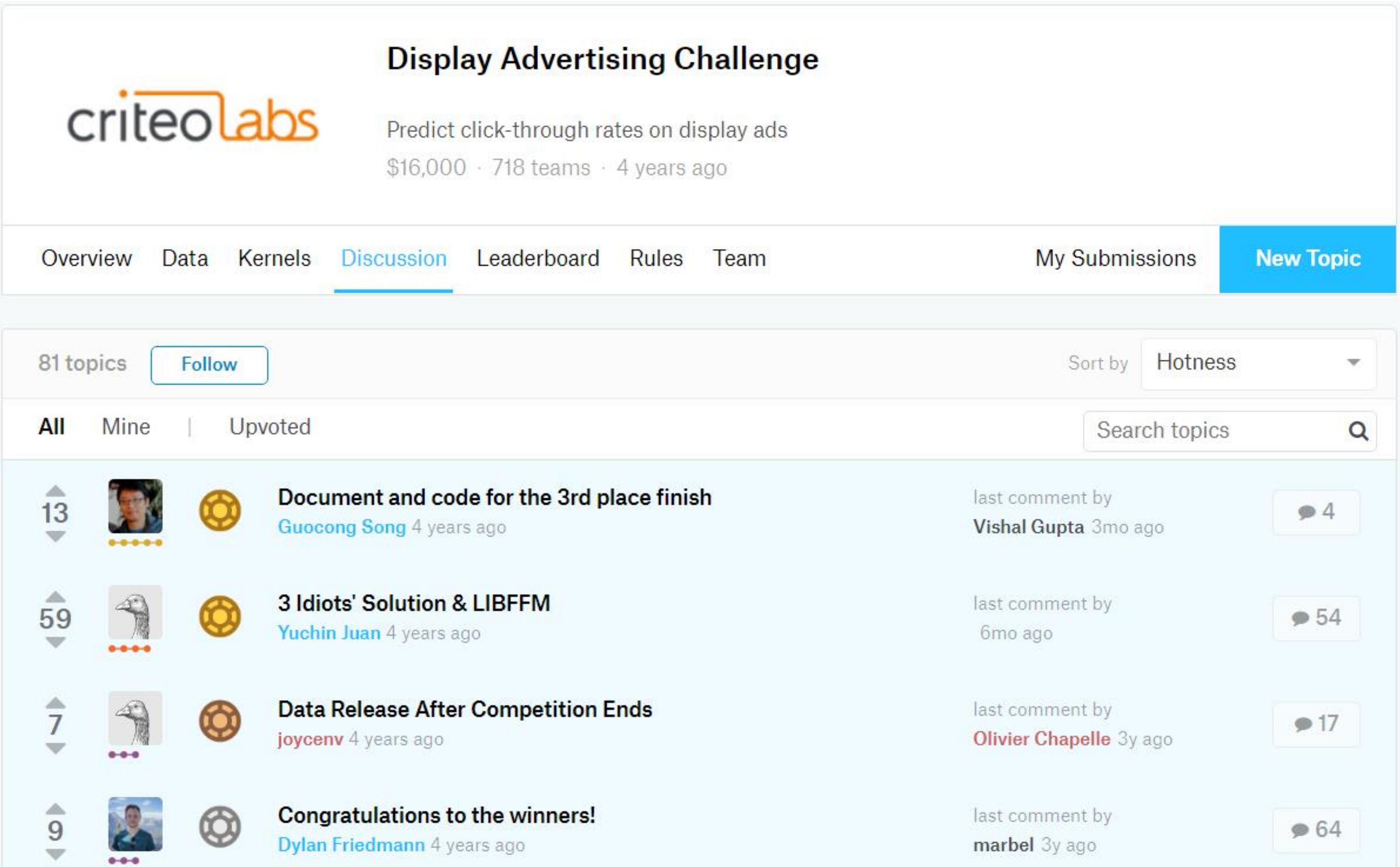
Datasets
Documentation
New Dataset
Public
Your Datasets
Favorites
Sort by
Hotness
14,368 Datasets
Sizes
File types
Licenses
Tags
Search datasets
150
Heart Disease UCI
https://archive.ics.uci.edu/ml/datasets/Heart+Disease
ronit updated 7 months ago (Version 1)
biology
health
classification
binary clas...
CSV
3.4 KB
Other
55
2
24k
441
Graduate Admissions
Predicting admission from important parameters
Mohan S Acharya updated a month ago (Version 2)
regression ...
model com...
random for...
+ 2 more...
CSV
9.4 KB
CC0
147
12
59k
48
Los Angeles Parking Citations
From Los Angeles Open Data
City of Los Angeles
Maintained by Kaggle
updated a day ago (Version 129)
socrata
Other
245.1 MB
ODbL
20
0
13k
351
FIFA 19 complete player dataset
18k+ FIFA 19 players, ~90 attributes extracted from the latest FIFA database
Karan Gadiya updated 2 months ago
sports
data visuali...
regression ...
+ 2 more...
CSV
2.1 MB
CC4
53
4
40k
907
Google Play Store Apps
Web scraped data of 10k Play Store apps for analysing the Android market.
Lavanya Gupta updated 3 days ago (Version 6)
video games
computer s...
internet
mobile web
CSV
1.9 MB
Other
191
22
208k

Kernels

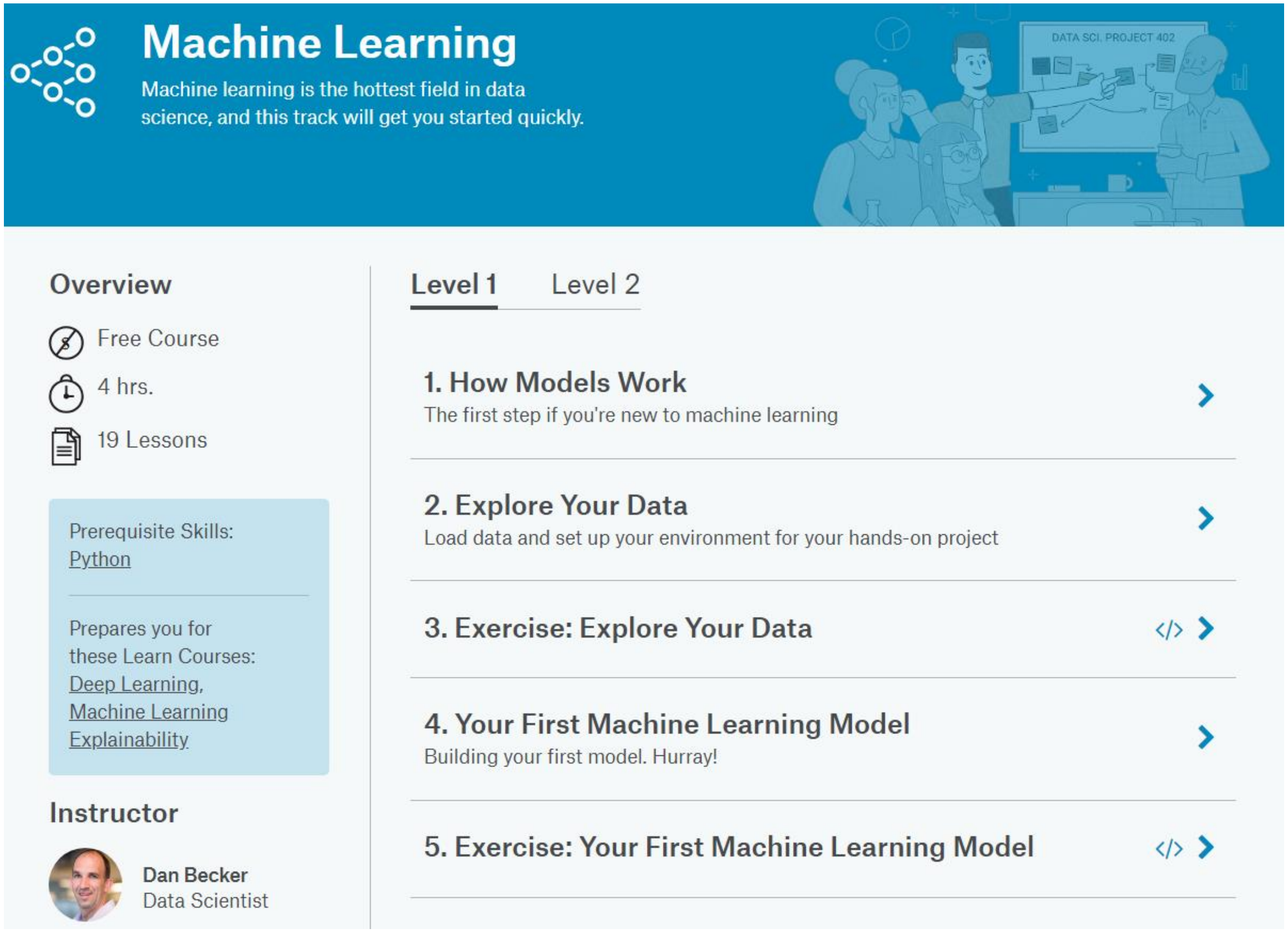
- Jupyter notebooks of mixed text and Python/R
  - Interleaved explanations and free runnable code
- E.g. https://www.kaggle.com/mjbahmani/a-comprehensive-ml-workflow-with-python



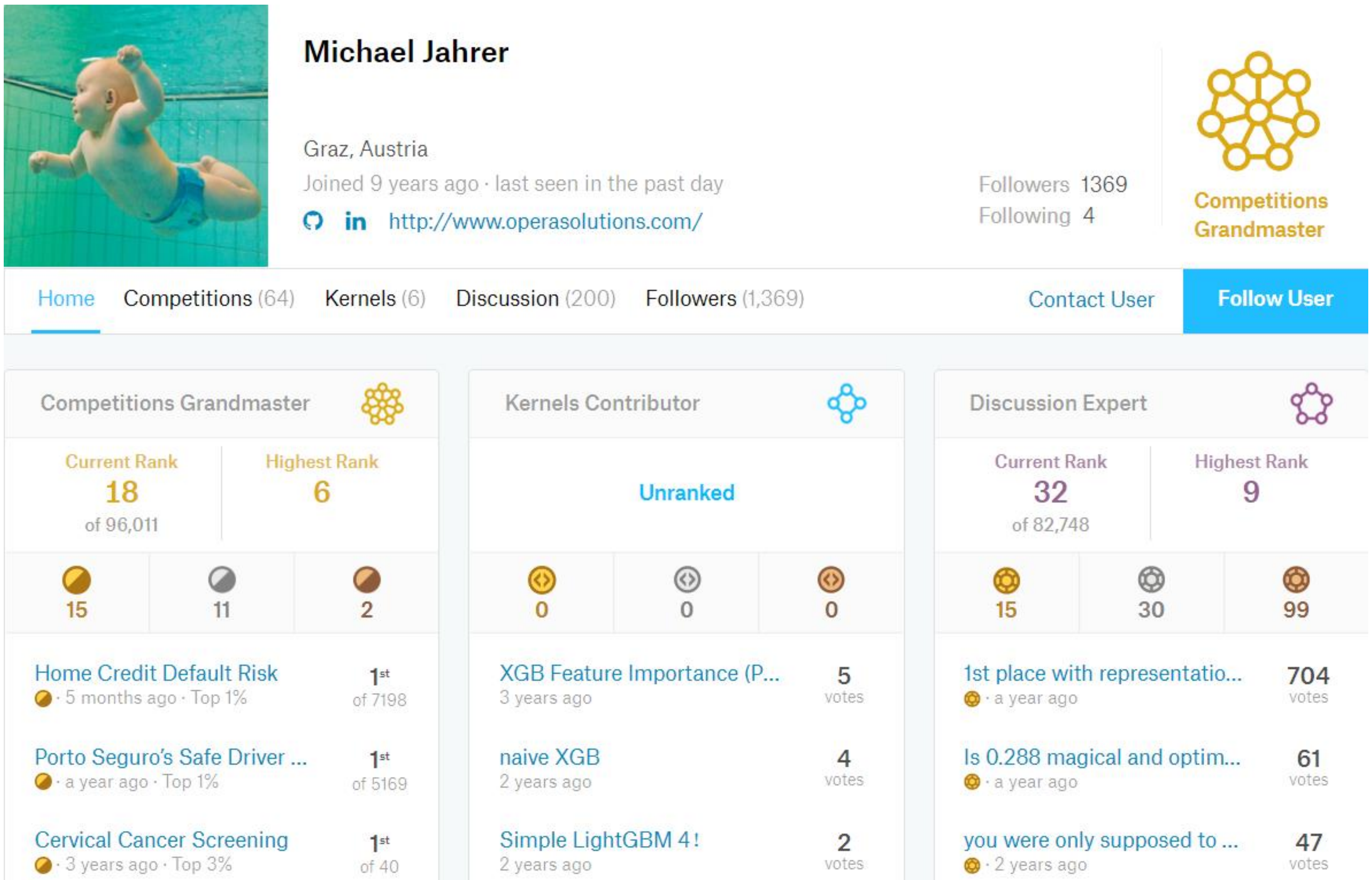
Discussions



Tutorials


Machine Learning
Machine learning is the hottest field in data science, and this track will get you started quickly.
DATA SCI. PROJECT 402
Overview
Free Course
4 hrs.
19 Lessons
Prerequisite Skills: Python
Prepares you for these Learn Courses: Deep Learning, Machine Learning Explainability
Instructor
Dan Becker
Data Scientist
Level 1
Level 2
1. How Models Work
The first step if you're new to machine learning
2. Explore Your Data
Load data and set up your environment for your hands-on project
3. Exercise: Explore Your Data
4. Your First Machine Learning Model
Building your first model. Hurray!
5. Exercise: Your First Machine Learning Model

Status Incentives



Kaggle CTRP Competitions

criteo.

Display Advertising Challenge
Predict click-through rates on display ads
Research · 4 years ago

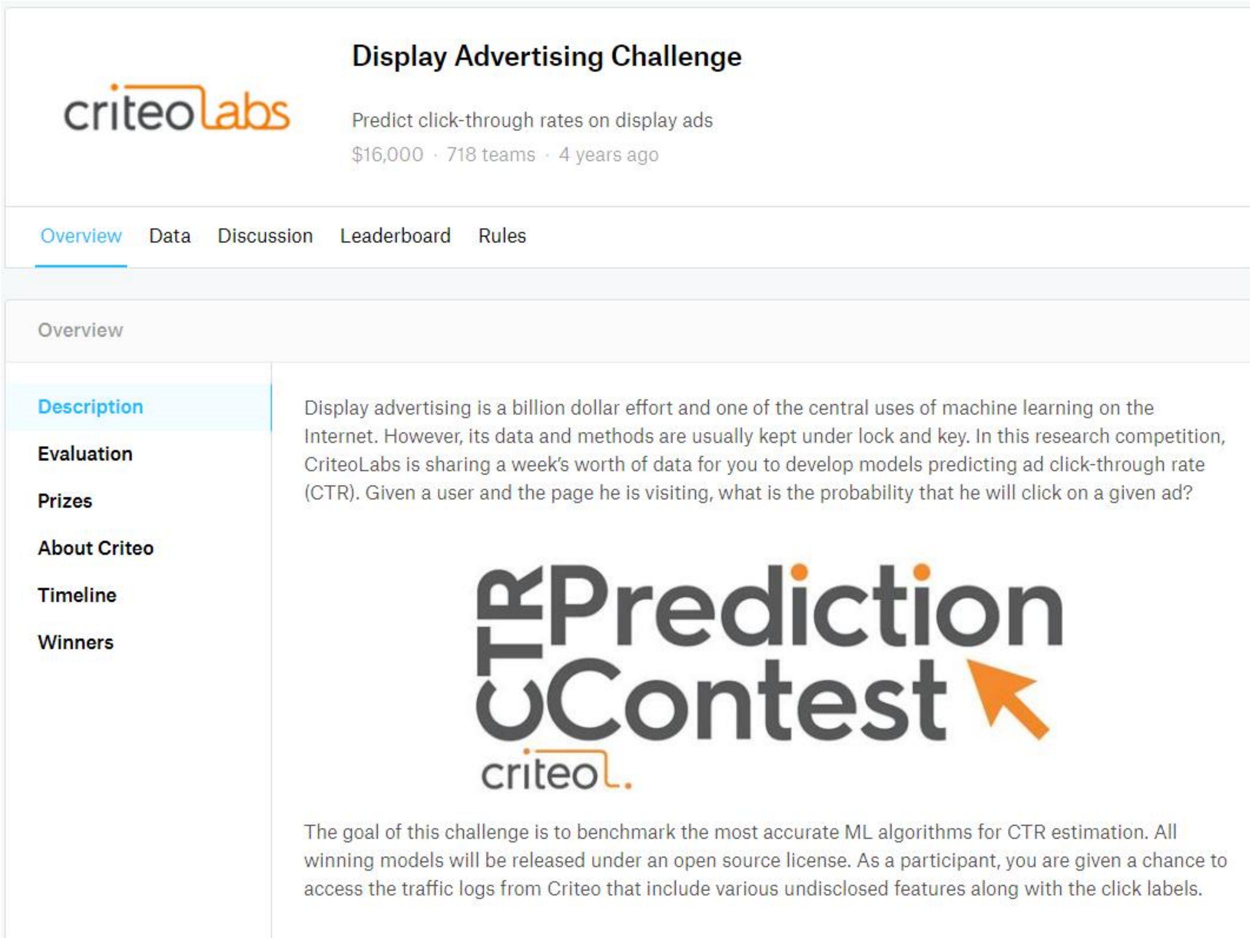
Avito Context Ad Clicks
Predict if context ads will earn a user's click
Featured · 4 years ago · marketing, tabular data, click prediction

Outbrain Click Prediction
Can you predict which recommended content each user will click?
Featured · 2 years ago · internet, tabular data, click prediction

Avazu

Click-Through Rate Prediction
Predict whether a mobile ad will be clicked
Featured · 4 years ago

Criteo Display Advertising Challenge



https://www.kaggle.com/c/criteo-display-ad-challenge

Criteo Display Advertising Challenge

- Criteo Display Advertising Challenge Data:
  - Features (inputs):
    - 13 numeric: unknown meanings, mostly counts, power laws evident
    - 26 categorical: unknown meanings, hashed (encoding without decoding), few dominant, many unique
  - Target (output): 0 / 1 (didn’t / did click through)

Mysterious Data

Label	I1	I2	···	I13	C1	C2	···	C26
1	3	20	···	2741	68fd1e64	80e26c9b	···	4cf72387
0	7	91	···	1157	3516f6e6	cfc86806	···	796a1a2e
0	12	73	···	1844	05db9164	38a947a1	···	5d93f8ab
					⋮			
?	9	62	···	1457	68fd1e64	cfc86806	···	cf59444f

#Train: $\approx$ 45M
#Test: $\approx$ 6M

Source: https://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf

Mysterious Data

Unknown Labels: meanings of numbers and categories not given

Label	I1	I2	⋯	I13	C1	C2	⋯	C26
1	3	20	⋯	2741	68fd1e64	80e26c9b	⋯	4cf72387
0	7	91	⋯	1157	3516f6e6	cfc86806	⋯	796a1a2e
0	12	73	⋯	1844	05db9164	38a947a1	⋯	5d93f8ab
					⋮			
?	9	62	⋯	1457	68fd1e64	cfc86806	⋯	cf59444f

#Train: $\approx$ 45M
#Test: $\approx$ 6M

Source: https://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf

Mysterious Data

Label	I1	I2	···	I13	C1	C2	···	C26
1	3	20	···	2741	68fd1e64	80e26c9b	···	4cf72387
0	7	91	···	1157	3516f6e6	cfc86806	···	796a1a2e
0	12	73	···	1844	05db9164	38a947a1	···	5d93f8ab
					⋮			
?	9	62	···	1457	68fd1e64	cfc86806	···	cf59444f

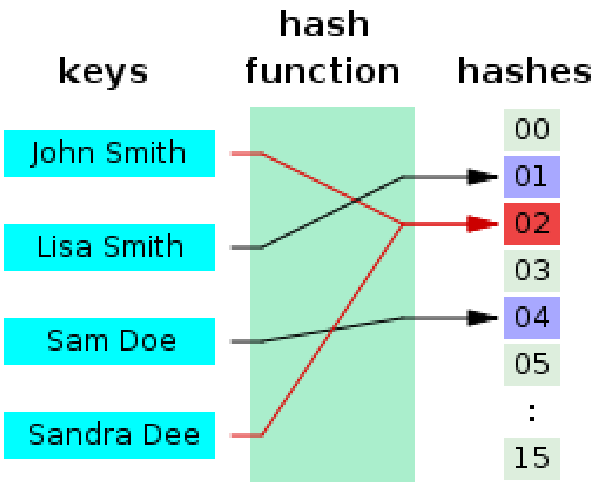
Categorical data is *hashed.*

#Train: $\approx$ 45M
#Test: $\approx$ 6M

Source: https://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf

Hashing

- A **hash** function takes some data and maps it to a number.
- Example: URL (web address)
  - Representation: string of characters
  - Character representation: a number (Unicode value)
  - Start with value 0.
  - Repeat for each character:
    - Multiply value by 31
    - Add next character Unicode to value
  - Don't worry about overflow – it's just a consistent "mathematical blender".



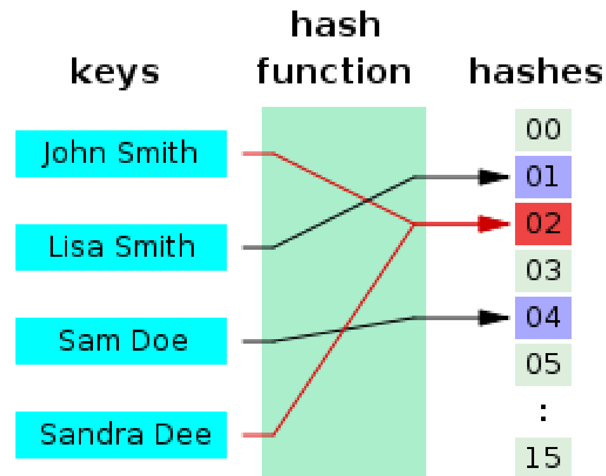
"Hi"

H = 72, i = 105

31 * 72 + 105 = 2337

Hash Function Characteristics

- **Mapping**: same input → same output
- **Uniform**: outputs have similar probabilities
  - **Collision**: two different inputs → same output
  - Collisions are allowable (inevitable if #data > #hash values) but not desirable.
- **Non-invertible**: can't get back input from output (e.g. cryptographic hashing, anonymization)



Missing Data

- The first 10 lines of the training data:

0	1	1	5	0	1382	4	15	2	181	1	2		2
0	2	0	44	1	102	8	2	2	4	1	1		4
0	2	0	1	14	767	89	4	2	245	1	3	3	45
0		893			4392		0	0	0		0		
0	3	-1		0	2	0	3	0	0	1	1		0
0		-1			12824		0	0	6		0		
0		1	2		3168		0	1	2		0		
1	1	4	2	0	0	0	1	0	0	1	1		0
0		44	4	8	19010	249	28	31	141		1		8
0		35		1	33737	21	1	2	3		1		1

68fd1e64	80e26c9b	fb936136	7b4723c4	25c83c98	7e0ccccf	de7995b8	1f89b562	a73ee510	a8cd5504	b2cb9c98	37c9c164	2824a5f6	1adce6ef	8ba8b39a	891b62e7	e5ba7672	f54016b9	21ddcdc9	b1252a9d	07b5194c		3a171ecb	c5c50484	e8b83407	9727dd16
68fd1e64	f0cf0024	6f67f7e5	41274cd7	25c83c98	fe6b92e5	922afcc0	0b153874	a73ee510	2b53e5fb	4f1b46f3	6.23E+11	d7020589	b28479f6	e6c5b5cd	c92f3b61	07c540c4	b04e4670	21ddcdc9	5840adea	60f6221e		3a171ecb	43f13e8b	e8b83407	731c3655
287e684f	0a519c5c	02cf9876	c18be181	25c83c98	7e0ccccf	c78204a1	0b153874	a73ee510	3b08e48b	5f5e6091	8fe001f4	aa655a2f	07d13a8f	6dc710ed	36103458	8efede7f	3412118d			e587c466	ad3062eb	3a171ecb	3b183c5c		
68fd1e64	2c16a946	a9a87e68	2e17d6f6	25c83c98	fe6b92e5	2e8a689b	0b153874	a73ee510	efea433b	e51ddf94	a30567ca	3516f6e6	07d13a8f	18231224	52b8680f	1e88c74f	74ef3502			6b3a5ca6		3a171ecb	9117a34a		
8cf07265	ae46a29d	c81688bb	f922efad	25c83c98	13718bbd	ad9fa255	0b153874	a73ee510	5282c137	e5d8af57	66a76a26	f06c53ac	1adce6ef	8ff4b403	01adbab4	1e88c74f	26b3c7a7			21c9516a		32c7478e	b34f3128		
05db9164	6c9c9cf3	2730ec9c	5400db8b	43b19349	6f6d9be8	53b5f978	0b153874	a73ee510	3b08e48b	91e8fc27	be45b877	9ff13f22	07d13a8f	06969a20	9bc7fff5	776ce399	92555263			242bb710	8ec974f4	be7c41b4	72c78f11		
439a44a4	ad4527a2	c02372d0	d34ebbaa	43b19349	fe6b92e5	4bc6ffea	0b153874	a73ee510	3b08e48b	a4609aab	14d63538	772a00d7	07d13a8f	f9d1382e	b00d3dc9	776ce399	cdfa8259			20062612		93bad2c0	1b256e61		
68fd1e64	2c16a946	503b9dbc	e4dbea90	f3474129	13718bbd	38eb9cf4	1f89b562	a73ee510	547c0ffe	bc8c9f21	60ab2f07	46f42a63	07d13a8f	18231224	e6b6bdc7	e5ba7672	74ef3502			5316a17f		32c7478e	9117a34a		
05db9164	d833535f	d032c263	c18be181	25c83c98	7e0ccccf	d5b6acf2	0b153874	a73ee510	2acdcf4e	086ac2d2	dfbb09fb	41a6ae00	b28479f6	e2502ec9	84898b2a	e5ba7672	42a2edb9			0014c32a		32c7478e	3b183c5c		
05db9164	510b40a5	d03e7c24	eb1fd928	25c83c98		52283d1c	0b153874	a73ee510	015ac893	e51ddf94	951fe4a9	3516f6e6	07d13a8f	2ae4121c	8ec71479	d4bb7bd8	70d0f5f9			0e63fca0		32c7478e	0e8fe315		

- Missing numeric and categorical features:

0	1	1	5	0	1382
0	2	0	44	1	102
0	2	0	1	14	767
0		893			4392
0	3	-1		0	2
0		-1			12824
0		1	2		3168
1	1	4	2	0	0
0		44	4	8	19010
0		35		1	33737

...

07b5194c		3a171ecb	c5c50484	e8b83407	9727dd16
60f6221e		3a171ecb	43f13e8b	e8b83407	731c3655
e587c466	ad3062eb	3a171ecb	3b183c5c		
6b3a5ca6		3a171ecb	9117a34a		
21c9516a		32c7478e	b34f3128		
242bb710	8ec974f4	be7c41b4	72c78f11		
20062612		93bad2c0	1b256e61		
5316a17f		32c7478e	9117a34a		
0014c32a		32c7478e	3b183c5c		
0e63fca0		32c7478e	0e8fe315		

Missing Data: Imputation

- One approach to dealing with missing data is to *impute* values, i.e. replace with reasonable values inferred from surrounding data.
- In other words, create predictors for each value based on other known/unknown values.
- Cons:
  - Difficult to validate.
  - In Criteo data, missing values are correlated.
  - So … we're writing predictors to impute data we're learning predictors from?

[General Introduction to Handling Missing Data]

Missing Data: Embrace the "Unknown"

- Retain "unknown" as data that contains valuable information.
- Does the lack of CTR context data caused by incognito browsing mode provide information on what a person is more likely to click?
- Categorical data: For each category C# with missing data, create a new category value "C#:unknown".

Missing Data: Embrace the "Unknown"

- Numeric data:
  - Create an additional feature that indicates whether the value for a feature is (un)known.
    - Additionally could impute mean, median, etc., for unknown value.
  - Convert to categorical and add "C#:unknown" category…

Numeric to Categorical: Binning

- Histogram-based
  - Uniform ranges: (+) simple (-) uneven distribution, poor for non-uniform data
  - Uniform ranges on transformation (e.g. log): (+) somewhat simple (-) transformation requires understanding of data distribution
- Quantiles
  - E.g. quartiles = 4-quantiles, quintiles = 5-quantiles
  - (+) simple, even distribution by definition, (-) preponderance of few values → duplicate bins (eliminate)

Categorical to Numeric: One-Hot Encoding

- For each categorical input variable:
  - For each possible category value, create a new numeric input variable that can be assigned numeric value 1 (“belongs to this category”) or 0 (“does not belong to this category).
  - For each input, replace the categorical value variable with these new numeric inputs.

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

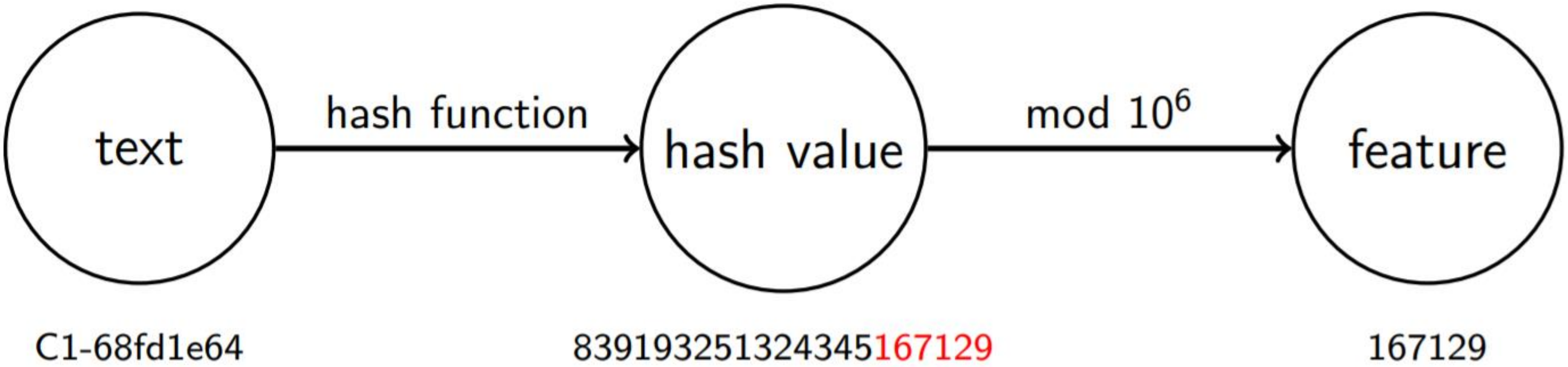
https://www.kaggle.com/dansbecker/using-categorical-data-with-one-hot-encoding

Categorical to Numeric: Hashing

- When there are a large number of categories, one-hot encoding isn't practical.
  - E.g. Criteo data category C3 in its small sample of CTR data had 10,131,226 distinct categorical values.
  - One approach (e.g. for power law data): one-hot encode few dominant values plus "rare" category.
  - Hashing trick:
    - Append category name and unusual character before category value and *hash* to an integer.
    - Create a one-hot-like category for each integer.

Hashing Trick Example

- From https://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf:

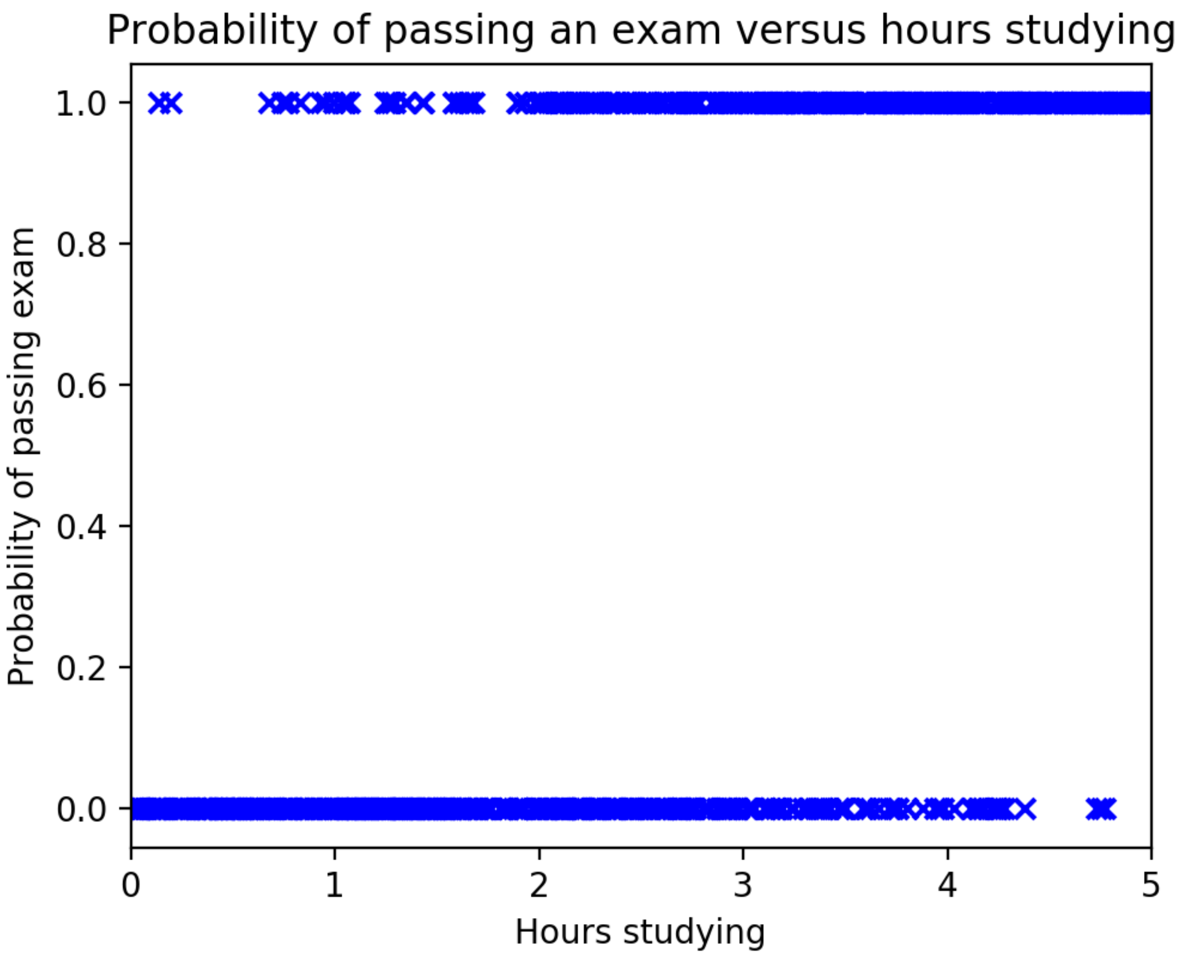


- Fundamental tradeoff: greater/lesser number hashed features results in …
  - … more/less expensive computation
  - … less/more frequent hash collisions (i.e. unlike categories treated as like)

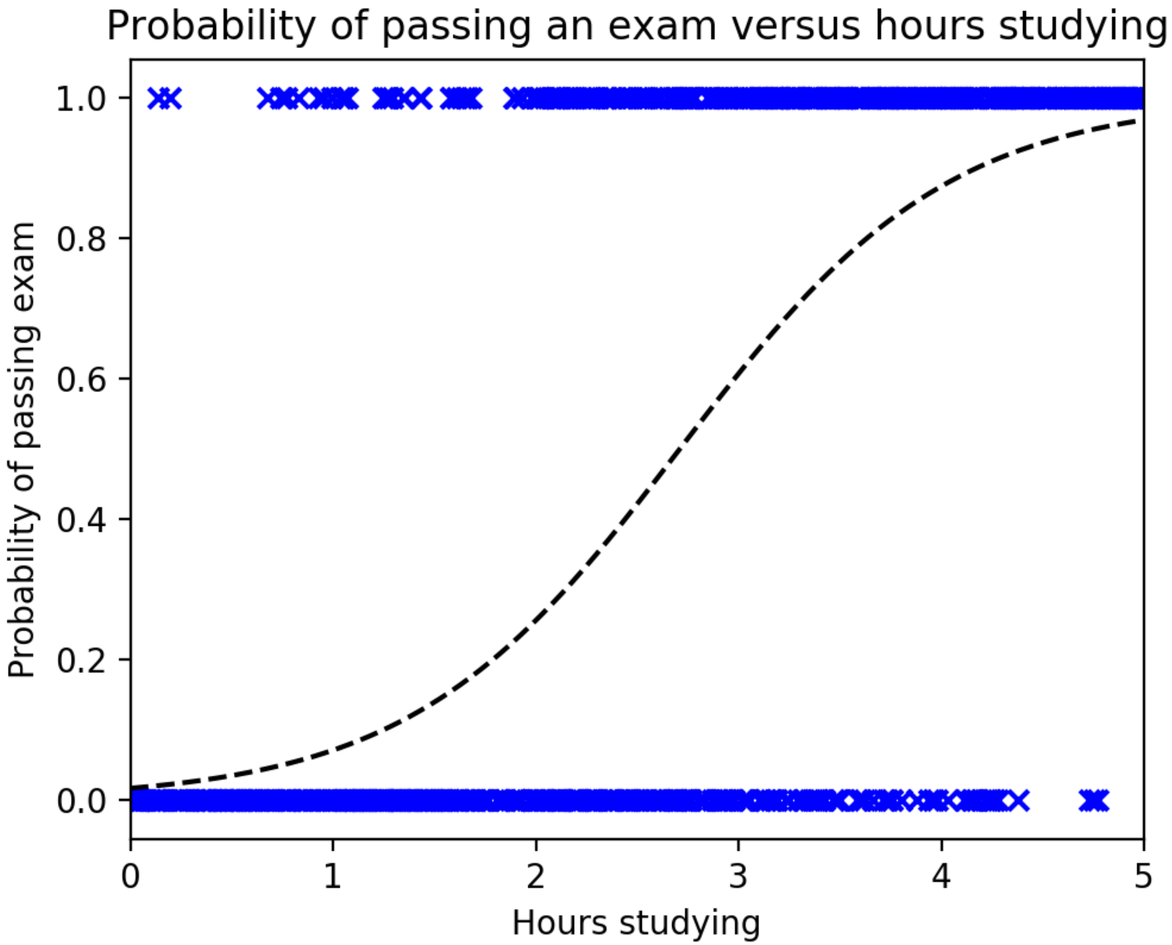
Logistic Regression Motivation

- Logistic regression is perhaps the simplest technique to beat the Criteo benchmark, scoring ~42nd percentile on leaderboard:
  - https://www.kaggle.com/c/criteo-display-ad-challenge/discussion/10322
  - 100 lines of Python, 200MB RAM, 30 min. training
  - Also: logistic regression recommended for CTRP by researchers of Criteo, Microsoft, LinkedIn, Google, and Facebook for practical, scalable implementation.

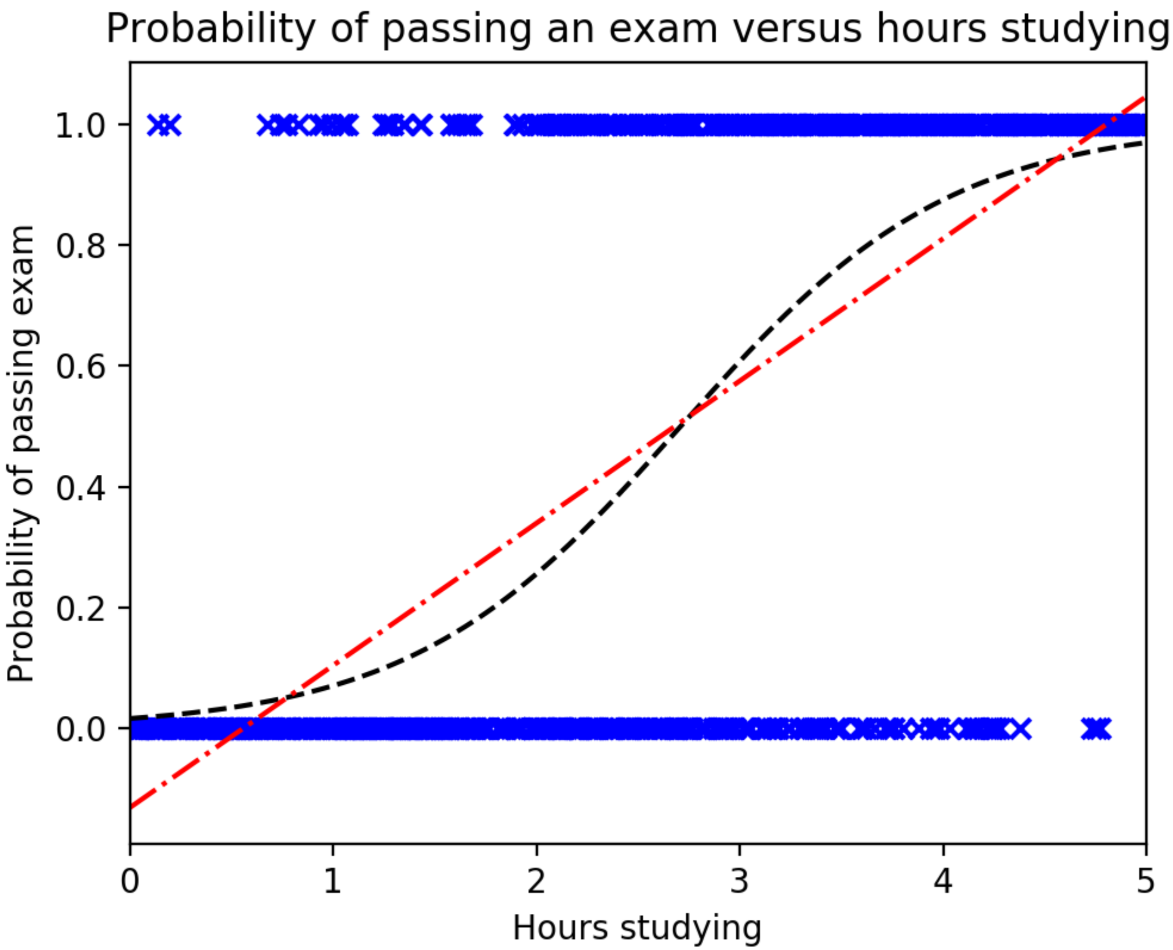
Example: Passing vs. Studying



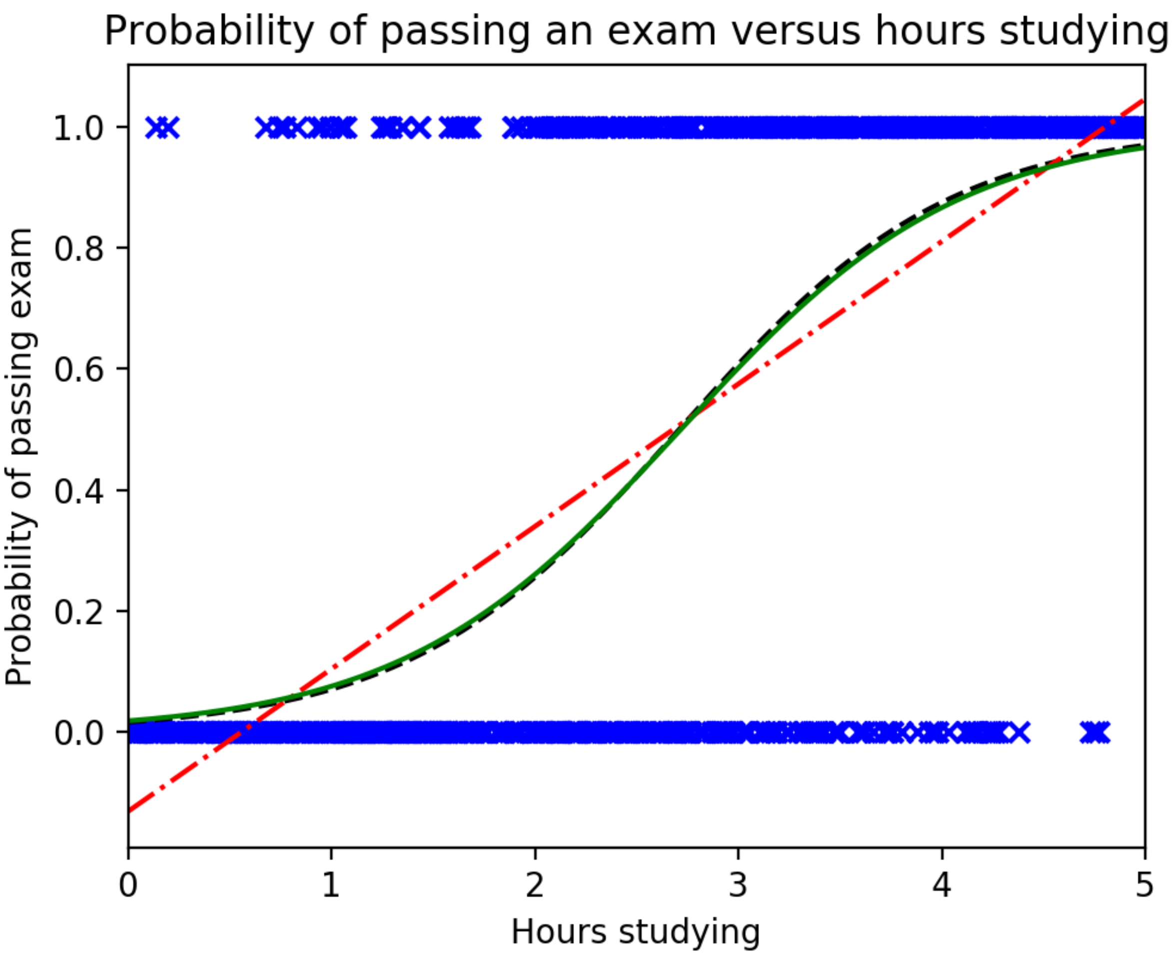
Unknown Logistic Model



Misapplication of Linear Regression



Logistic Regression Recovering Model



Logistic Regression with Stochastic Gradient Descent

- Output: $p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$
- Initially: $\beta_0 = \beta_1 = 0$
- Repeat:
  – For each input x,
    - Adjust intercept β_0 by learning rate * error * p′(x)
    - Adjust coefficient β_1 by learning rate * error * p′(x) * x
- Note:
  – Error = y - p(x)
  – p′(x) = p(x) * (1 – p(x))   (the slope of p at x)
  – This is neural network learning with a single logistic neuron with bias input of 1

Logistic Regression Takeaways

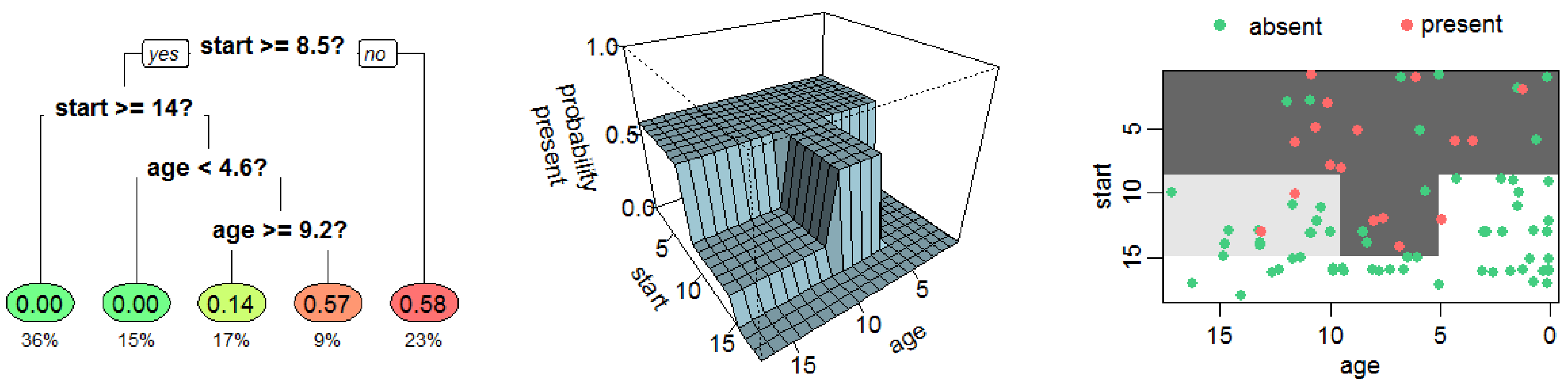
- The previous algorithm doesn’t require complex software. (12 lines raw Python code)
- Easy and effective for CTR prediction.
- Key to good performance: *skillful feature engineering of numeric features*
- Foreshadowing: Since logistic regression is a simple special case of neural network learning, I would expect deep learning tools to make future inroads here.

Maximizing Info with Decisions

- Number Guessing Game example:
  - “I’m thinking of a number from 1 to 100.”
  - Number guess → “Higher.” / “Lower.” / “Correct.”
  - What is the best strategy and why?
- Good play maximizes information according to some measure (e.g. entropy).

Decision Trees for Regression (Regression Trees)

- Numeric features (missing values permitted)
- At each node in the tree, a branch is decided on according to a features value (or lack thereof)



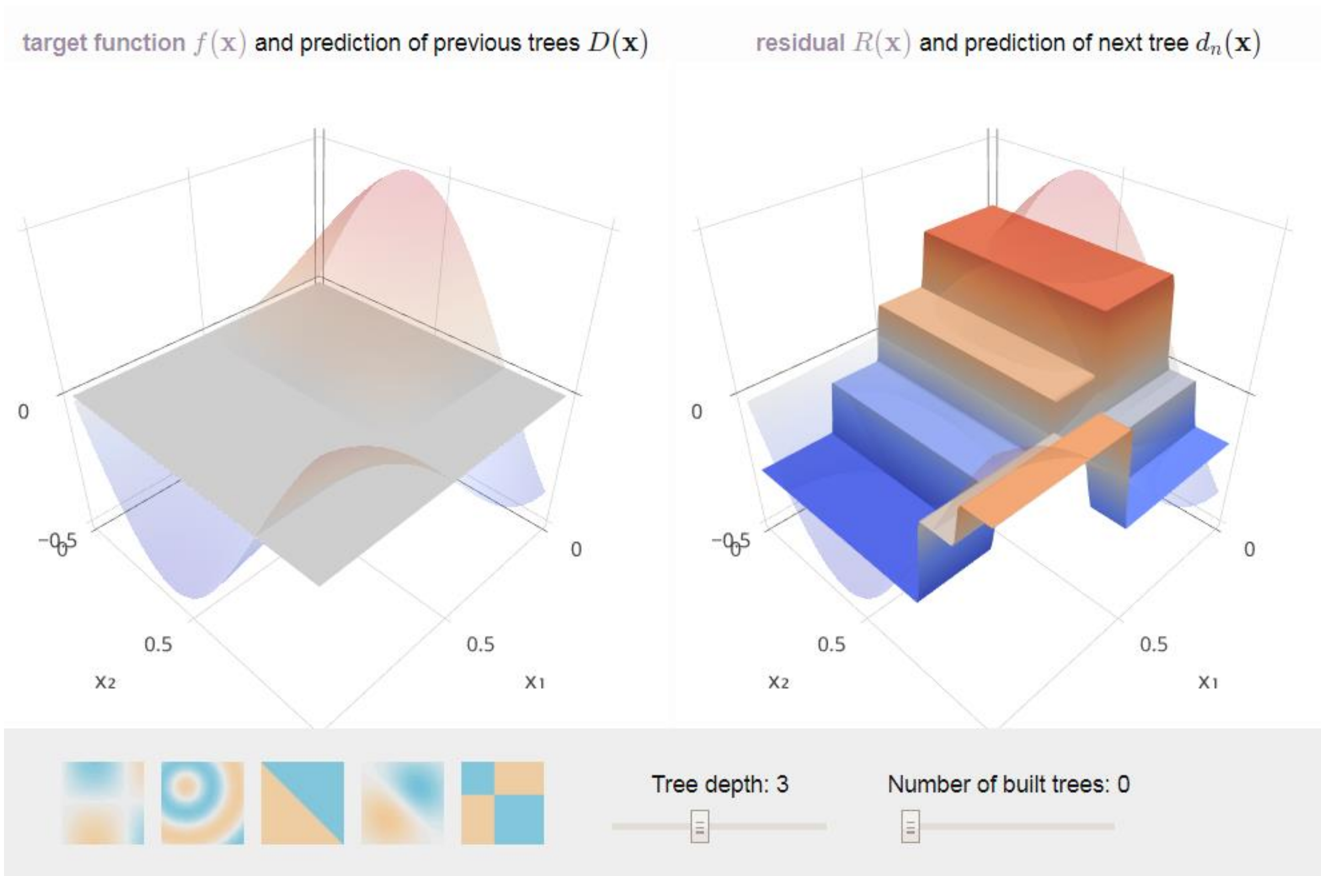
A regression tree estimating the probability of kyphosis (hunchback) after surgery, given the age of the patient and the vertebra at which surgery was started. Source: https://en.wikipedia.org/wiki/Decision_tree_learning

The Power of Weak Classifiers

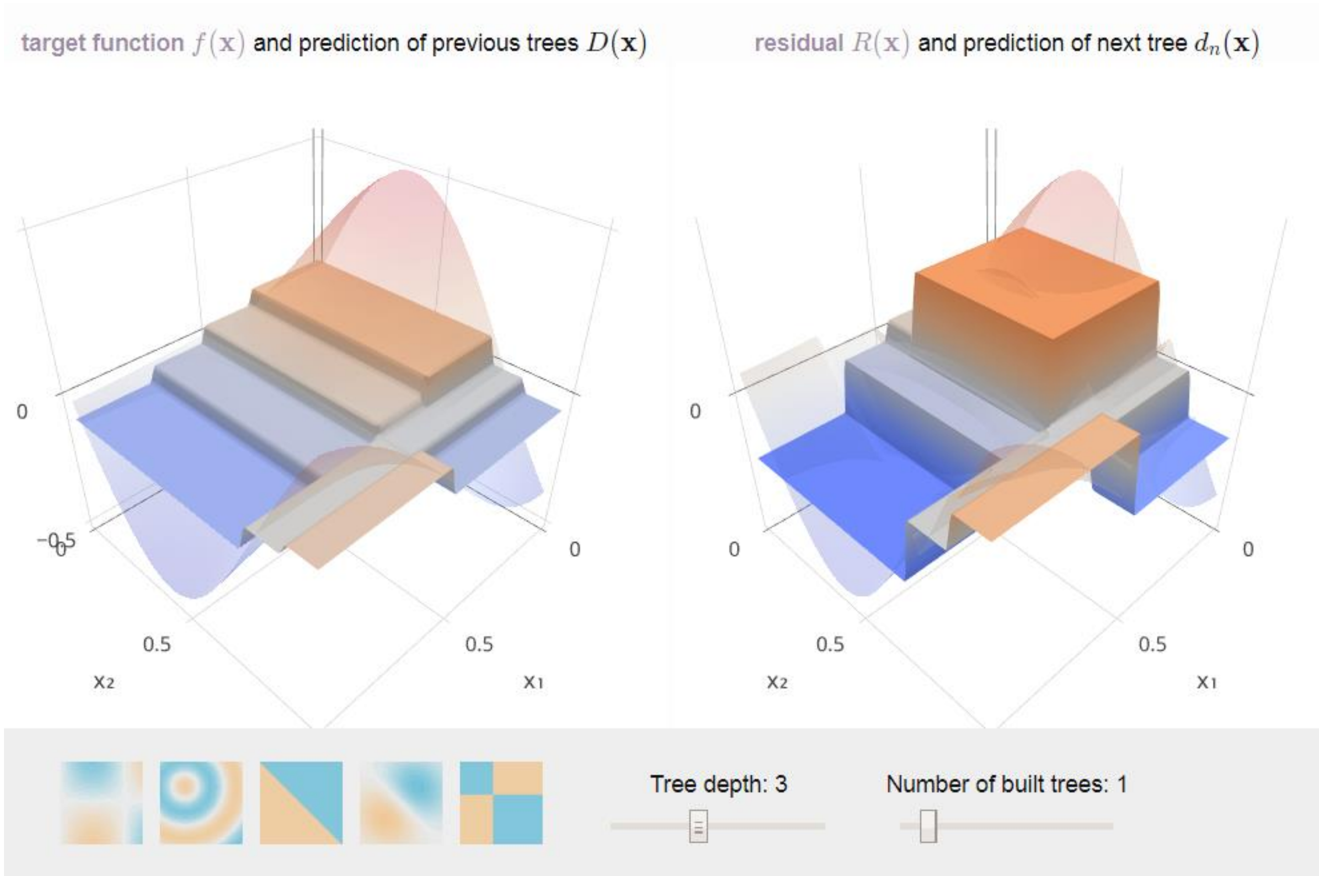
- Caveats:
  - Too deep: Single instance leafs → overfitting; similar to nearest neighbor (n=1)
  - Too shallow: Large hyperrectangular sets → underfitting; poor, blocky generalization
- Many weak classifiers working together can achieve good fit and generalization.
  - "Plans fail for lack of counsel, but with many advisers they succeed." – Proverbs 15:22
- Ensemble methods: **boosting**, bagging, stacking

Gradient Boosting of Regression Trees

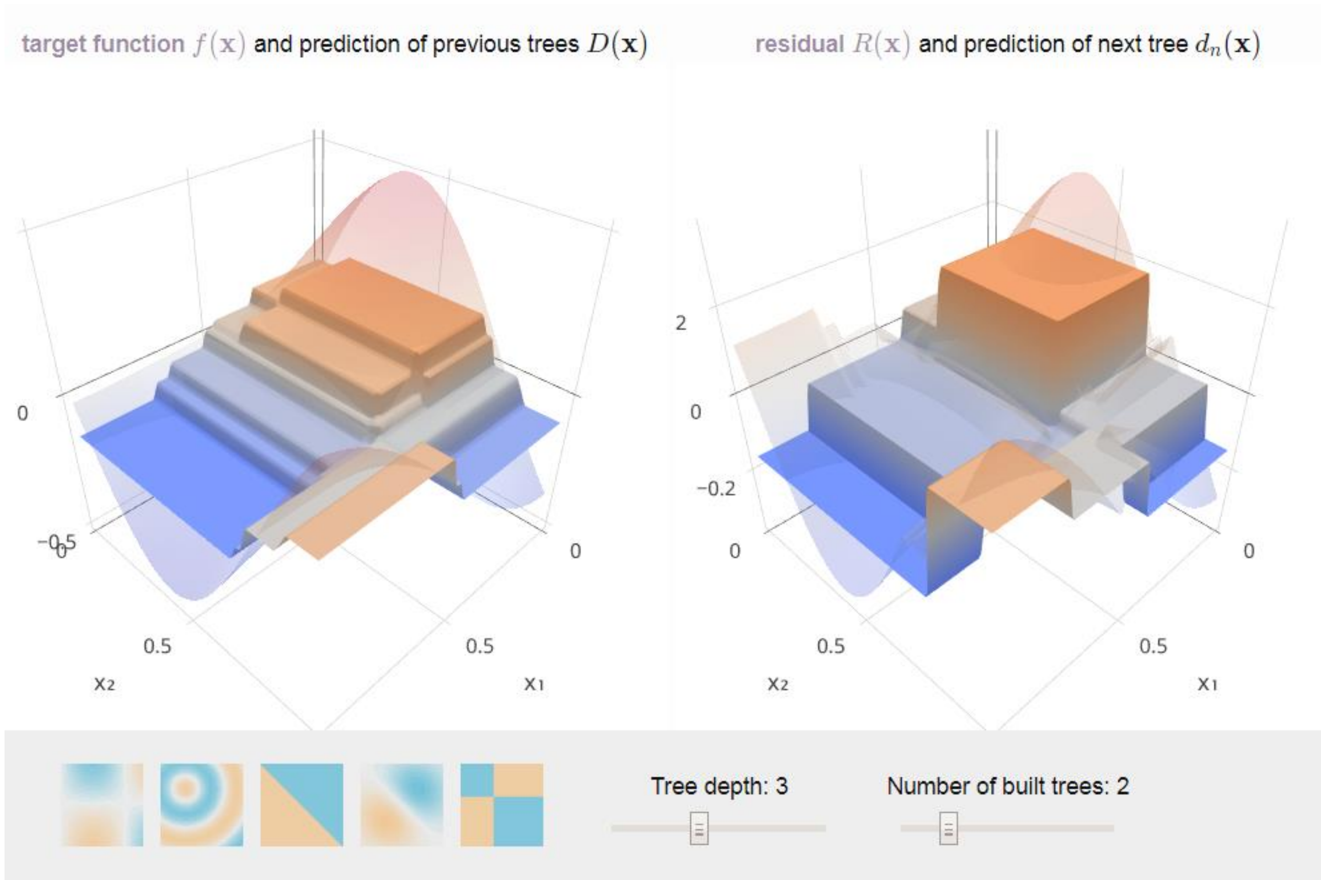
- Basic boosting idea:
  - Initially, make a 0 or constant prediction.
  - Repeat:
    - Compute prediction errors from the weighted sum of our weak-learner predictions.
    - Fit a new weak-learner to predict these errors and *add* its weighted error-prediction to our model.
- Alex Rogozhnikov's beautiful demonstration: https://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html



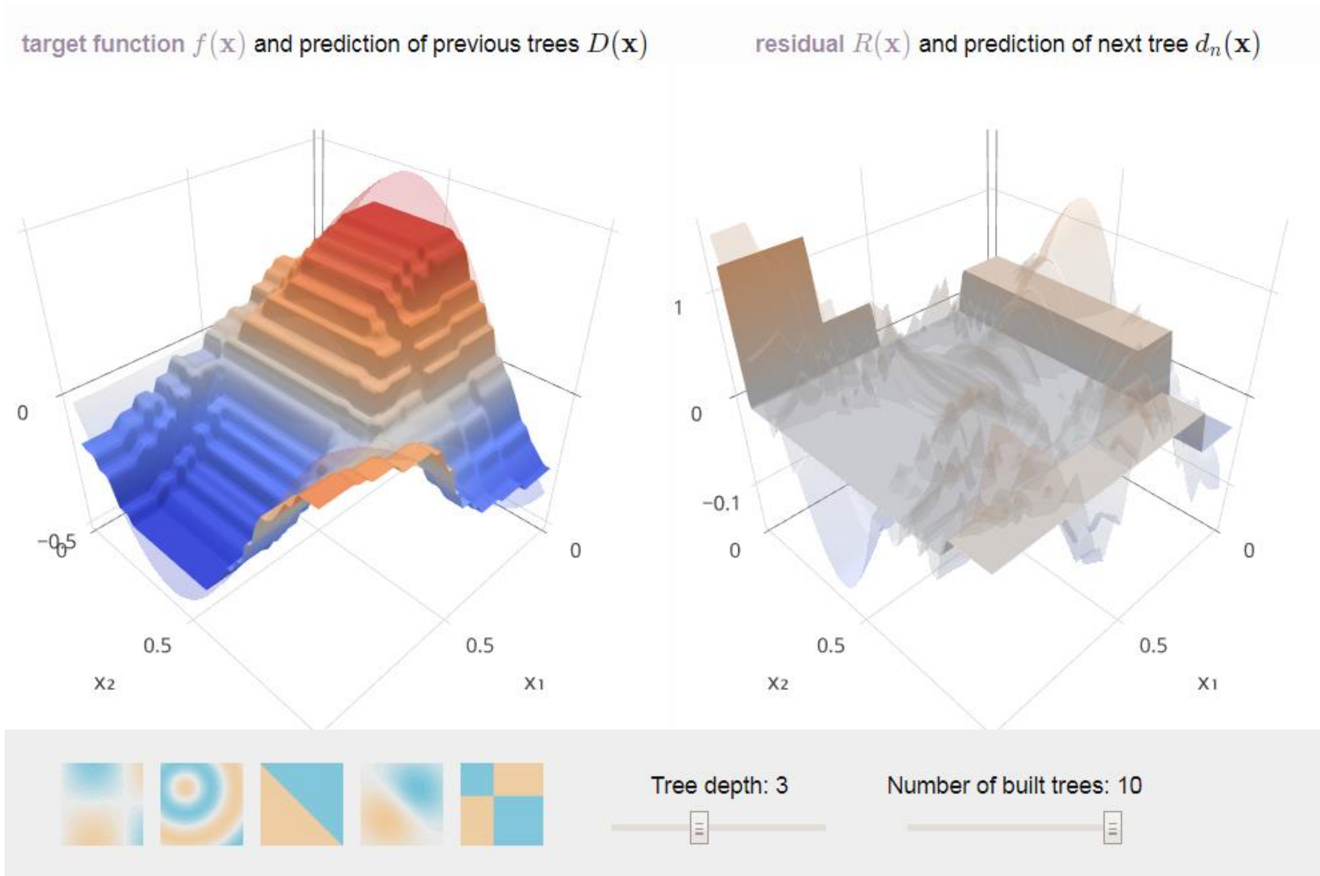
target function $f(\mathbf{x})$ and prediction of previous trees $D(\mathbf{x})$
residual $R(\mathbf{x})$ and prediction of next tree $d_n(\mathbf{x})$
0
−0.5
0
0.5
0.5
x2
x1
Tree depth: 3
Number of built trees: 0



target function $f(\mathbf{x})$ and prediction of previous trees $D(\mathbf{x})$
residual $R(\mathbf{x})$ and prediction of next tree $d_n(\mathbf{x})$
0
−0.5
0
0.5
0.5
x2
x1
0
0
0
0.5
0.5
x2
x1
Tree depth: 3
Number of built trees: 1



target function $f(\mathbf{x})$ and prediction of previous trees $D(\mathbf{x})$
residual $R(\mathbf{x})$ and prediction of next tree $d_n(\mathbf{x})$
0
−0.5
0.5
x2
0.5
x1
0
2
0
−0.2
0
0.5
x2
0.5
x1
0
Tree depth: 3
Number of built trees: 2



target function $f(\mathbf{x})$ and prediction of previous trees $D(\mathbf{x})$
residual $R(\mathbf{x})$ and prediction of next tree $d_n(\mathbf{x})$
0
−0.5
0.5
x2
0
0.5
x1
1
0
−0.1
0
0.5
x2
0
0.5
x1
Tree depth: 3
Number of built trees: 10

XGBoost

- *"Among the 29 challenge winning solutions published at Kaggle's blog during 2015, 17 solutions [~59%] used XGBoost. Among these solutions, eight [~28%] solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles."* - Tianqi Chen, Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System"

XGBoost Features

- **XGBoost** is a specific implementation of **gradient boosted decision trees** that:
  - Supports a command-line interface, C++, Python (scikit-learn), R (caret), Java/JVM languages + Hadoop platform
  - A range of computing environments with parallelization, distributed computing, etc.
  - Handles sparse, missing data
  - Is *fast* and *high-performance* across diverse problem domains
  - https://xgboost.readthedocs.io

Field-aware Factorization Machines (FFMs)

- Top-performing technique in 3 of 4 Kaggle CTR prediction competitions plus RecSys 2015:
  - Criteo: https://www.kaggle.com/c/criteo-display-ad-challenge
  - Avazu: https://www.kaggle.com/c/avazu-ctr-prediction
  - Outbrain: https://www.kaggle.com/c/outbrain-click-prediction
  - RecSys 2015: http://dl.acm.org/citation.cfm?id=2813511&dl=ACM&coll=DL&CFID=941880276&CFTOKEN=60022934

What's Different? Field-Aware Latent Factors

- Latent factor
  - learned weight; tuned variable
  - How much an input contributes to an output
- Many techniques learn "latent factors":
  - Linear regression: one per feature + 1

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

  - Logistic regression: one per feature + 1

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}$$

What's Different? Field-Aware Latent Factors (cont.)

- Many techniques learn "latent factors":
  - Degree-2 polynomial regression: one per pair of features

$$\phi_{\mathrm{Poly2}}(\boldsymbol{w}, \boldsymbol{x}) = \sum_{j_1=1}^{n} \sum_{j_2=j_1+1}^{n} w_{h(j_1, j_2)} x_{j_1} x_{j_2}$$

  - Factorization machine (FM):
    - *k* per feature
    - "latent factor vector", a.k.a. "latent vector"

$$\phi_{\mathrm{FM}}(\boldsymbol{w}, \boldsymbol{x}) = \sum_{j_1=1}^{n} \sum_{j_2=j_1+1}^{n} (\boldsymbol{w}_{j_1} \cdot \boldsymbol{w}_{j_2}) x_{j_1} x_{j_2}$$

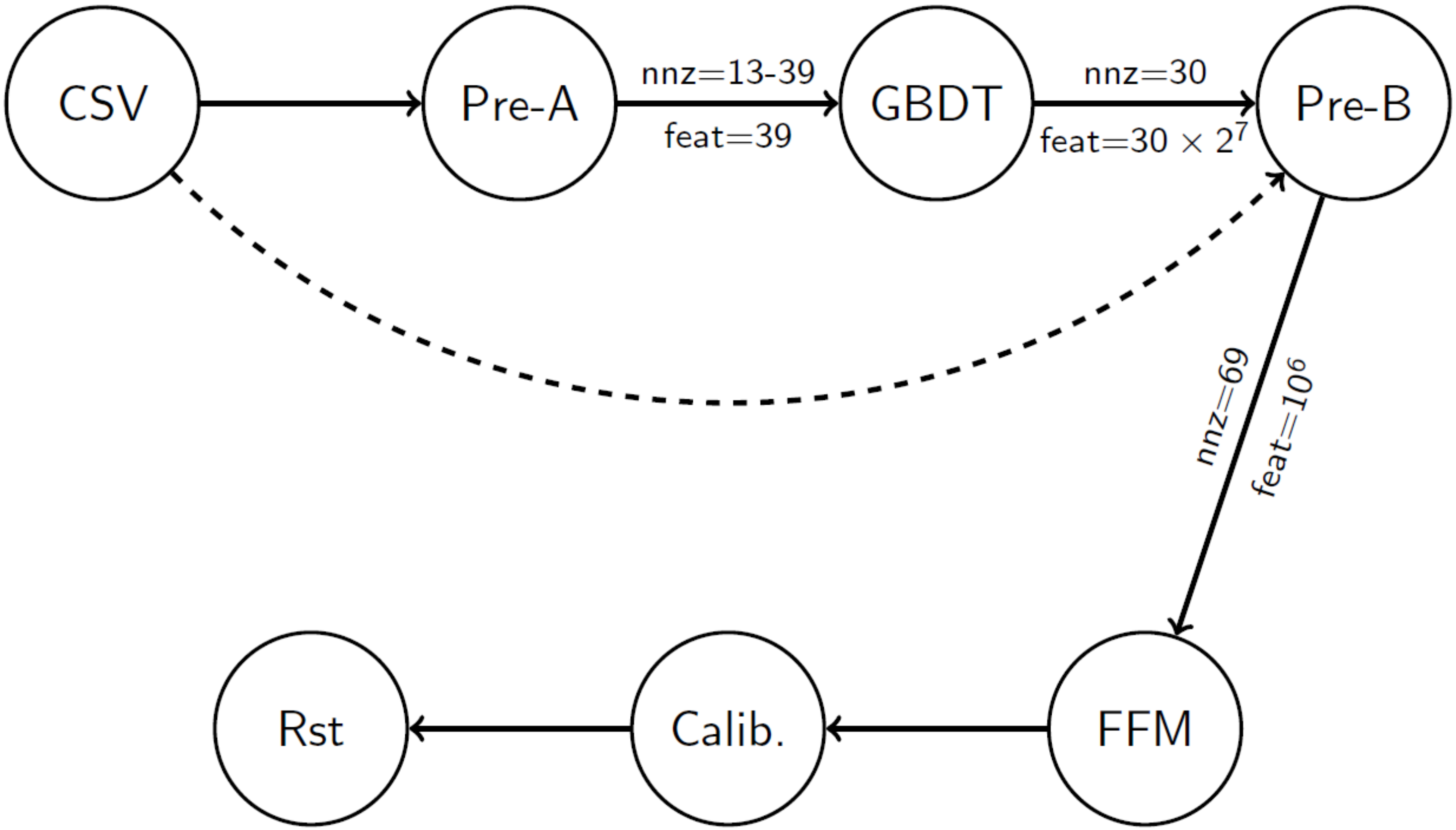
What's Different? Field-Aware Latent Factors (cont.)

- Many techniques learn "latent factors":
  - Field-aware Factorization machine (FFM):
    - *k* per feature *and field* pair
    - Field:
      - Features are often one-hot encoded
      - Continuous block of binary features often represent different values for the same underlying "field"
      - E.g. Field: "OS", features: "Windows", "MacOS", "Android"
      - libffm: FFM library (https://github.com/guestwalk/libffm)

$$\phi_{\mathrm{FFM}}(\boldsymbol{w}, \boldsymbol{x}) = \sum_{j_1=1}^{n} \sum_{j_2=j_1+1}^{n} (\boldsymbol{w}_{j_1,f_2} \cdot \boldsymbol{w}_{j_2,f_1}) x_{j_1} x_{j_2}$$

Winning Team Process

- From https://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf:



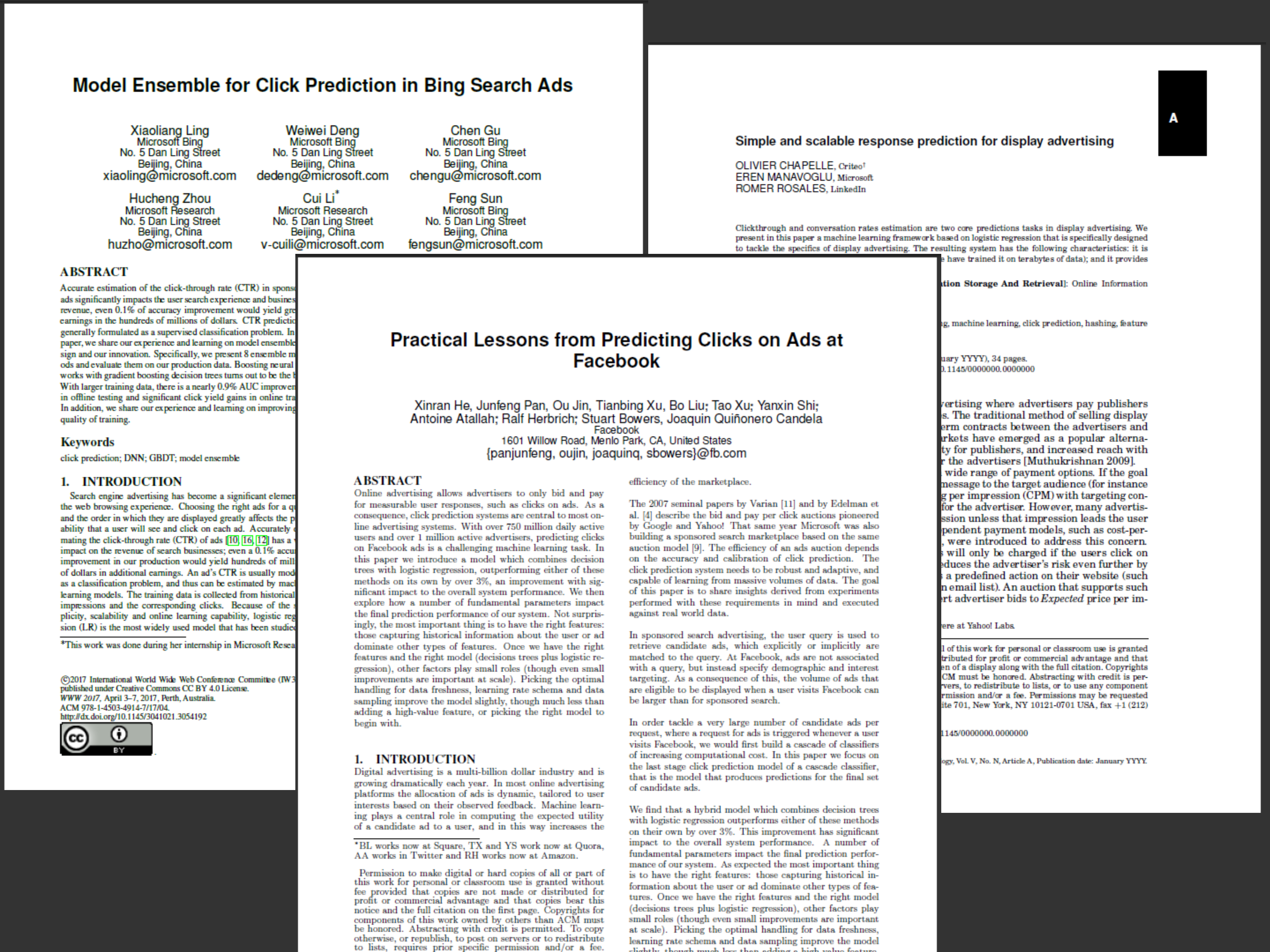
"nnz" means the number of non-zero elements of each impression; "feat" represents the size of feature space.

Is the Extra Engineering Worth it?

- Kaggle Criteo leaderboard based on **logarithmic loss** (a.k.a. logloss)
  - 0.69315 → 50% correct in binary classification (random guessing baseline)
- Simple logistic regression with hashing trick:
  - 0.46881 (private leaderboard) ~62.6% correct
- FFM with feature engineering using GBDT:
  - 0.44463 (private leaderboard) ~64.1% correct

Computational Cost

- ~1.5% increase in correct prediction, but greater computational complexity:
  - Logistic regression: n factors to learn and **re**learn in **dynamic** context
  - FFM: kn^2 factors to learn and **re**learn



Model Ensemble for Click Prediction in Bing Search Ads

Xiaoliang Ling, Microsoft Bing, No. 5 Dan Ling Street, Beijing, China, xiaoling@microsoft.com
Weiwei Deng, Microsoft Bing, No. 5 Dan Ling Street, Beijing, China, dedeng@microsoft.com
Chen Gu, Microsoft Bing, No. 5 Dan Ling Street, Beijing, China, chengu@microsoft.com
Hucheng Zhou, Microsoft Research, No. 5 Dan Ling Street, Beijing, China, huzho@microsoft.com
Cui Li*, Microsoft Research, No. 5 Dan Ling Street, Beijing, China, v-cuili@microsoft.com
Feng Sun, Microsoft Bing, No. 5 Dan Ling Street, Beijing, China, fengsun@microsoft.com

ABSTRACT

Accurate estimation of the click-through rate (CTR) in spons
ads significantly impacts the user search experience and busines
revenue, even 0.1% of accuracy improvement would yield gre
earnings in the hundreds of millions of dollars. CTR predictio
generally formulated as a supervised classification problem. In
paper, we share our experience and learning on model ensemble
sign and our innovation. Specifically, we present 8 ensemble m
ods and evaluate them on our production data. Boosting neural
works with gradient boosting decision trees turns out to be the b
With larger training data, there is a nearly 0.9% AUC improve
in offline testing and significant click yield gains in online tra
In addition, we share our experience and learning on improving
quality of training.

Keywords

click prediction; DNN; GBDT; model ensemble

1. INTRODUCTION

Search engine advertising has become a significant elemen
the web browsing experience. Choosing the right ads for a q
and the order in which they are displayed greatly affects the p
ability that a user will see and click on each ad. Accurately
mating the click-through rate (CTR) of ads [10, 16, 12] has a
impact on the revenue of search businesses; even a 0.1% accu
improvement in our production would yield hundreds of mill
of dollars in additional earnings. An ad's CTR is usually mod
as a classification problem, and thus can be estimated by mac
learning models. The training data is collected from historical
impressions and the corresponding clicks. Because of the
plicity, scalability and online learning capability, logistic reg
sion (LR) is the most widely used model that has been studied

*This work was done during her internship in Microsoft Resea

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4914-7/17/04.
http://dx.doi.org/10.1145/3041021.3054192

Simple and scalable response prediction for display advertising

OLIVIER CHAPELLE, Criteo†
EREN MANAVOGLU, Microsoft
ROMER ROSALES, LinkedIn

Clickthrough and conversation rates estimation are two core predictions tasks in display advertising. We present in this paper a machine learning framework based on logistic regression that is specifically designed to tackle the specifics of display advertising. The resulting system has the following characteristics: it is
have trained it on terabytes of data); and it provides

tion Storage And Retrieval]: Online Information
g, machine learning, click prediction, hashing, feature
uary YYYY), 34 pages.
0.1145/0000000.0000000

ertising where advertisers pay publishers
s. The traditional method of selling display
erm contracts between the advertisers and
rkets have emerged as a popular alterna-
ty for publishers, and increased reach with
r the advertisers [Muthukrishnan 2009].
wide range of payment options. If the goal
message to the target audience (for instance
g per impression (CPM) with targeting con-
for the advertiser. However, many advertis-
ssion unless that impression leads the user
pendent payment models, such as cost-per-
, were introduced to address this concern.
will only be charged if the users click on
duces the advertiser's risk even further by
a predefined action on their website (such
n email list). An auction that supports such
rt advertiser bids to *Expected* price per im-

ere at Yahoo! Labs.

1145/0000000.0000000
ogy, Vol. V, No. N, Article A, Publication date: January YYYY.

Practical Lessons from Predicting Clicks on Ads at Facebook

Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu; Tao Xu; Yanxin Shi; Antoine Atallah; Ralf Herbrich; Stuart Bowers, Joaquin Quiñonero Candela
Facebook
1601 Willow Road, Menlo Park, CA, United States
{panjunfeng, oujin, joaquinq, sbowers}@fb.com

ABSTRACT

Online advertising allows advertisers to only bid and pay for measurable user responses, such as clicks on ads. As a consequence, click prediction systems are central to most online advertising systems. With over 750 million daily active users and over 1 million active advertisers, predicting clicks on Facebook ads is a challenging machine learning task. In this paper we introduce a model which combines decision trees with logistic regression, outperforming either of these methods on its own by over 3%, an improvement with significant impact to the overall system performance. We then explore how a number of fundamental parameters impact the final prediction performance of our system. Not surprisingly, the most important thing is to have the right features: those capturing historical information about the user or ad dominate other types of features. Once we have the right features and the right model (decisions trees plus logistic regression), other factors play small roles (though even small improvements are important at scale). Picking the optimal handling for data freshness, learning rate schema and data sampling improve the model slightly, though much less than adding a high-value feature, or picking the right model to begin with.

1. INTRODUCTION

Digital advertising is a multi-billion dollar industry and is growing dramatically each year. In most online advertising platforms the allocation of ads is dynamic, tailored to user interests based on their observed feedback. Machine learning plays a central role in computing the expected utility of a candidate ad to a user, and in this way increases the efficiency of the marketplace.

*BL works now at Square, TX and YS work now at Quora, AA works in Twitter and RH works now at Amazon.

The 2007 seminal papers by Varian [11] and by Edelman et al. [4] describe the bid and pay per click auctions pioneered by Google and Yahoo! That same year Microsoft was also building a sponsored search marketplace based on the same auction model [9]. The efficiency of an ads auction depends on the accuracy and calibration of click prediction. The click prediction system needs to be robust and adaptive, and capable of learning from massive volumes of data. The goal of this paper is to share insights derived from experiments performed with these requirements in mind and executed against real world data.

In sponsored search advertising, the user query is used to retrieve candidate ads, which explicitly or implicitly are matched to the query. At Facebook, ads are not associated with a query, but instead specify demographic and interest targeting. As a consequence of this, the volume of ads that are eligible to be displayed when a user visits Facebook can be larger than for sponsored search.

In order tackle a very large number of candidate ads per request, where a request for ads is triggered whenever a user visits Facebook, we would first build a cascade of classifiers of increasing computational cost. In this paper we focus on the last stage click prediction model of a cascade classifier, that is the model that produces predictions for the final set of candidate ads.

We find that a hybrid model which combines decision trees with logistic regression outperforms either of these methods on their own by over 3%. This improvement has significant impact to the overall system performance. A number of fundamental parameters impact the final prediction performance of our system. As expected the most important thing is to have the right features: those capturing historical information about the user or ad dominate other types of features. Once we have the right features and the right model (decisions trees plus logistic regression), other factors play small roles (though even small improvements are important at scale). Picking the optimal handling for data freshness, learning rate schema and data sampling improve the model

Published Research from the Trenches

- Initial efforts focused on **logistic regression**
- Most big production systems reportedly kept it simple in the final stage of prediction:
  - Google (2013): prob. feature inclusion + Bloom filters → **logistic regression**
  - Facebook (2014): boosted decision trees → **logistic regression**
  - Yahoo (2014): hashing trick → **logistic regression**
- However…

Towards Neural Network Prediction

- More recently, Microsoft (2017) research
  - reports "factorization machines (FMs), gradient boosting decision trees (GBDTs) and deep neural networks (DNNs) have also been evaluated and gradually adopted in industry."
  - recommends boosting **neural networks** with gradient boosting decision trees

Perspective

- The last sigmoid layer of a neural network (deep or otherwise) for binary classification is logistic regression.
- Previous layers of a deep neural network learn an internal representation of inputs, i.e. perform automatic *feature engineering*.
- Thus, most efforts to engineer successful, modern CTR prediction systems focus on layered feature engineering using:
  - Hashing tricks
  - Features engineered with GBDTs, FFMs, and deep neural networks (DNNs), or a layered/ensembled combination thereof.
- Future: Additional automated feature representation learning with deep neural networks

CTRP Conclusions

- To get prediction performance quickly and easily, hash data to binary features and apply logistic regression.
- For + few % of accuracy, dig into **Kaggle forums** and the **latest industry papers** for a variety of means to engineer features most helpful to CTR prediction. We've surveyed a number here.
- Knowledge is power. (↑ data → ↑ predictions)
- Priority of effort: ↑ data > ↑ feature engineering > ↑ learning/regression algorithms.

Next Steps

- Interested in learning more about Data Science and Machine Learning?
  - Create a Kaggle Account
  - Enter a learning competition, e.g. "[Titanic: Machine Learning from Disaster]"
  - Take related tutorials, learn from kernels and discussions, steadily work to improve your skills, and share it forward