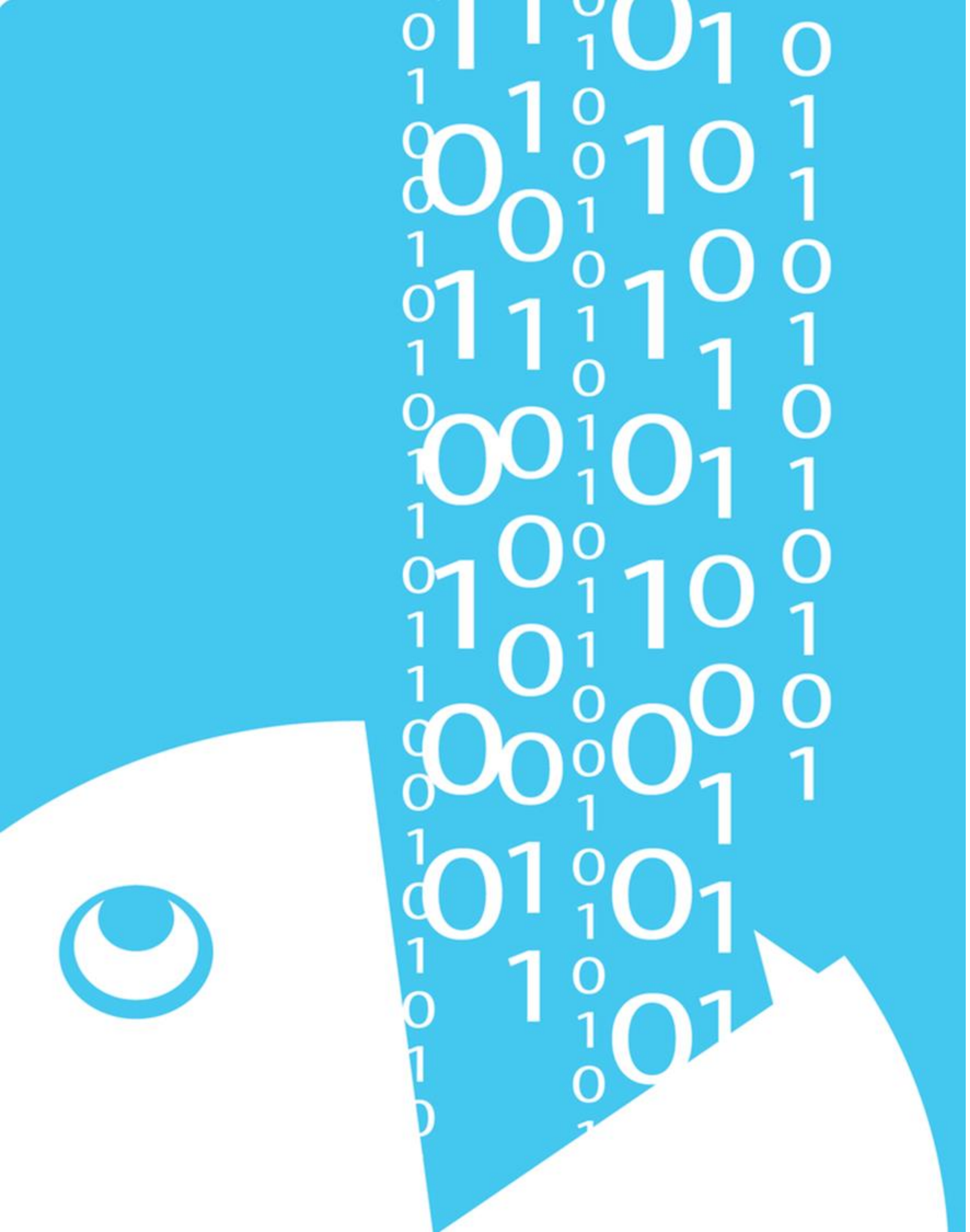
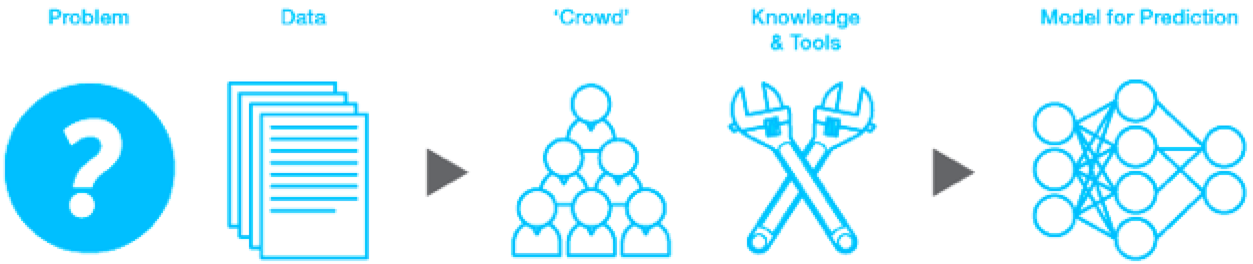




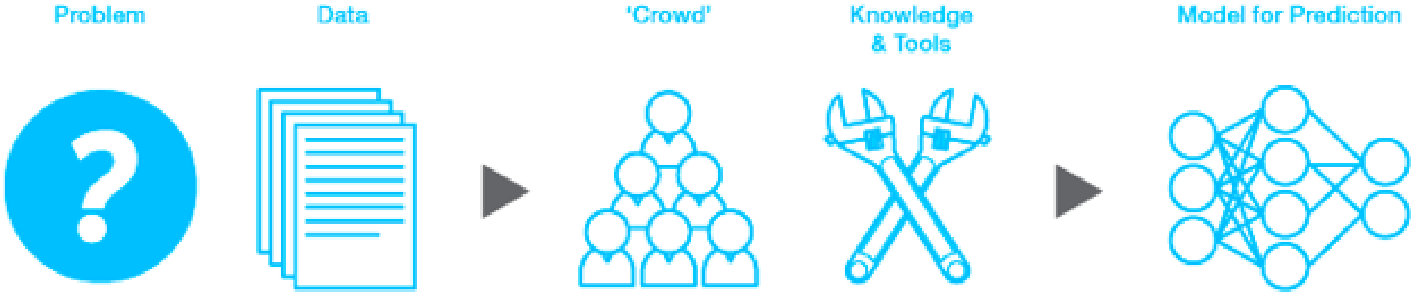
kaggle

A path to becoming an AI expert & winning data science competitions

DARIUS BARUŠAUSKAS @ OXIPIT

kaggle

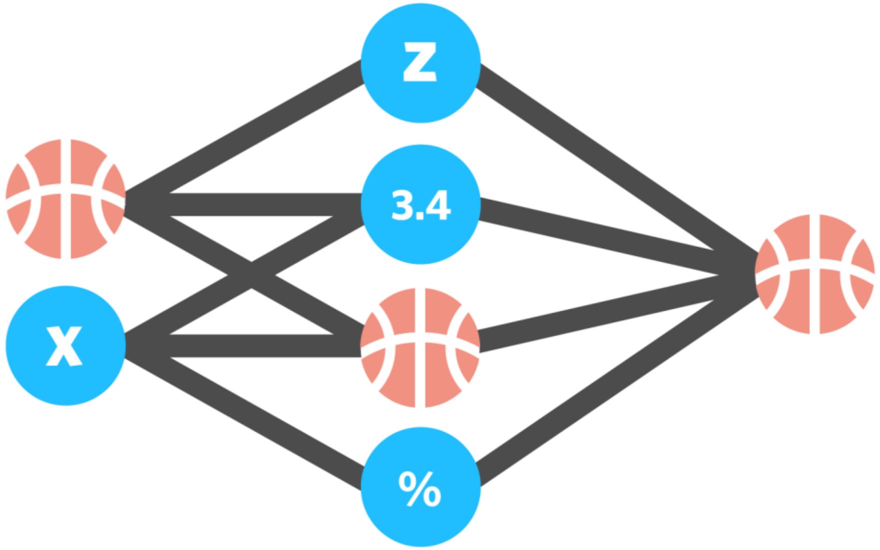
A platform for predictive modeling competitions.



"We're making data science into a sport."

Problem:

Predict outcomes of basketball matches



Expected result:

Beat the odds & profit in gambling markets

Solution:

- Gather enough data
- Define the problem in mathematical way
- Apply data science skills & create predictive models

March Machine Learning Mania 2017

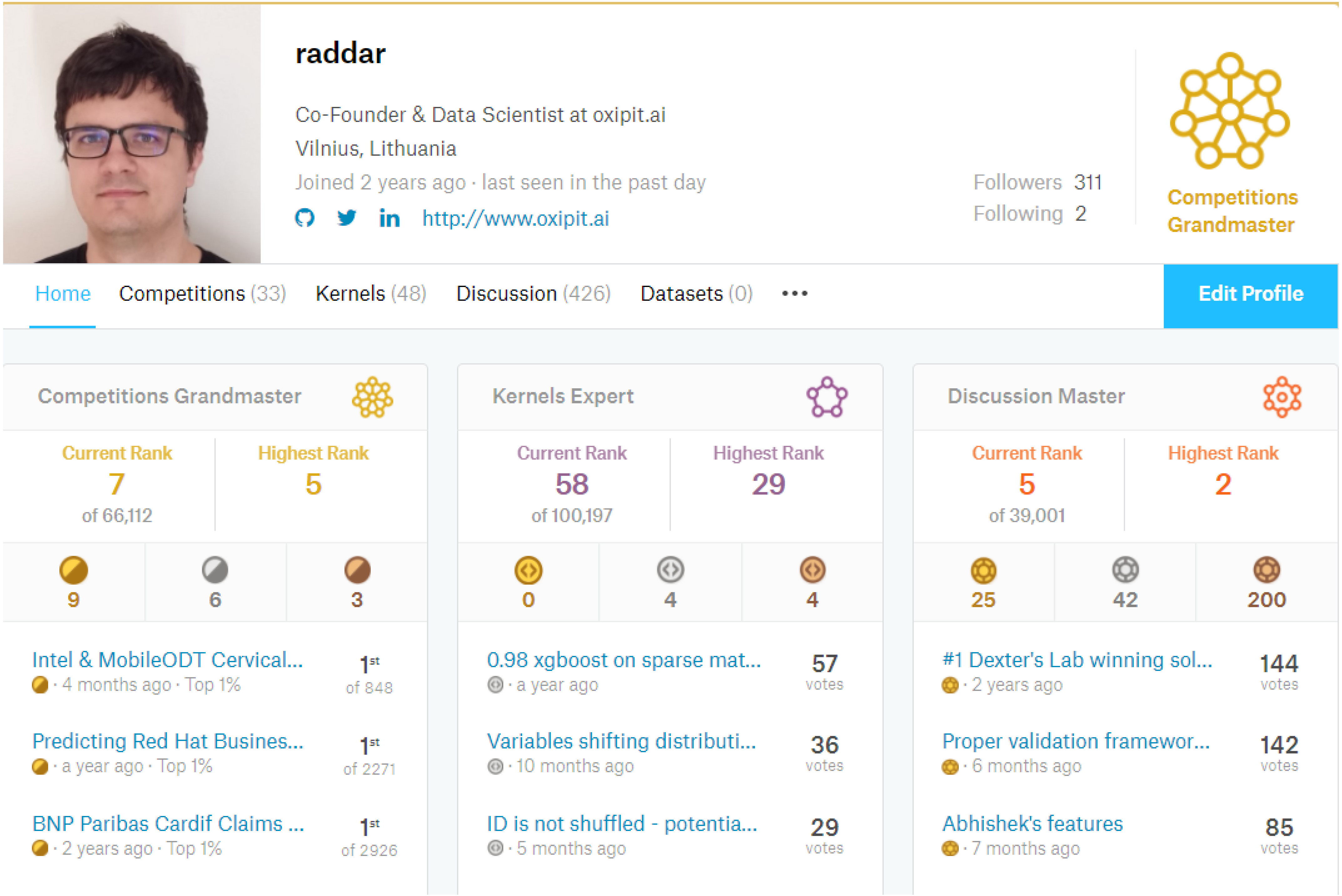
Predict the 2017 NCAA Basketball Tournament

442 teams · 5 months ago

#	∆pub	Team Name	Score
1	—	Andrew Landgraf	0.438575
2	—	skellert	0.449816
3	—	Monte McNair	0.453737
4	—	Erik	0.461005
5	—	djscott1909	0.461073
6	—	SachinKashyap	0.464925
7	—	octonion	0.466444
8	—	Bracket Hacker	0.467004
9	—	Team Derek	0.467550
10	—	James Trotman	0.467759
11	—	Don't know what I'm doing...	0.467948
12	—	Zach	0.468622
13	—	Naus	0.470295
14	—	ShedrickBridgeforth	0.470566
15	—	Ayala	0.470798
16	—	EVBettor	0.471954
17	—	Brian E	0.472076
18	—	PEC	0.472089
19	—	raddar	0.472186

Kaggle:

- Submit a model and get feedback of its performance
- Compete with other people
- Get rewarded based on your results


raddar
Co-Founder & Data Scientist at oxipit.ai
Vilnius, Lithuania
Joined 2 years ago · last seen in the past day
http://www.oxipit.ai
Followers 311
Following 2
Competitions Grandmaster
Home
Competitions (33)
Kernels (48)
Discussion (426)
Datasets (0)
Edit Profile
Competitions Grandmaster
Current Rank 7 of 66,112
Highest Rank 5
9
6
3
Intel & MobileODT Cervical... · 4 months ago · Top 1% 1st of 848
Predicting Red Hat Busines... · a year ago · Top 1% 1st of 2271
BNP Paribas Cardif Claims ... · 2 years ago · Top 1% 1st of 2926
Kernels Expert
Current Rank 58 of 100,197
Highest Rank 29
0
4
4
0.98 xgboost on sparse mat... · a year ago 57 votes
Variables shifting distributi... · 10 months ago 36 votes
ID is not shuffled - potentia... · 5 months ago 29 votes
Discussion Master
Current Rank 5 of 39,001
Highest Rank 2
25
42
200
#1 Dexter's Lab winning sol... · 2 years ago 144 votes
Proper validation framewor... · 6 months ago 142 votes
Abhishek's features · 7 months ago 85 votes

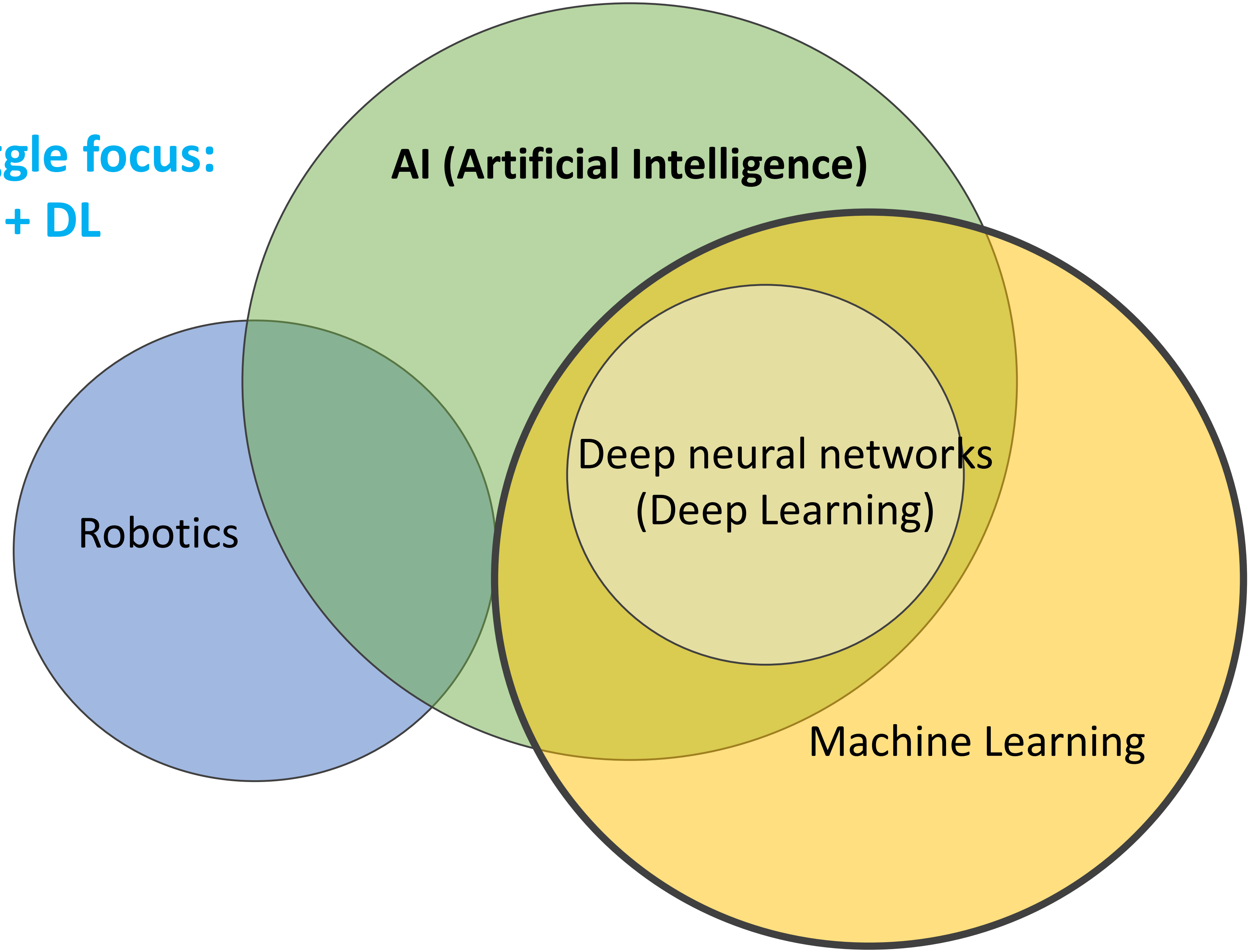
Kaggle –
What's so fun about it?

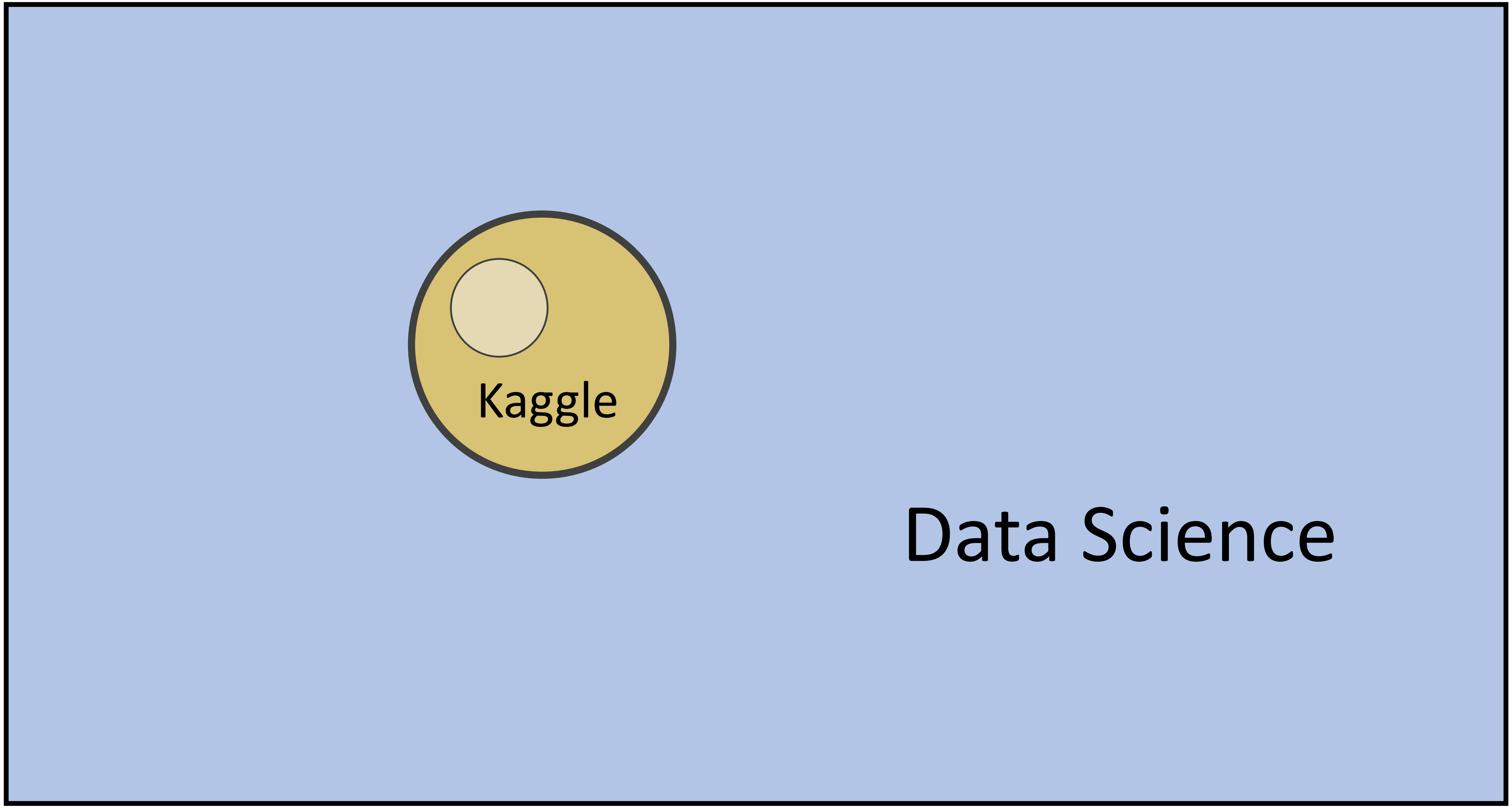
- Real-time standings - competitive spirit of a game
- Cooperate with anyone to learn from each other
- Improve or learn new skills from shared scripts in forums
- Overall profile rankings on past competition performance – added value to data science profiles
- Friendly and helping community

- And of course nice prizes for top finishers ☺

#	∆1w	Team Name ✱ in the money	Kernel	Team Members	Score	Entries	Last
1	—	✱ raddar			0.995124	169	5mo
2	—	✱ Victor			0.994862	220	5mo
3	—	✱ Joshua Havelka			0.994596	182	5mo
4	—	menny			0.994010	84	5mo
5	—	No Hat			0.993830	28	5mo
6	—	Mickey			0.993813	183	5mo
7	—	A Series Of Unlikely Explanations			0.993721	98	5mo
8	—	idle_speculation			0.993574	6	5mo
9	▲3	Nickel			0.993554	64	5mo
10	▼1	BM (aka BatMan)			0.993544	103	5mo
11	—	SK			0.993534	10	5mo
12	▼2	Darragh			0.993518	43	5mo
13	▲1	Mikhail			0.993438	28	5mo
14	▼1	KazAnova | Faron | SRK			0.993401	102	5mo

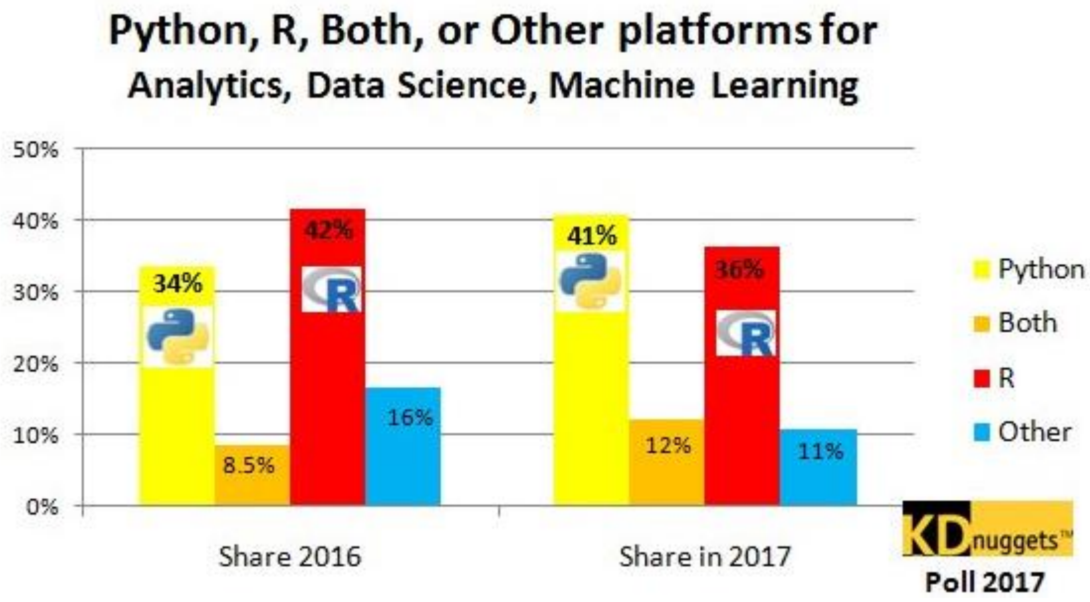
Kaggle focus:
ML + DL

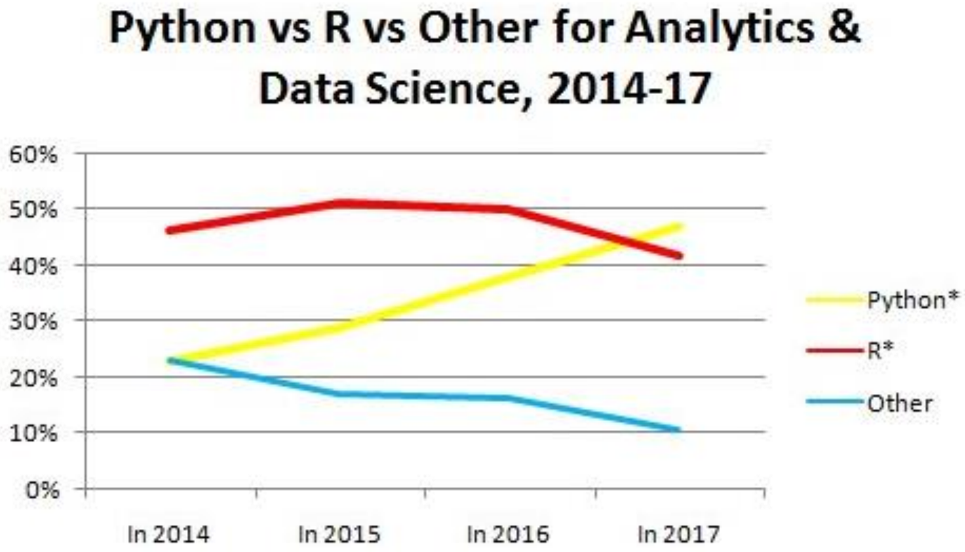



Kaggle
Data Science

Typical Kaggle problems

- Regression
- Classification
- Segmentation
- Recommendation
- Time series
- Optimization



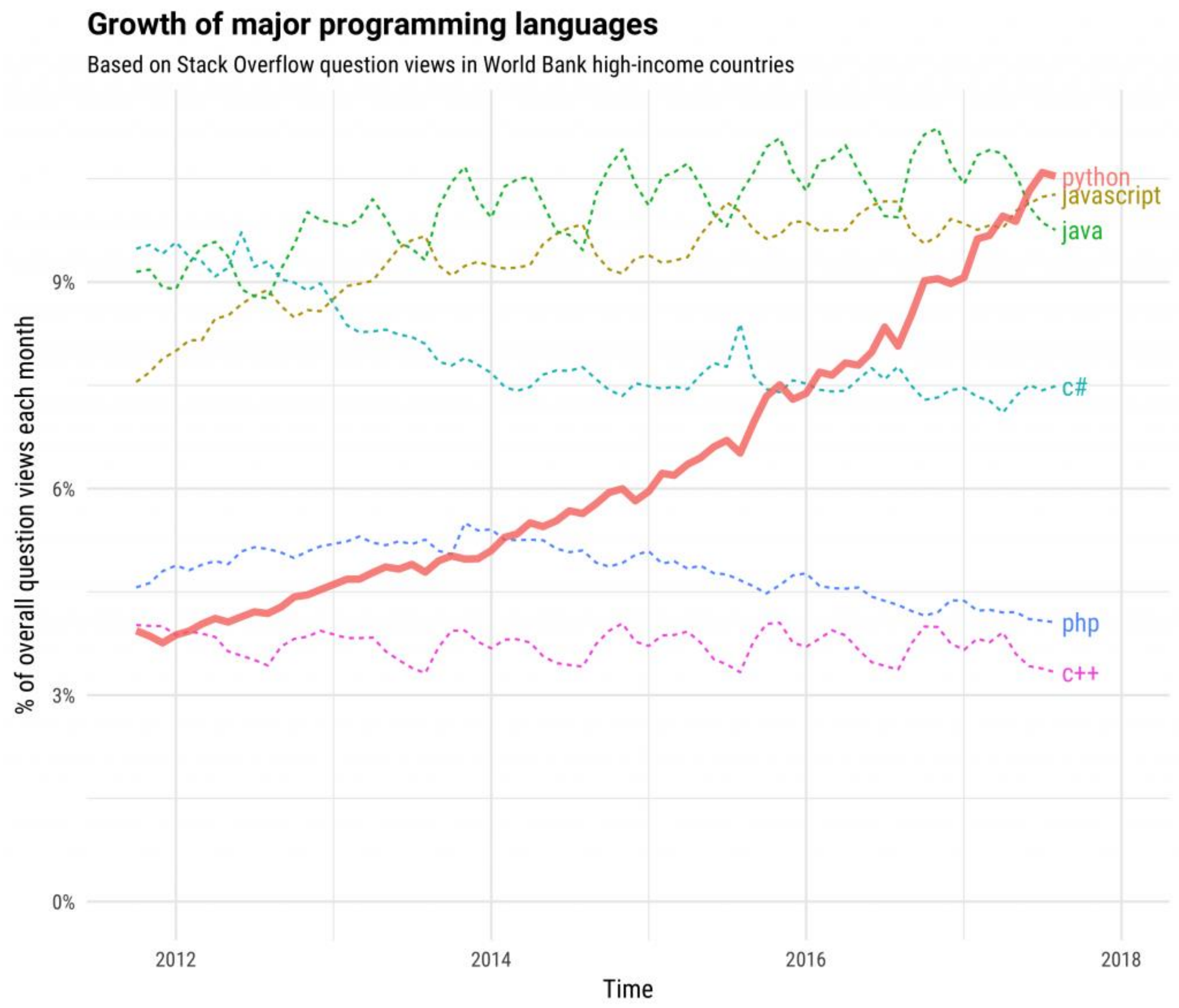


Tools

- ML – python and/or R
- DL – python

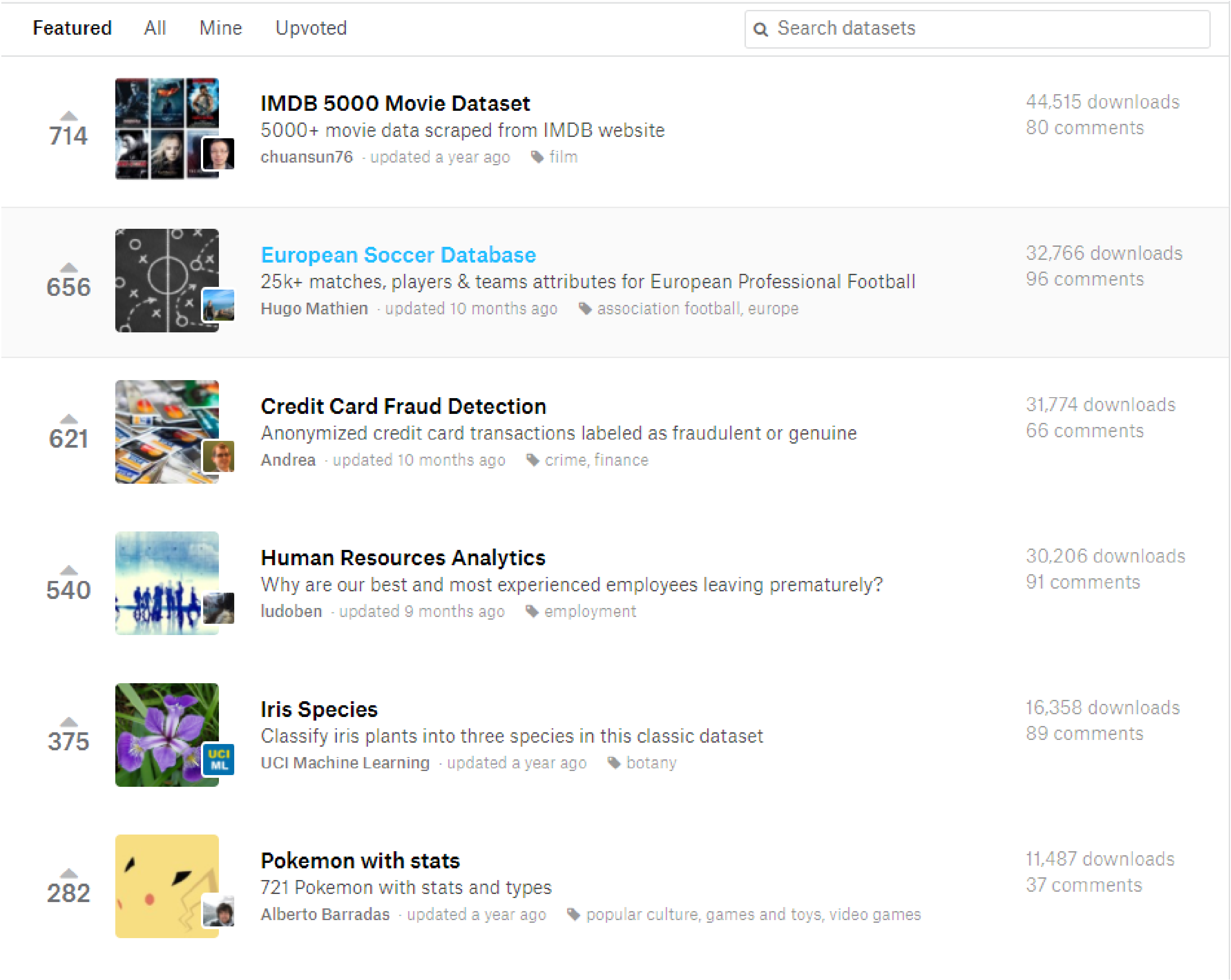
Getting started?
pick python

Kaggle 80%-90% python


Growth of major programming languages
Based on Stack Overflow question views in World Bank high-income countries
% of overall question views each month
9%
6%
3%
0%
2012
2014
2016
2018
Time
python
javascript
java
c#
php
c++

How to get started

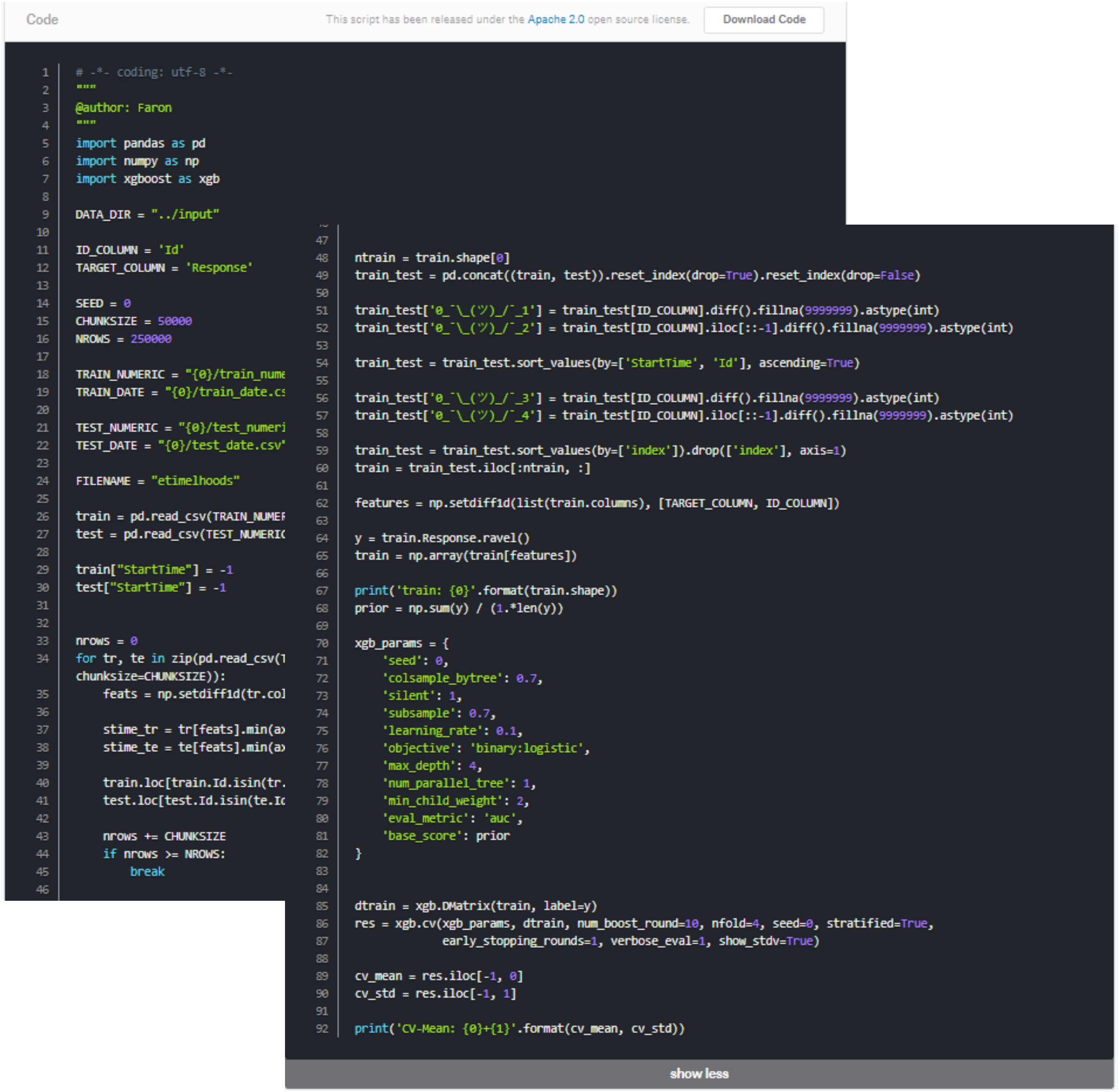
- Over 1000 playground datasets
- Over 250 competition datasets
- Pick a few which you find interesting



Learn from existing code

- People are rewarded for sharing working code
- Good code is up-voted
- Easy to fork, run & edit the code

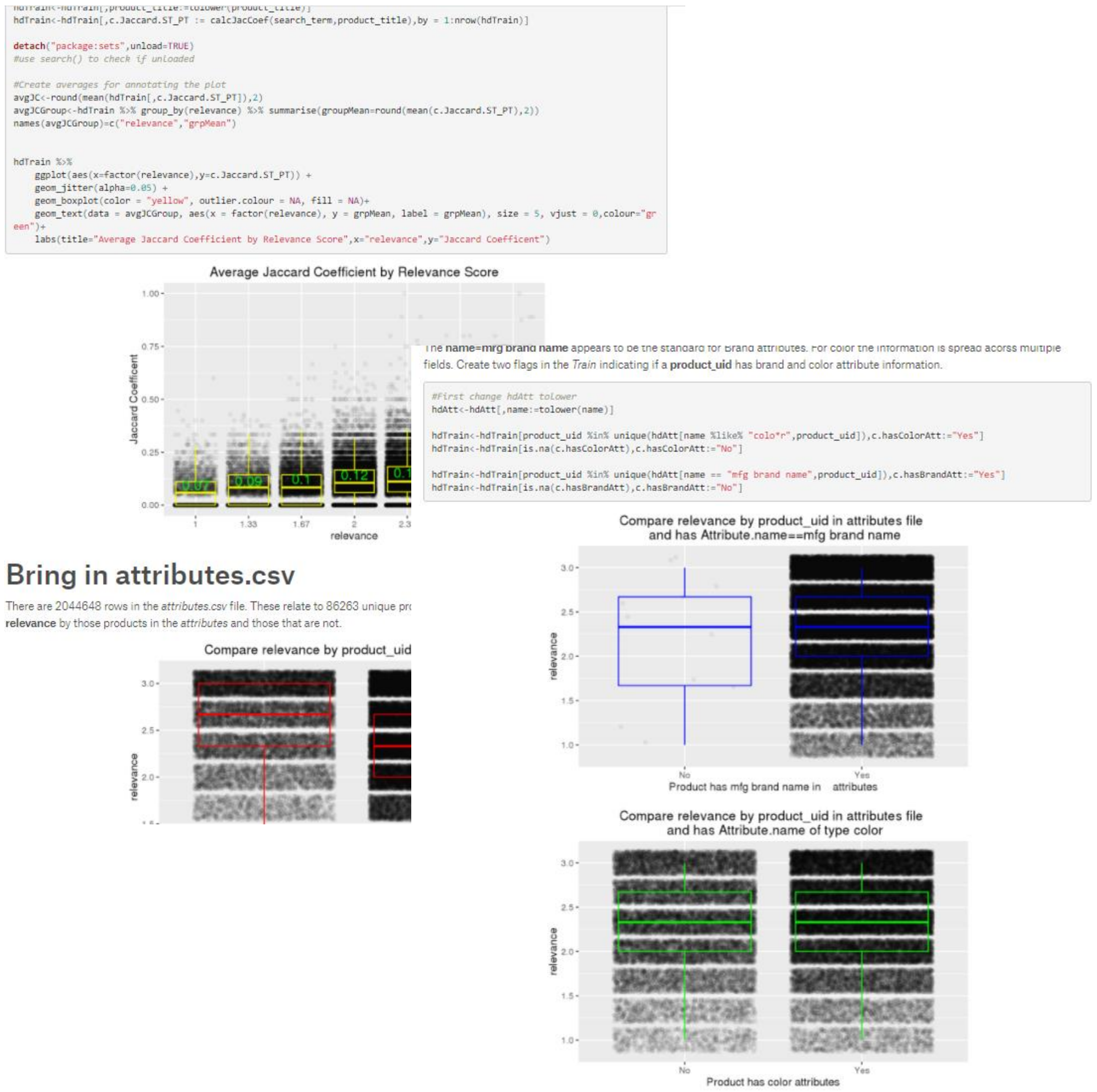
Kaggle provides an environment to run your code online



Learn from notebooks

- Visualize data with notebooks
- They tell stories how people approach problems
- Easy to reproduce

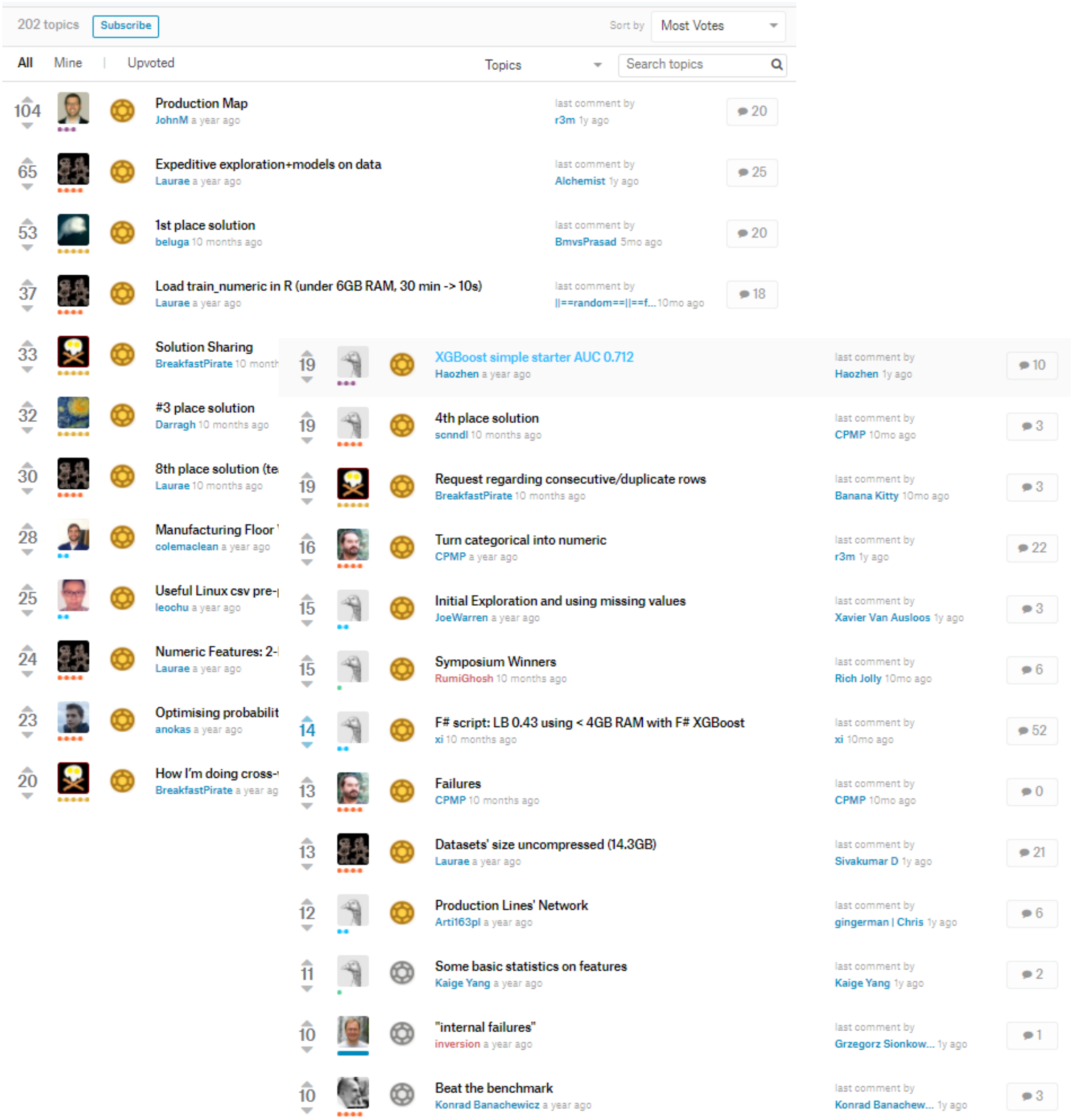
Kaggle provides an environment to write your own notebooks



Learn from discussions

- Get involved in discussions to learn which methods work
- Friendly community for newcomers in data science field

People get rewarded for contributing to informative discussions



Essential things to know to have a competitive model

- Cross-validation
- Bagging
- Ensembling

March Machine Learning Mania 2017

Predict the 2017 NCAA Basketball Tournament

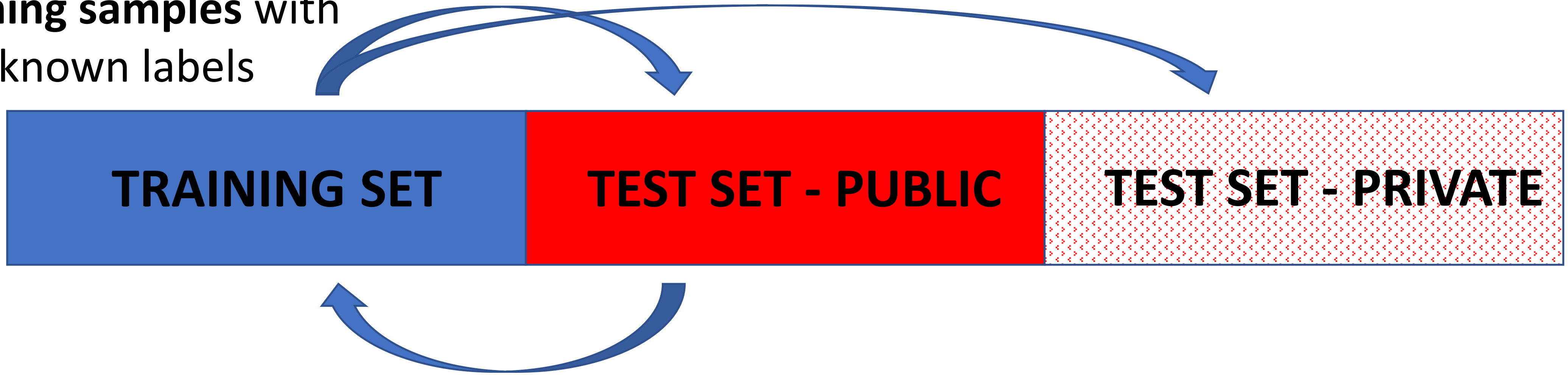
442 teams · 5 months ago

#	∆pub	Team Name	Score
1	—	**Andrew Landgraf**	0.438575
2	—	**skellert**	0.449816
3	—	**Monte McNair**	0.453737
4	—	**Erik**	0.461005
5	—	**djscott1909**	0.461073
6	—	**SachinKashyap**	0.464925
7	—	**octonion**	0.466444
8	—	**Bracket Hacker**	0.467004
9	—	**Team Derek**	0.467550
10	—	**James Trotman**	0.467759
11	—	**Don't know what I'm doing...**	0.467948
12	—	**Zach**	0.468622
13	—	**Naus**	0.470295
14	—	**ShedrickBridgeforth**	0.470566
15	—	**Ayala**	0.470798
16	—	**EVBettor**	0.471954
17	—	**Brian E**	0.472076
18	—	**PEC**	0.472089
19	—	**raddar**	0.472186

How does it all work?

1. Build model using **training samples** with known labels

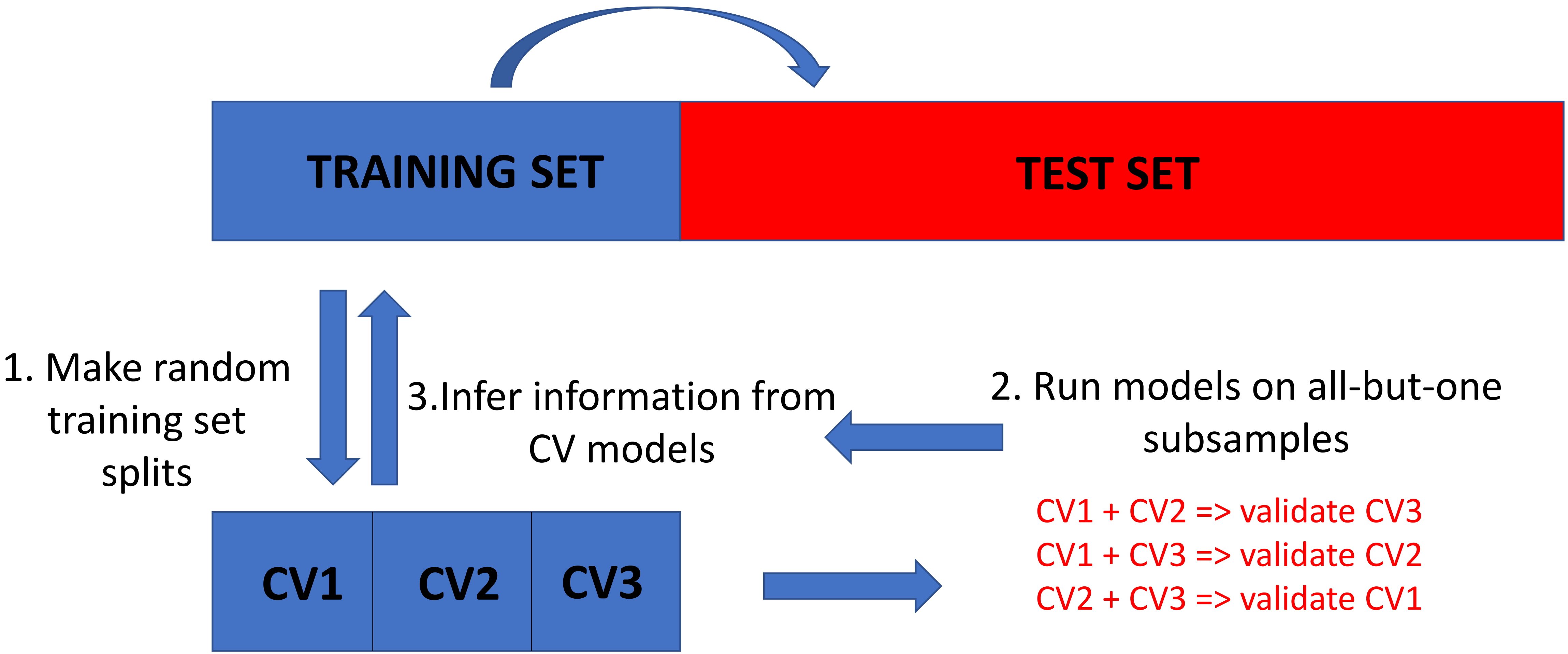
2. Make predictions on **test samples**



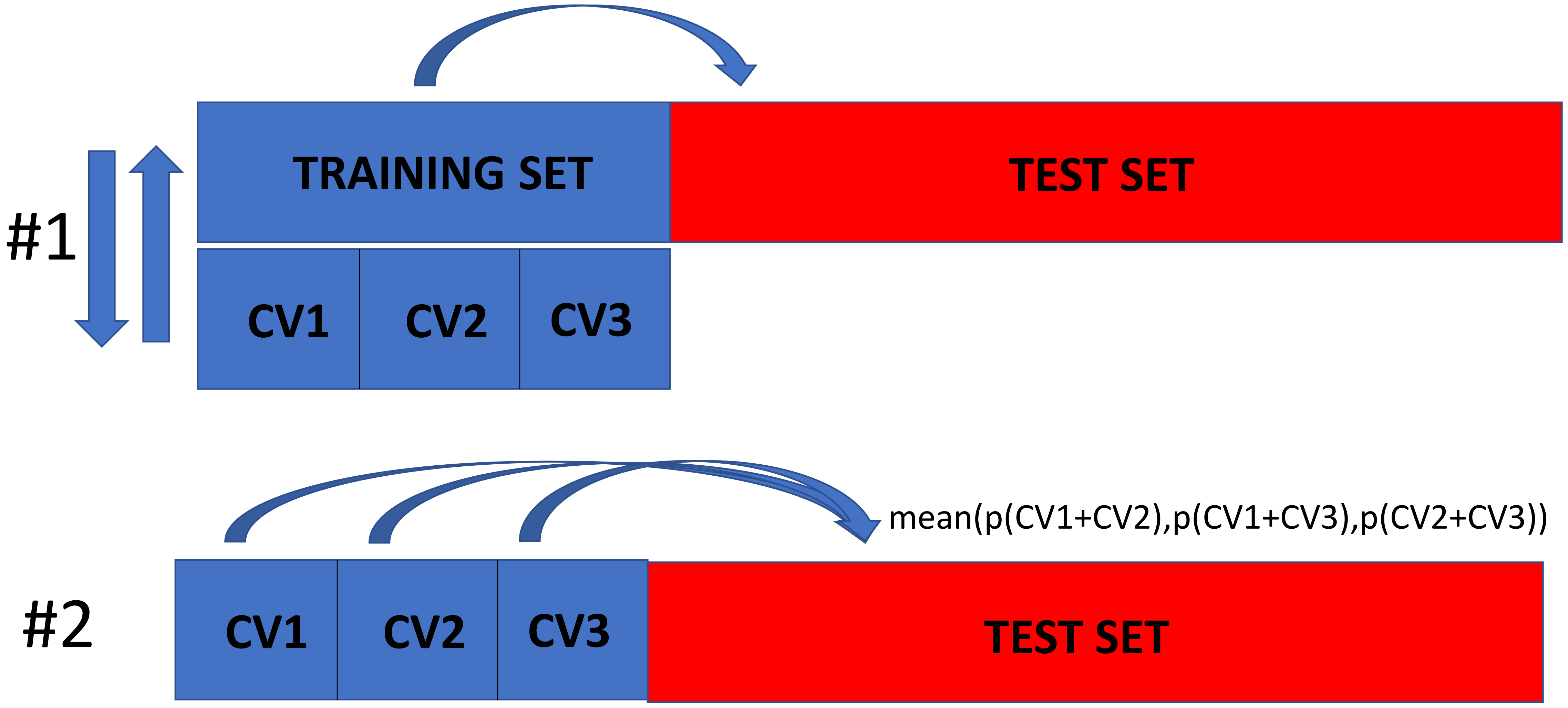
3. Submit model predictions on **“unseen”** data and get quick response in public leaderboard standings

4. Final standings based on Private test predictions

Cross-validation – the basics



Using CV results



Public leaderboard – don’t be fooled

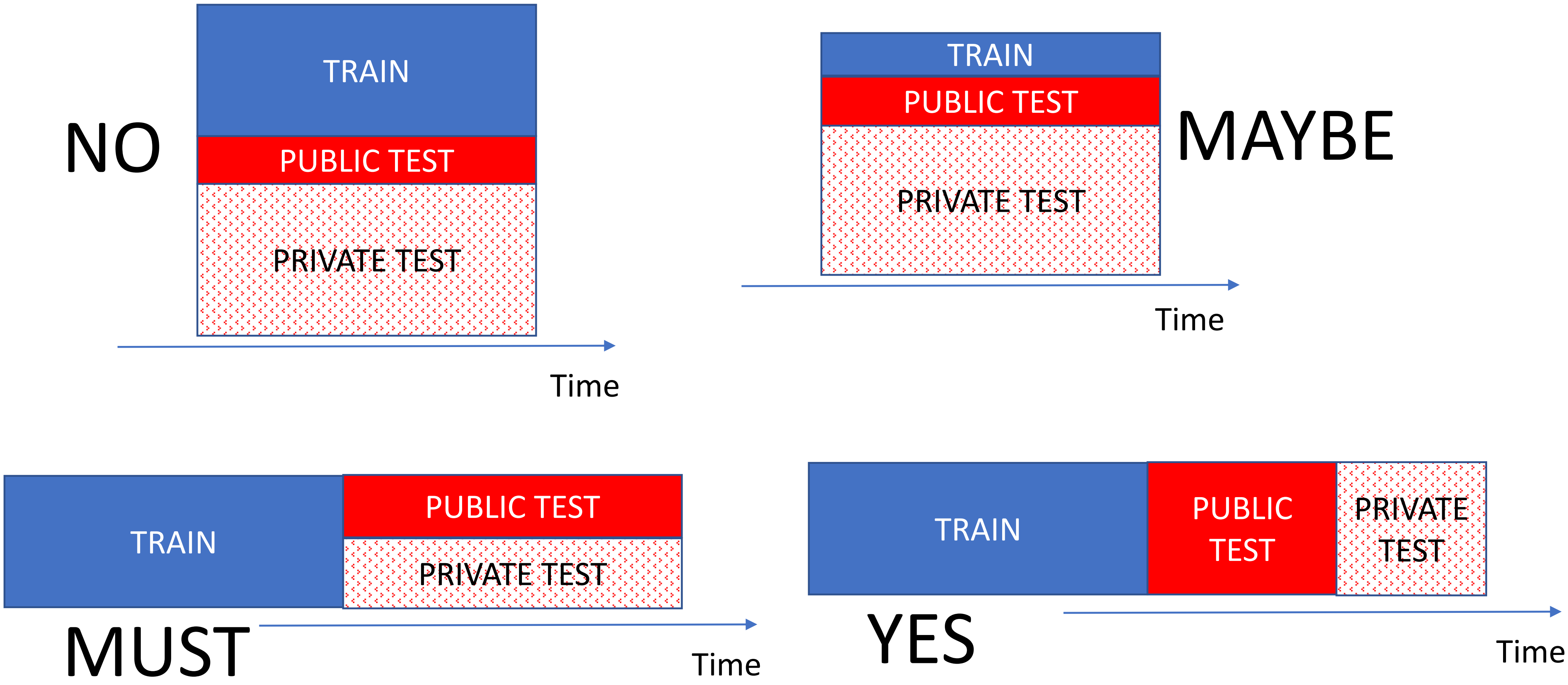
Random forest example – tuning number of trees with no proper validation

Model	Public test score	Private test score
Number trees 200	0.812	0.815
Number trees 300	0.814	0.819
Number trees 400	0.816	0.820
Number trees 500	0.814	**0.821**
Number trees 450	0.817	0.819
Number trees 430	0.818	0.816
… many attempts later …	Maximum public LB score	Seems good right? – often NO!
Number trees 437	0.820	0.814

#500 < #400 => test if #450 is good? => public LB score confirms it! => spurious results

Rule of thumb – the less submissions you make, less likely you have overfitted to LB solution!

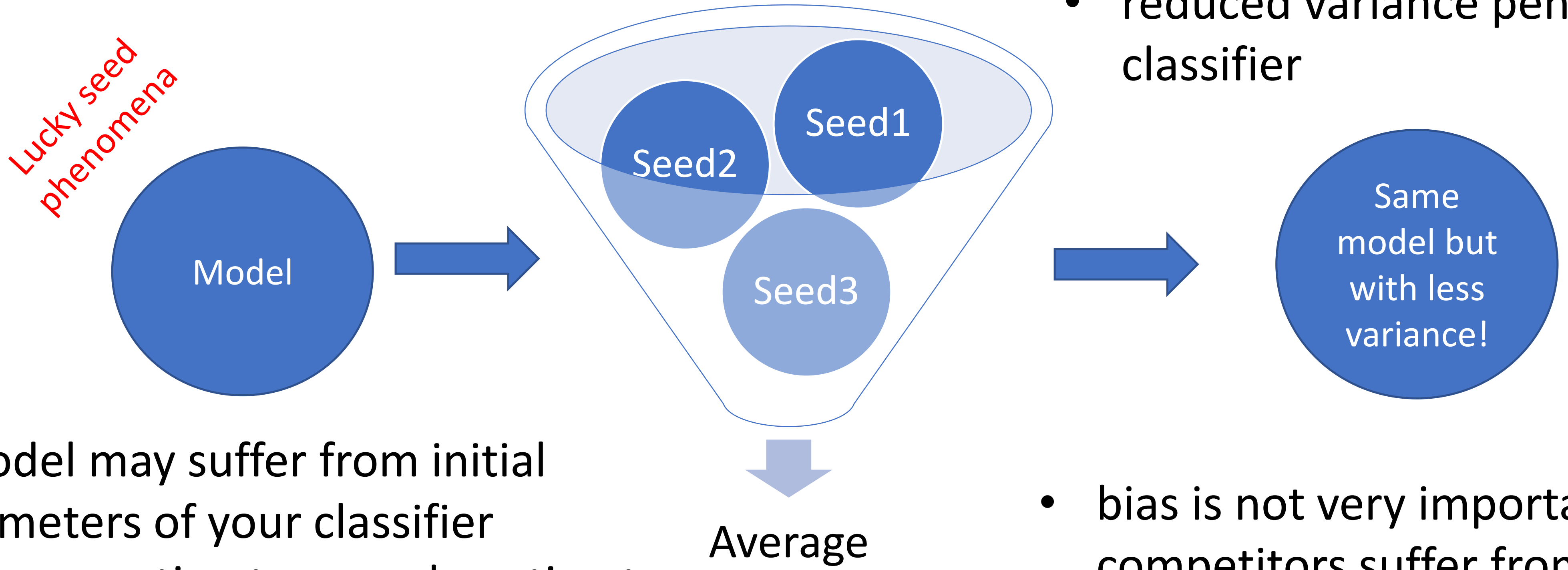
Should I even trust public leaderboard?



Bootstrap aggregating – in short: bagging

Bagging ~= averaging similar models

Bias-variance tradeoff:
- reduced variance penalty of a classifier



A model may suffer from initial parameters of your classifier (may overestimate or underestimate your potential score)

- bias is not very important – competitors suffer from bias too (with similar degree)

Numeric features

Feature transformations to consider:

- Scaling – min/max, N(0,1), root/power scaling, log scaling, Box-Cox, quantiles
- Rounding (too much precision might be noise!)
- Interactions {+,-,*,/}
- Row counters – #0’s, #NA’s, #negatives
- Row stats (if makes sense) – min, max, median, skewness…
- Row similarity scoring – cosine similarity
- Clustering
- …

Sometimes numeric features are categorical or even ordinal by nature - always consider that!

Tree methods are *almost* invariant to **scaling.**
Linear models & neural networks need careful raw data treatment!

Ways to deal with categorical features

- *One hot encoding* – create bag-of-words (bag the size of unique categories) and assign 0/1's for each representing column. Will fail on new categories.
- *Label encoding* – label each category into $\mathbb{N}$ space. Will fail for new categories

Features with many categories - rows:categories ratio 20:1 or less

- *Count encoding* – count number of each category and use counter as a value. Iterate counter for each CV fold
- *Hash encoding* – mapping categories to reduced category space and do one hot encoding. Produces good results but may be sub-optimal.
- *Category embedding* – use a function (autoencoder) to map each category to Euclidian space
- ***Likelihood encoding* – apply label average (likelihood) for each category**

Categorical features – interactions!

- If interactions are natural for a problem - ML only does approximations! => sub-optimal
- Always test your method with all explicitly created possible 2-way interactions

- Interactions feature selection may be important due to exploding feature space (20 categorical features => 20*19/2 interactions!)
- If 2-way interactions help – go even further (3-way, 4-way, …)

Interaction example

X1	X2	X1*X2
BMW	6	BMW-6
AUDI	9	AUDI-9
BMW	4	BMW-4
AUDI	13	AUDI-13
BMW	7	BMW-7
BMW	21	BMW-21

Likelihood encoding

Imagine having categorical feature with 100k different values on 1M dataset – one hot encoding each category => dimensionality curse for any model

How not to do it!

X1	Label	X1m
“BMW”	0	0.5
“BMW”	1	0.5
“BMW”	1	0.5
“BMW”	0	0.5
“BMW”	NA	0.5

Label information of each row was transferred into independent variable X1m – label now has indirect dependence to itself - label leakage introduced!

Likelihood encoding

How to do it? Might work but risky

X1	Label	X1m
“BMW”	0	0.666 + rand(0,sigma)
“BMW”	1	0.333 + rand(0,sigma)
“BMW”	1	0.333 + rand(0,sigma)
“BMW”	0	0.666 + rand(0,sigma)
“BMW”	NA	0.5

Problem: selecting optimum sigma value is problematic and depends on how much you regularize your model

Holdout set might help but not always

Likelihood encoding

X1	Label	X1m
"BMW"	0	CV out-of-fold prediction
"BMW"	1	CV out-of-fold prediction
"BMW"	1	CV out-of-fold prediction
"BMW"	0	CV out-of-fold prediction
"BMW"	NA	CV predictions average

Nested cross-validation and high number of CV folds for your model is highly recommended
Additional CV calculations are done within each model CV fold. Tricky and prone to errors method.

Problem: some degree of loss of information due to CV scheme

Good option for larger datasets

ML Methods

- Regularized GLMs (underperforming but good ensembling material)
- XGBoost (top pick for traditional data)
- LightGBM (top pick for traditional data)
- Keras (NN's are always good with good pre-processing)
- LinearSVM (non-linear is resource hungry and usually not worth it)
- Vowpal Wabbit (extremely fast online learning algorithms)
- Random forests (used to be popular, underperforming to GBM's now)

To achieve maximum, each method requires good understanding of how they work and how they should be tuned

Tuning parameters

Naive approach : apply grid search on all parameter space

- Zero effort and no supervision
- Enormous parameters' space
- Very time consuming

Bayesian optimization methods : trade-off between expert and grid-search approach

- Zero effort and no supervision
- Grid space reduced on previous iterations' results (mimic expert decisions)
- Time consuming (still)

Competitive advantage

- Get good score in public LB as fast as possible (if you intend to team up with experienced people)
- Fail fast & often / agile sprint / fast iterations
- Use many different methods and remember to save all your models (both CV and test runs)

- Invest into having generic modelling scripts for every method – time saver long-term
- Debug, Debug, Debug – nothing worse is finding a silly preprocessing error at the last day of the competition
- Write reproducible code - setting seeds before any RNG tasks
- Learn to write efficient code (vectorization, parallelization; use C, FORTRAN, etc. based libs)
- Learn how to look at the data and do feature engineering
- Get access to reasonable hardware (8CPU threads, 32GB RAM minimum); AWS is always an option

Competitive advantage – data flow pipeline

Ideally you would want to have a framework which could be applied to any competition;
It should be:

- Data friendly (sparse/dense data, missing values, larger than memory)
- Problem friendly (classification, regression, clustering, ranking, etc.)
- Memory friendly (garbage collection, avoid using *swap* partition, etc.)
- Automated (runs unsupervised from data reading to generating submission files)

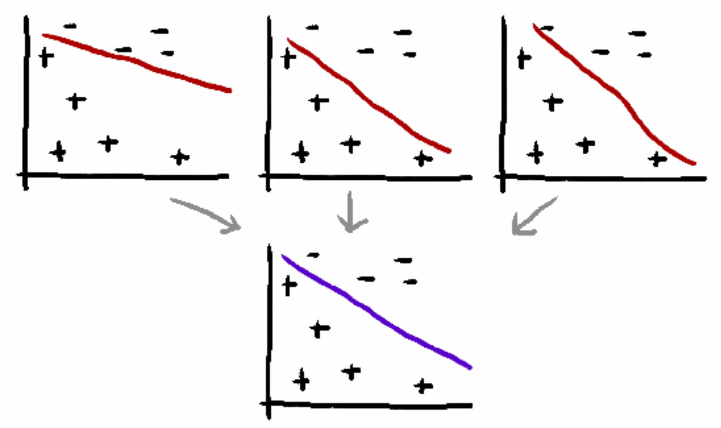
Good framework can save hours of repetitive work when going from one competition to another

Success formula (personal opinion)

50% - feature engineering
30% - model diversity
10% - luck
10% - proper ensembling:

- Voting
- Averaging
- Bagging
- Boosting
- Binning
- Blending
- Stacking

Although 10%, but it is the key what separates top guys from the rest



MODEL	PUBLIC MAE	PRIVATE MAE
Random Forests 500 estimators	6.156	6.546
Extremely Randomized Trees 500 estimators	6.317	6.666
KNN-Classifier with 5 neighbors	6.828	7.460
Logistic Regression	6.694	6.949
Stacking with Extremely Randomized Trees	**4.772**	**4.718**

Ensembling added efficiency +30%

Ensembling by voting

Common application – recommendation models

```
1111111100 = 80% accuracy
0111011101 = 70% accuracy
1000101111 = 60% accuracy
```

Majority vote

```
1111111101 = 90% accuracy
```

Ensembling by averaging

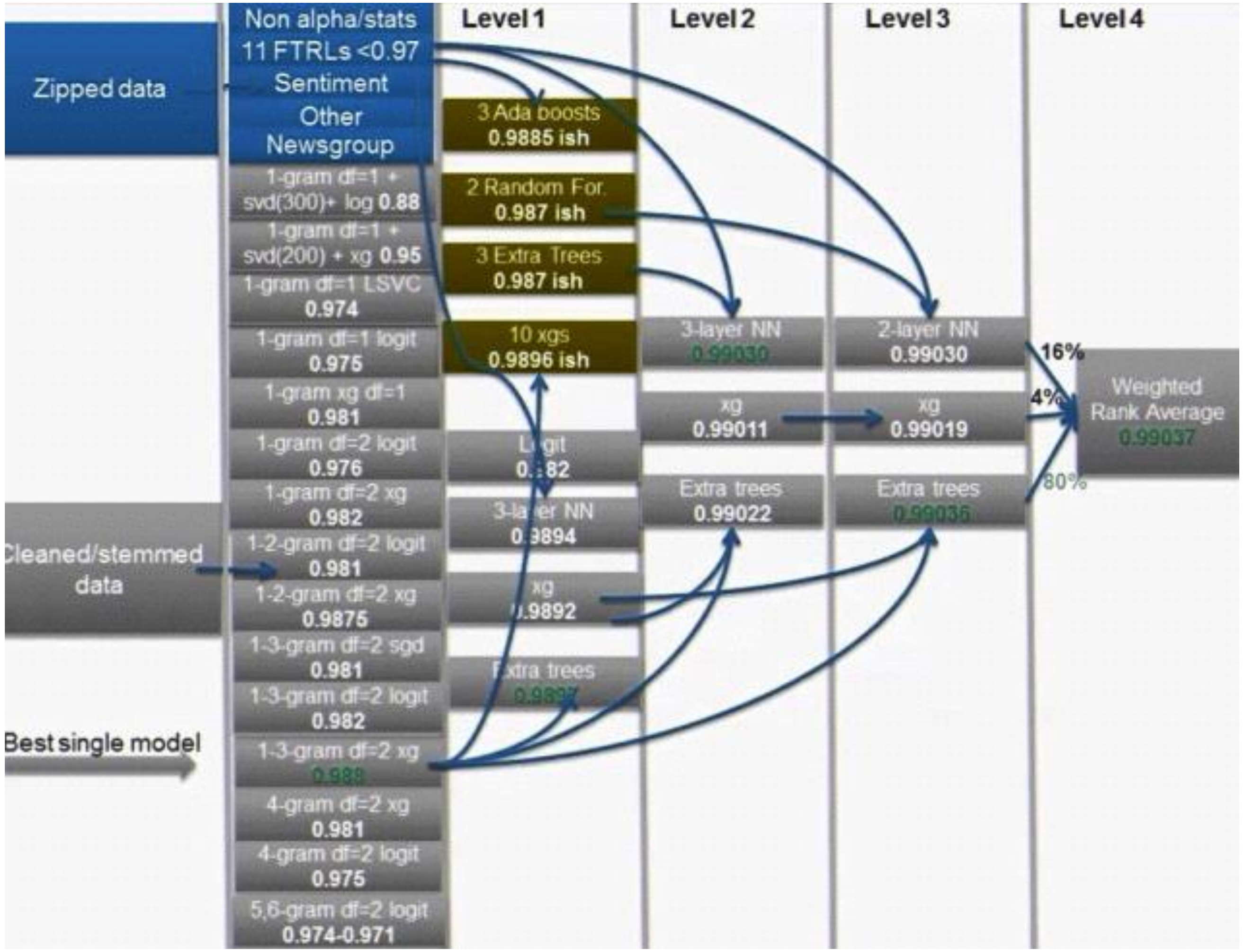
Let's say we have N classifiers: X1, X2, … , XN

We want to make a single prediction using weighted average:
B1*X1+B2*X2+B3*X3+…+BN*XN

Solve the problem using CV predictions with optimization algorithms

optim(B1*X1+B2*X2+B3*X3+…+BN*XN) with starting weights Bi=1/N

Structure of stacked models



- Weaker models often struggle to compensate fine-tuned model’s weaknesses
- “More is better” motivates create monster ensembles of sub-optimal models
- **Model selection for stacker > fine-tuning**

Model selection for ensemble

Having more models than necessary in ensemble may hurt.

Lets say we have a library of created models. Usually greedy-forward approach works well:

- Start with a few well-performing models’ ensemble
- Loop through each other model in a library and add to current ensemble
- Determine best performing ensemble configuration
- Repeat until metric converged

During each loop iteration it is wise to consider only a subset of library models, which could work as a regularization for model selection.

Repeating procedure few times and bagging results reduces the possibility of overfitting by doing model selection.

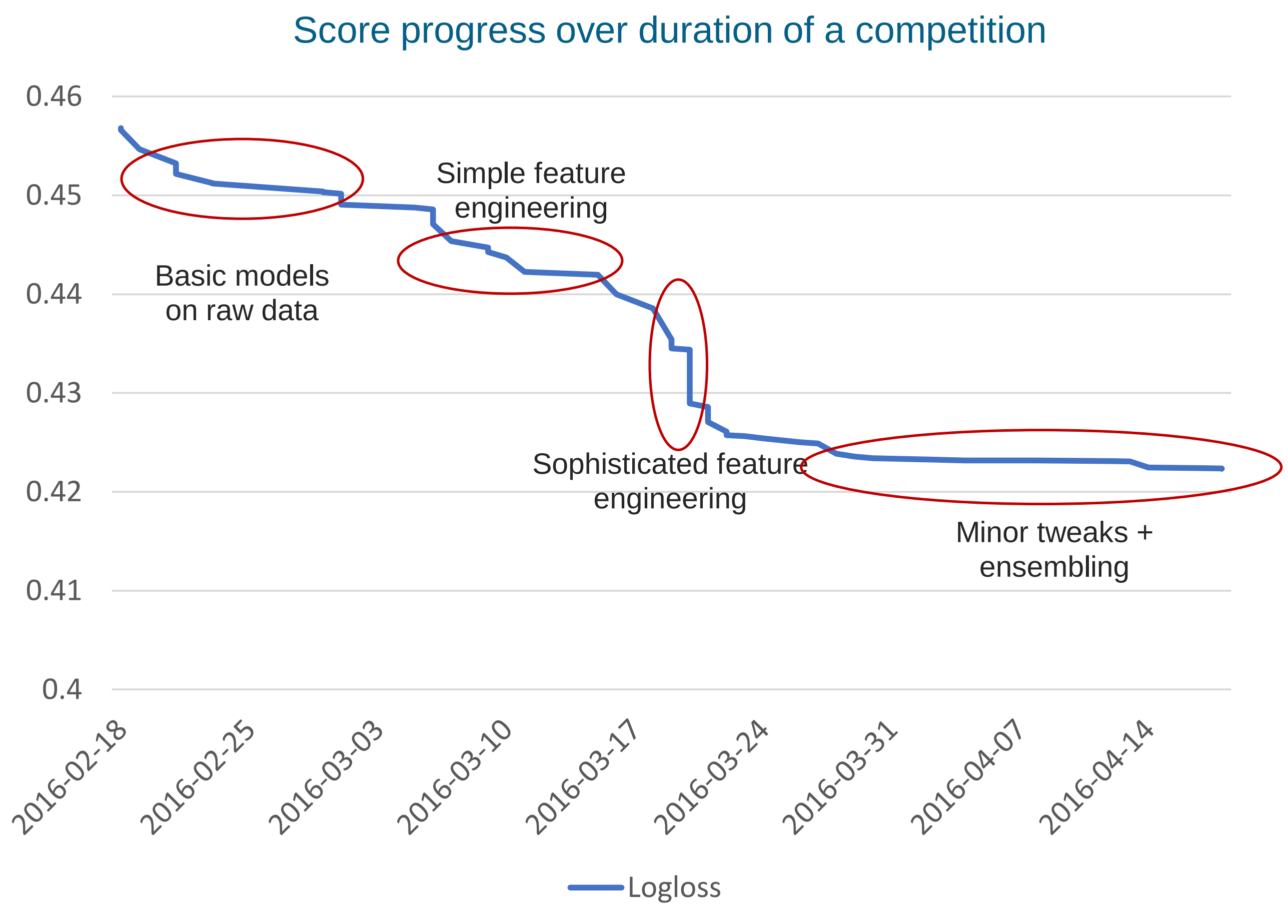
Rule of thumb: data first, ML later

1. Create few simple models first – having a good dataflow pipeline early is very convenient

2. Data exploration and visualization. Might be boring and frustrating, but pays off well. Excel is underrated in this aspect☺

3. Think of how to make **smart** features; avoid using linear combinations, {/,*} is superior

4. Winners usually find something that most people struggle to see in data. Not many people look at the data at all!



Feature engineering - summary

Feature engineering is hard and is the most creative part in Kaggle – not many enjoy it

Typical feature transformations:

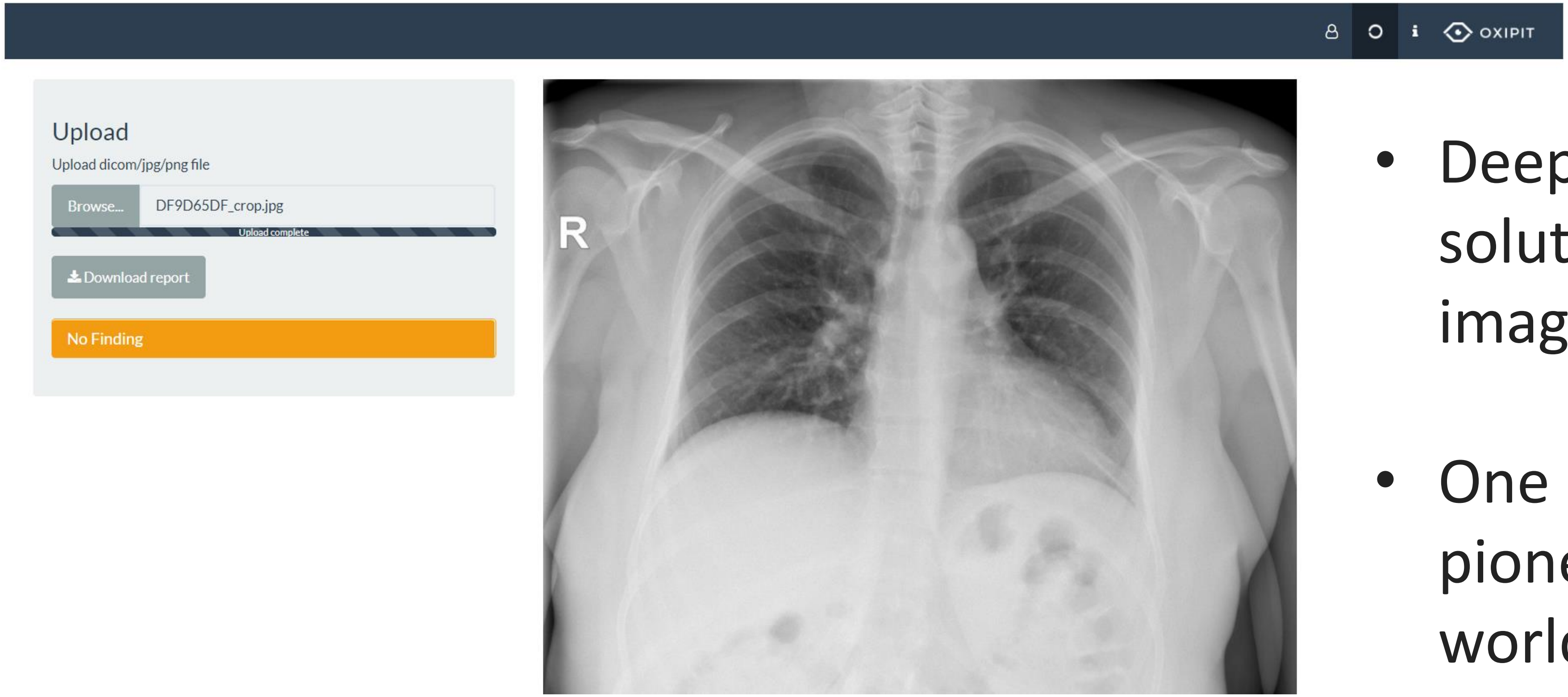
- Multiplication & ratios of numeric features
- Log, Box-Cox transformations
- PCA, t-SNE outputs
- Percentile ranks
- Interactions of categorical features (A,B => “AB”), multiple levels of interactions
- Lag & lead features (for timetable data)
- Target likelihood ratios of categorical features
- One-hot encoding/label encoding of categorical features
- …

Having strong explicit features in simple models often beat monster ensembles with no feature engineering (personal experience ☺)

Final remarks

- Kaggle is a playground for hyper-optimization and stacking – for business any solution in 10% rankings is sufficient.
- Teaming up is important – simple average of i.e. 9th and 10th place finishes can sometimes beat 1st place solution.
- No winning solution go straight to production due to their complexity or occasional leakages.
- Sponsors benefit from forum discussions and provided data visualizations.

oxipit.ai



- Deep Learning solutions for medical imaging
- One of the first pioneers in the field worldwide