

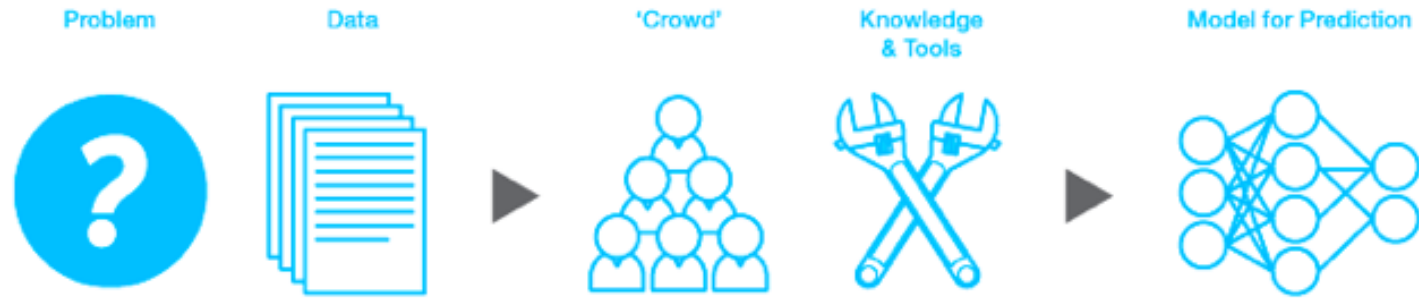
kaggle™

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

DARIUS BARUŠAUSKAS @ OXIPIT



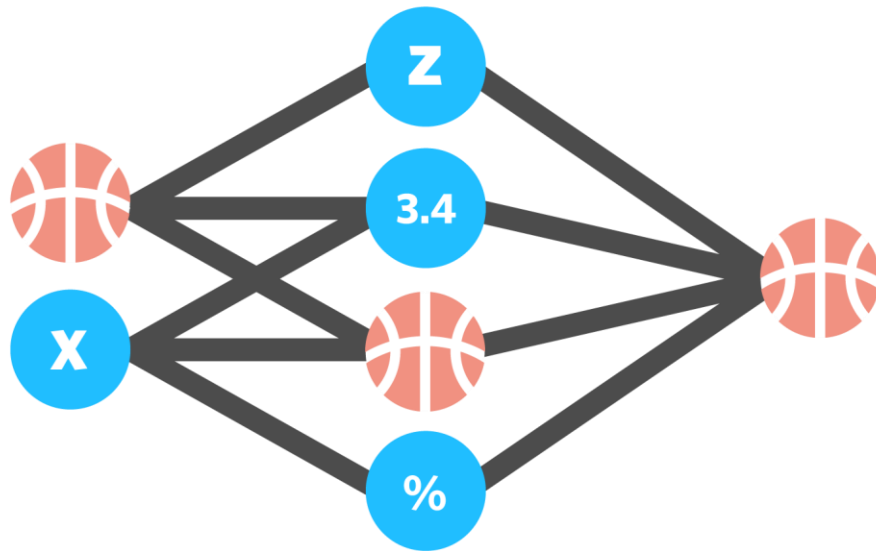
A platform for predictive modeling competitions.



“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

#	Δ 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
[GitHub](#) [Twitter](#) [LinkedIn](#) <http://www.oxipit.ai>



**Competitions
Grandmaster**

Followers 311

Following 2

[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

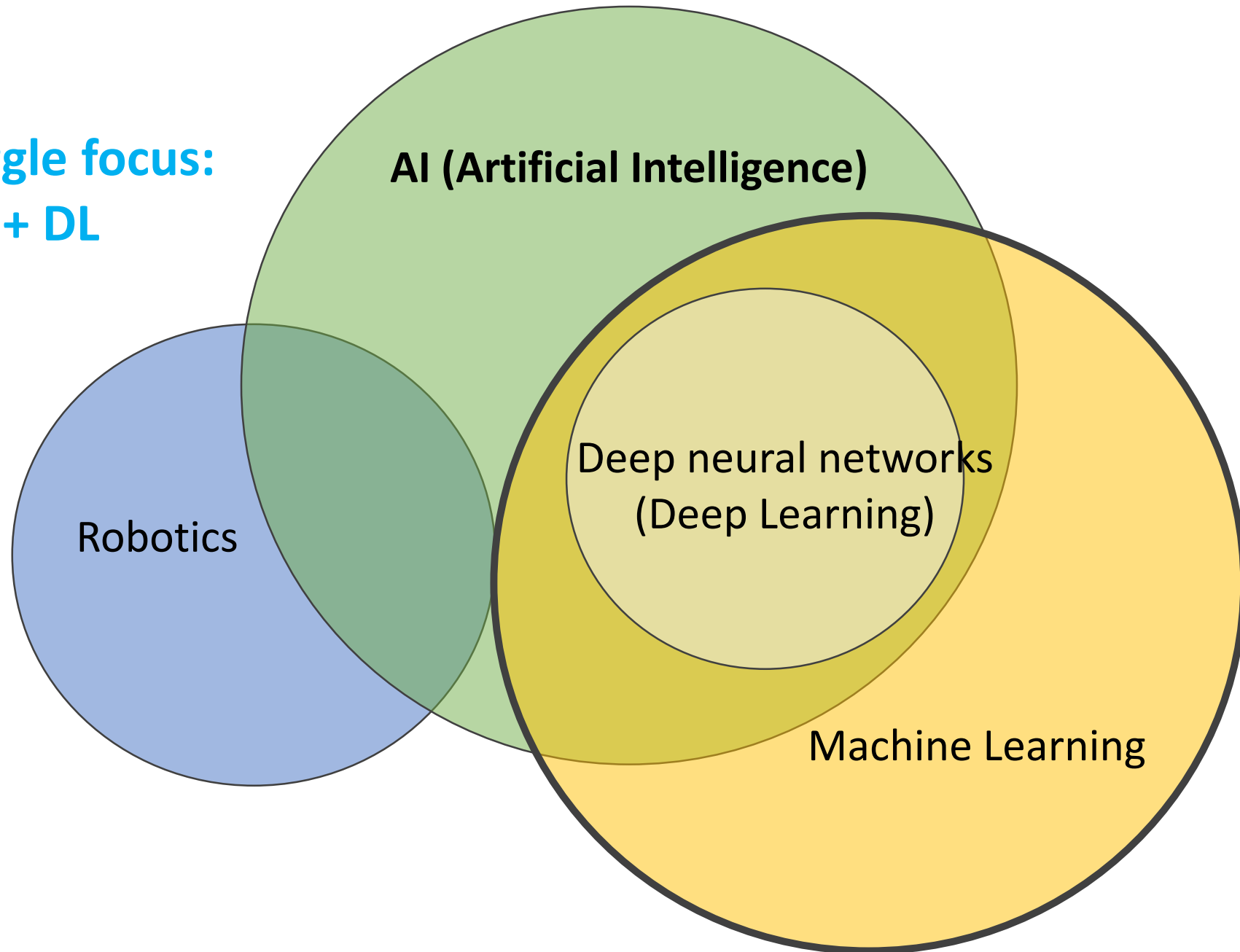
votes

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 😊

#	Δ1w	Team Name <small>★ in the money</small>	Kernel	Team Members	Score <small>🔍</small>	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



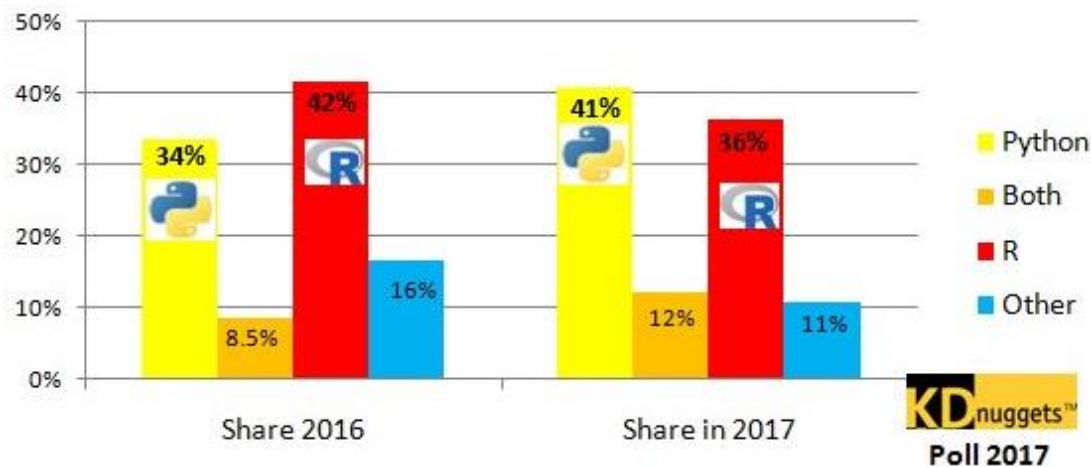


Data Science

Typical Kaggle problems

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

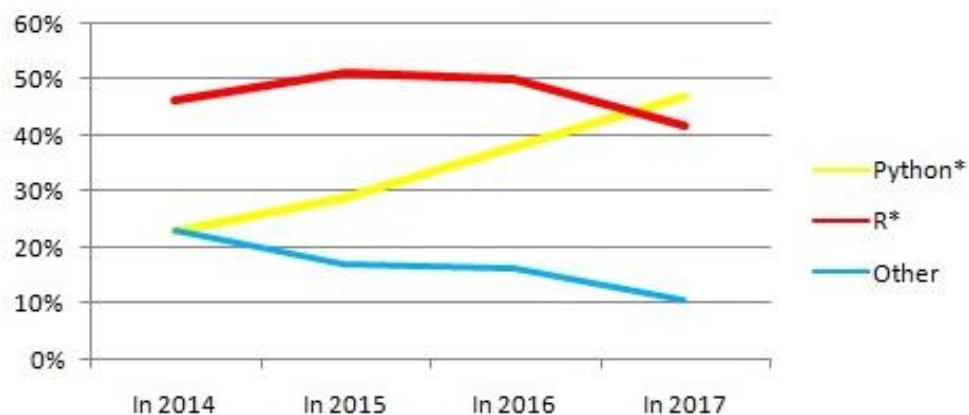
Python, R, Both, or Other platforms for Analytics, Data Science, Machine Learning



Tools

- ML – python and/or R
- DL – python

Python vs R vs Other for Analytics & Data Science, 2014-17

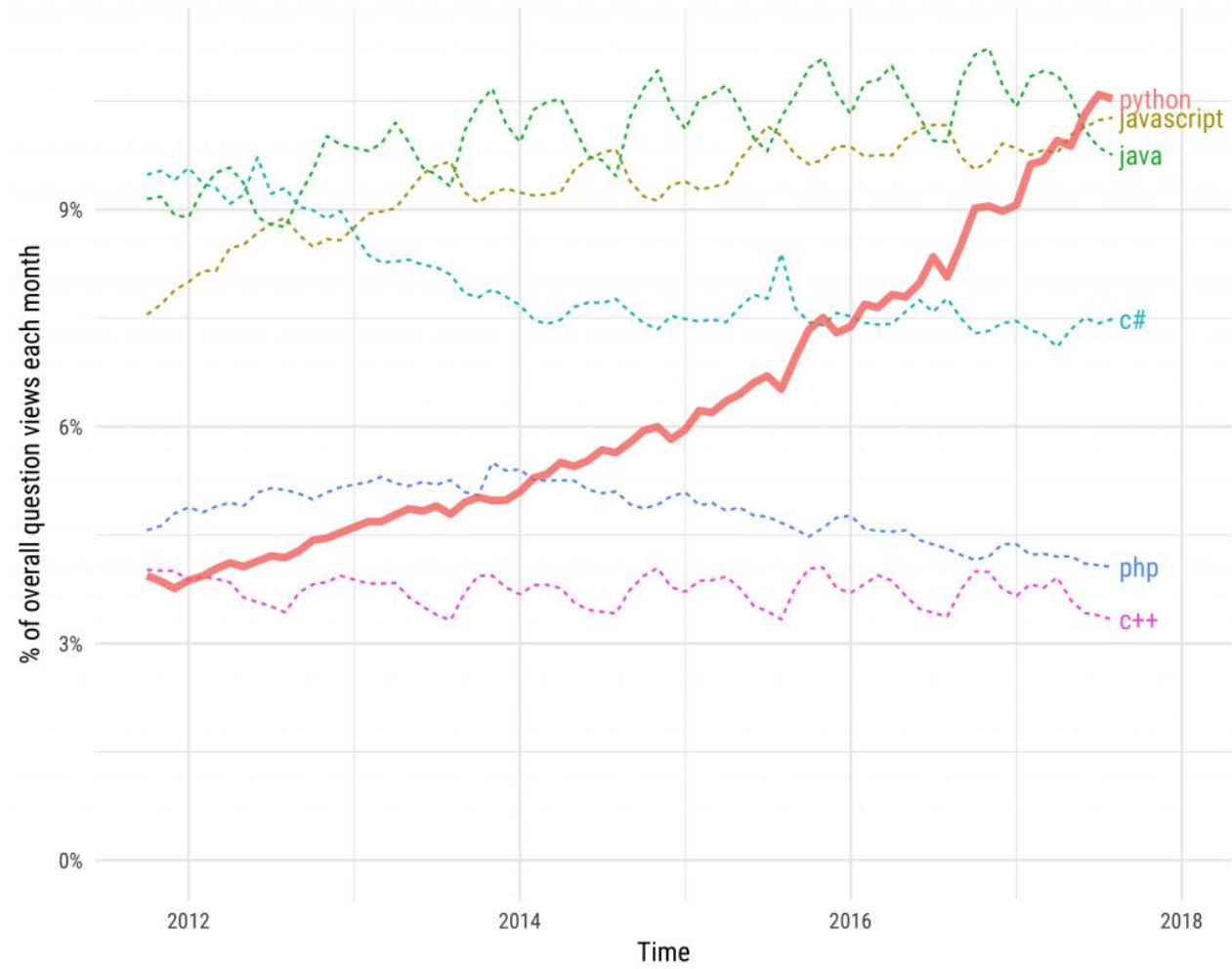


Getting started?
pick python

Kaggle 80%-90% python






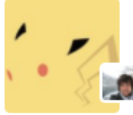
Growth of major programming languages

Based on Stack Overflow question views in World Bank high-income countries



How to get started

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

Featured	All	Mine	Upvoted	<input type="text" value="Search datasets"/>	
714		IMDB 5000 Movie Dataset 5000+ movie data scraped from IMDB website chuansun76 · updated a year ago · film	44,515 downloads 80 comments		
656		European Soccer Database 25k+ matches, players & teams attributes for European Professional Football Hugo Mathien · updated 10 months ago · association football, europe	32,766 downloads 96 comments		
621		Credit Card Fraud Detection Anonymized credit card transactions labeled as fraudulent or genuine Andrea · updated 10 months ago · crime, finance	31,774 downloads 66 comments		
540		Human Resources Analytics Why are our best and most experienced employees leaving prematurely? Iudoben · updated 9 months ago · employment	30,206 downloads 91 comments		
375		Iris Species Classify iris plants into three species in this classic dataset UCI Machine Learning · updated a year ago · botany	16,358 downloads 89 comments		
282		Pokemon with stats 721 Pokemon with stats and types Alberto Barradas · updated a year ago · popular culture, games and toys, video games	11,487 downloads 37 comments		

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

Code

This script has been released under the [Apache 2.0](#) open source license.

[Download Code](#)

```

1 # -*- coding: utf-8 -*-
2 """
3 @author: Faron
4 """
5 import pandas as pd
6 import numpy as np
7 import xgboost as xgb
8
9 DATA_DIR = "../input"
10
11 ID_COLUMN = 'Id'
12 TARGET_COLUMN = 'Response'
13
14 SEED = 0
15 CHUNKSIZE = 50000
16 NROWS = 250000
17
18 TRAIN_NUMERIC = "{0}/train_numeri"
19 TRAIN_DATE = "{0}/train_date.csv"
20
21 TEST_NUMERIC = "{0}/test_numeri"
22 TEST_DATE = "{0}/test_date.csv"
23
24 FILENAME = "etemelhoods"
25
26 train = pd.read_csv(TRAIN_NUMERIC)
27 test = pd.read_csv(TEST_NUMERIC)
28
29 train["StartTime"] = -1
30 test["StartTime"] = -1
31
32
33 nrows = 0
34 for tr, te in zip(pd.read_csv(
35     chunksize=CHUNKSIZE)):
36     feats = np.setdiff1d(tr.columns,
37                          test.columns)
38     stime_tr = tr[feats].min()
39     stime_te = te[feats].min()
40     train.loc[train.Id.isin(tr.Id),
41               test.Id.isin(te.Id)] = stime_tr
42
43     nrows += CHUNKSIZE
44     if nrows >= NROWS:
45         break
46
47 ntrain = train.shape[0]
48 train_test = pd.concat([train, test]).reset_index(drop=True).reset_index(drop=False)
49
50 train_test['0_\'\'\'\'_1'] = train_test[ID_COLUMN].diff().fillna(9999999).astype(int)
51 train_test['0_\'\'\'\'_2'] = train_test[ID_COLUMN].iloc[:-1].diff().fillna(9999999).astype(int)
52
53 train_test = train_test.sort_values(by=['StartTime', 'Id'], ascending=True)
54
55 train_test['0_\'\'\'\'_3'] = train_test[ID_COLUMN].diff().fillna(9999999).astype(int)
56 train_test['0_\'\'\'\'_4'] = train_test[ID_COLUMN].iloc[:-1].diff().fillna(9999999).astype(int)
57
58 train_test = train_test.sort_values(by=['index']).drop(['index'], axis=1)
59 train = train_test.iloc[:ntrain, :]
60
61 features = np.setdiff1d(list(train.columns), [TARGET_COLUMN, ID_COLUMN])
62
63 y = train.Response.ravel()
64 train = np.array(train[features])
65
66 print('train: {0}'.format(train.shape))
67 prior = np.sum(y) / (1.*len(y))
68
69
70 xgb_params = {
71     'seed': 0,
72     'colsample_bytree': 0.7,
73     'silent': 1,
74     'subsample': 0.7,
75     'learning_rate': 0.1,
76     'objective': 'binary:logistic',
77     'max_depth': 4,
78     'num_parallel_tree': 1,
79     'min_child_weight': 2,
80     'eval_metric': 'auc',
81     'base_score': prior
82 }
83
84 dtrain = xgb.DMatrix(train, label=y)
85 res = xgb.cv(xgb_params, dtrain, num_boost_round=10, nfold=4, seed=0, stratified=True,
86             early_stopping_rounds=1, verbose_eval=1, show_stdv=True)
87
88 cv_mean = res.iloc[-1, 0]
89 cv_std = res.iloc[-1, 1]
90
91
92 print('CV-Mean: {0}+{1}'.format(cv_mean, cv_std))

```

[show less](#)

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

```

hdtrain=hdtrain[product_cat==tolower(product_cat)]
hdtrain[,c("Jaccard_ST_PT")] := calcJacCoef(search_term,product_title),by = 1:nrow(hdtrain))

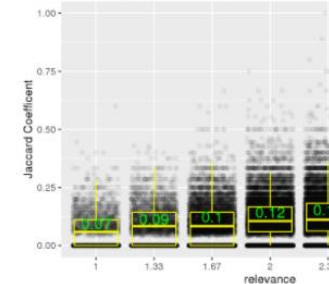
detach("package:sets",unload=TRUE)
#use search() to check if unloaded

#Create averages for annotating the plot
avgJC<-round(mean(hdtrain[,c("Jaccard_ST_PT")],2)
avgJCGroup<-hdtrain %>% group_by(relevance) %>% summarise(groupMean=round(mean(c("Jaccard_ST_PT"),2))
names(avgJCGroup)=c("relevance","grpMean")

hdtrain %>%
  ggplot(aes(x=factor(relevance),y=c("Jaccard_ST_PT"))) +
  geom_jitter(alpha=0.05) +
  geom_boxplot(color = "yellow", outlier.colour = NA, fill = NA)+
  geom_text(data = avgJCGroup, aes(x = factor(relevance), y = grpMean, label = grpMean), size = 5, vjust = 0,colour="green")+
  labs(title="Average Jaccard Coefficient by Relevance Score",x="relevance",y="Jaccard Coefficient")

```

Average Jaccard Coefficient by Relevance Score



The name=mfg brand name appears to be the standard for brand attributes. For color the information is spread across multiple fields. Create two flags in the Train indicating if a product_uid has brand and color attribute information.

```

#First change hdAtt to lower
hdAtt<-hdAtt[,name:=tolower(name)]

hdtrain<-hdtrain[product_uid %in% unique(hdAtt[name %like% "color",product_uid]),c.hasColorAtt:= "Yes"]
hdtrain<-hdtrain[is.na(c("hasColorAtt")),c.hasColorAtt:= "No"]

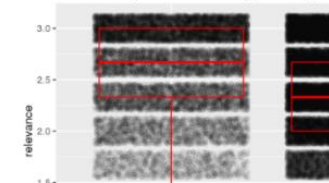
hdtrain<-hdtrain[product_uid %in% unique(hdAtt[name == "mfg brand name",product_uid]),c.hasBrandAtt:= "Yes"]
hdtrain<-hdtrain[is.na(c("hasBrandAtt")),c.hasBrandAtt:= "No"]

```

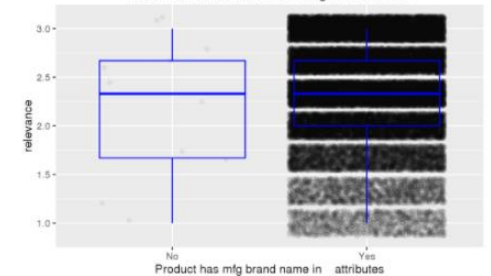
Bring in attributes.csv

There are 2044648 rows in the attributes.csv file. These relate to 86263 unique products by those products in the attributes and those that are not.

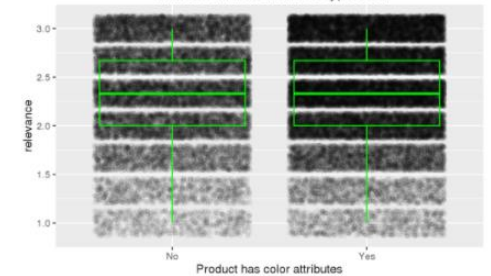
Compare relevance by product_uid



Compare relevance by product_uid in attributes file and has Attribute.name==mfg brand name



Compare relevance by product_uid in attributes file and has Attribute.name of type color



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

202 topics

Subscribe

Sort by

Most Votes

All

Mine

Upvoted

Topics

Search topics

104

Production Map

JohnM a year ago

last comment by r3m 1y ago

20

65

Expeditive exploration+models on data

Laurae a year ago

last comment by Alchemist 1y ago

25

53

1st place solution

beluga 10 months ago

last comment by BmvsPrasad 5mo ago

20

37

Load train_numeric in R (under 6GB RAM, 30 min -> 10s)

Laurae a year ago

last comment by j|=random==j|=f... 10mo ago

18

33

Solution Sharing

BreakfastPirate 10 months ago

19

XGBoost simple starter AUC 0.712

Haozhen a year ago

last comment by Haozhen 1y ago

10

32

#3 place solution

Darragh 10 months ago

19

4th place solution

sonndi 10 months ago

last comment by CPMP 10mo ago

3

30

8th place solution (te

Laurae 10 months ago

19

Request regarding consecutive/duplicate rows

BreakfastPirate 10 months ago

last comment by Banana Kitty 10mo ago

3

28

Manufacturing Floor'

colemaclean a year ago

16

Turn categorical into numeric

CPMP a year ago

last comment by r3m 1y ago

22

25

Useful Linux csv pre-

leochu a year ago

15

Initial Exploration and using missing values

JoeWarren a year ago

last comment by Xavier Van Ausloos 1y ago

3

24

Numeric Features: 2-

Laurae a year ago

15

Symposium Winners

RumiGhosh 10 months ago

last comment by Rich Jolly 10mo ago

6

23

Optimising probabilit

anokas a year ago

14

F# script: LB 0.43 using < 4GB RAM with F# XGBoost

xi 10 months ago

last comment by xi 10mo ago

52

20

How I'm doing cross-

BreakfastPirate a year ago

13

Failures

CPMP 10 months ago

last comment by CPMP 10mo ago

0

13

Datasets' size uncompressed (14.3GB)

Laurae a year ago

last comment by Sivakumar D 1y ago

21

12

Production Lines' Network

Arti163pl a year ago

last comment by gingerman | Chris 1y ago

6

11

Some basic statistics on features

Kaige Yang a year ago

last comment by Kaige Yang 1y ago

2

10

"internal failures"

inversion a year ago

last comment by Grzegorz Sionkow... 1y ago

1

10

Beat the benchmark

Konrad Banachewicz a year ago

last comment by Konrad Banachew... 1y ago

3

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

#	Δ 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

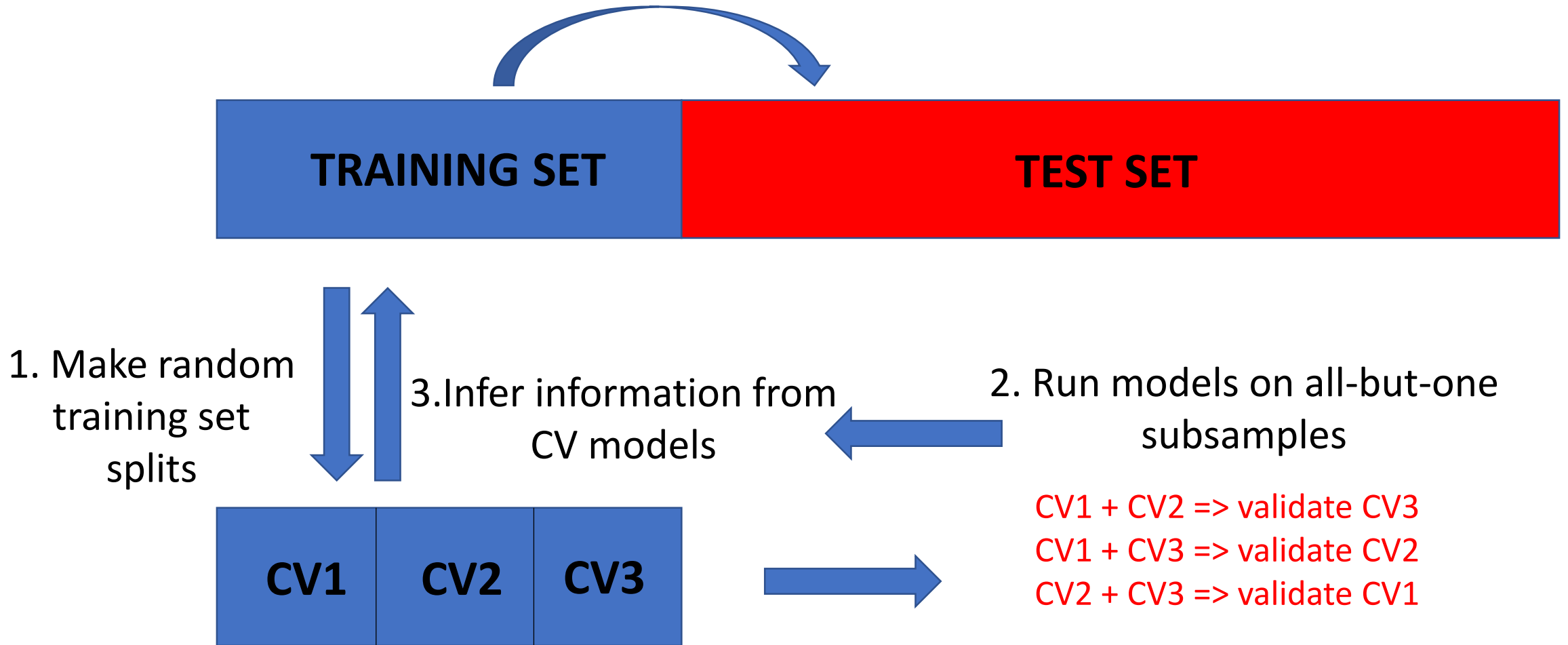
2. Make predictions on **test samples**



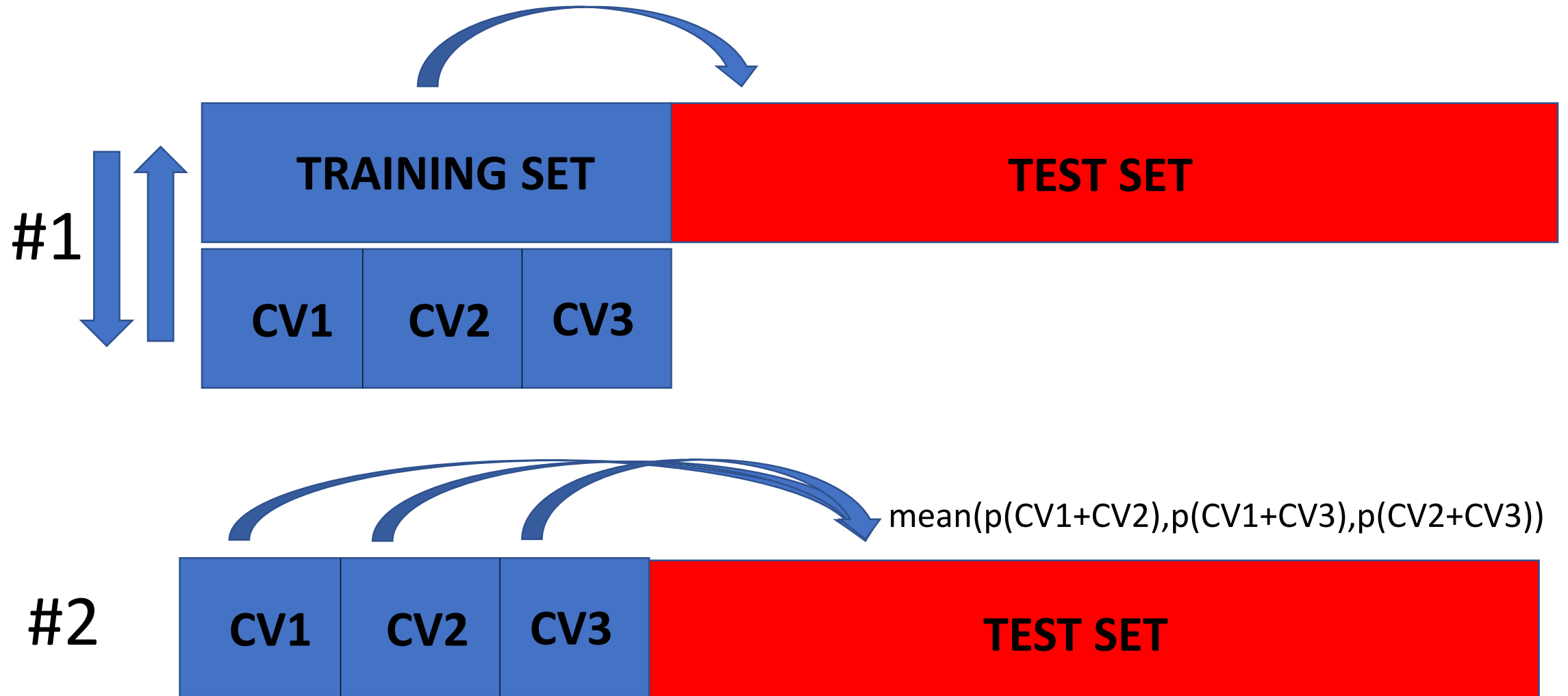
3. Submit model predictions on **“unseen”** data and get quick response in public leaderboard standings

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



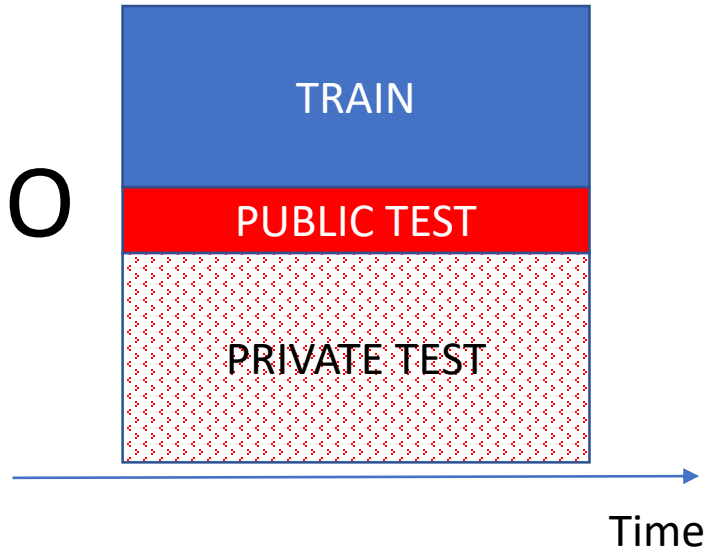
Model	Public test score	Private test score
Number trees 200	0.812	0.815
Number trees 300	0.814	0.819
Number trees 400	0.816	0.820
Number trees 500	0.814	0.821
Number trees 450	0.817	0.819
Number trees 430	0.818	0.816
... many attempts later ...	Maximum public LB score	Seems good right? – often NO!
Number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

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

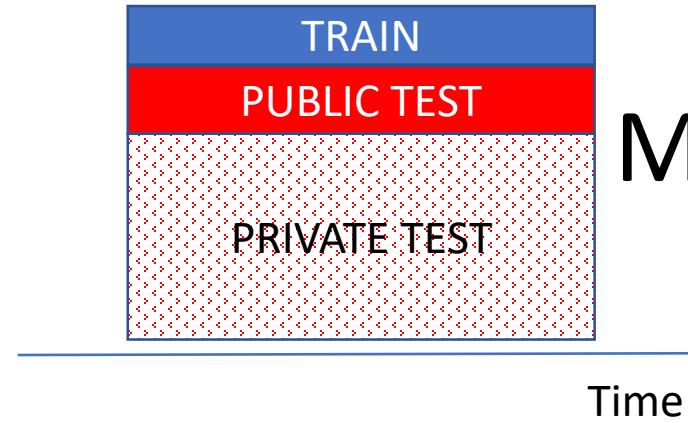
Rule of thumb – the less submissions you make, less likely you have overfitted to LB solution!

Should I even trust public leaderboard?

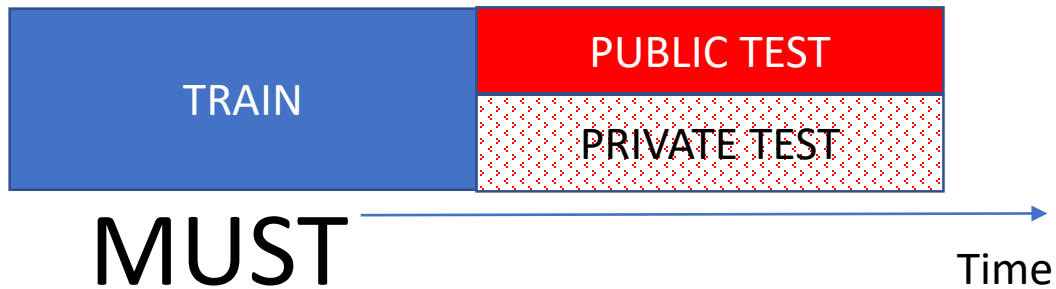
NO



MAYBE



MUST

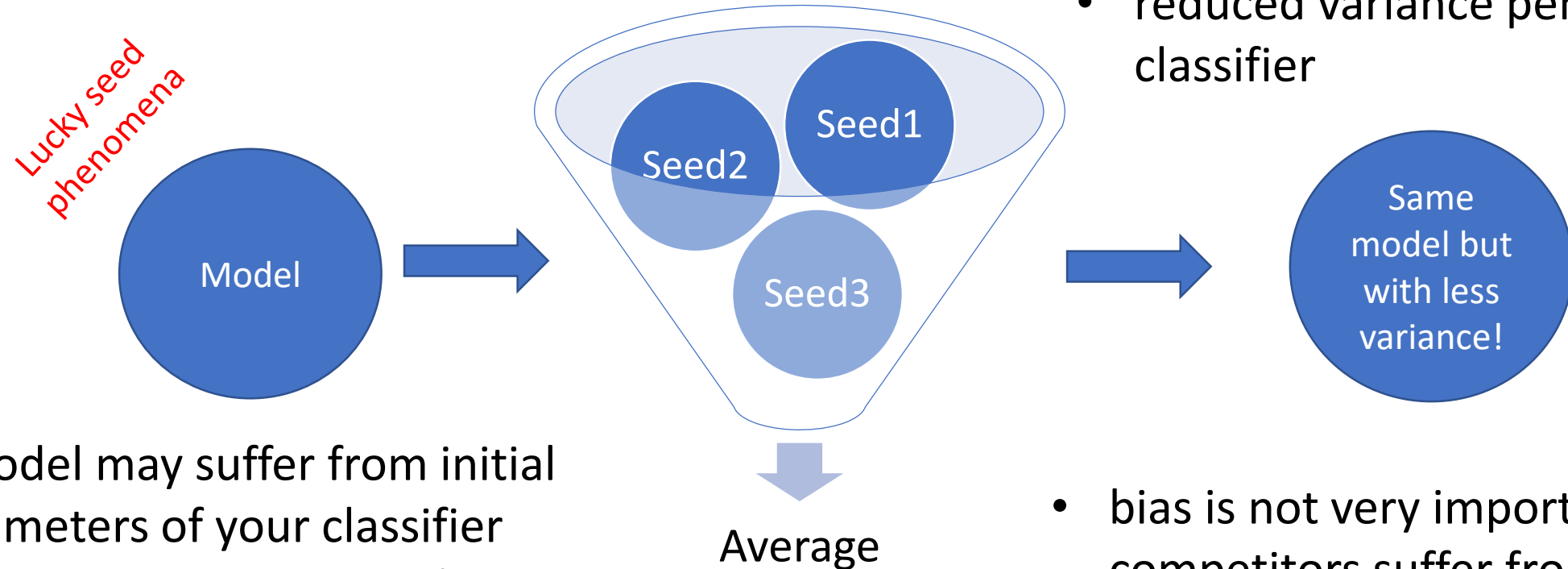


YES



Bootstrap aggregating – in short: bagging

Bagging \sim averaging similar models



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

Bias-variance tradeoff:

- reduced variance penalty of a classifier
- 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 => $20 \times 19 / 2$ 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	$0.666 + \text{rand}(0, \text{sigma})$
"BMW"	1	$0.333 + \text{rand}(0, \text{sigma})$
"BMW"	1	$0.333 + \text{rand}(0, \text{sigma})$
"BMW"	0	$0.666 + \text{rand}(0, \text{sigma})$
"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

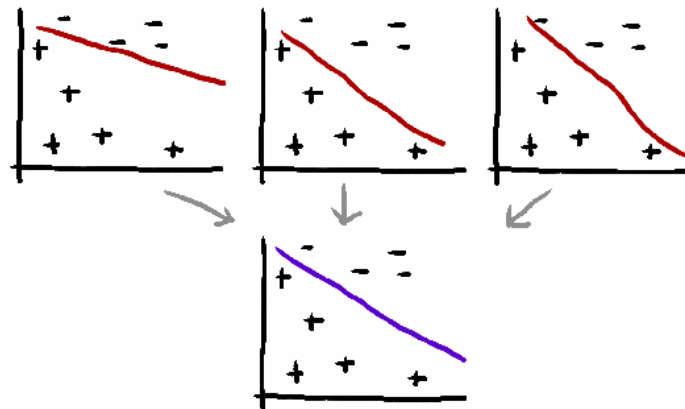
10% - luck

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-Classifer 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: X_1, X_2, \dots, X_N

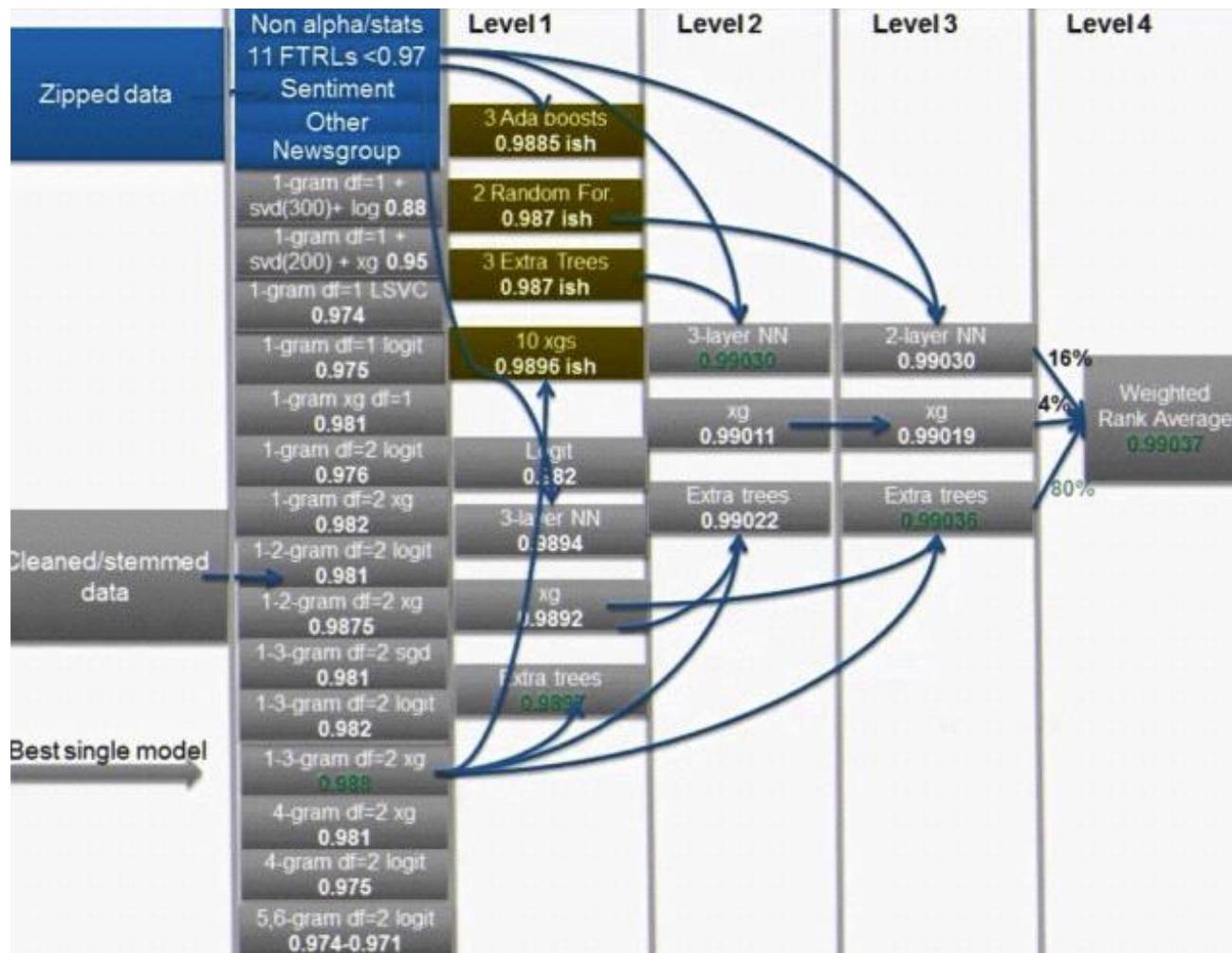
We want to make a single prediction using weighted average:

$$B_1 * X_1 + B_2 * X_2 + B_3 * X_3 + \dots + B_N * X_N$$

Solve the problem using CV predictions with optimization algorithms

optim($B_1 * X_1 + B_2 * X_2 + B_3 * X_3 + \dots + B_N * X_N$) with starting weights $B_i = 1/N$

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:

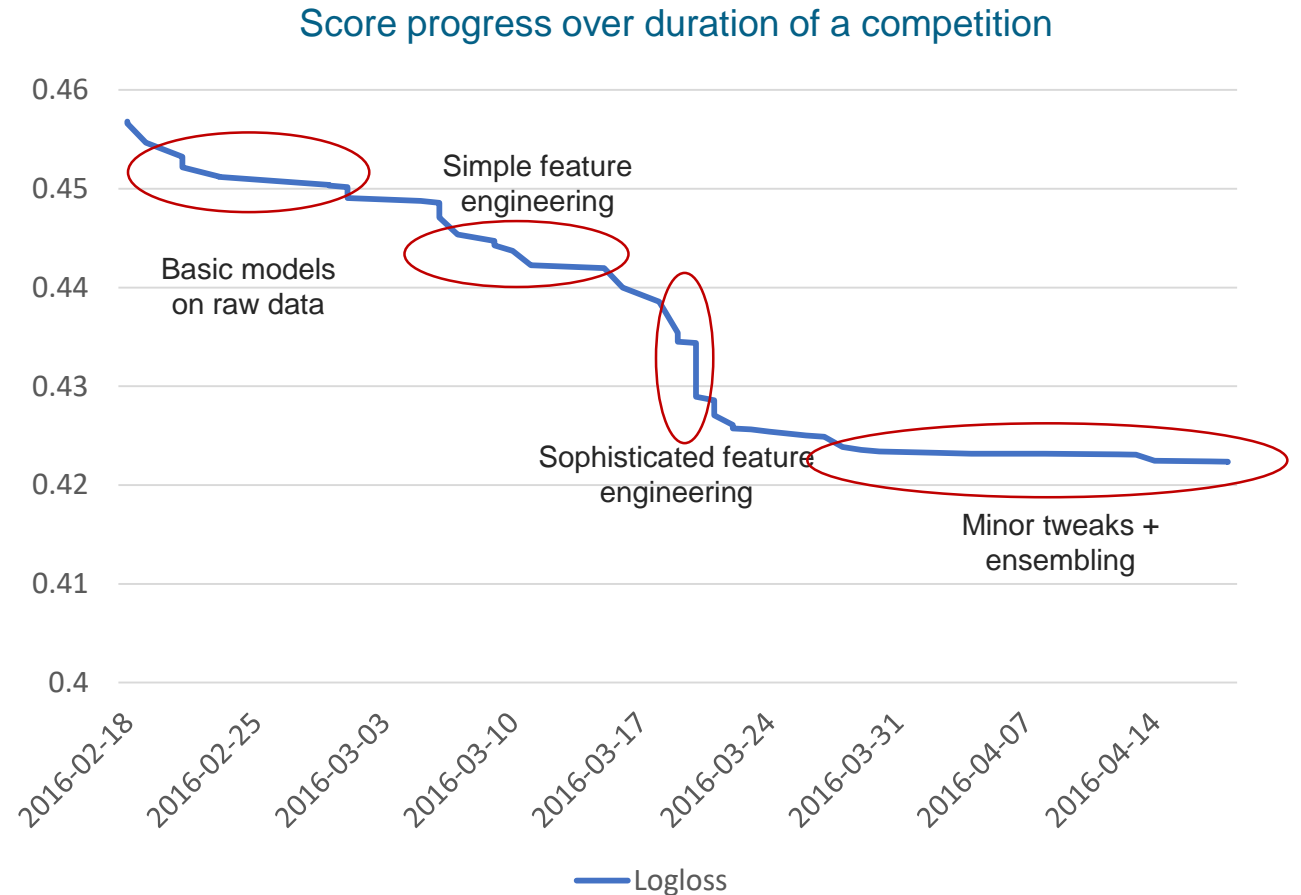
- Start with a few well-performing models' ensemble
- Loop through each other model in a library and add to current ensemble
- Determine best performing ensemble configuration
- Repeat until metric converged

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 ($A, B \Rightarrow "AB"$), 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



Upload

Upload dicom/jpg/png file

Browse... DF9D65DF_crop.jpg

Upload complete

Download report

No Finding



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