1

(http://

winners

intervie

1st-

place-

mad-

profess

Dato Winners' Interview: 1st place, Mad Professors

Kaggle Team (http://blog.kaggle.com/author/kaggleteam/) | 12.03.2015

This contest was organized by Dato (https://dato.com/) (from GraphLab Create (https://dato.com/products/create/)).

The task of the Truly Native? (https://www.kaggle.com/c/dato-native) contest was:

Given the HTML of ~337k websites served to users of StumbleUpon, identify the paid content disguised as real content.

#	Δrank	Team Name ‡ model uploaded * in the money	Score
1	↑1	Mad Professors ‡ *	0.99037
2	↓1	mortehu	0.99017
3	—	bibaze	0.98916

(http://blog.kaggle.com/wp-content/uploads/2015/12/dato1.png)

The Basics

What was your background prior to entering this challenge?

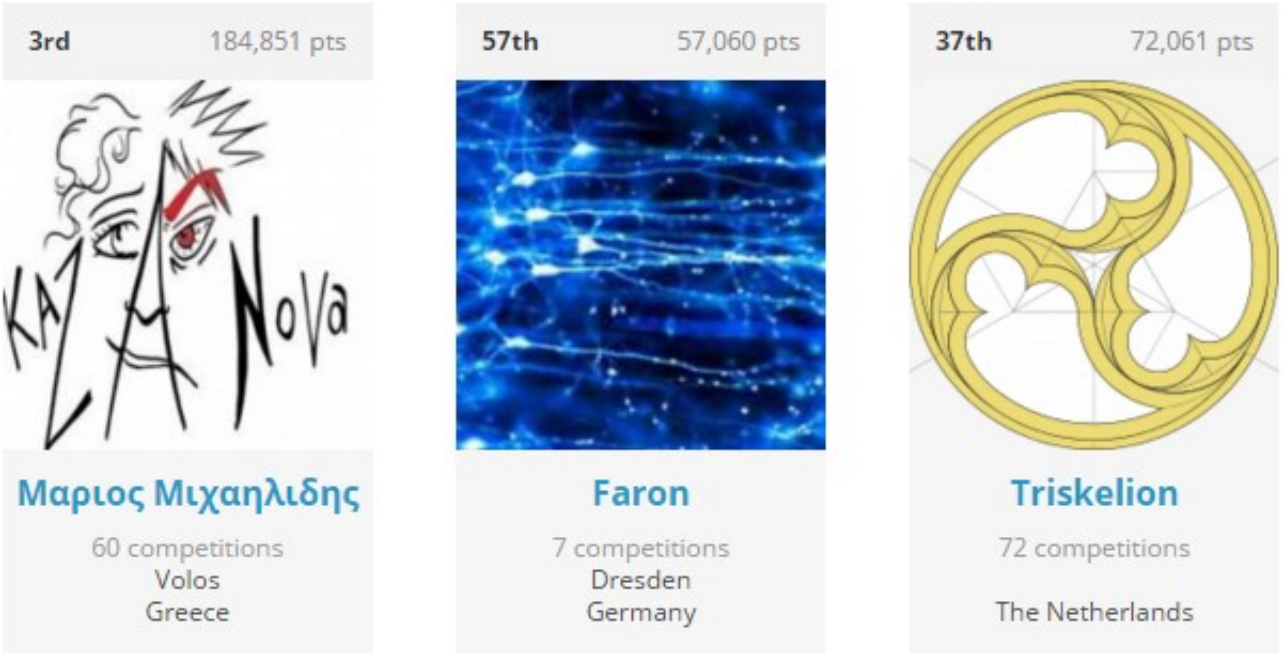
Marios Michailidis: I am a PhD. student (on improving recommender systems) and a sr. personalization data scientist at dunnhumby (https://www.dunnhumby.com/).

Mathias Müller: I have a Master's in computer science (focus areas cognitive robotics and AI) and work as a machine learning engineer at FSD (https://www.fsd-web.de/index.php/en/).

HJ van Veen: I study machine learning and artificial intelligence. I work in R&D for a Dutch healthcare IT startup and write for MLWave (http://mlwave.com).

How did you get started with Kaggle?

Marios Michailidis: I wanted a new challenge and learn from the best (http://blog.kaggle.com/2015/05/07/profiling-top-kagglers-kazanovacurrently-2-in-the-world/).
Mathias Müller: I found Kaggle after doing a TopCoder (https://www.topcoder.com/) Marathon Match.
HJ van Veen: I was looking for cool challenges and read a post by Rob Renaud (https://github.com/rrenaud).



(http://blog.kaggle.com/wp-content/uploads/2015/12/team.jpg)

Let's Get Technical

During the competition we tried a wide variety of models and approaches.

Once the fog of war cleared these essentials remained:

1. A deep XGBoost on text with tokenizer, tfidf-vectorizer, cleaning, stemming and n-grams,

2. A weighted rank average of multi-layer meta-model networks (StackNet).

The XGBoost model got us to top 10. The meta-modelling then got us to the first position.

What worked

Meta-modelling. Stacked generalization in a multi-layered fashion. As described in the Stacked Generalization (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.56.1533&rep=rep1&type=pdf) paper, the output of a stacker model can serve as the input for yet another stacker model. We did this for 3 levels as it kept increasing our AUC score. The first level predictions are also fed to the highest (3rd) level of stacking (together with a subset of raw features).

Cross-validation. With the large dataset we were able to keep stacking out of fold predictions and improve the AUC score. 5-fold CV for every model was very close (and always a little below) our public leaderboard score. We went up one position on the private leaderboard.

Cleaning. Basic lowercasing, removing double spaces, removing non-alphanumeric characters, repairing documents where the characters are separated by 3 spaces.

Ngrams, chargrams, and skipgrams. 1-3-grams in combination with cleaning captures dates, url's, and natural language indicative of native advertising.

Stemming. Both stemming with the Snowball stemmer (http://www.nltk.org/_modules/nltk/stem/snowball.html) (NLTK (http://www.nltk.org/)) and the Mineiro stemmer (http://blogs.technet.com/b/machinelearning/archive/2014/09/24/online-learning-and-sub-linear-debugging.aspx) worked for detecting the root of tokens or to lessen noise.

Tf-idf. Term frequency can prevent a bias to longer documents. Inverse document frequency can prevent a bias to common tokens. tfidf (http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html) outperformed tf, which outperformed binary frequency.

Small and **large memory**. We had access to machines ranging from 2-core 4GB laptops to 30-core 256GB servers. Both types of hardware were able to contribute.

Non-alphanumeric features. Building on the forum post by David Shinn (https://www.kaggle.com/c/dato-native/forums/t/16626/beat-the-benchmark-0-90388-with-simple-model) we created count features for every document (number of dots, spaces, tabs, URL's, script tags, commas etc.). These counts were fed to the highest levels of stackers as raw features.

XGBoost. As the winner of an increasing amount of Kaggle competitions, XGBoost (https://github.com/dmlc/xgboost) showed us again to be a great all-round algorithm worth having in your toolbox. With a max_depth of over 30 XGBoost was allowed to build deep trees with the text tokens.

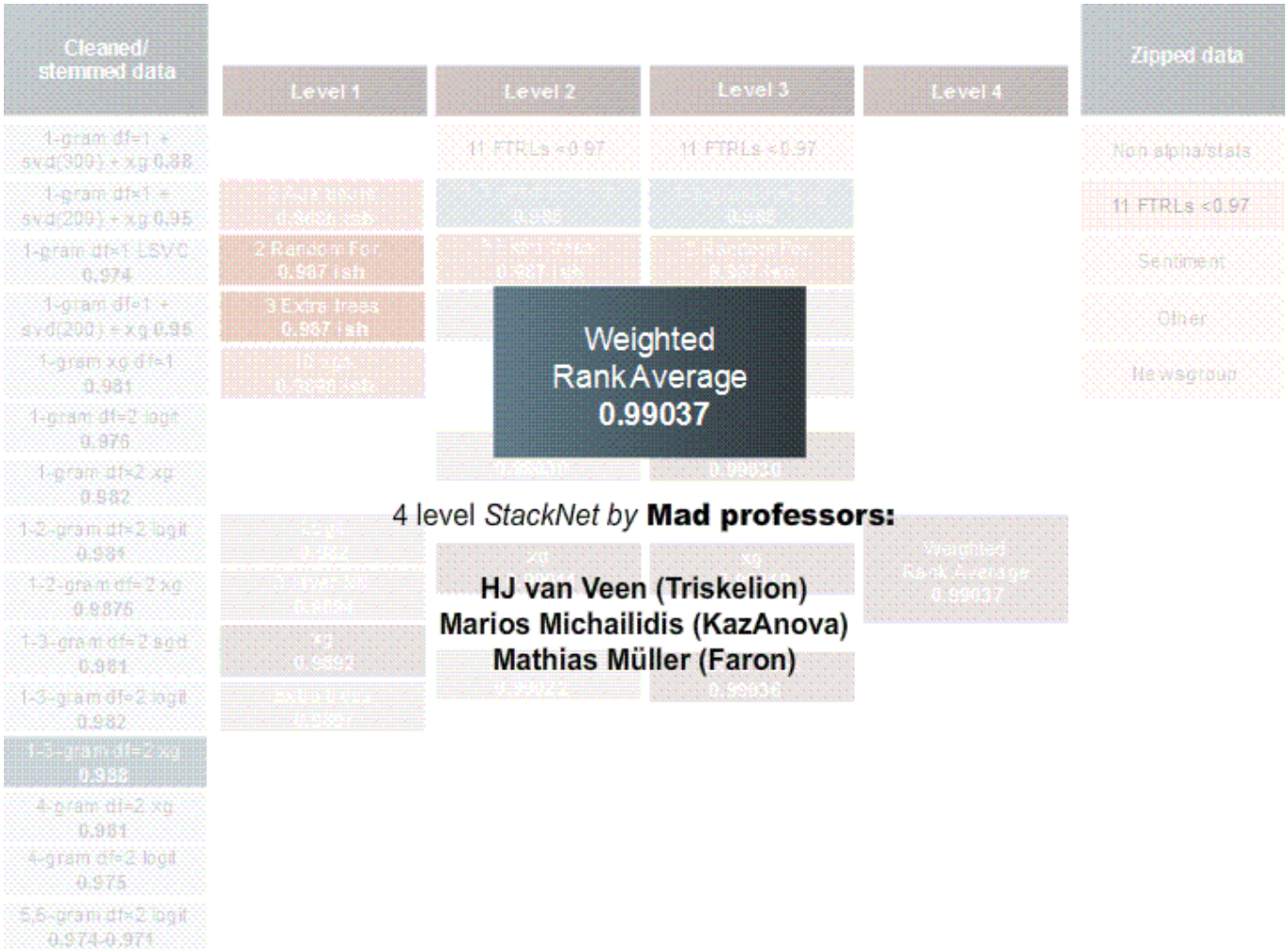
Artificial Neural Nets. We further dipped our toes in the lake of deep learning with single-layer, dual-layer and three-layer neural nets. The shallow nets, like Perceptron were trained on the raw features. The multi-layer nets (Theano (http://deeplearning.net/software/theano/), Lasagne (https://github.com/Lasagne/Lasagne) via NoLearn (https://github.com/dnouri/nolearn)) were used for stacking.

Extremely Randomized Trees. ExtraTreesClassifier (http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html) from Scikit-learn was used in all levels of stacking. Especially in the higher levels it proved to be a good algorithm (http://link.springer.com/article/10.1007%2Fs10994-006-6226-1) with the best variance and bias reduction.

LibLinear. Regularized Logistic Regression with LibLinear (https://www.csie.ntu.edu.tw/~cjlin/liblinear/) (via Scikit-learn) was used in the first layer of generalization as a "grunt" linear algorithm. It did not beat XGBoost in accuracy, but it did so in speed.

Sub-linear debugging. Parameter tuning, feature extraction and model selection was done with sublinear debugging (http://www.machinedlearnings.com/2013/06/productivity-is-about-not-waiting.html). For online learning algorithms we looked at progressive validation loss, and for all models we looked at the first round of out-of-fold-guessing.

Patience. The larger deep models took days to train. We resisted early ensembling and kept focus on creating out-of-fold predictions.



Animation by Nadja Rößger. Click here (http://i.imgur.com/GfWipUH.gif) to replay

What more or less worked

Online learning. For online learning we experimented with the tinrtgu's FTRL (https://www.kaggle.com/c/avazu-ctr-prediction/forums/t/10927/beat-the-benchmark-with-less-than-1mb-of-memory) and SGD (https://www.kaggle.com/c/criteo-display-ad-challenge/forums/t/10322/beat-the-benchmark-with-less-then-200mb-of-memory/53674) scripts and Vowpal Wabbit (http://hunch.net/~vw/). The FTRL script is able to get to ~97 using multiple passes/epochs as shown by Subhajit Mandal (https://www.kaggle.com/c/dato-native/forums/t/17009/congratulations-to-winners/96189#post96189).

AdaBoost. In the second generalization layer we added AdaBoosting (http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html) with decision tree stumps. Random Forests did a little better.

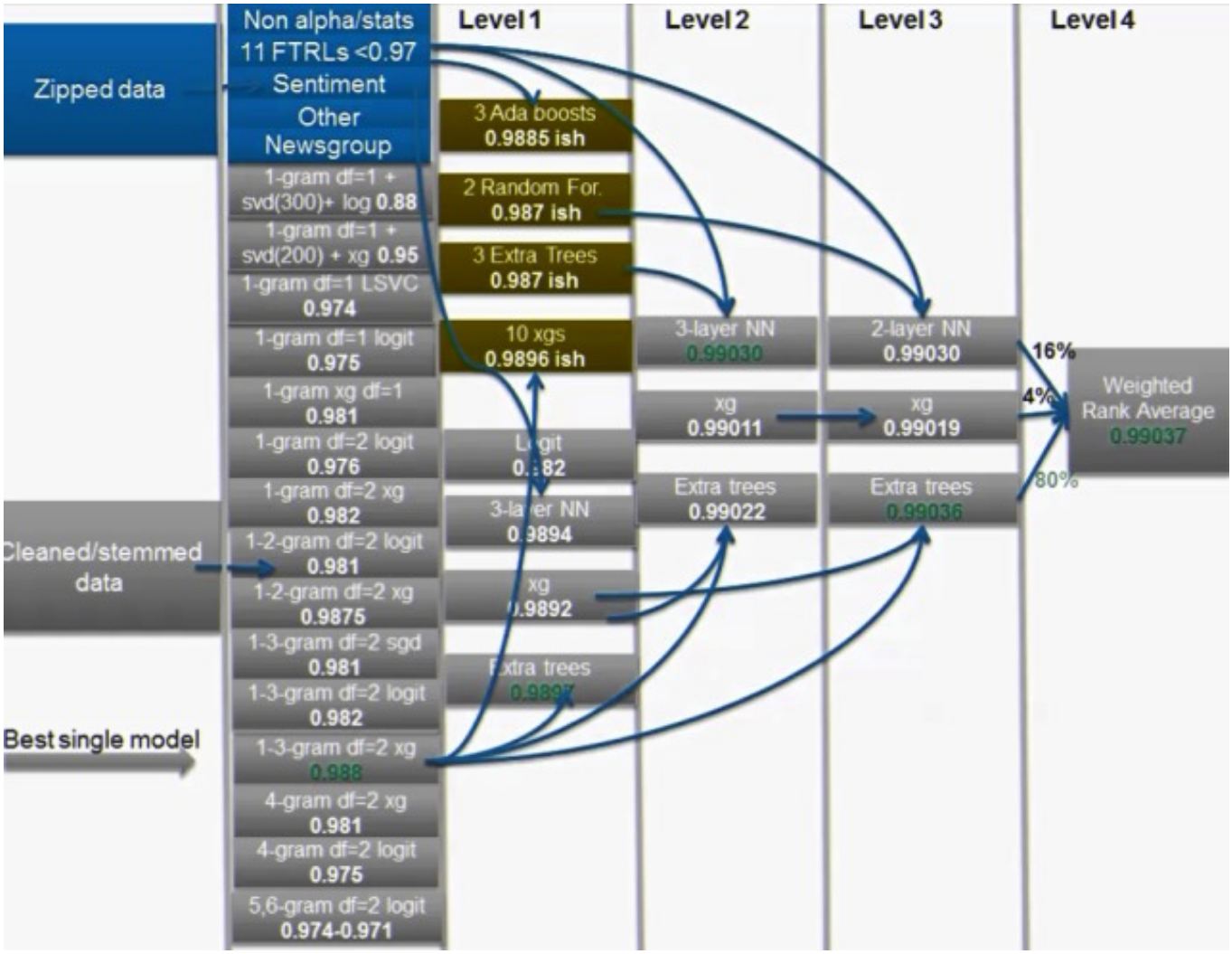
Random Forests. Also from the second generalization layer, and also slightly beaten by Extremely Randomized Trees.

Latent Semantic Analysis. We used TF-IDF followed by truncatedSVD (http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html) to generate a new feature set.

Models trained on external data. We trained models on both the IMDB movie review dataset (http://dl.acm.org/citation.cfm?id=2002491) (from the Kaggle Word2Vec tutorial (https://www.kaggle.com/c/word2vec-nlp-tutorial)) and on the 20 newsgroups (http://qwone.com/~jason/20Newsgroups/) datasets. These models generated predictions for both the train and test set.

What did not make it

- Vowpal Wabbit. Models trained before the reset were not updated, as we were further along in the modeling process by then.
- Matlab NN. We had some troubles with instability/overfit using Matlab NN's for stacking.
- Normalized Compression Distance. Though promising for creating informative features this technique was left out. (code (https://github.com/MLWave/normalized-compression-neighbors))
- Approximate IDF-weighing. Using a count-min sketch we generated approximate inverse document frequency for online learning.
- Naive Bayes. Using the same count-min sketch we generated approximate Naive Bayes vectors as used in NBSVM (https://github.com/mesnilgr/nbsvm). We did not want to add information from the entire train set for out-of-fold predictors.
- We never tried LDA, word vectors, POS-tagging or RNN's.



(http://blog.kaggle.com/wp-content/uploads/2015/12/models1.jpg)

Different models' performance maps

What did not work

Fast & simple solution. Our entire solution takes 4 weeks to run on ~55 cores and may take, at peak, around 128GB of memory. Mortehu (http://blog.kaggle.com/2015/10/30/dato-winners-interview-2nd-place-mortehu/) definitely won, in this regard, with his simpler production-friendly models (https://github.com/mortehu/text-classifier).

Headers. There was a brief conflict in the beginning of the competition about the "correct" way to store and share out-of-fold predictions. This spawned a meme than ran for the entirety of the contest. Let's just say that it doesn't matter how you store these (.npy, .txt, .csv, with/without headers, with/without ID's, ordered by ID or sample submission), you should agree on one, and only one way to do this.

```
 Triskelion: I put the oof's in the svn in .csv format, with
             headers (file,prediction), just like you asked
Michailidis: ...
     Mathias: :)
```

Blacklists. Before the restart simple "blacklists" seemed to work a bit. After the reset this effect was gone. All URL's on page were parsed and URL's that appeared only within positively labelled documents were added to the blacklist.

Restart. Not all modelling techniques, ideas and vectorizing approaches survived the restart.

	A	B	C	D	E	F	G
1	Interaction	Gain	FScore	wFScore	Average wFScore	Average Gain	Expected Gain
2	ET|NN|RF	8643,78906	14	1,8244240573	0,1303160041	617,4135042857	2053,5520050028
3	ET|NN|train_3gram_xgb	6150,27789	12	1,6534659895	0,1377888325	512,5231575	1005,160132734
4	ET|RF|train_3gram_xgb	4955,3814	4	0,9063621442	0,2265905361	1238,84535	3462,1790338967
5	NN|RF|train_3gram_xgb	4590,07026	20	3,089436659	0,154471833	229,503513	502,5007130517
6	ET|RF|RF	4447,5906	1	0,108986777	0,108986777	4447,5906	484,7285647742
7	ET|NN|pred11	3850,44126	7	1,6455307391	0,2350758199	550,0630371429	1399,4551499371
8	NN|pred11|train_3gram_xgb	3816,422984	25	7,4984922375	0,2999396895	152,65691936	1400,7665705894
9	ET|pred11|train_4gram_logit	3053,340665	9	0,9509670061	0,1056630007	339,2600738889	663,2769315677
10	ET|pred11|pred11	2851,9602	5	0,7159508621	0,1431901724	570,39204	895,1420212349
11	ET|NN|SGD	2767,79973	6	1,5664318124	0,2610719687	461,299955	891,0012386754
12	ET|pred11|train_3gram_xgb	2663,453436	19	4,841847313	0,2548340691	140,1817597895	823,7985994551
13	ET|NN|pred10	2651,00366	6	0,9827700977	0,1637950163	441,8339433333	390,0298844204
14	pred11|RF|train_3gram_xgb	2624,37092	12	3,4269720017	0,2855810001	218,6975766667	1378,8935778957
15	NN|pred11|RF	2327,87673	11	3,5190168818	0,3199106256	211,6251572727	742,6123284163
16	NN|pred11|pred11	2124,73724	10	1,9383164662	0,1938316466	212,473724	608,3670915329
17	ET|train_3gram_xgb|train_3gram_xgb	2075,6807	5	0,426057836	0,0852115672	415,13614	132,1124494593
18	NN|RF|train_4gram_logit	1621,71886	10	1,1678524864	0,1167852486	162,171886	436,4242787133
19	NN|pred7|RF	1619,889034	8	0,7106771	0,0888346375	202,48612925	151,8309218165
20	AdaBoost|ET|RF	1605,8833	7	0,1348139637	0,0192591377	229,4119	49,7055686035
21	ET|NN|NN	1526,921452	4	0,3043123205	0,0760780801	381,730363	129,3066903729
22	NN|NN|RF	1390,17673	2	0,4253418237	0,2126709119	695,088365	373,1355854657
23	NN|pred5|RF	1258,91564	8	0,668185246	0,0835231558	157,364455	388,2560009057

(http://blog.kaggle.com/wp-content/uploads/2015/12/xgbfi-large.png)

Ranking feature and model interactions with XGBfi (https://github.com/far0n/xgbfi). (click to enlarge)

Futuristic

Team discussion often ventured beyond standard meta-modelling. Interesting questions and ideas popped up:

- stacking features: Use the leaf ID's from XGB as features. Also try to use the residual (Y_real-Y_pred). See Jeong-Yoon Lee presentation (https://www.youtube.com/watch?v=ClAZQI_B4t8).
- multi-layer stacking: Just how deep can we go for a significant improvement?
- transfer learning: Use XGBoost as a feature-interaction extractor for linear models like those from VW. Train a shallow net on multi-class predictions from ensemble.
- net architecture: We did a form of human backprop. Can this be automated? Likewise, can we implement dropout?
- model averaging: Practical sharing of insights with geometric mean vs. mean vs. rank averaging.
- reusable holdout set: Can we fully eliminate the need for leaderboard feedback (http://googleresearch.blogspot.nl/2015/08/the-reusable-holdout-preserving.html) and rank

models on a ladder (http://arxiv.org/abs/1502.04585)?
- reusable trained models: Can we store models trained on different data and use their predictions for a new train and test set as features?
- reusable training information: Can we store and reuse the knowledge gained during training on data, models, pipelines, problems and generalization?
- self-guessing:

> **Rather one constructs a generalizer from scratch, requiring it to have zero cross-validation error. Instead of coming up with a set of generalizers and then observing their behavior, one takes the more enlightened approach of specifying the desired behavior first, and then solving the inverse problem of calculating the generalizer with that desired behavior. This approach is called "self-guessing". -Wolpert (1992) Stacked Generalization**

We Learned

- Basic NLP still works very well for this task.
- HTML is a messy "natural" language.
- Bigger datasets allow for bigger ensembles.
- Bigger ensembles require bigger hardware.
- Multi-layer stacking can keep squeezing out AUC.
- The higher the stacking level, the shallower the models need to be.
- Teaming up is a good thing to do.
- Diverse independent teams work well with an agile management style.
- Patience. Don't get lost in premature optimization with ensembles.
- Use solid development tools: code/model repositories, issue tracker, communication.
- You should always store out of fold predictions with headers and sample ID's.

Just For Fun

What kind of Kaggle competition would you like to see?

Marios Michailidis: Predicting (Kaggle) rank from (LinkedIn) profiles.
Mathias Müller: a ML corewars (http://www.corewars.org/).
HJ van Veen: Predicting pixel values for partly obscured images.

Mad Professors

Our team name is a nod to the Mad Professor -- a creative dub artist twisting hundreds of knobs to mix music -- and pays homage to a real driving force behind (academic) research: all teachers, assistants, and professors.

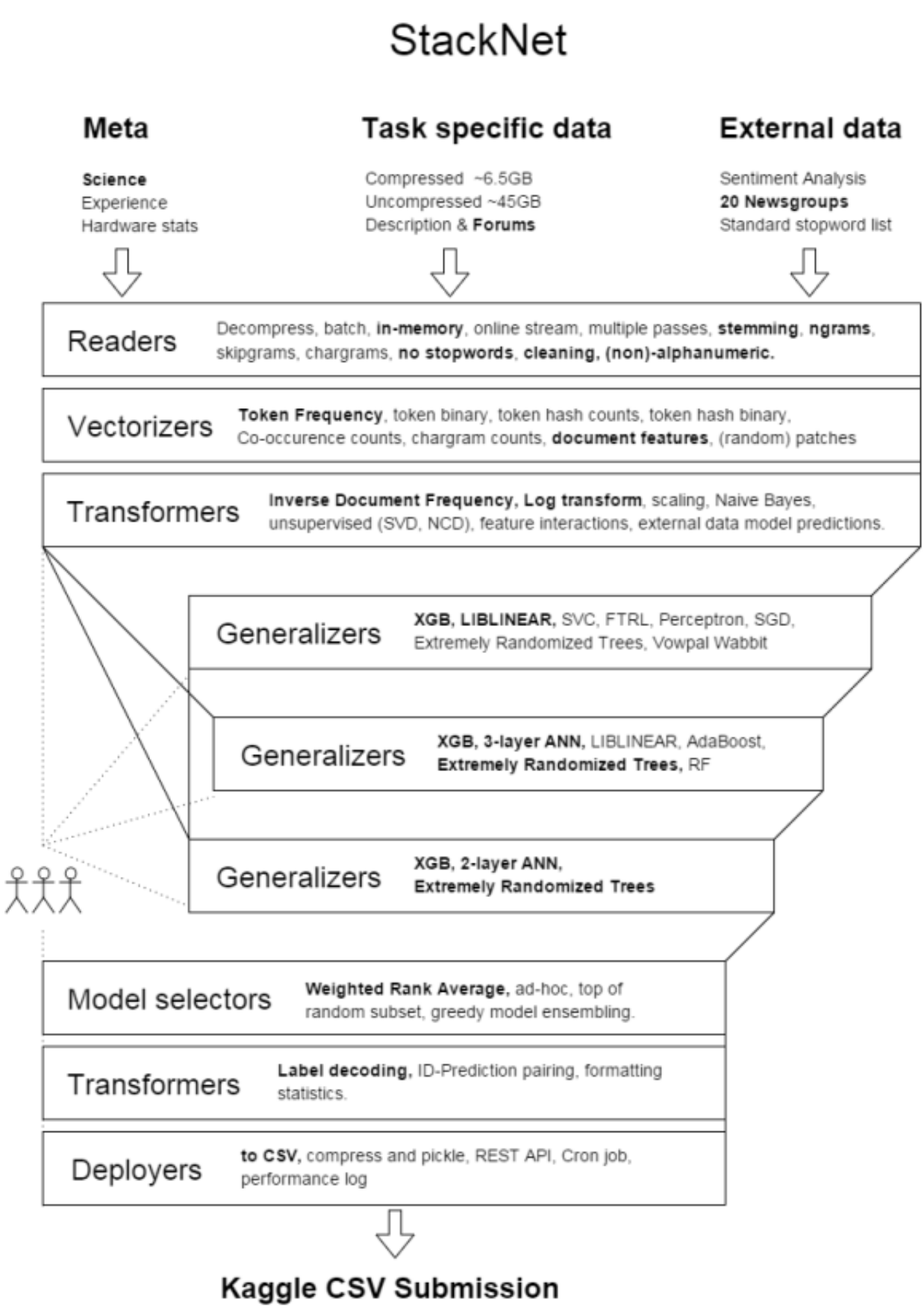
By name we would like to mention: Abu-Mostafa, Bishop, Caruana, Dean, Graham, Langford, Lecun, Hinton, Huttunen, Ng, Schmidhuber, Vitanyi, Wiering, Wolpert, Zajac.

Followed by a randomly shuffled list, (capped at 99), of people who have, directly or indirectly, contributed brain power to the computation of StackNet.

```
random.shuffle(l)
print l[:99]
```

Thanks

Thanks to the competitors for the challenge, Kaggle (https://www.kaggle.com/) for hosting, Dato (https://dato.com/) for organizing, and StumbleUpon (https://www.stumbleupon.com/) for providing the data. Thanks to the open source community and the research that makes it all possible.



(http://blog.kaggle.com/wp-content/uploads/2015/12/stacknet1.png)
(http://blog.kaggle.com/wp-content/uploads/2015/12/stacknet.png)

You know you like ExtraTreesClassifier, when you feed the output of a logistic regression algorithm A trained on a different dataset to an ExtraTreesClassifier B, its output to another ExtraTreesClassifier C, its output to another ExtraTreesClassifier D, its output to a weighted rank average E. Thanks Gilles Louppe and Geurts et al.!

Bios

(https://www.linkedin.com/in/mariosmichailidis) (http://blog.kaggle.com/wp-content/uploads/2015/12/mariosmichailidis.png)**Marios Michailidis (https://www.linkedin.com/in/mariosmichailidis)** (KazAnova (https://www.kaggle.com/kazanova)) is Senior Data Scientist in dunnhumby and part-time PhD in machine learning at University College London (UCL) with a focus on improving recommender systems. He has worked in both marketing and credit sectors in the UK Market and has led many analytics projects with various themes including: Acquisition, Retention, Uplift, fraud detection, portfolio optimization and more. In his spare time he has created KazAnova, a GUI for credit scoring 100% made in Java.

(https://de.linkedin.com/in/mathias-m%C3%BCller-69763b103) (http://blog.kaggle.com/wp-content/uploads/2015/12/mathiasmueller.png)**Mathias Müller (https://de.linkedin.com/in/mathias-m%C3%BCller-69763b103)** (Faron (https://www.kaggle.com/mmueller)) is a machine learning engineer for FSD Fahrzeugsystemdaten. He has a Master's in Computer Science from the Humboldt University. His thesis was about 'Bio-Inspired Visual Navigation of Flying Robots'.

(https://www.linkedin.com/in/hjvanveen) (http://blog.kaggle.com/wp-content/uploads/2015/12/hjvanveen.png)**HJ van Veen (https://www.linkedin.com/in/hjvanveen)** (Triskelion (https://www.kaggle.com/triskelion)) is a data engineer and researcher at Zorgon, a Dutch Healthcare & IT startup. He studies machine learning and artificial intelligence, and uses Kaggle to learn more about predictive modelling. He is also the author of MLWave.com.

DATO TRULY NATIVE? (HTTP://BLOG.KAGGLE.COM/TAG/DATO-TRULY-NATIVE/)

NATURAL LANGUAGE PROCESSING (HTTP://BLOG.KAGGLE.COM/TAG/NATURAL-LANGUAGE-PROCESSING/)

1 Comment No Free Hunch 1 Login

Recommend Share Sort by Best

Join the discussion…

KK • 2 months ago

Thanks for the article. Very insightful.

• Reply • Share ›

Subscribe Add Disqus to your site Add Disqus Add Privacy

(https://www.facebook.com/kaggle)

(https://twitter.com/kaggle)