

No Free Hunch (<http://blog.kaggle.com/>)



[\(HTTP://BLOG.KAGGLE.COM\)](http://blog.kaggle.com/) > STACKING MADE EASY: AN INTRODUCTION TO STACKNET BY COMPETITIONS GRANDMASTER MARIOS MICHAILIDIS (KAZANOVA)

→ ([HTTP://BLOG.KAGGLE.COM/2017/06/06/WEVE-PASSED-1-MILLION-MEMBERS/](http://blog.kaggle.com/2017/06/06/WEVE-PASSED-1-MILLION-MEMBERS/))

Stacking Made Easy: An Introduction to StackNet by Competitions Grandmaster Marios Michailidis (KazAnova)

Megan Risdal (<http://blog.kaggle.com/author/mrisdal/>) | 06.15.2017

4

(<http://kaggle.com/marios-michailidis/stacking-made-easy-an-introduction-to-stacknet-by-competitions-grandmaster-marios-michailidis-kazanov>)

[made-](#)

[easy-](#)

[an-](#)

[introdu-](#)

[to-](#)

[stackne](#)

[by-](#)

[competi](#)

[grandm](#)

[marios-](#)

[michaili](#)


[kazanov](#)



You've probably heard the adage "two heads are better than one." Well, it applies just as well to machine learning where the combination of diverse approaches leads to better results. And if you've followed Kaggle competitions, you probably also know that this approach, called *stacking*, has become a staple technique among top Kagglers.

In this interview, [Marios Michailidis \(AKA Competitions Grandmaster KazAnova on Kaggle\)](https://www.kaggle.com/kazanov) (<https://www.kaggle.com/kazanov>) gives an intuitive overview of stacking, including its rise in use on Kaggle, and how the resurgence of neural networks led to the genesis of his stacking library introduced here, StackNet. He shares how to make StackNet—a computational, scalable and analytical, meta-modeling framework—part of your toolkit and explains why machine learning practitioners shouldn't always shy away from complex solutions in their work.

Have questions about stacking, StackNet, or Marios' illustrious career on and off Kaggle? He's holding an AMA (Ask Me Anything) on the Kaggle Forums. (<https://www.kaggle.com/general/34802>)




Μαριος Μιχαηλιδης **KazAnova**

Data Scientist at H2O ai
Volos, Greece
Joined 4 years ago · last seen in the past day
<http://www.kazanovaforanalytics.com/>

Followers 103

Following 5



Competitions Grandmaster

[Home](#)
[Competitions \(94\)](#)
[Kernels \(5\)](#)
[Discussion \(471\)](#)
[Organizations \(1\)](#)
[Followers \(103\)](#)
[Contact User](#)
[Follow User](#)

Competitions Grandmaster

Current Rank

3

of 60,430

Highest Rank

1

24

23

21

Homesite Quote Conversion

1st

a year ago · Top 1%

of 1764

Truly Native?

1st

2 years ago · Top 1%

of 274

Acquire Valued Shoppers C...

1st

3 years ago · Top 1%

of 952

Kernels Contributor

Unranked

0

0

0

Xgboost python scores arou...

4

3 months ago

votes

Your Second Round vs the F...

3

a year ago

votes

enhanced

2

2 years ago

votes

Discussion Master

Rank

2

of 31,266

30

37

234

The 'Magic' (Leak) feature is...

221

2 months ago

votes

Score 0.53776 (or 0.52879) ...

146

3 months ago

votes

My Approach

92

2 months ago

votes

MARIOS (AKA KAZANOVA) ON KAGGLE ([HTTPS://WWW.KAGGLE.COM/KAZANOVA](https://www.kaggle.com/kazanova)).



Overview of Stacking and StackNet

Can you give a brief introduction to stacking and why it's important?

Stacking or *Stacked Generalization* is the process of combining various machine learning algorithms using holdout data. It is attributed to [Wolpert 1992](http://www.machine-learning.martinsewell.com/ensembles/stacking/Wolpert1992.pdf) (<http://www.machine-learning.martinsewell.com/ensembles/stacking/Wolpert1992.pdf>). It normally involves a four-stage process. Consider 3 datasets A, B, C. For A and B we know the ground truth (or in other words the target variable y). We can use stacking as follows:

1. We train various machine learning algorithms (regressors or classifiers) in dataset A
2. We make predictions for each one of the algorithms for datasets B and C and we create new datasets B1 and C1 that contain only these predictions. So if we ran 10 models then B1 and C1 have 10 columns each.
3. We train a new machine learning algorithm (often referred to as *Meta learner* or *Super learner*) using B1
4. We make predictions using the Meta learner on C1

Still confusing? Consider this animation:

A				
X0	x1	x2	xn	y
0.17	0.25	0.93	0.79	1
0.35	0.61	0.93	0.57	0
0.44	0.59	0.56	0.46	0
0.37	0.43	0.74	0.28	1
0.96	0.07	0.57	0.01	1

B				
X0	x1	x2	xn	y
0.89	0.72	0.50	0.66	0
0.58	0.71	0.92	0.27	1
0.10	0.35	0.27	0.37	0
0.47	0.68	0.30	0.98	0
0.39	0.53	0.59	0.18	1

C				
X0	x1	x2	xn	y
0.29	0.77	0.05	0.09	?
0.38	0.66	0.42	0.91	?
0.72	0.66	0.92	0.11	?
0.70	0.37	0.91	0.17	?
0.59	0.98	0.93	0.65	?

Train algorithm 0 on A and make predictions for B and C and save to B1, C1

Train algorithm 1 on A and make predictions for B and C and save to B1, C1

Stacking is a machine learning technique that combines the predictions of multiple models to improve performance.

B1		C1	
pred0	pred1	pred0	pred1
0.24	0.72	0.50	0.50
0.95	0.25	0.62	0.59
0.64	0.80	0.22	0.31
0.89	0.58	0.90	0.47
0.11	0.20	0.20	0.09

For a **large scale** implementation of stacking, one may further read or use [stacked ensembles](https://h2o-release.s3.amazonaws.com/h2o/rel-ueno/2/docs-website/h2o-docs/data-science/stacked-ensembles.html) (<https://h2o-release.s3.amazonaws.com/h2o/rel-ueno/2/docs-website/h2o-docs/data-science/stacked-ensembles.html>).

Stacking is important because (experimentally) it has been found to **improve performance** in various machine learning problems. I believe most winning solutions on Kaggle the past 4 years included some form of stacking.

Additionally, the advent of **increased computing power** and parallelism has made it possible to run many algorithms together. Most algorithms rely on certain parameters or assumptions to perform best, hence each one has advantages and disadvantages. Stacking is a mechanism that tries to leverage the benefits of each algorithm while disregarding (to some extent) or *correcting* for their disadvantages. In its most abstract form, stacking can be seen as a mechanism that *corrects the errors of your algorithms*.

Great, so what is StackNet?

StackNet is a computational, scalable and analytical, meta-modeling framework implemented in Java that resembles a feedforward neural network and uses Wolpert's stacked generalization on multiple levels to improve accuracy in machine learning predictive problems.

StackNet was created as part of my [PhD at UCL](http://www.cs.ucl.ac.uk/home/) (<http://www.cs.ucl.ac.uk/home/>) which was sponsored by [dunnhumby](https://www.dunnhumby.com/) (<https://www.dunnhumby.com/>). It can be downloaded from the [GitHub repo](https://github.com/kaz-Anova/StackNet) (<https://github.com/kaz-Anova/StackNet>):

The supervisors are:

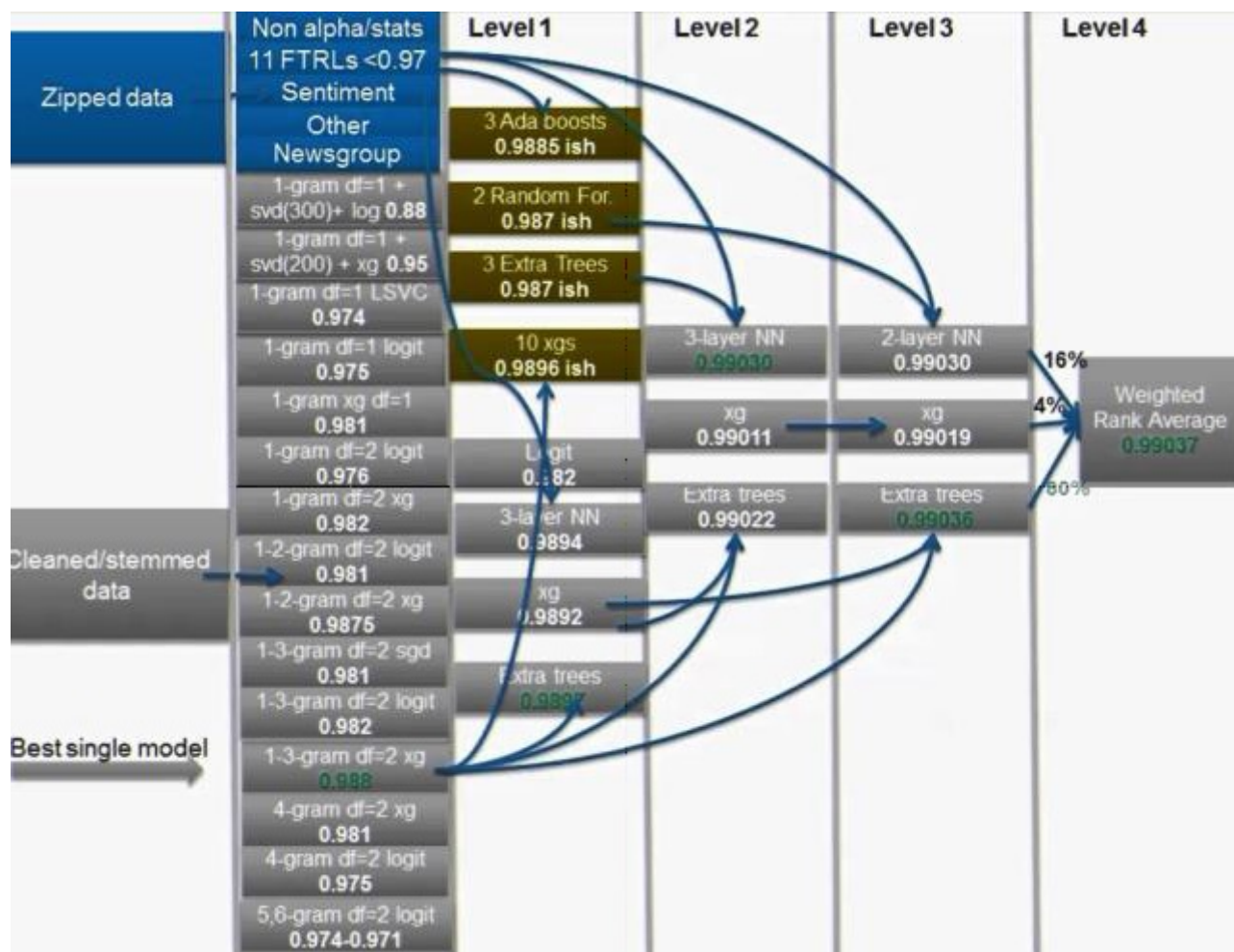
- [Giles Pavey](https://www.linkedin.com/in/giles-pavey-924426/) (<https://www.linkedin.com/in/giles-pavey-924426/>)

- [Prof. Philip Treleaven \(http://www0.cs.ucl.ac.uk/staff/P.Treleaven/\)](http://www0.cs.ucl.ac.uk/staff/P.Treleaven/)

Breaking down the definition of StackNet to its basic rudiments we have:

1. **Computational:** Because it involves heavy computing.
2. **Scalable:** Because many models can be run in parallel. More threads lead to faster results.
3. **Analytical:** Because it is heavily based on the principles of data analysis (or data science), especially when it comes to data preprocessing, cross validating, measuring performance through various metrics.
4. **Meta-modelling:** Because it uses the notion of meta learners. In other words, it uses predictions of some algorithms as features to other algorithms.
5. **Wolpert's stacked generalization:** Because the meta-learners are created using this technique of combining predictions in hold-out datasets.
6. **Feedforward neural network and multiple levels:** Because stacking is not limited to just the 4 stages mentioned in the stacking section, but have the ability to be repeated multiple times creating more datasets from predictions (like B2,C2...until Bn,Cn).
7. **Java:** Because the first implementation of StackNet is in the Java programming language.

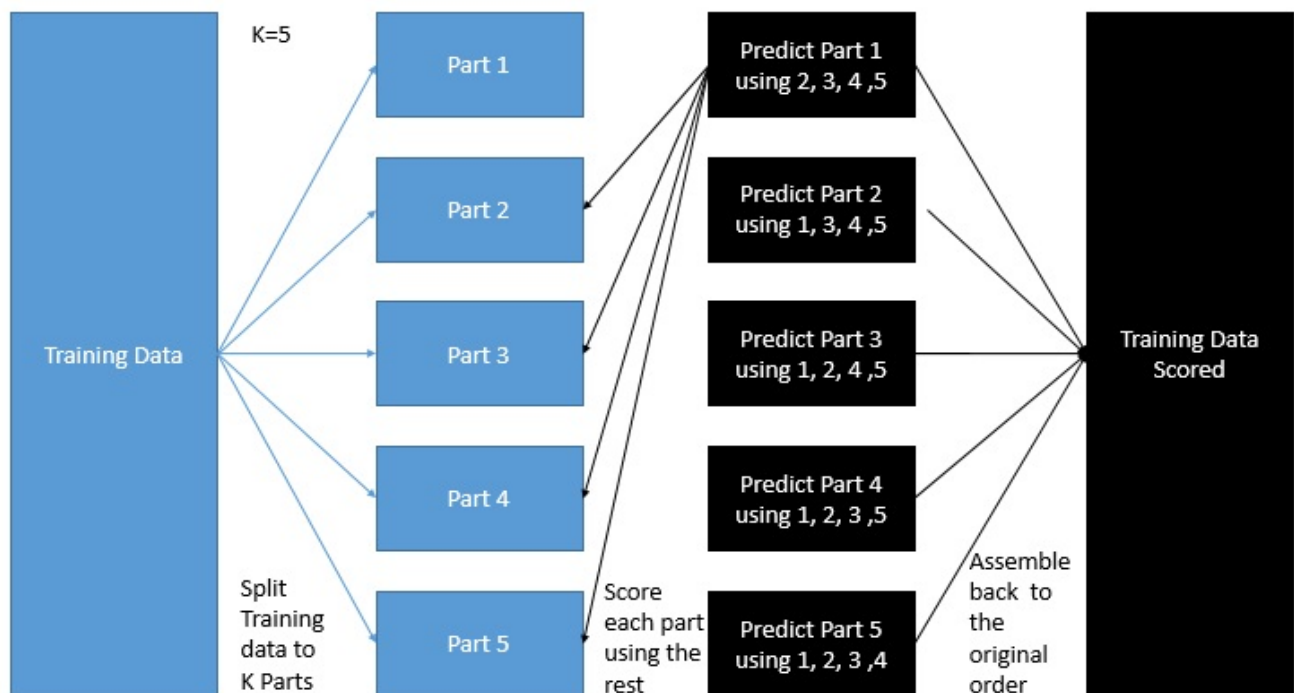
StackNet has (already) been used to win machine learning challenges. A typical implementation may be viewed in [the winning solution of the Truly Native Kaggle challenge \(http://blog.kaggle.com/2015/12/03/dato-winners-interview-1st-place-mad-professors/\)](http://blog.kaggle.com/2015/12/03/dato-winners-interview-1st-place-mad-professors/). In that challenge, the winning StackNet had 4 layers of meta (neuron) models to achieve the best score. The final architecture is illustrated below:



A typical neural network is commonly trained with a form of *back propagation*; however, stacked generalization—as stated before—requires a forward training methodology that splits the data **into 2 parts** (A and B)—one of which is used for training (A) and the other for predictions (B).

The reason this split is necessary is to **avoid over-fitting**. However, splitting the data in just 2 parts would mean that in each new layer in the second part needs to be further dichotomized or in general more datasets would be required (D,F...Z). This has the effect of increasing the bias of overfitting even more as each algorithm will have to be trained and validated on increasingly **less data**.

To overcome this drawback, the algorithm utilizes **a K-fold cross validation** (where **K** is a hyperparameter). This process implies that we split the data K times and run models to output predictions for each K part and then bring the K parts back together to the original order so that the output predictions can be used in later stages of the model. This process is illustrated below in figure



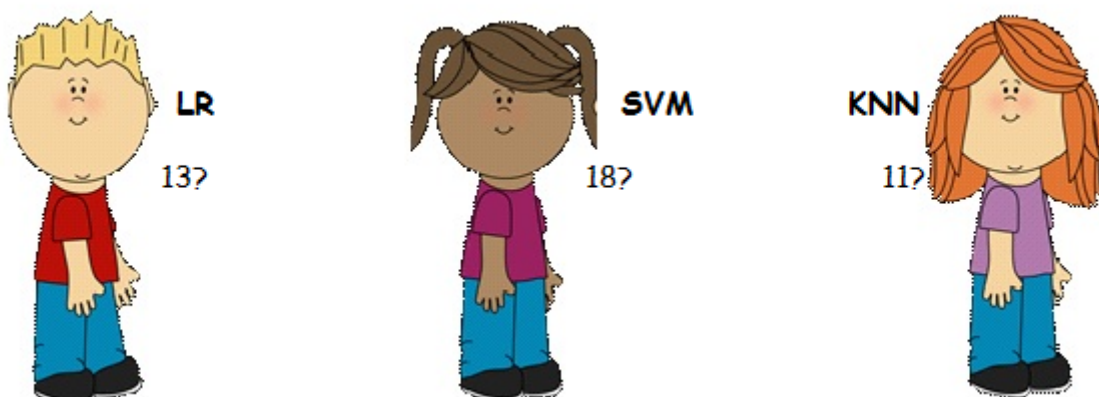
Optionally instead of predicting the test data at the same time for each K model, we may choose to train another algorithm on the whole training data (after K-fold is done). There is no reason to limit the ability of the model to learn using less than 100% of the training data since the output scoring is already unbiased.

“I still don’t get it–why does building so many models on different levels help? Besides they look ‘ugly’. I prefer the era when building one good model was enough.”

It is based on the principle that no model is perfect. Almost every time the models make mistakes. Plus, each model has different advantages and disadvantages and they tend to seize the data from different angles. Leveraging the uniqueness of each model is of the essence for building very predictive models.

I often like to explain stacking on multiple levels with the following (albeit simplistic) example:

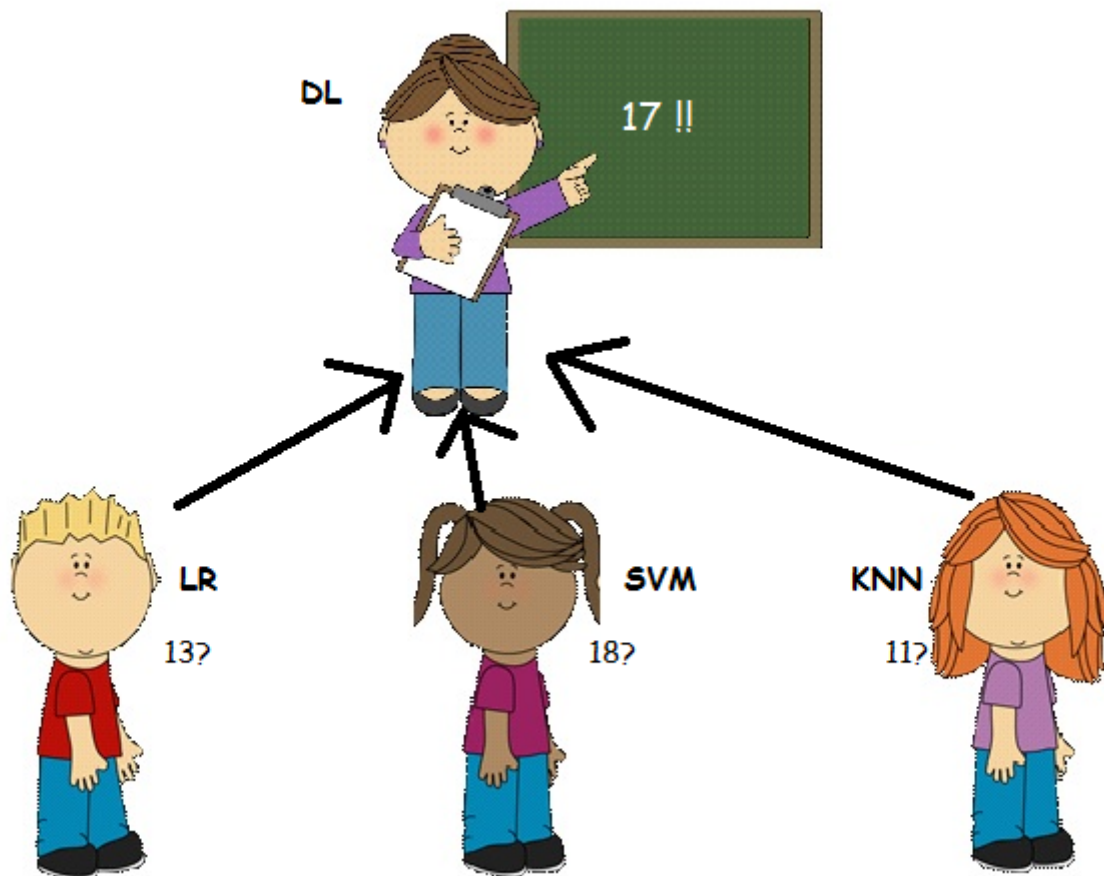
Let’s assume there are three students named **LR**, **SVM**, **KNN** and they argue about a physics question where they have different opinions of what the right answer might be:



They decide there is no way to convince one another about their case and they do the democratic thing via taking an **average** of their estimates which in this case is **14**. They used one of the simplest form of ensembling–AKA **model averaging**.

Their teacher, **Miss DL**–a maths teacher–bears witness to the argument the students are having and decides to help. She asks “what was the question?”, but the students refuse to tell her (because they know it is not in their benefit to give all the information, besides they think she might find silly they are arguing for such a trivial matter). However they do tell her that it is a physics related argument.

In this scenario, the teacher does not have access to the initial data as she does not know what the question was. However she does know the students very well–their **strengths** and **weakness** and she decides she can still help in solving this matter. Using historical information of how well the students have done in the past, plus the fact that she knows SVM loves physics and always does well in this subject (plus her father worked in a physics institute of excellence for young scientists), she thinks the most appropriate answer would be more like **17**.

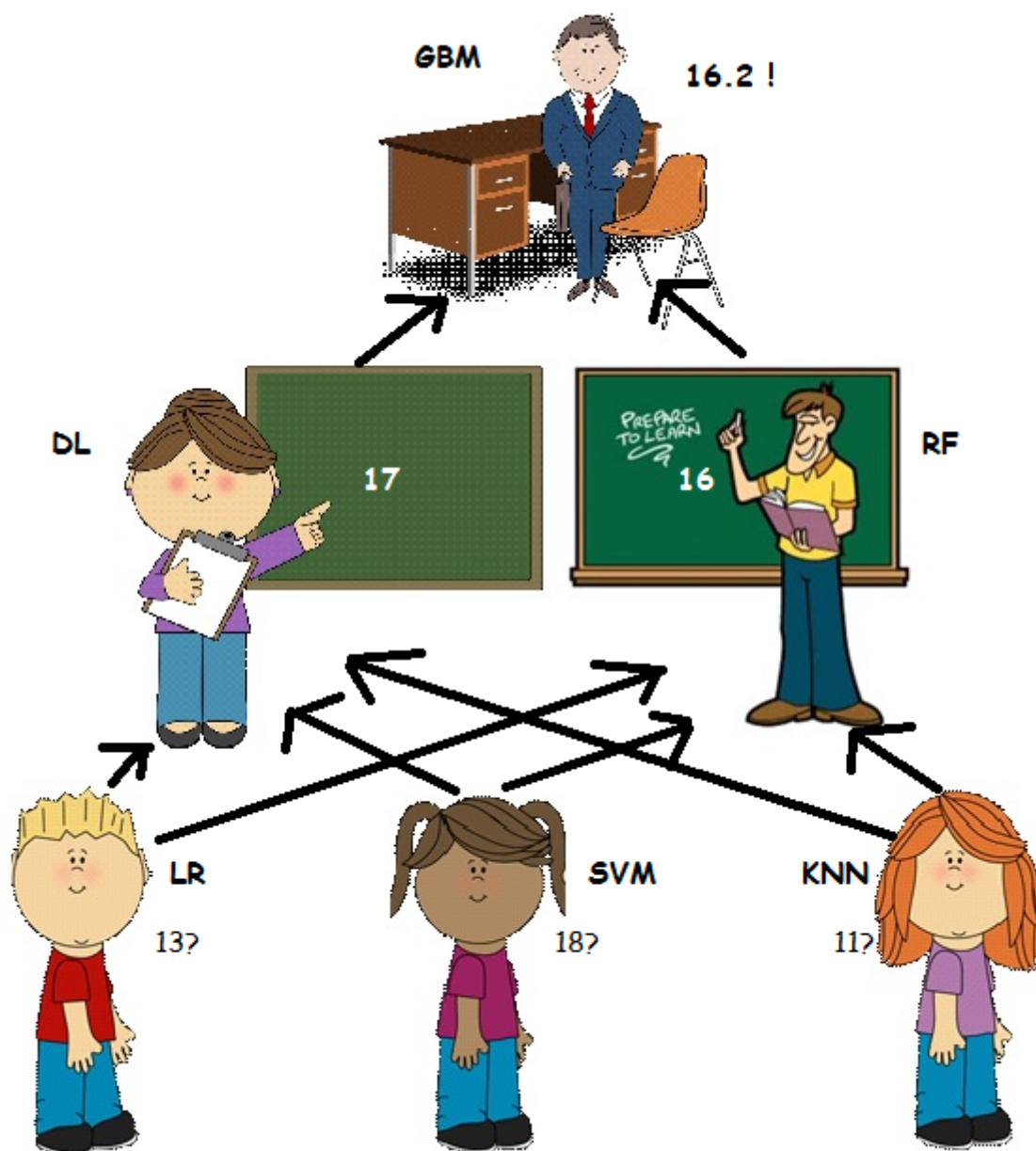


In this instance the teacher (DL) is a **meta learner**. She uses as input data the results of what other models (the students) are outputting. Then she combines it with historical information of how well the students have done in the past to get a better estimate (and help resolve the conflict).

However... **Mr RF**, a physics teacher has a slightly different opinion. He was there the whole time, but he waited until this moment to make his move! Mr RF has been teaching LR private lessons in physics lately to boost his grades (something that miss DL did not know) and he thinks LR's contribution to the final estimate should be bigger. Therefore he claims the right answer to be more like **16**!

In this case Mr RF is **also a Meta learner** and he processes the historical data with different logic–plus he has access to additional sources (or different historical information) than Miss DL.

This dispute can only be resolved if **Headmaster GBM** makes a decision! GBM does not know what the children have said, but knows his teachers quite well and he is more keen to trust his physics teacher (RF). He concludes the answer to be more like **16.2**.



In this scenario the headmaster is a **level 2 meta learner** or a meta learner of meta learners and via processing historical information of his teachers he may still be able to provide a better estimate than a simple average of their results.



The history of stacking and StackNet on Kaggle

What's the inspiration behind StackNet?

There were 4 main drives behind the idea of StackNet:

1. **To win.** When I first started competing on Kaggle, I was trying to come up with ideas to improve. I first tried something like stacked generalization in a [bird classification challenge in 2013](https://www.kaggle.com/c/multilabel-bird-species-classification-nips2013) (<https://www.kaggle.com/c/multilabel-bird-species-classification-nips2013>). This was my first top

10 result and I briefly [explained my approach there \(https://www.kaggle.com/c/multilabel-bird-species-classification-nips2013/discussion/6383\)](https://www.kaggle.com/c/multilabel-bird-species-classification-nips2013/discussion/6383), without really knowing that what I did was stacked-generalization—I just tried it out of intuition to get my best score. After this I learnt more about it and I tried to build a strategy to make the best of it—hence coming up with StackNet.

2. **To leverage my experience** in some areas. Before I started competing on kaggle, my hobby was to do predictive modelling in the credit sector. I had [built a tool \(http://www.kazanovaforanalytics.com/\)](http://www.kazanovaforanalytics.com/) that helps to build [credit scorecards \(https://en.wikipedia.org/wiki/Credit_scorecards\)](https://en.wikipedia.org/wiki/Credit_scorecards)—using various machine learning algorithms but with a focus on logistic regression and linear models. When I first joined kaggle, everything was about Random Forests and Gradient Boosting Machines and I quickly realised that my expertise was not enough to perform competitively in this environment. However I had spent quite a lot of time developing methodologies and processes that could make these models work well and I was constantly trying to find ways to include them into my pipeline. StackNet was the process that allowed me to find use for these weaker models as they still add information in this multi-modeling context.
3. The return of neural networks—AKA **deep learning**. The architecture of deep learning and the notion of being able to build deep models scalably, boosted the idea of using it in tandem with stacked generalization.
4. **To build something novel and give back to the community**. I have learnt a lot from open source material, from people sharing tips, tricks, codes, methodologies. I have learnt a lot from Kaggle. I have seen it many times, people building tools and sharing it back to the community making everyone better and I wanted to do a small part towards that direction. In a way, it is like paying my debt to my (anonymous) mentors—to your everyday, friendly neighborhood data scientists/coders/programmers that via sharing have made this world a better place.

How did your experiences competing on Kaggle shape the development of StackNet?

As I have already mentioned above, Kaggle was integral to developing StackNet as it was the product of trying to beat the state-of-the-art methodologies that are often applied on Kaggle among thousands of participants in order to win data science challenges. Apart from that I have learnt a lot from the participants (Kagglers) either from them posting in forums or kernels or via direct collaborations.

Also competing in 100+ diverse challenges in areas such as image classification, sound classification, retention, recommendations, credit, marketing, text, etc. meant building and experimenting with more than 100K machine learning models—Kaggle has provided a great data science playground to invent or improve techniques such as this. It won't be extreme to say that StackNet is in a way a *"Kaggle's baby"*!

Some people would claim that building so many or complex machine learning models is "a waste of resources" given that the chance of winning is low. I think they probably ignore the potential contribution to science and the personal development you experience as a data scientist via participating. Also what may seem complex today, may not be that complex tomorrow if there is a demand for it—see deep learning! Not to mention the apparent boost in connectivity with the

professional market plus all the fun that comes out via competing. Even my current role at [H2O](https://www.h2o.ai/) (<https://www.h2o.ai/>) was a product of developing StackNet and trying it on Kaggle, since the company already builds very innovative software (like [H2O-3](https://github.com/h2oai/h2o-3) (<https://github.com/h2oai/h2o-3>)) in the data science space and we got the chance to meet each other through this platform.



Let's get technical: Using StackNet

How to use StackNet

All the information required to run StackNet can be found in its [main GitHub repository](https://github.com/kaz-Anova/StackNet) (<https://github.com/kaz-Anova/StackNet>) along with [examples from Kaggle competitions](https://github.com/kaz-Anova/StackNet/tree/master/example) (<https://github.com/kaz-Anova/StackNet/tree/master/example>) on how to get competitive scores.

In a few simple steps, what you need to do to run StackNet is:

1. Clone the repository with `git clone https://github.com/kaz-Anova/StackNet.git` (<https://github.com/kaz-Anova/StackNet.git>) or just download the **StackNet.jar** from the repo which is already compiled.
 - Optionally if you want XGBoost or other external tools apart from the .jar you also need the **lib.zip** file and unzip it in the same directory where the .jar is.
2. Make sure you have **java 1.6** installed or higher.
3. Get a dataset and convert it to .libsvm format. You can see some examples of such format [here](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/) (<https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>), but in general it is [target_variable col0:value0 col1:value1 coln:valuen]
4. Consider the following command (in cmd) for training a regression problem. For simplicity all parameters have a new entry in the following list, but in reality they need to be space delimited as one long sequence:
 1. **Java -jar stacknet.jar train** : tells to execute the .jar file and train a model
 2. **task=regression** : we specify if we want regression or classification
 3. **sparse=true** : means our data is in sparse (libsvm) format
 4. **train_file=x.csv** : the name of the training file
 5. **test_file= X_test.csv** : the name of the test file to make predictions for
 6. **pred_file=pred.csv** : the name of the file containing the predictions for the test_file
 7. **params=params.txt** : A file containing the structure of the StackNet, which and how many algorithms to use

8. **metric**=rmse : the metric to validate performance of algorithms per fold

9. **folds**=5 : Number of K-folds

10. **seed** = 1 : random seed to replicate results

11. **threads**=3 : number of models to run in parallel . Ideally should not exceed cores for best performance.

12. **model**=model.mod :name of the model file. If we want to use it later on for other predictions, we could dump it.

The params file has the following structure. Each line is a model. When there is an empty line then any new algorithm is used in the next level:

```
LogisticRegression C:1 Type:Liblinear maxim_Iteration:100 scale:true verbose:false
RandomForestClassifier bootstrap:false estimators:100 threads:5
GradientBoostingForestClassifier estimators:100 shrinkage:0.05
LSVC Type:Liblinear threads:1 C:1.0 maxim_Iteration:100 seed:1
LibFmClassifier lfeatures:3 init_values:0.035 smooth:0.05 learn_rate:0.1
NaiveBayesClassifier usescale:true threads:1 Shrinkage:0.1 seed:1 verbose:false
XgboostRegressor booster:gbtree objective:reg:linear num_round:100 eta:0.015
RandomForestClassifier estimators=1000 rounding:3 threads:4 max_depth:6
```

Tip: To tune a single model, one may choose an algorithm for the first layer and a dummy one for the second layer. StackNet expects at least two algorithms, so with this format the user can visualize the performance of single algorithm inside the K-fold. For example, if I wanted to tune a Random Forest Classifier, I would put it in the first line (layer) and also put any model (let's say Logistic Regression) in the second layer and could break the process immediately after the first layer kfold is done:

```
RandomForestClassifier bootstrap:false estimators:100 threads:5
LogisticRegression verbose:false
```

All the [available algorithms and their advised tunable parameters \(https://github.com/kaz-Anova/StackNet/blob/master/parameters/PARAMETERS.MD\)](https://github.com/kaz-Anova/StackNet/blob/master/parameters/PARAMETERS.MD) are in the repo.

As a quick note, one should try a few diverse models. To my experience, a good stacking solution is often composed of at least:

- 2 or 3 GBMs (one with low depth, one with medium and one with high)
- 1 or 2 Random Forests (again as diverse as possible—one low depth, one high)
- 1 or 2 NNs (one deeper, one smaller)
- 1 linear model

There is a video presenting StackNet in [the data science festival in London 2017](https://www.youtube.com/watch?v=9Vk1rXLhG48) (<https://www.youtube.com/watch?v=9Vk1rXLhG48>) which features a top 10 submission using it in the Amazon employee classification challenge (<https://www.kaggle.com/c/amazon-employee-access-challenge>). The example details are here (https://github.com/kaz-Anova/StackNet/blob/master/example/example_amazon/EXAMPLE.MD). One may look *after minute 14* where StackNet is presented.

Marios Michailidis: How to become a Kaggle #1: An introduction to model sta...



Why Java?

- Java is less verbose than C to write and therefore more suited for data scientists to use.
- It is very popular (as it is included in 3 billion devices).
- Can be used in any operating system without serious modifications or compilers.
- Statically typed and better defined than other languages, hence is more suited for development (in comparison to Python for instance).

- Also Java does not have tools with an easy python API like [sklearn \(http://scikit-learn.org/\)](http://scikit-learn.org/) does. StackNet contains its own algorithms (as well as other implementations) and can be further used in development too. These algorithms included in StackNet have [sklearn-like APIs \(http://scikit-learn.org/stable/modules/classes.html\)](http://scikit-learn.org/stable/modules/classes.html), (however most of them are NOT the same algorithms). In other words, using Java for this means people who use Java for their data analysis could benefit from such an API, therefore it potentially extends the reach of data science to more people.

What's next for StackNet? What algorithms and features can we look forward to?

The most immediate improvements will refer to extending its coverage with world-class machine learning tools (like the XGBoost that got recently added). StackNet will (almost) always give you a bit better than your best single model, so it is vital to include the best tools so that the strongest ensembles can be constructed.

Tools expected to be added soon are:

- [H \(https://h2o-release.s3.amazonaws.com/h2o/rel-turan/4/docs-website/h2o-py/docs/modeling.html\)](https://h2o-release.s3.amazonaws.com/h2o/rel-turan/4/docs-website/h2o-py/docs/modeling.html) [2O algorithms \(https://h2o-release.s3.amazonaws.com/h2o/rel-turan/4/docs-website/h2o-py/docs/modeling.html\)](https://h2o-release.s3.amazonaws.com/h2o/rel-turan/4/docs-website/h2o-py/docs/modeling.html)
- [LightGBM \(https://github.com/Microsoft/LightGBM\)](https://github.com/Microsoft/LightGBM)
- [LIBFFM \(https://www.csie.ntu.edu.tw/~cjlin/libffm/\)](https://www.csie.ntu.edu.tw/~cjlin/libffm/)
- [Vowpal Wabbit \(https://github.com/JohnLangford/vowpal_wabbit/wiki\)](https://github.com/JohnLangford/vowpal_wabbit/wiki)
- [Weka Algorithms \(http://www.cs.waikato.ac.nz/ml/weka/\)](http://www.cs.waikato.ac.nz/ml/weka/)

If I could add other tools like the [sklearn \(http://scikit-learn.org/\)](http://scikit-learn.org/) library that would have been great, but I am not sure how feasible is to be done efficiently from Java.

Other elements that are expected be added come from the following areas:

- Wrappers for other programming languages
- Data pre-processing
- Feature Selection
- Hyper parameter optimization and grid search
- Possibly model decompression (from ensemble back to simple model) - *ensemble pruning*
- Possibly GUI for easier usage

Additionally StackNet model will be presented at [infiniteconf 2017 \(https://skillsmatter.com/conferences/7983-infiniteconf-2017-the-conference-on-big-data-data-science-and-engineering#program\)](https://skillsmatter.com/conferences/7983-infiniteconf-2017-the-conference-on-big-data-data-science-and-engineering#program) [6th-7th July]

When may StackNet not work well?

StackNet can overfit when there are **strong temporal elements** in the data—in situations like predicting the stock market or when the test data seem like they have been drawn from a **different or altered distribution** in comparison to the train data. StackNet's ability to exhaust information within the training data (through multiple algorithms) means may not do very well when the future is significantly different from the past.

Do you have any advice on using StackNet outside of competitions?

The first thing to mention here is that organizations should not be afraid to use blackbox solutions. It is true that a StackNet model may be very complex and tough to understand, but this should not intimidate people. One should focus on its performance via setting up a **reliable testing framework**.

Obviously it takes some time to get used to the tool and learn most of the algorithms and their parameters but it is worth the investment if maximum prediction performance is sought. Since StackNet's predict and train tasks are **scalable**, time could be reduced significantly.

Also by having the ability to control the size of the ensemble, an organization may find it easy to specify the right threshold of **model complexity, performance and cost**.

How can readers contribute to the project? What kinds of contributions are you looking for?

Apart from reporting bugs, a great help would be for people to assist with making wrappers for other tools or programming languages. It would be great if they could use the implemented API for estimators (<https://github.com/kaz-Anova/StackNet/blob/master/src/ml/estimator.java>), classifiers (<https://github.com/kaz-Anova/StackNet/blob/master/src/ml/classifier.java>), and regressors (<https://github.com/kaz-Anova/StackNet/blob/master/src/ml/regressor.java>) to add to this project.

General performance optimization advice is very welcome, too.



Acknowledgments

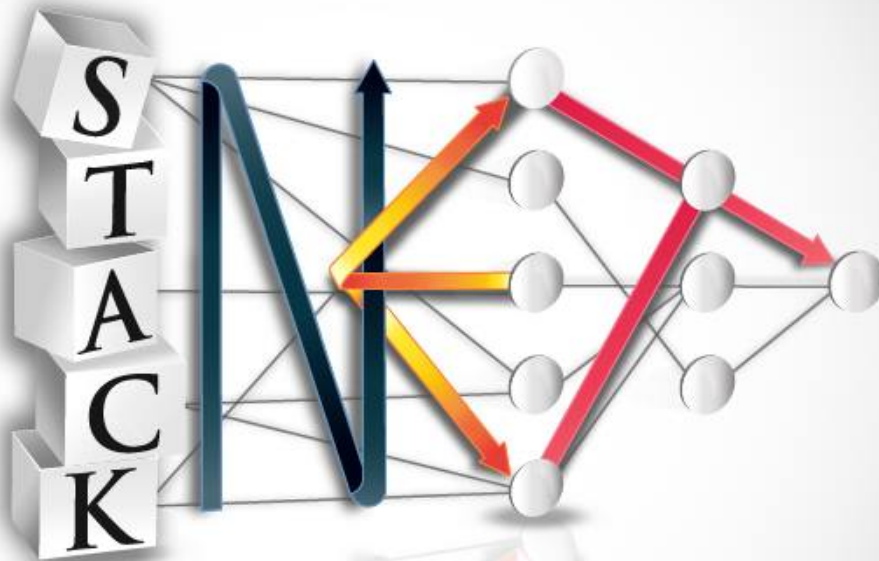
References to other similar work, projects, or blogs

- Wolpert 1992 (<http://www.machine-learning.martinsewell.com/ensembles/stacking/Wolpert1992.pdf>) Stacked generalization
- xcessiv (<https://github.com/reiinakano/xcessiv>)
- Deepwater-stacked ensembles (<https://www.youtube.com/watch?v=vFzdVlzwjwY&t=4667s>) video
- Kaggle ensemble guide (<https://mlwave.com/kaggle-ensembling-guide/>)
- Kaggler's practical ensemble guide (<http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/>)

Special thanks to

Apart from my supervisors, I would like to thank:

- Obviously [dunnhumby](https://www.dunnhumby.com/) (<https://www.dunnhumby.com/>) for supporting the project
- [UCL](https://www.ucl.ac.uk/) (<https://www.ucl.ac.uk/>) for the same reason
- My Kaggle buddies [Triskelion](https://www.kaggle.com/triskelion) (<https://www.kaggle.com/triskelion>) and [Faron](https://www.kaggle.com/mmueller) (<https://www.kaggle.com/mmueller>) for building thousands of stacked models together!
- My sister [Afroditi Michailidi](https://www.linkedin.com/in/afroditi-michailidi-445561a6/) (<https://www.linkedin.com/in/afroditi-michailidi-445561a6/>) for designing the logo



About the author



(<https://www.linkedin.com/in/mariosmichailidis>)

Marios Michailidis (<https://www.linkedin.com/in/mariosmichailidis>) | [KazAnova](https://www.kaggle.com/kazanova) (<https://www.kaggle.com/kazanova>) is Research Data Scientist at [H](https://www.h2o.ai/) (<https://www.h2o.ai/>) 2O (<https://www.h2o.ai/>) and part-time PhD student in machine learning at University College London ([UCL](https://www.ucl.ac.uk/) (<https://www.ucl.ac.uk/>)) with a focus on ensemble modelling. He is the creator of [StackNet](https://github.com/kaz-Anova/StackNet) (<https://github.com/kaz-Anova/StackNet>) and other freeware machine learning tools like [KazAnova](http://www.kazanovaforanalytics.com/software.html) (<http://www.kazanovaforanalytics.com/software.html>) [GUI](http://www.kazanovaforanalytics.com/software.html) (<http://www.kazanovaforanalytics.com/software.html>). Marios is former [Kaggle #1](#)

[\(http://blog.kaggle.com/2016/02/10/profiling-top-kagglers-kazanov-new-1-in-the-world/\)](http://blog.kaggle.com/2016/02/10/profiling-top-kagglers-kazanov-new-1-in-the-world/), having competed in over 100 Kaggle competitions to challenge himself, learn from the best, and improve his research work.

[MODEL STACKING \(HTTP://BLOG.KAGGLE.COM/TAG/MODEL-STACKING/\)](http://blog.kaggle.com/tag/model-stacking/)

[PROFILING TOP KAGGLERS
\(HTTP://BLOG.KAGGLE.COM/TAG/PROFILING-TOP-KAGGLERS/\)](http://blog.kaggle.com/tag/profiling-top-kagglers/)

[STACKNET \(HTTP://BLOG.KAGGLE.COM/TAG/STACKNET/\)](http://blog.kaggle.com/tag/stacknet/)



LOG IN WITH

OR SIGN UP WITH DISQUS ?

**Satadru SK Kundu** • 8 days ago

Hi,

Can stacknet be used in Python or it has any Python API?

^ | ▾ • Reply • Share ›

**KazA nova** → Satadru SK Kundu • 7 days ago

Not Yet. There will be in the future though . For now you could use it either through the command line or via Java.

^ | ▾ • Reply • Share ›

**Simran Singh** • 8 days ago

Thanks for the great article.

I have a stupid question though... The gif shows the case for binary classification. The features in B1 (and C1) are the probabilities of each training example being in either class 0 or 1. How will B1 look like in case of multi class classification?? (say we have 4 classes W, X, Y, Z)

^ | ▾ • Reply • Share ›

**KazA nova** → Simran Singh • 7 days ago

This is not stupid - no worries. You will have 4 predictions per model. So probability to be class W,X,Y,Z for each model. In the example above you will end up with 12 columns (4 for each model).

^ | ▾ • Reply • Share ›

**Reii Nakano** • 8 days ago

Creator of xcessiv here! Thanks for the mention!

This is a great article. I am interested in the use of the HOG / SIFT features for classification.

<https://www.facebook.com/kaggle><https://twitter.com/kaggle>

