How to become kaggle #1:

An introduction to model stacking

Marios Michailidis

Marios.michailidis@dunnhumby.com

dunnhumby

What is kaggle

- **World's biggest predictive modelling competition platform**
- **Half a million** members
- Companies host **data challenges**.
- Usual tasks include:
  - Predict **topic or sentiment** from text.
  - Predict species/type from **image**.
  - Predict s**tore**/product/area **sales**
  - **Marketing** response

amazon.com

facebook

dunnhumby

Microsoft

Inspired by Horse races…!

- At the **University of Southampton**, an entrepreneur talked to us about how he was able to **predict** the **horse races** with **regression**!

Was curious, wanted to learn more

- learned **statistical tools** (Like SAS, SPSS, R)
- I became more **passionate**!
- Picked up **programming skills**

Built KazAnova

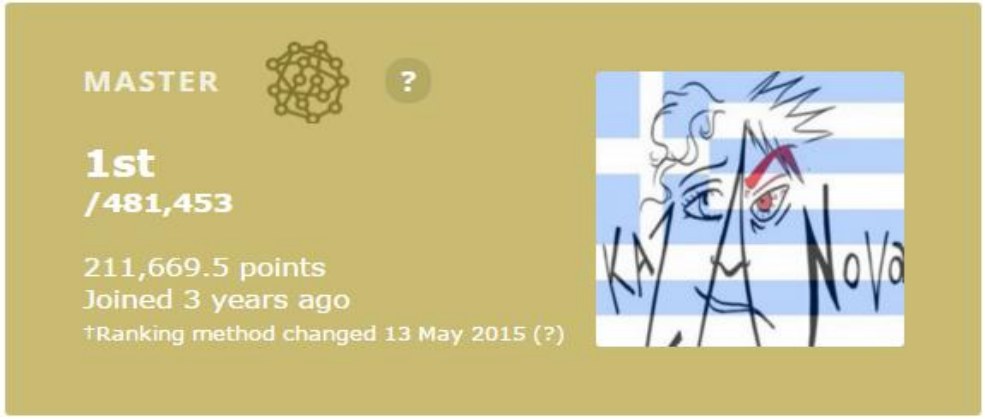
- Generated a couple of algorithms and data techniques and decided to **make them public** so that others can gain from it.
- I released it at **www.kazanovaforanalytics.com/**
- Named it after **ANOVA** (Statistics) and
- **KAZANI** , mom's last name.

Joined Kaggle!

- Was **curious** about Kaggle.
- Joined a few **contests and learned lots** ☺ .
- The community was very open to **sharing** and **collaboration**.

3 Years of modelling competitions

- Over **100** competitions
- Participated with **46** different teams
- **25** top 10 finishes
- **14** times prize winner
- **3** different modelling platforms
- **Ranked 1st** out of 480k data scientists



So… what wins competitions?

In short:

- **Understand** the **problem** (functions,metrics,features)
- **Discipline** (especially in when testing)
- try **problem-specific** things or new approaches
- The **hours** you put in
- the **right tools**
- **Collaboration**.
- **Ensembling**

What's next

- **PhD** (UCL) about using ensemble methods to improve **recommender systems**.
- Developed **StackNet** , *a scalable meta-modelling framework*

dunnhumby

The **StackNet** Model

A scalable meta-modelling framework

Supervisors

Professor Philip Treleaven
Giles Pavey

dunnhumby

What is StackNet?

- StackNet is…
  - A **meta modelling** methodology that
  - utilizes **Wolpert's stacked generalization** (1992) of combining multiple models assuming
  - a feedforward **neural network** architecture of multiple levels
  - Each **node represents a machine learning algorithm**
  - A version of it with several algorithms is available in **Java**

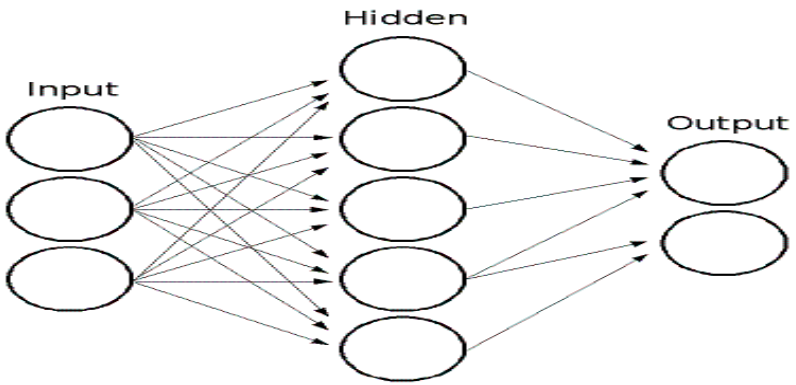
Inspiration - Stacking

- Wolpert in 1992 introduced *stacking* – a Meta-modelling technique.

1. Split the training set into two disjoint sets.
2. Train several base learners on the first part.
3. Test the base learners on the second part.
4. Using the predictions from (3) as the inputs, and the correct responses as the outputs, train a higher level learner.

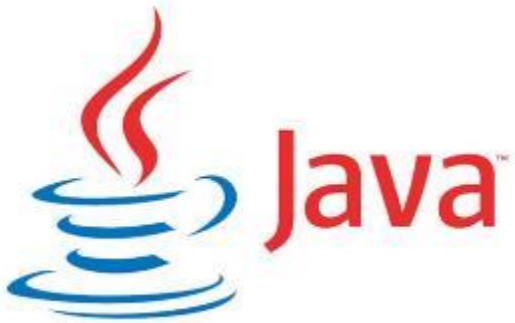
Inspiration – Neural Networks

- Artificial networks were first created in an attempt to mimic the biological neural networks in the human Brain. [Rosenblatt ,1958] was the first to create – the **perceptron**.
- The advances in computing power and specifically the usages of GPUs has allowed them to be run at greater speeds in comlex structures taking the form of today's **deep learning** [Schmidhuber, 2015] .
- Their **structure** is considered **state-of-the-art** for many tasks



Inspiration – Why Java

- is **less verbose** than C and very **popular**
- Can be used in **any operational system**
- Almost **every** computer/device has it by default
- **Statically typed** and better defined
- Java Does **not** have **Scikit-learn**!



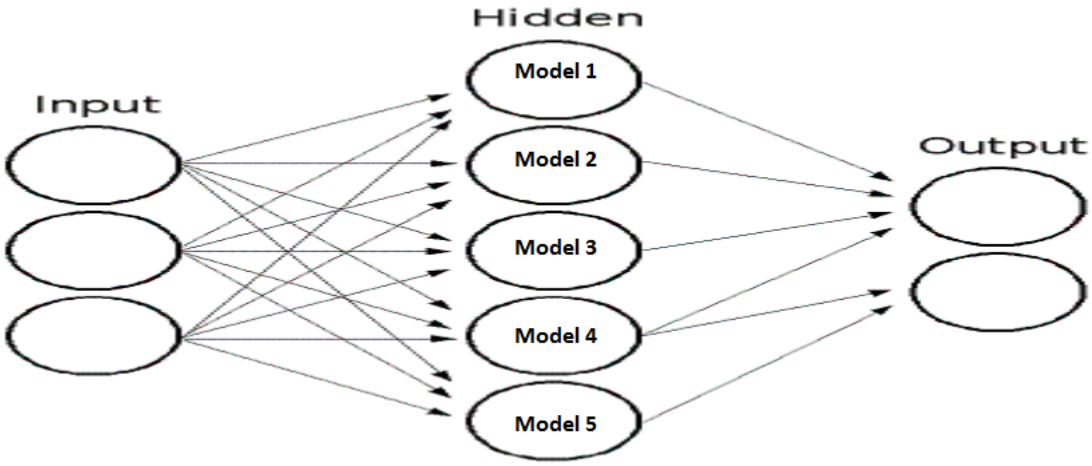
How it works - General

- In a neural network , every node is a **simple linear model** (like linear regression) maybe with some non linear transformation.
- Instead of a linear model , StackNet proposes **any modelling function**.
- In other words:

$$f_1(x_i) = \sum_{h=1}^{H} (g_1(\hat{x_i}) beta_{1h} + bias_{1h})$$

$$f_1(x_i, s) = \sum_{h=1}^{H} (g_1 s_h(\hat{x_i}))$$

How it works - General



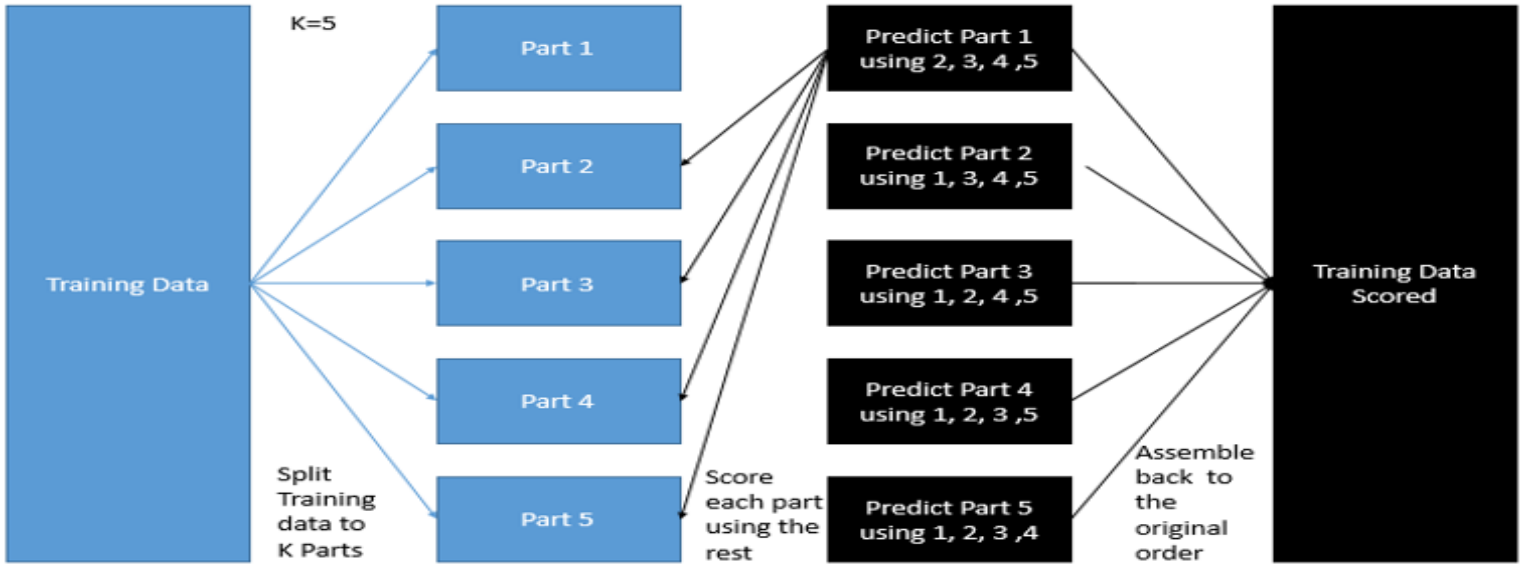
First batch of models includes

- Linear Regression
- Logistic regression
- Kernel models
- K nearest neighbours
- GBMs

- Naïve Bayes
- LibFm
- Multilayer Perceptron
- Decision trees
- Random Forests

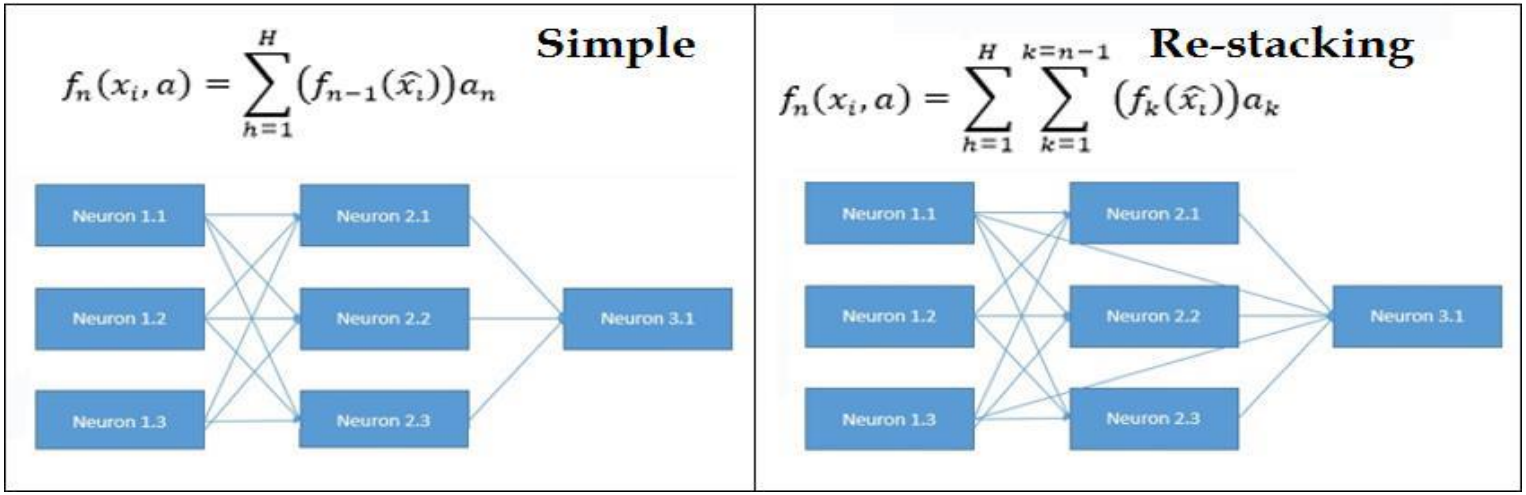
Training – Reusable Holdout

- Limited data based on which multiple models must be built on , enhances the notion of **a re-usable holdout**
- It uses stratified k-folding – which is a **hyper parameter**.



Training - Modes

- The training process is a straight **one-pass**. There is no notion of re-optimizing in multiple epochs. Convergence needs to reached within that 1 epoch.



Command Line parameters

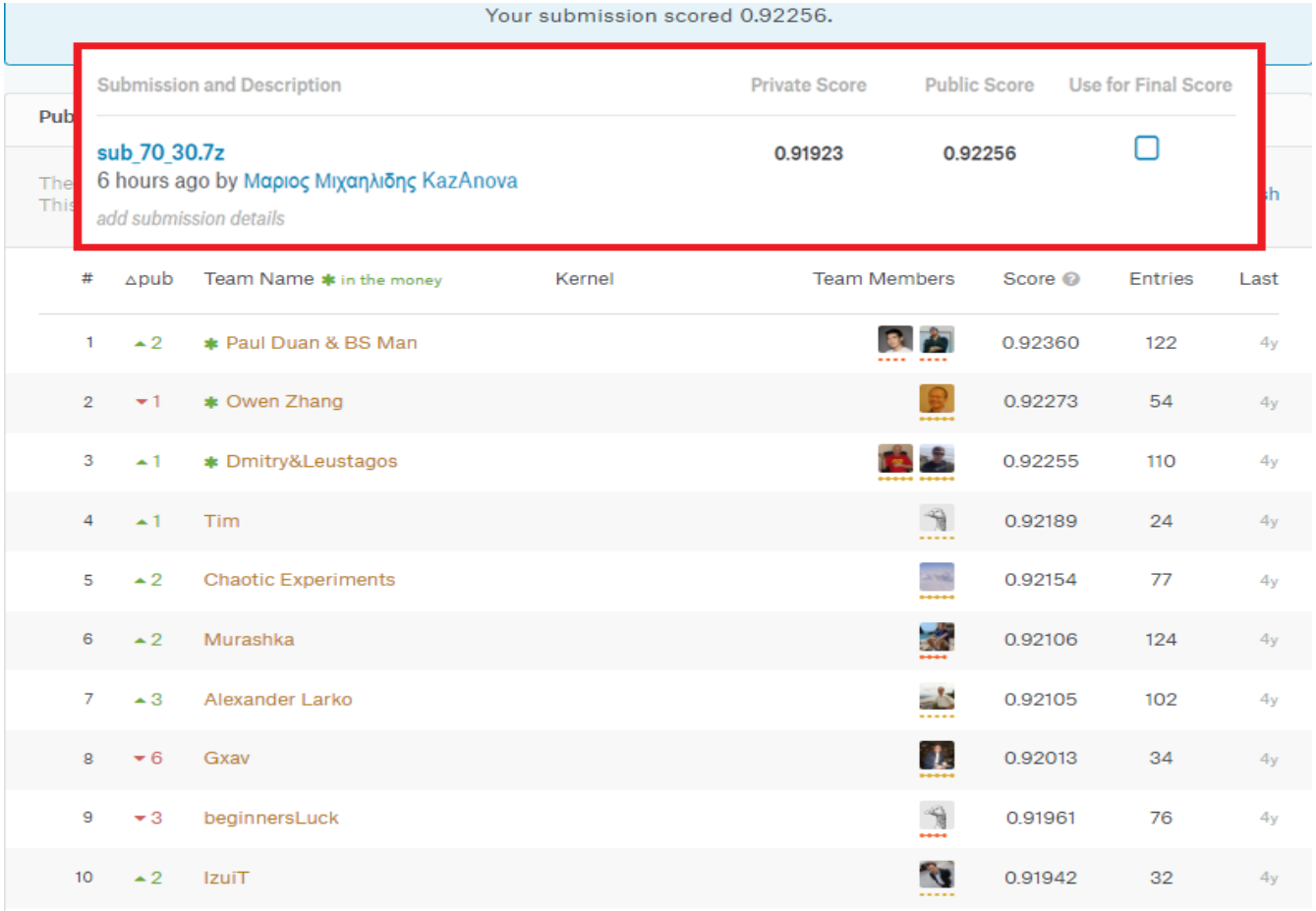
Command	Explanation
sparse	True if the data to be imported are in sparse format (libsvm)
has_head	True if train_file and test_file have headers else false
model	Name of the output model file.
pred_file	Name of the output prediction file.
train_file	Name of the training file.
test_file	Name of the test file.
test_target	True if the test file has a target variable in the beginning
params	Parameter file where each line is a model.
verbose	True if we need StackNet to output its progress else false
threads	Number of models to run in parallel.
metric	Logloss, Rmse, accuracy or auc (for binary only)
stackdata	True for restacking else false
seed	Integer for randomised procedures
folds	Number of folds for re-usable kfold

Sample Parameter's File

LogisticRegression Type:Liblinear C:2.0 threads:1 usescale:True
GradientBoostingForestClassifier estimators:300 shrinkage:0.10 max_depth:6 max_features:0.5
RandomForestClassifier estimators:300 threads:5 max_depth:16 max_features:0.25

RandomForestClassifier estimators:1500 max_depth:7 max_features:0.2 min_leaf:1.0

Example Amazon – get top 10 with StackNet



Your submission scored 0.92256.

Submission and Description	Private Score	Public Score	Use for Final Score
sub_70_30.7z 6 hours ago by Μαριος Μιχαηλιδης KazAnova *add submission details*	0.91923	0.92256	☐

#	△pub	Team Name ✱ in the money	Kernel	Team Members	Score	Entries	Last
1	▲2	✱ Paul Duan & BS Man			0.92360	122	4y
2	▼1	✱ Owen Zhang			0.92273	54	4y
3	▲1	✱ Dmitry&Leustagos			0.92255	110	4y
4	▲1	Tim			0.92189	24	4y
5	▲2	Chaotic Experiments			0.92154	77	4y
6	▲2	Murashka			0.92106	124	4y
7	▲3	Alexander Larko			0.92105	102	4y
8	▼6	Gxav			0.92013	34	4y
9	▼3	beginnersLuck			0.91961	76	4y
10	▲2	IzuiT			0.91942	32	4y

How the experiment was run

- **Amazon.com - Employee Access Challenge** was a popular Kaggle competition (around 1700 teams) and they first one I entered! I finished 100ish after spending 3 weeks.
- This experiment will get you to **top10** within a few hours (including data preparation and modelling.
- The interesting about this competition is it has **only 8 Variables** (and 1 duplicate)!
- All **categorical** with **high cardinality**
- The purpose is to build a model, learned using historical data, that will **determine an employee's access needs**
- The metric to optimize is Area Under The Roc Curve or simply **AUC**
- Data is downloaded from : https://www.kaggle.com/c/amazon-employee-access-challenge
- To get all the code , visit : https://github.com/kaz-Anova/StackNet/tree/master/example/example_amazon
- Run with **python** the **prepare_data.py** script to create the **modelling data**.

Run first StackNet on sparse data

- Compute best 4 way interactions with a linear model (python script)
- Use logistic regression to assess good interractions
- Then run StackNet with the following command:

java -Xmx3048m -jar StackNet.jar **train**

train_file=train.sparse

test_file=test.sparse

params=param_amazon_linear.txt

pred_file=amazon_linear_pred.csv

test_target=false

verbose=true

Threads=1

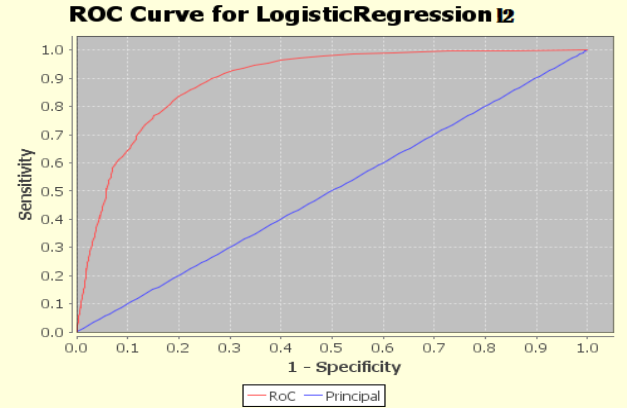
sparse=true

folds=5

seed=1

metric=auc

Level 1		
MODEL	**AUC**	**GINI**
LogisticRegression L2	0.893	78.70%
LogisticRegression SGD	0.885	76.95%
LSVC L2	0.891	78.18%
LinearRegression	0.879	75.80%
LibFmClassifier	0.891	78.28%
softmaxnnclassifier	0.882	76.38%
GradientBoostingForestClassifier	0.851	70.14%
LogisticRegression L1	0.88	75.97%
LSVC L1	0.873	74.52%
Level 2		
RandomForestClassifier - 0.901		



dunnhumby

Run first StackNet on dense data…per fold

- Compute all 3 way interactions
- Compute counts and likelihood (woe features) per fold
- Create five pairs of train/cv files and a train and a test file too.
- Run the command :

java -Xmx3048m -jar StackNet.jar **train**

data_prefix=amazon_counts

test_file=amazon_counts_test.txt

params=param_amazon_count.txt

pred_file=amazon_count_pred.csv

test_target=false

verbose=true

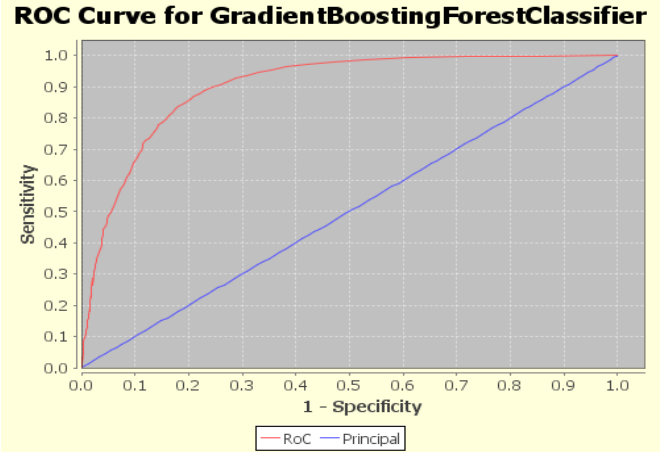
Threads=1

folds=5

seed=1

metric=auc

Level 1		
MODEL	**AUC**	**GINI**
LogisticRegression	0.889	77.86%
GradientBoostingForestClassifier	0.9	80.19%
RandomForestClassifier	0.899	79.78%
softmaxnnclassifier	0.866	73.14%
LSVC	0.888	77.59%
LibFmClassifier	0.89	77.93%
GradientBoostingForestRegressor	0.858	71.61%
LinearRegression	0.901	80.19%
Level 2		
RandomForestClassifier - 0.904		



dunnhumby

Blend the 2 predictions to get the top10 score

- Use the **blend_script.py** to rank average the two prediction files produced with the previous 2 models to achieve the top score
- Submit it to Kaggle.

Things to be mindful when using StackNet

- StackNet cannot do wonders! But it can give you **better results than your single models involved or simple ensemble methods** (like averaging). So…StackNet will be a bit stronger than your best models.
- StackNet may underperform when there are **Strong temporal elements** in the data and it is better to supply your own train/cv datasets.
- When supplying **own files**, you are responsible for **controlling overfitting**. Never have overlapping samples across folds or within folds.

StackNet was used in other challenges…

- It was used to **win** Dato's Truly native classification challenge (Kaggle) : http://blog.kaggle.com/2015/12/03/dato-winners-interview-1st-place-mad-professors/
- It was also used to **win** the Homesite Quote Conversion challenge on Kaggle : http://blog.kaggle.com/2016/04/08/homesite-quote-conversion-winners-write-up-1st-place-kazanova-faron-clobber/
- There is a big discussion and how it has helped different people in the ongoing Kaggle competition hosted by Twosigma : https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries/discussion/30012

Next Steps

- Include other **prominent machine learning tools** such as Xgboost , Lightgbm , H2O, Weka.
- Make it available to more **programming languages**
- Include **feature engineering** steps
- Include **hyper parameter optimization**
- Implement **feature selection**
- Implement **model selection** and **dropouts**

Tools to include vol1

- **Liblinear** : for linear models http://www.csie.ntu.edu.tw/~cjlin/liblinear/
- **LibSvm** for Support Vector machines www.csie.ntu.edu.tw/~cjlin/libsvm/
- **Scikit** package in python for text classification, random forests and gradient boosting machines scikit-learn.org/stable/
- **Xgboost** for fast scalable gradient boosting https://github.com/tqchen/xgboost
- **LightGBM** https://github.com/Microsoft/LightGBM
- **Vowpal Wabbit** hunch.net/~vw/ for fast memory efficient linear models
- http://www.heatonresearch.com/encog **encog** for neural nets
- **H2O** for many models

Tools to include vol 2

- **LibFm** www.libfm.org
- **LibFFM** : https://www.csie.ntu.edu.tw/~cjlin/libffm/
- **Weka** in Java (has everything) http://www.cs.waikato.ac.nz/ml/weka/
- **Graphchi** for factorizations : https://github.com/GraphChi
- **GraphLab** for lots of stuff. https://dato.com/products/create/open_source.html
- **Cxxnet** : One of the best implementation of convolutional neural nets out there. Difficult to install and requires GPU with NVDIA Graphics card. https://github.com/antinucleon/cxxnet
- **RankLib**: The best library out there made in java suited for ranking algorithms (e.g. rank products for customers) that supports optimization fucntions like NDCG. people.cs.umass.edu/~vdang/ranklib.html
- **Keras (** http://keras.io/) and **Lasagne(https**://github.com/Lasagne/Lasagne) for nets. This assumes you have **Theano** (http://deeplearning.net/software/theano/) or **Tensorflow** https://www.tensorflow.org/ .