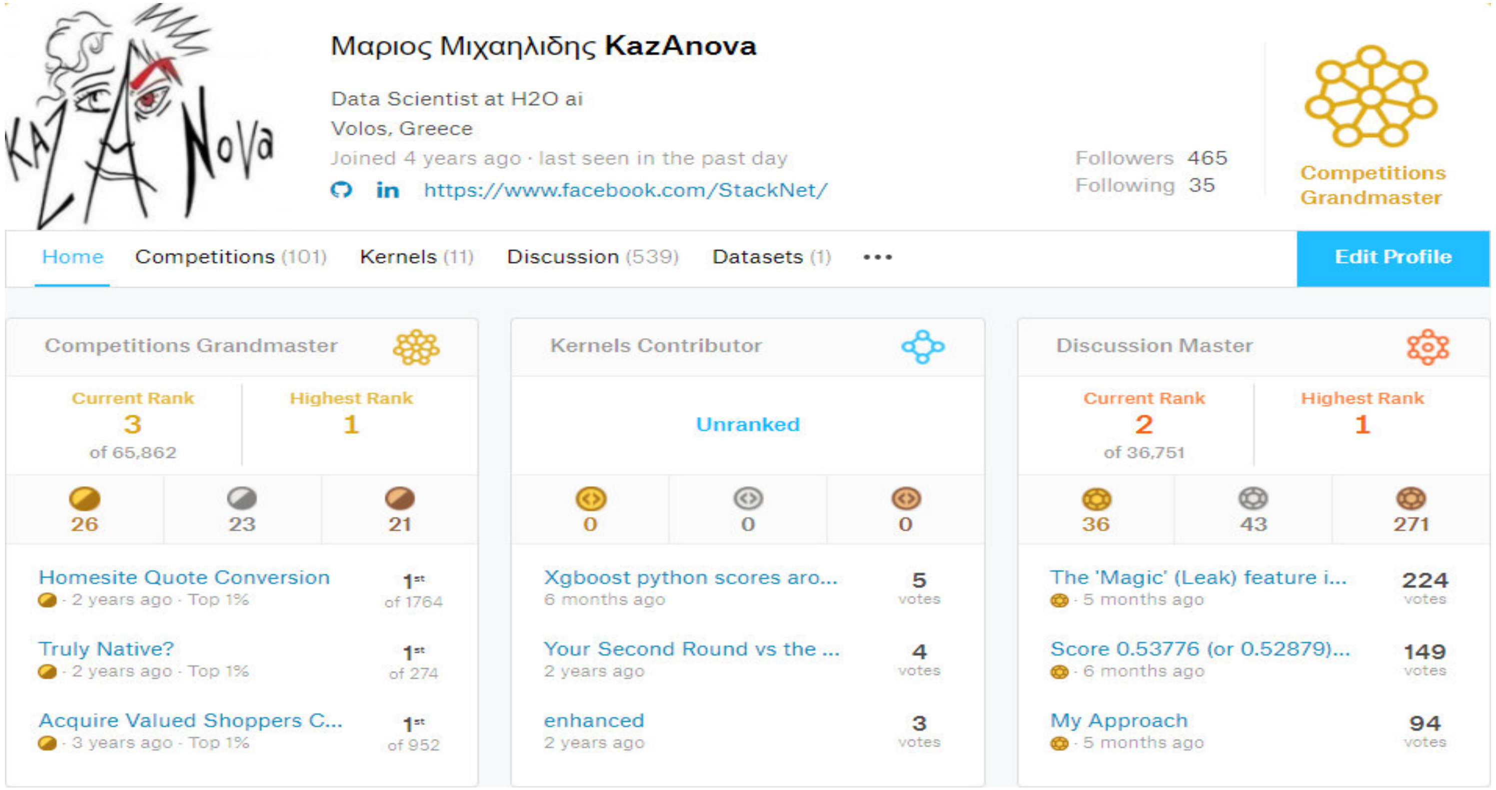
Intro to ensemble methods

By Marios Michailidis

Background

- Research data scientist at H_2O.ai
- PhD in ensemble methods
- Former kaggle #1



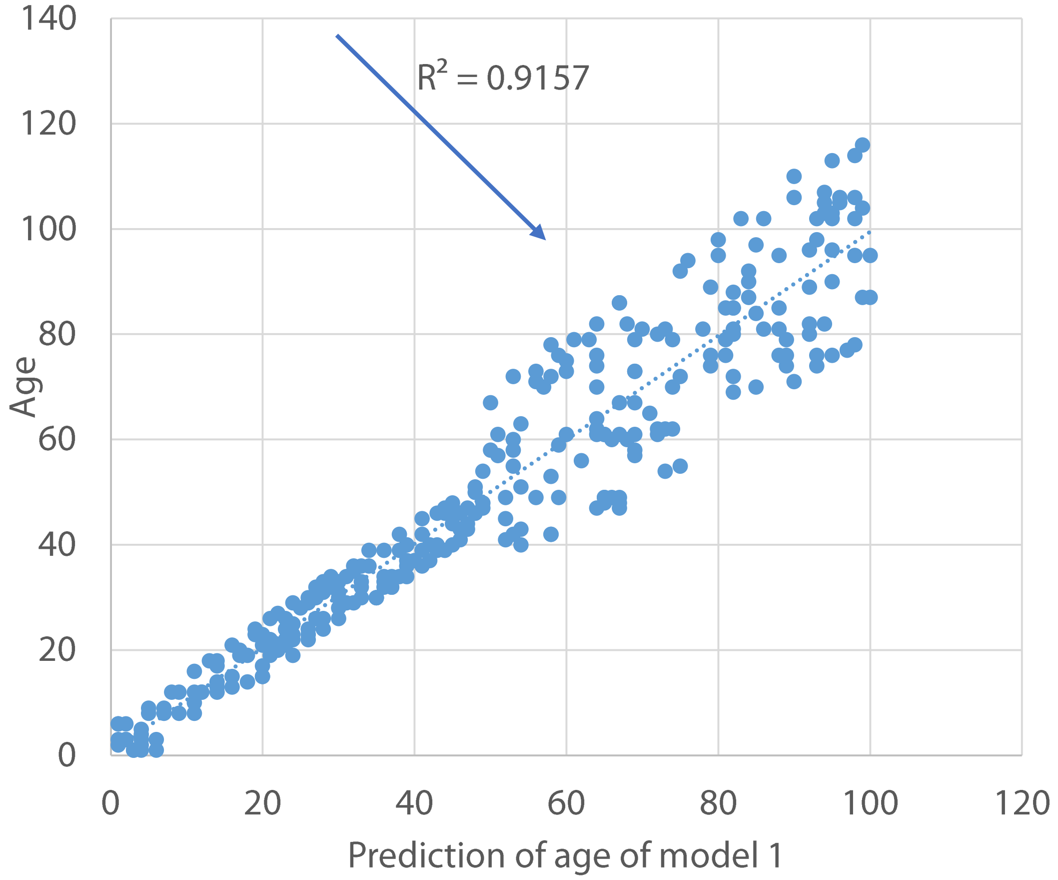
What is ensemble modelling?

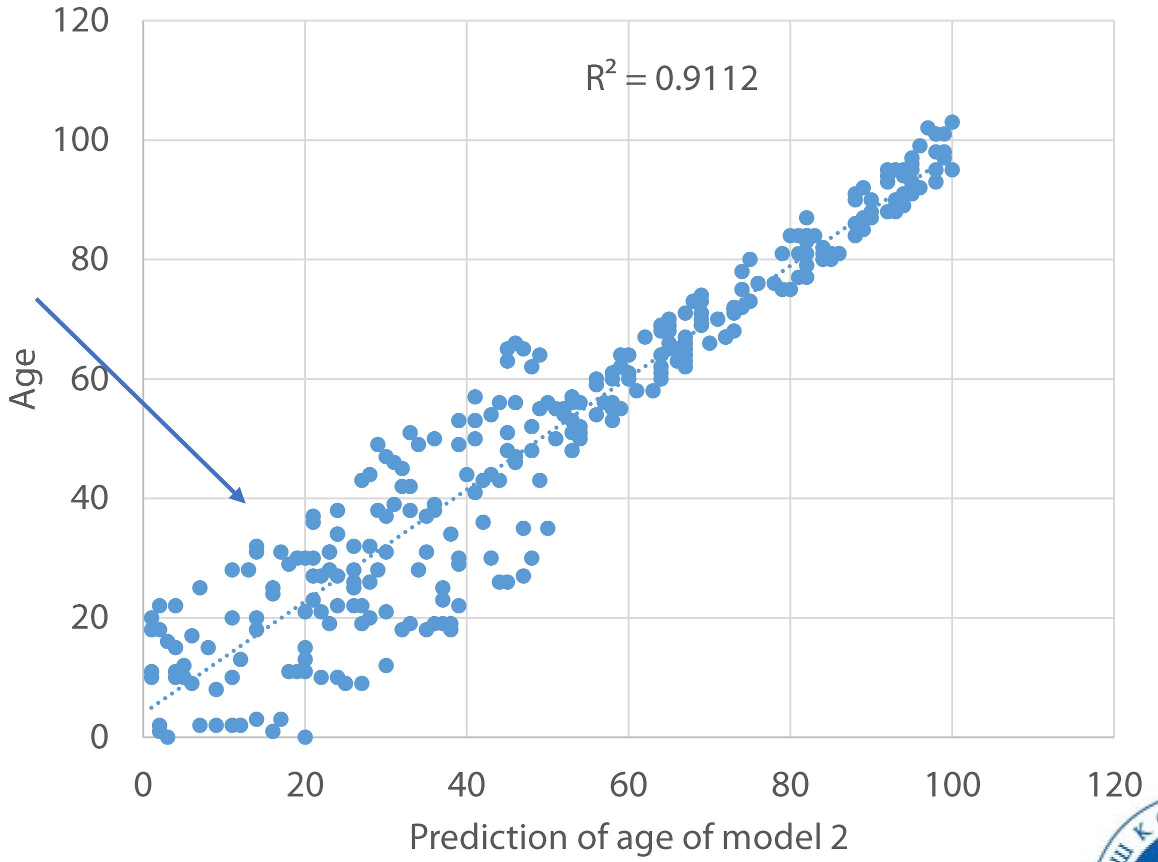
It means combining different machine learning models to get a better prediction.

Examined ensemble methods

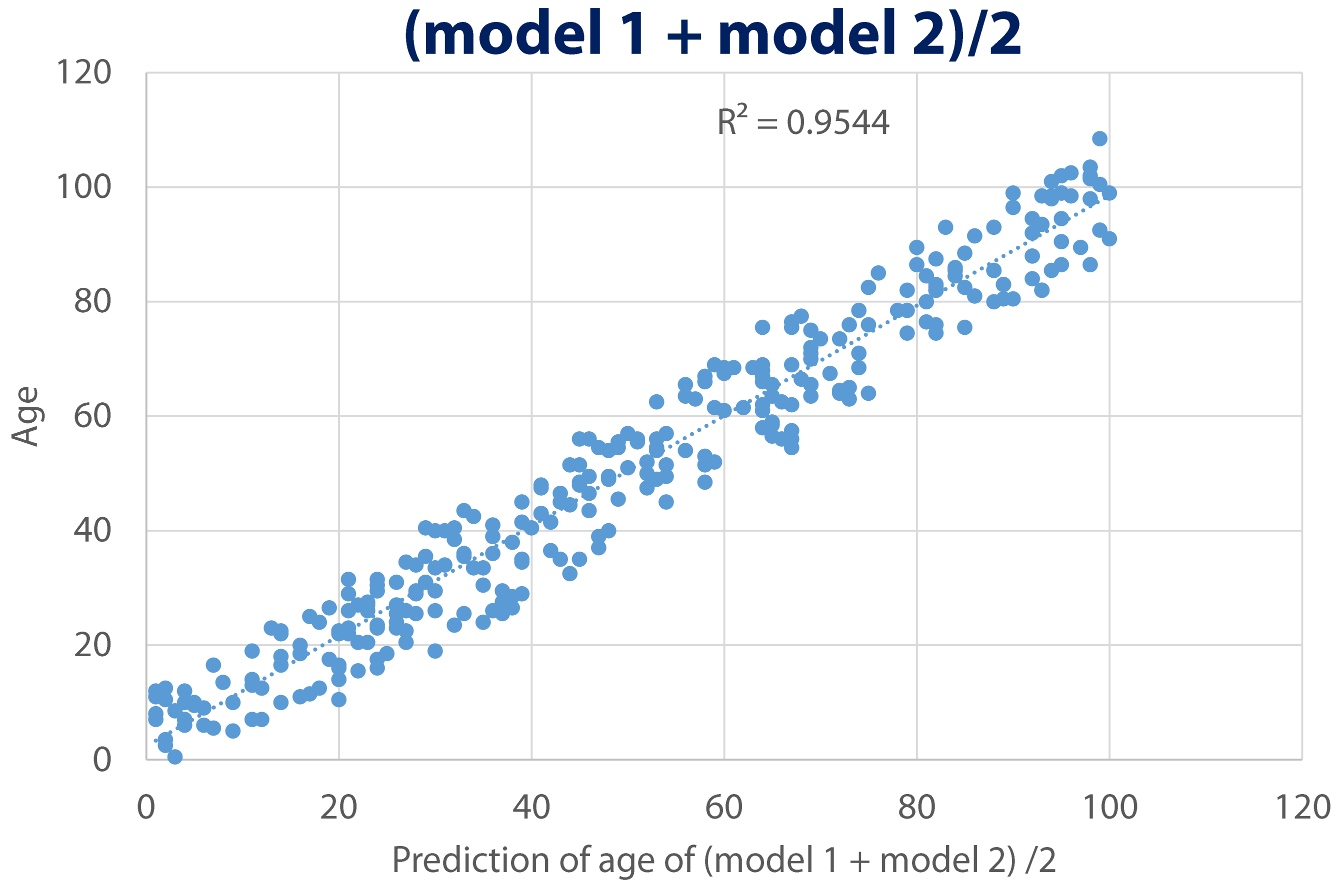
- Averaging (or blending)
- Weighted averaging
- Conditional averaging
- Bagging
- Boosting
- Stacking
- StackNet

Averaging ensemble methods

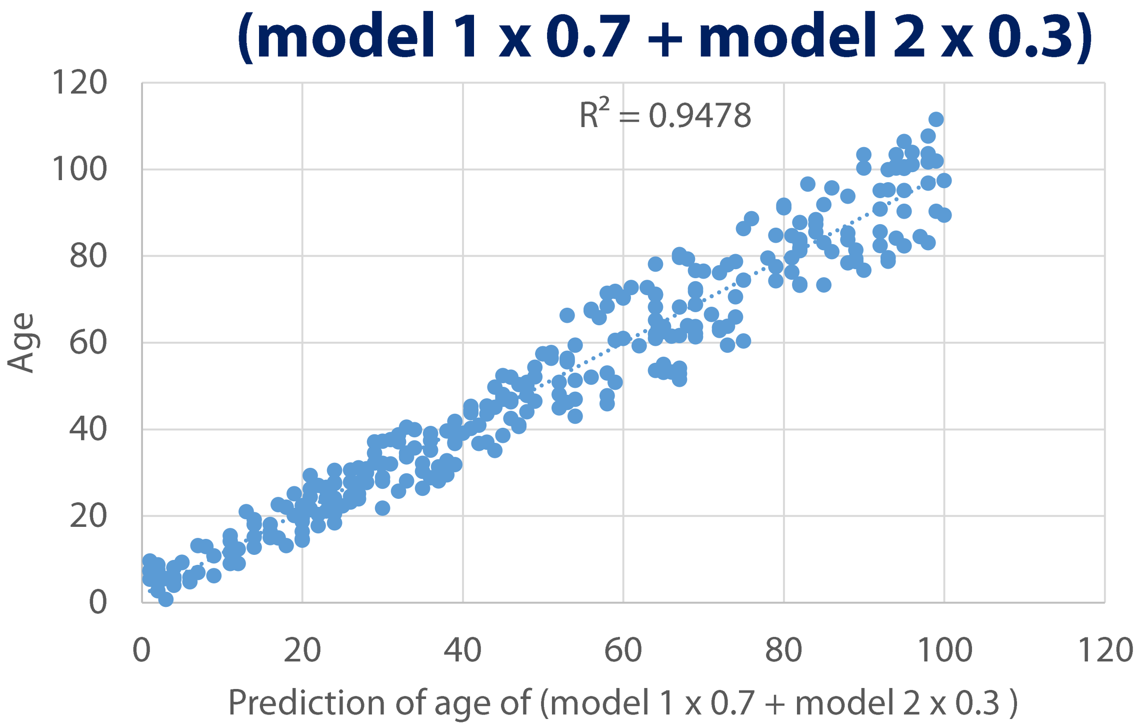




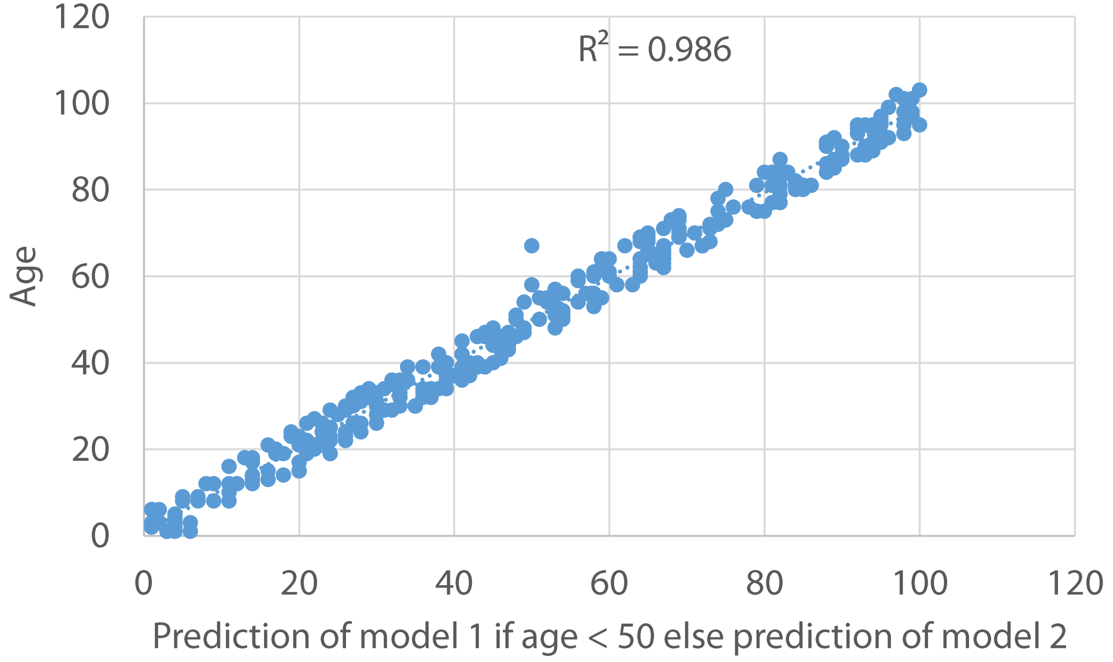
Averaging ensemble methods



Averaging ensemble methods



Averaging ensemble methods



Ensemble methods: bagging

By Marios Michailidis

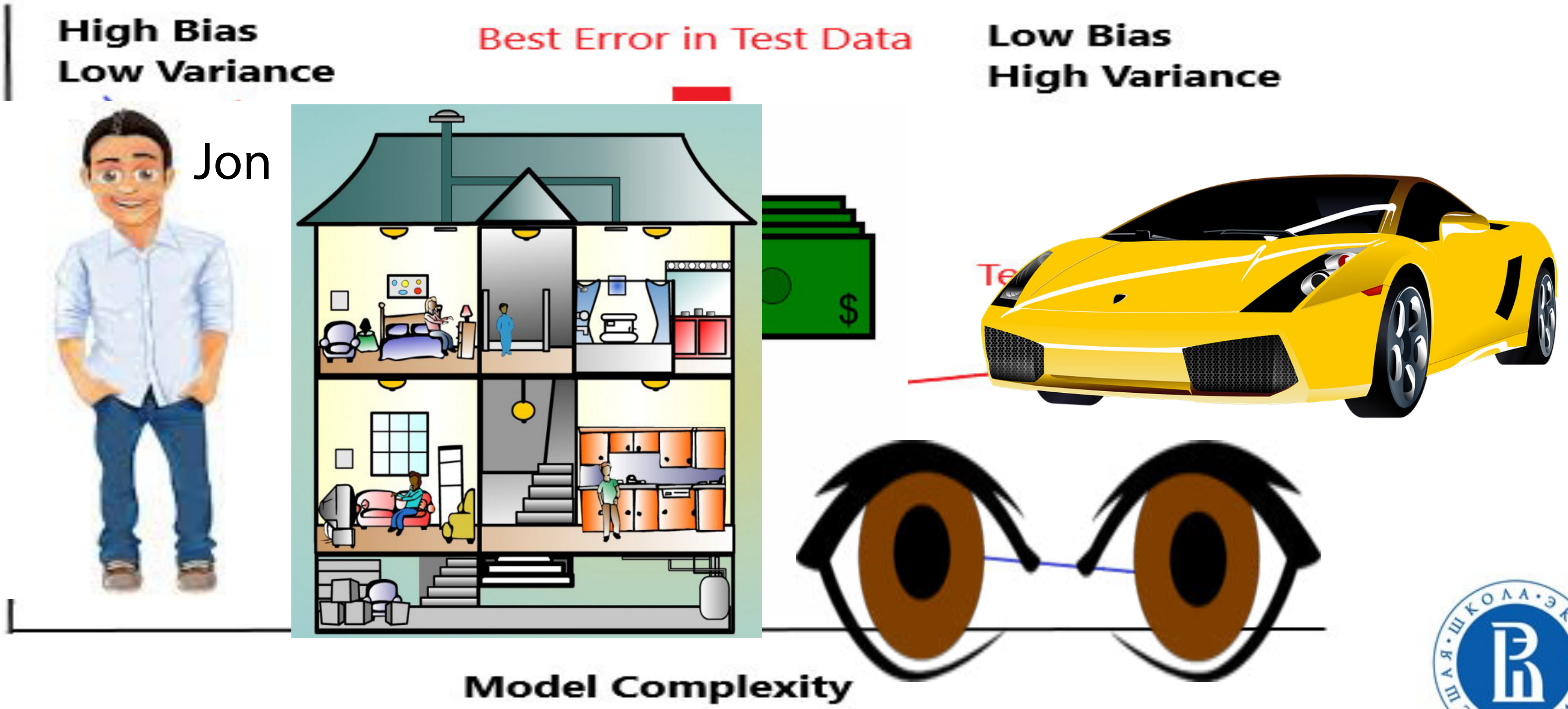
Examined ensemble methods

- Averaging (or blending)
- Weighted averaging
- Conditional averaging
- Bagging
- Boosting
- Stacking
- StackNet

What is Bagging

Means **averaging** slightly different versions of the same model to improve accuracy

Why Bagging


High Bias
Low Variance
Best Error in Test Data
Low Bias
High Variance
Jon
$
Model Complexity

Parameters that control bagging?

- Changing the seed
- Row (Sub) sampling or Bootstrapping
- Shuffling
- Column (Sub) sampling
- Model-specific parameters
- Number of models (or bags)
- (Optionally) parallelism

Examples of bagging

BaggingClassifier and **BaggingRegressor** from Sklearn

```
# train is the training data
# test is the test data
# y is the target variable
model=RandomForestRegressor()
bags=10
seed=1
# create array object to hold bagged predictions
bagged_prediction=np.zeros(test.shape[0])
#loop for as many times as we want bags
for n in range (0, bags):
    model.set_params(random_state=seed + n)# update seed
    model.fit(train,y) # fit model
    preds=model.predict(test) # predict on test data
    bagged_prediction+=preds # add predictions to bagged predictions
#take average of predictions
bagged_prediction/= bags
```

Ensemble methods: boosting

By Marios Michailidis

Examined ensemble methods

- Averaging (or blending)
- Weighted averaging
- Conditional averaging
- Bagging
- Boosting
- Stacking
- StackNet

What is Boosting

A form of weighted averaging of models where each model is built sequentially via taking into account the past model performance.

Main boosting types

- Weight based
- Residual based

Weight based boosting

Rownum	x0	x1	x2	x3	y
0	0.94	0.27	0.80	0.34	1
1	0.84	0.79	0.89	0.05	1
2	0.83	0.11	0.23	0.42	1
3	0.74	0.26	0.03	0.41	0
4	0.08	0.29	0.76	0.37	0
5	0.71	0.76	0.43	0.95	1
6	0.08	0.72	0.97	0.04	0

Weight based boosting

Rownum	x0	x1	x2	x3	y	pred
0	0.94	0.27	0.80	0.34	1	0.80
1	0.84	0.79	0.89	0.05	1	0.75
2	0.83	0.11	0.23	0.42	1	0.65
3	0.74	0.26	0.03	0.41	0	0.40
4	0.08	0.29	0.76	0.37	0	0.55
5	0.71	0.76	0.43	0.95	1	0.34
6	0.08	0.72	0.97	0.04	0	0.02

Weight based boosting

Rownum	x0	x1	x2	x3	y	pred	abs.error
0	0.94	0.27	0.80	0.34	1	0.80	0.20
1	0.84	0.79	0.89	0.05	1	0.75	0.25
2	0.83	0.11	0.23	0.42	1	0.65	0.35
3	0.74	0.26	0.03	0.41	0	0.40	0.40
4	0.08	0.29	0.76	0.37	0	0.55	0.55
5	0.71	0.76	0.43	0.95	1	0.34	0.66
6	0.08	0.72	0.97	0.04	0	0.02	0.02

Weight based boosting

Rownum	x0	x1	x2	x3	y	pred	abs.error	weight
0	0.94	0.27	0.80	0.34	1	0.80	0.20	1.20
1	0.84	0.79	0.89	0.05	1	0.75	0.25	1.25
2	0.83	0.11	0.23	0.42	1	0.65	0.35	1.35
3	0.74	0.26	0.03	0.41	0	0.40	0.40	1.40
4	0.08	0.29	0.76	0.37	0	0.55	0.55	1.55
5	0.71	0.76	0.43	0.95	1	0.34	0.66	1.66
6	0.08	0.72	0.97	0.04	0	0.02	0.02	1.02

Weight based boosting

Rownum	x0	x1	x2	x3	y	weight
0	0.94	0.27	0.80	0.34	1	1.20
1	0.84	0.79	0.89	0.05	1	1.25
2	0.83	0.11	0.23	0.42	1	1.35
3	0.74	0.26	0.03	0.41	0	1.40
4	0.08	0.29	0.76	0.37	0	1.55
5	0.71	0.76	0.43	0.95	1	1.66
6	0.08	0.72	0.97	0.04	0	1.02

Weight based boosting parameters

- Learning rate (or shrinkage or eta)
- Number of estimators
- Input model – can be anything that accepts weights
- Sub boosting type:
  - AdaBoost – Good implementation in sklearn (python)
  - LogitBoost - Good implementation in Weka (Java)

Residual based boosting

Rownum	x0	x1	x2	x3	y
0	0.94	0.27	0.80	0.34	1
1	0.84	0.79	0.89	0.05	1
2	0.83	0.11	0.23	0.42	1
3	0.74	0.26	0.03	0.41	0
4	0.08	0.29	0.76	0.37	0
5	0.71	0.76	0.43	0.95	1
6	0.08	0.72	0.97	0.04	0

Residual based boosting

Rownum	x0	x1	x2	x3	y	pred
0	0.94	0.27	0.80	0.34	1	0.80
1	0.84	0.79	0.89	0.05	1	0.75
2	0.83	0.11	0.23	0.42	1	0.65
3	0.74	0.26	0.03	0.41	0	0.40
4	0.08	0.29	0.76	0.37	0	0.55
5	0.71	0.76	0.43	0.95	1	0.34
6	0.08	0.72	0.97	0.04	0	0.02

Residual based boosting

Rownum	x0	x1	x2	x3	y	pred	error
0	0.94	0.27	0.80	0.34	1	0.80	0.20
1	0.84	0.79	0.89	0.05	1	0.75	0.25
2	0.83	0.11	0.23	0.42	1	0.65	0.35
3	0.74	0.26	0.03	0.41	0	0.40	-0.40
4	0.08	0.29	0.76	0.37	0	0.55	-0.55
5	0.71	0.76	0.43	0.95	1	0.34	0.66
6	0.08	0.72	0.97	0.04	0	0.02	-0.02

Residual based boosting

Rownum	x0	x1	x2	x3	y
0	0.94	0.27	0.80	0.34	0.2
1	0.84	0.79	0.89	0.05	0.25
2	0.83	0.11	0.23	0.42	0.35
3	0.74	0.26	0.03	0.41	-0.4
4	0.08	0.29	0.76	0.37	-0.55
5	0.71	0.76	0.43	0.95	0.66
6	0.08	0.72	0.97	0.04	-0.02

Residual based boosting

Rownum	x0	x1	x2	x3	y	new pred
0	0.94	0.27	0.80	0.34	0.2	0.15
1	0.84	0.79	0.89	0.05	0.25	0.20
2	0.83	0.11	0.23	0.42	0.35	0.40
3	0.74	0.26	0.03	0.41	-0.4	-0.30
4	0.08	0.29	0.76	0.37	-0.55	-0.20
5	0.71	0.76	0.43	0.95	0.66	0.24
6	0.08	0.72	0.97	0.04	-0.02	-0.01

Residual based boosting

Rownum	x0	x1	x2	x3	y	new pred	old pred
0	0.94	0.27	0.80	0.34	0.2	0.15	0.80
1	0.84	0.79	0.89	0.05	0.25	0.20	0.75
2	0.83	0.11	0.23	0.42	0.35	0.40	0.65
3	0.74	0.26	0.03	0.41	-0.4	-0.30	0.40
4	0.08	0.29	0.76	0.37	-0.55	-0.20	0.55
5	0.71	0.76	0.43	0.95	0.66	0.24	0.34
6	0.08	0.72	0.97	0.04	-0.02	-0.01	0.02

Residual based boosting

Rownum	x0	x1	x2	x3	y	new pred	old pred
0	0.94	0.27	0.80	0.34	0.2	0.15	0.80
1	0.84	0.79	0.89	0.05	0.25	0.20	0.75
2	0.83	0.11	0.23	0.42	0.35	0.40	0.65
3	0.74	0.26	0.03	0.41	-0.4	-0.30	0.40
4	0.08	0.29	0.76	0.37	-0.55	-0.20	0.55
5	0.71	0.76	0.43	0.95	0.66	0.24	0.34
6	0.08	0.72	0.97	0.04	-0.02	-0.01	0.02

To predict Rownum=1 we would say :

Final prediction = 0.75 + 0.20 = **0.95**

Residual based boosting

Rownum	x0	x1	x2	x3	y	new pred	old pred
0	0.94	0.27	0.80	0.34	0.2	0.15	0.80
1	0.84	0.79	0.89	0.05	0.25	0.20	0.75
2	0.83	0.11	0.23	0.42	0.35	0.40	0.65
3	0.74	0.26	0.03	0.41	-0.4	-0.30	0.40
4	0.08	0.29	0.76	0.37	-0.55	-0.20	0.55
5	0.71	0.76	0.43	0.95	0.66	0.24	0.34
6	0.08	0.72	0.97	0.04	-0.02	-0.01	0.02

To predict Rownum=1 we would say :

Final prediction = 0.75 + 0.20 = **0.95**

Residual based boosting parameters

- Learning rate (or shrinkage or eta)
- Number of estimators
- Row (sub) sampling
- Column (sub) sampling
- Input model – better be trees.
- Sub boosting type:
    - Fully gradient based
    - Dart

Residual based favourite implementations

- Xgboost
- Lightgbm
- H2O's GBM
- Catboost
- Sklearn's GBM

Ensemble methods: stacking

By Marios Michailidis

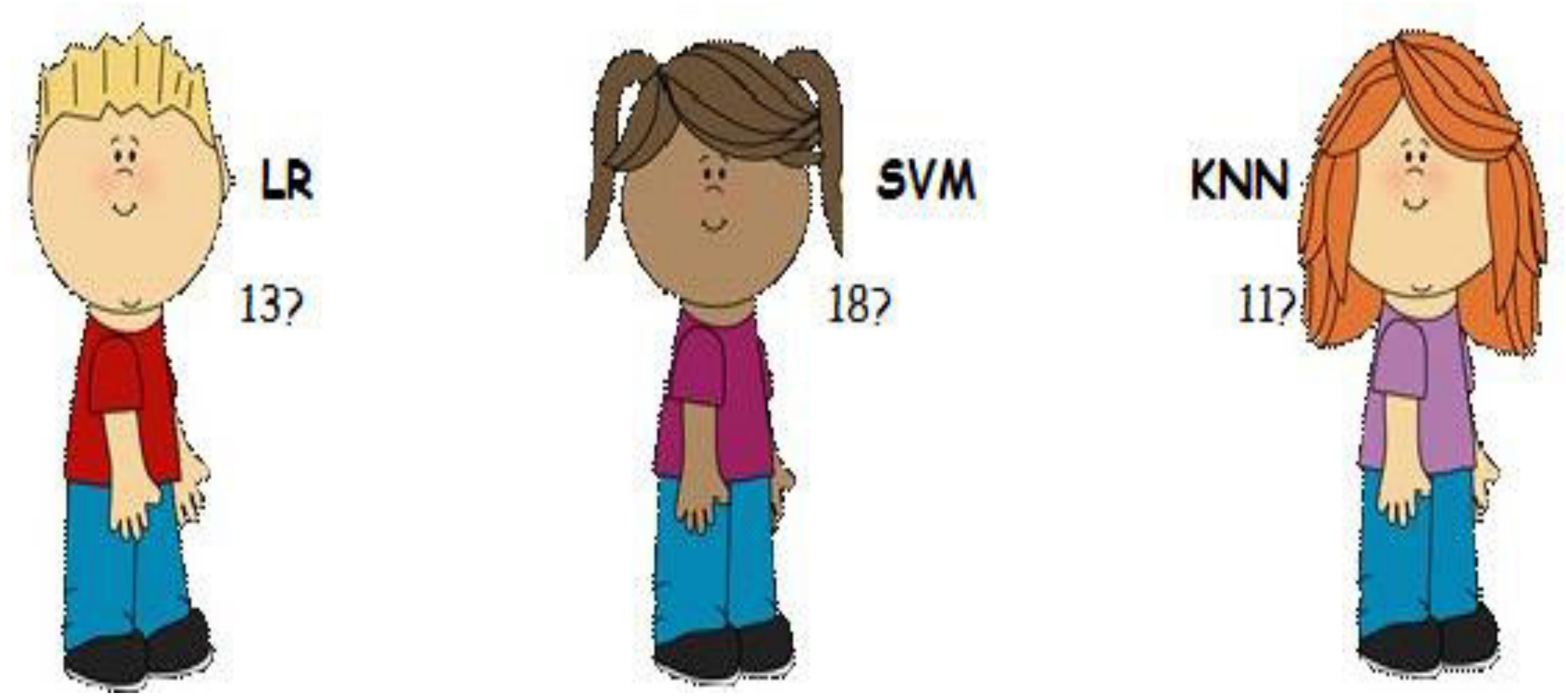
Examined ensemble methods

- Averaging (or blending)
- Weighted averaging
- Conditional averaging
- Bagging
- Boosting
- Stacking
- StackNet

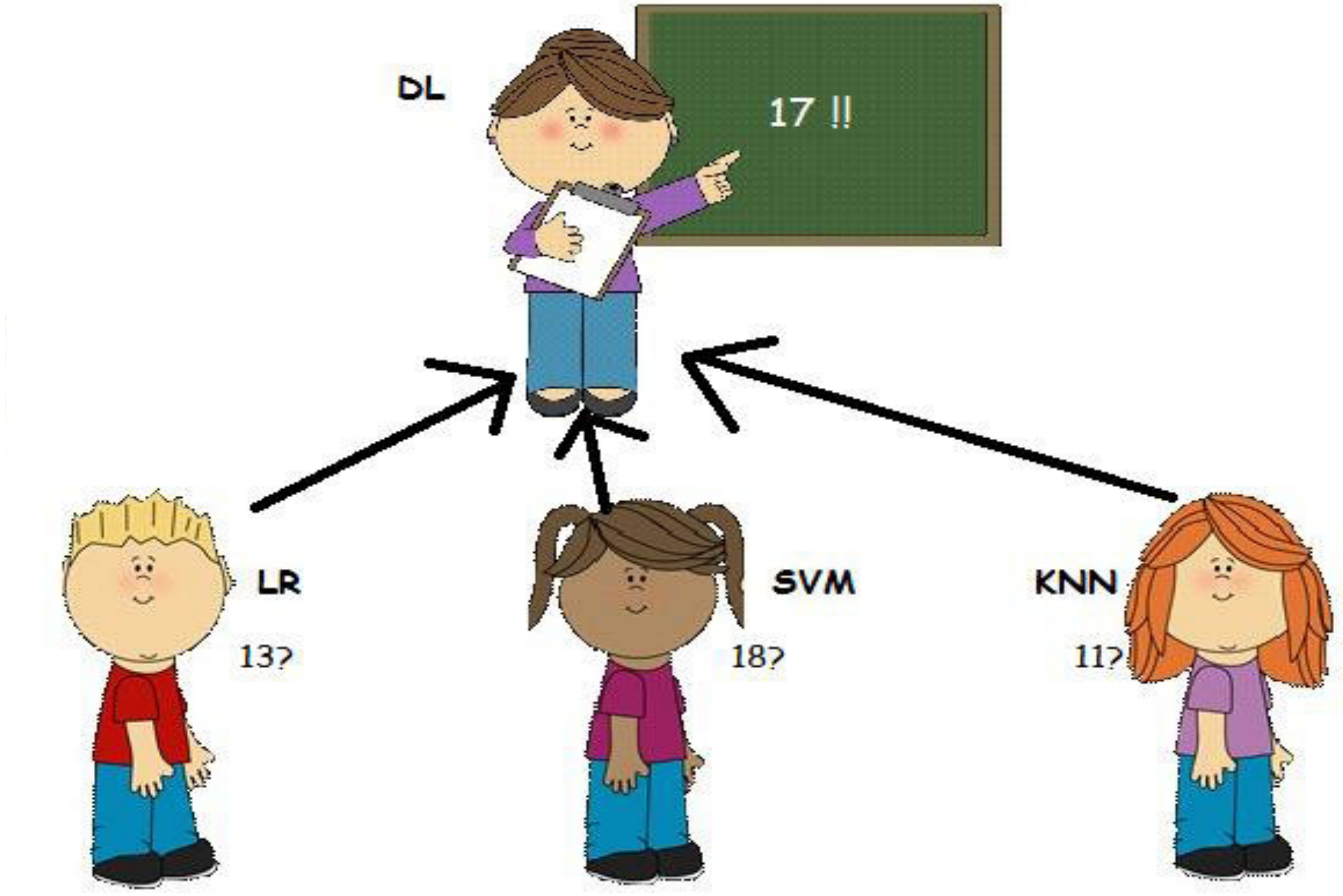
What is Stacking

Means making predictions of a number of models in a hold-out set and then using a different (Meta) model to train on these predictions.

Naïve example



Naïve example



Methodology

- Wolpert in 1992 introduced stacking. It involves:

1. **Splitting** the train set into two disjoint sets.
2. **Train** several base learners on the first part.
3. **Make predictions** with the base learners on the second (validation) part.
4. Using the **predictions** from (3) **as the inputs** to train a higher level learner.

Still confused about Stacking?

Consider datasets A,B,C. Target variable (y) is known for A,B

Still confused about Stacking?

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	?

Still confused about Stacking?

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**

B1
pred0
0.24
0.95
0.64
0.89
0.11

C1
pred0
0.50
0.62
0.22
0.90
0.20

Still confused about Stacking?

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**

B1	
pred0	pred1
0.24	0.72
0.95	0.25
0.64	0.80
0.89	0.58
0.11	0.20

C1	
pred0	pred1
0.50	0.50
0.62	0.59
0.22	0.31
0.90	0.47
0.20	0.09

Still confused about Stacking?

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**

Train algorithm **2** on A and make predictions for B and C and save to **B1**, **C1**

B1			
pred0	pred1	pred2	y
0.24	0.72	0.70	0
0.95	0.25	0.22	1
0.64	0.80	0.96	0
0.89	0.58	0.52	0
0.11	0.20	0.93	1

C1			
pred0	pred1	pred2	y
0.50	0.50	0.39	?
0.62	0.59	0.46	?
0.22	0.31	0.54	?
0.90	0.47	0.09	?
0.20	0.09	0.61	?

Still confused about Stacking?

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**

Train algorithm **2** on A and make predictions for B and C and save to **B1**, **C1**

B1			
pred0	pred1	pred2	y
0.24	0.72	0.70	0
0.95	0.25	0.22	1
0.64	0.80	0.96	0
0.89	0.58	0.52	0
0.11	0.20	0.93	1

C1				
pred0	pred1	pred2	y	Preds3
0.50	0.50	0.39	?	0.45
0.62	0.59	0.46	?	0.23
0.22	0.31	0.54	?	0.99
0.90	0.47	0.09	?	0.34
0.20	0.09	0.61	?	0.05

Train algorithm **3** on B1 and make predictions for C1

Stacking example

```
from sklearn.ensemble import RandomForestRegressor #import model
from sklearn.linear_model import LinearRegression #import model
import numpy as np #import numpy for stats
from sklearn.model_selection import train_test_split # split the training data

# train is the training data
# y is the target variable for the train data
# test is the test data
```

Stacking example

```
from sklearn.ensemble import RandomForestRegressor #import model
from sklearn.linear_model import LinearRegression #import model
import numpy as np #import numpy for stats
from sklearn.model_selection import train_test_split # split the training data

# train is the training data
# y is the target variable for the train data
# test is the test data
```

Stacking example

```
from sklearn.ensemble import RandomForestRegressor #import model
from sklearn.linear_model import LinearRegression #import model
import numpy as np #import numpy for stats
from sklearn.model_selection import train_test_split # split the training data
```

```
# train is the training data
# y is the target variable for the train data
# test is the test data
```

Stacking example

```
#split train data in 2 parts, training and  valdiation.
training,valid,ytraining,yvalid = train_test_split(train,y,test_size=0.5)
#specify models
model1=RandomForestRegressor()
model2=LinearRegression()
#fit models
model1.fit(training,ytraining)
model2.fit(training,ytraining)
#make predictions for validation
preds1=model1.predict(valid)
preds2=model2.predict(valid)
#make predictions for test data
test_preds1=model1.predict(test)
test_preds2=model2.predict(test)
#Form a new dataset for valid and test via stacking the predictions
stacked_predictions=np.column_stack((preds1,preds2))
stacked_test_predictions=np.column_stack((test_preds1,test_preds2))
#specify meta model
meta_model=LinearRegression()
#fit meta model on stacked predictions
meta_model.fit(stacked_predictions,yvalid)
#make predictions on the stacked predictions of the test data
final_predictions=meta_model.predict(stacked_test_predictions)
```

Stacking example

```
#split train data in 2 parts, training and  valdiation.
training,valid,ytraining,yvalid = train_test_split(train,y,test_size=0.5)
#specify models
model1=RandomForestRegressor()
model2=LinearRegression()
#fit models
model1.fit(training,ytraining)
model2.fit(training,ytraining)
#make predictions for validation
preds1=model1.predict(valid)
preds2=model2.predict(valid)
#make predictions for test data
test_preds1=model1.predict(test)
test_preds2=model2.predict(test)
#Form a new dataset for valid and test via stacking the predictions
stacked_predictions=np.column_stack((preds1,preds2))
stacked_test_predictions=np.column_stack((test_preds1,test_preds2))
#specify meta model
meta_model=LinearRegression()
#fit meta model on stacked predictions
meta_model.fit(stacked_predictions,yvalid)
#make predictions on the stacked predictions of the test data
final_predictions=meta_model.predict(stacked_test_predictions)
```

Stacking example

```
#split train data in 2 parts, training and  valdiation.
training,valid,ytraining,yvalid = train_test_split(train,y,test_size=0.5)
#specify models
model1=RandomForestRegressor()
model2=LinearRegression()
#fit models
model1.fit(training,ytraining)
model2.fit(training,ytraining)
#make predictions for validation
preds1=model1.predict(valid)
preds2=model2.predict(valid)
#make predictions for test data
test_preds1=model1.predict(test)
test_preds2=model2.predict(test)
#Form a new dataset for valid and test via stacking the predictions
stacked_predictions=np.column_stack((preds1,preds2))
stacked_test_predictions=np.column_stack((test_preds1,test_preds2))
#specify meta model
meta_model=LinearRegression()
#fit meta model on stacked predictions
meta_model.fit(stacked_predictions,yvalid)
#make predictions on the stacked predictions of the test data
final_predictions=meta_model.predict(stacked_test_predictions)
```

Stacking example

```
#split train data in 2 parts, training and  valdiation.
training,valid,ytraining,yvalid = train_test_split(train,y,test_size=0.5)
#specify models
model1=RandomForestRegressor()
model2=LinearRegression()
#fit models
model1.fit(training,ytraining)
model2.fit(training,ytraining)
#make predictions for validation
preds1=model1.predict(valid)
preds2=model2.predict(valid)
#make predictions for test data
test_preds1=model1.predict(test)
test_preds2=model2.predict(test)
#Form a new dataset for valid and test via stacking the predictions
stacked_predictions=np.column_stack((preds1,preds2))
stacked_test_predictions=np.column_stack((test_preds1,test_preds2))
#specify meta model
meta_model=LinearRegression()
#fit meta model on stacked predictions
meta_model.fit(stacked_predictions,yvalid)
#make predictions on the stacked predictions of the test data
final_predictions=meta_model.predict(stacked_test_predictions)
```

Stacking example

```
#split train data in 2 parts, training and  valdiation.
training,valid,ytraining,yvalid = train_test_split(train,y,test_size=0.5)
#specify models
model1=RandomForestRegressor()
model2=LinearRegression()
#fit models
model1.fit(training,ytraining)
model2.fit(training,ytraining)
#make predictions for validation
preds1=model1.predict(valid)
preds2=model2.predict(valid)
#make predictions for test data
test_preds1=model1.predict(test)
test_preds2=model2.predict(test)
#Form a new dataset for valid and test via stacking the predictions
stacked_predictions=np.column_stack((preds1,preds2))
stacked_test_predictions=np.column_stack((test_preds1,test_preds2))
#specify meta model
meta_model=LinearRegression()
#fit meta model on stacked predictions
meta_model.fit(stacked_predictions,yvalid)
#make predictions on the stacked predictions of the test data
final_predictions=meta_model.predict(stacked_test_predictions)
```

Stacking example

```
#split train data in 2 parts, training and  valdiation.
training,valid,ytraining,yvalid = train_test_split(train,y,test_size=0.5)
#specify models
model1=RandomForestRegressor()
model2=LinearRegression()
#fit models
model1.fit(training,ytraining)
model2.fit(training,ytraining)
#make predictions for validation
preds1=model1.predict(valid)
preds2=model2.predict(valid)
#make predictions for test data
test_preds1=model1.predict(test)
test_preds2=model2.predict(test)
#Form a new dataset for valid and test via stacking the predictions
stacked_predictions=np.column_stack((preds1,preds2))
stacked_test_predictions=np.column_stack((test_preds1,test_preds2))
#specify meta model
meta_model=LinearRegression()
#fit meta model on stacked predictions
meta_model.fit(stacked_predictions,yvalid)
#make predictions on the stacked predictions of the test data
final_predictions=meta_model.predict(stacked_test_predictions)
```

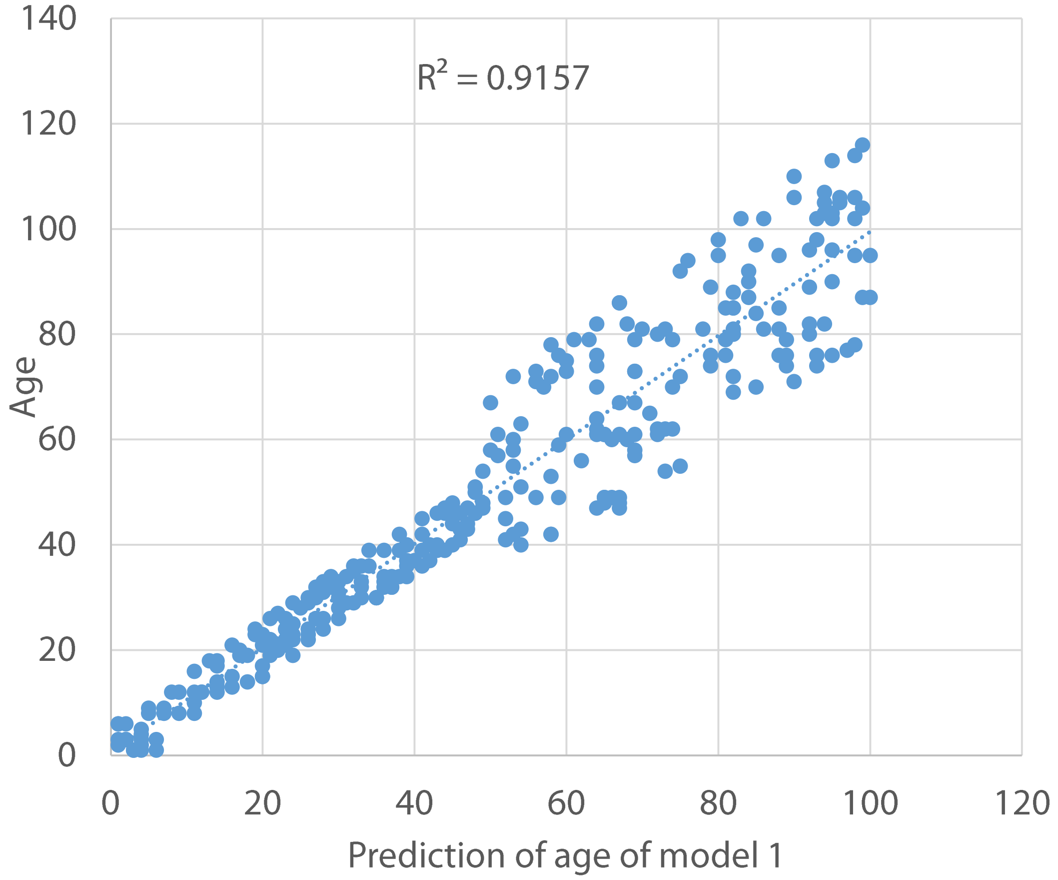
Stacking example

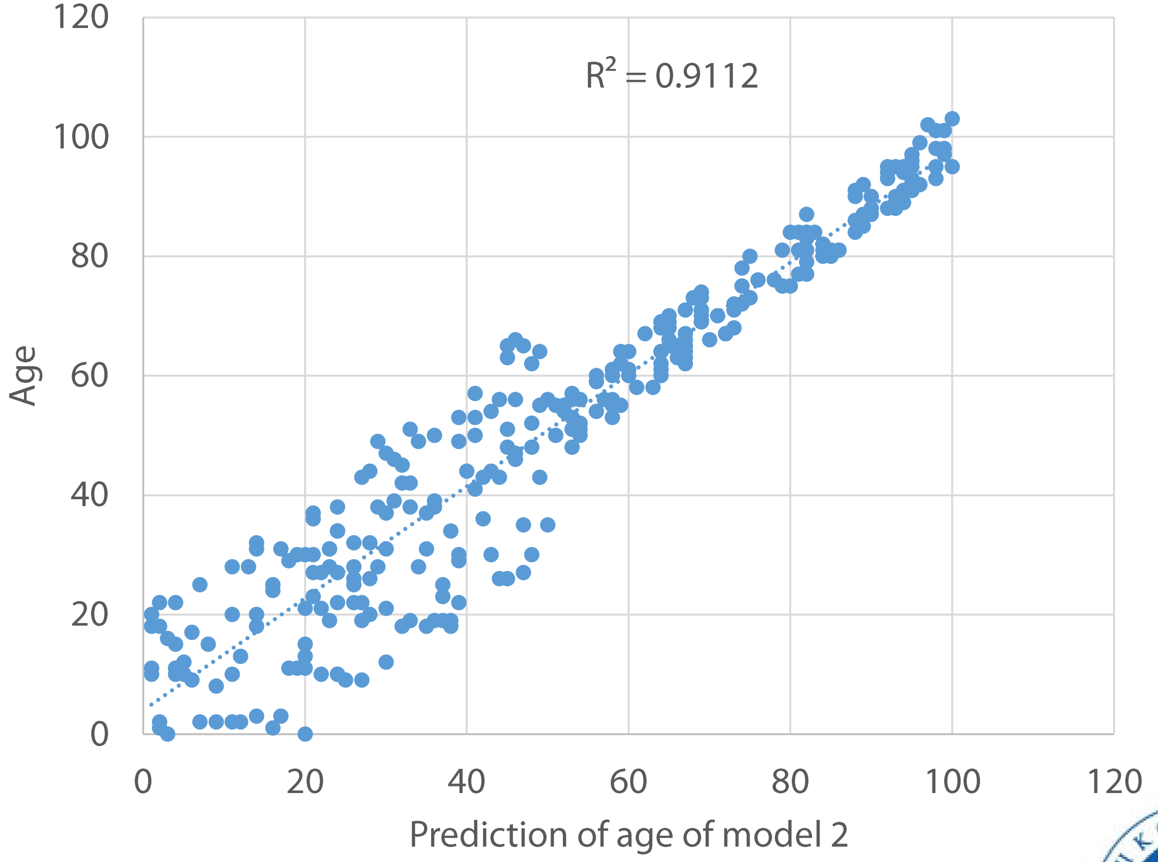
```
#split train data in 2 parts, training and  valdiation.
training,valid,ytraining,yvalid = train_test_split(train,y,test_size=0.5)
#specify models
model1=RandomForestRegressor()
model2=LinearRegression()
#fit models
model1.fit(training,ytraining)
model2.fit(training,ytraining)
#make predictions for validation
preds1=model1.predict(valid)
preds2=model2.predict(valid)
#make predictions for test data
test_preds1=model1.predict(test)
test_preds2=model2.predict(test)
#Form a new dataset for valid and test via stacking the predictions
stacked_predictions=np.column_stack((preds1,preds2))
stacked_test_predictions=np.column_stack((test_preds1,test_preds2))
#specify meta model
meta_model=LinearRegression()
#fit meta model on stacked predictions
meta_model.fit(stacked_predictions,yvalid)
#make predictions on the stacked predictions of the test data
final_predictions=meta_model.predict(stacked_test_predictions)
```

Stacking example

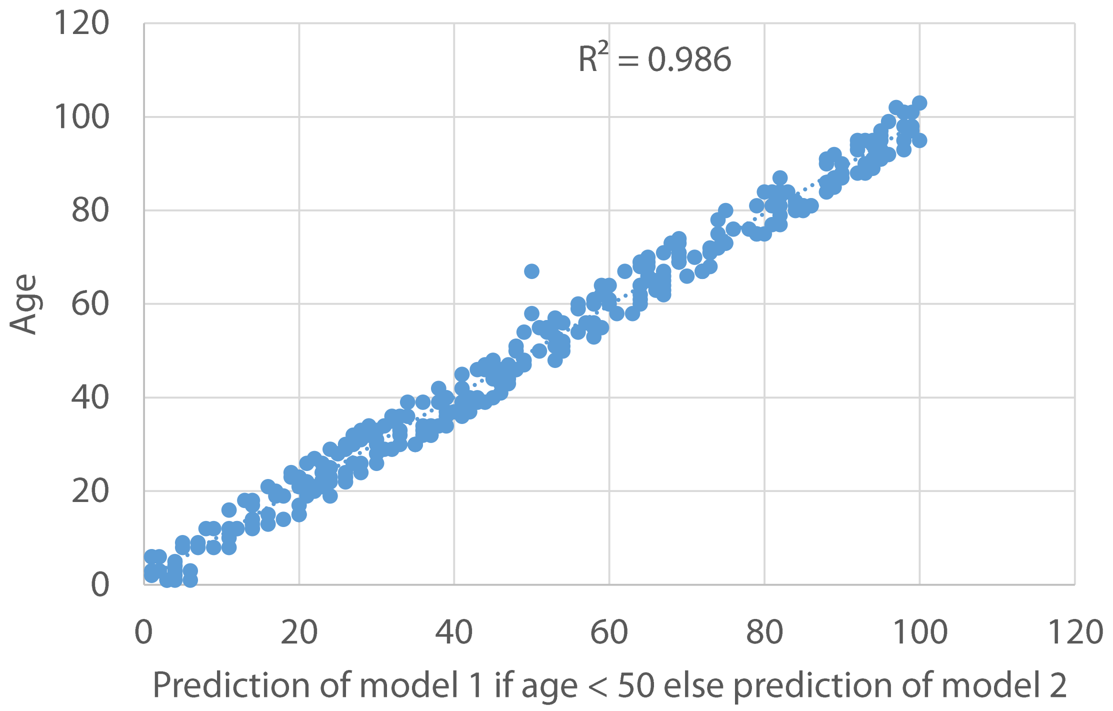
```
#split train data in 2 parts, training and  valdiation.
training,valid,ytraining,yvalid = train_test_split(train,y,test_size=0.5)
#specify models
model1=RandomForestRegressor()
model2=LinearRegression()
#fit models
model1.fit(training,ytraining)
model2.fit(training,ytraining)
#make predictions for validation
preds1=model1.predict(valid)
preds2=model2.predict(valid)
#make predictions for test data
test_preds1=model1.predict(test)
test_preds2=model2.predict(test)
#Form a new dataset for valid and test via stacking the predictions
stacked_predictions=np.column_stack((preds1,preds2))
stacked_test_predictions=np.column_stack((test_preds1,test_preds2))
#specify meta model
meta_model=LinearRegression()
#fit meta model on stacked predictions
meta_model.fit(stacked_predictions,yvalid)
#make predictions on the stacked predictions of the test data
final_predictions=meta_model.predict(stacked_test_predictions)
```

Stacking (past) example

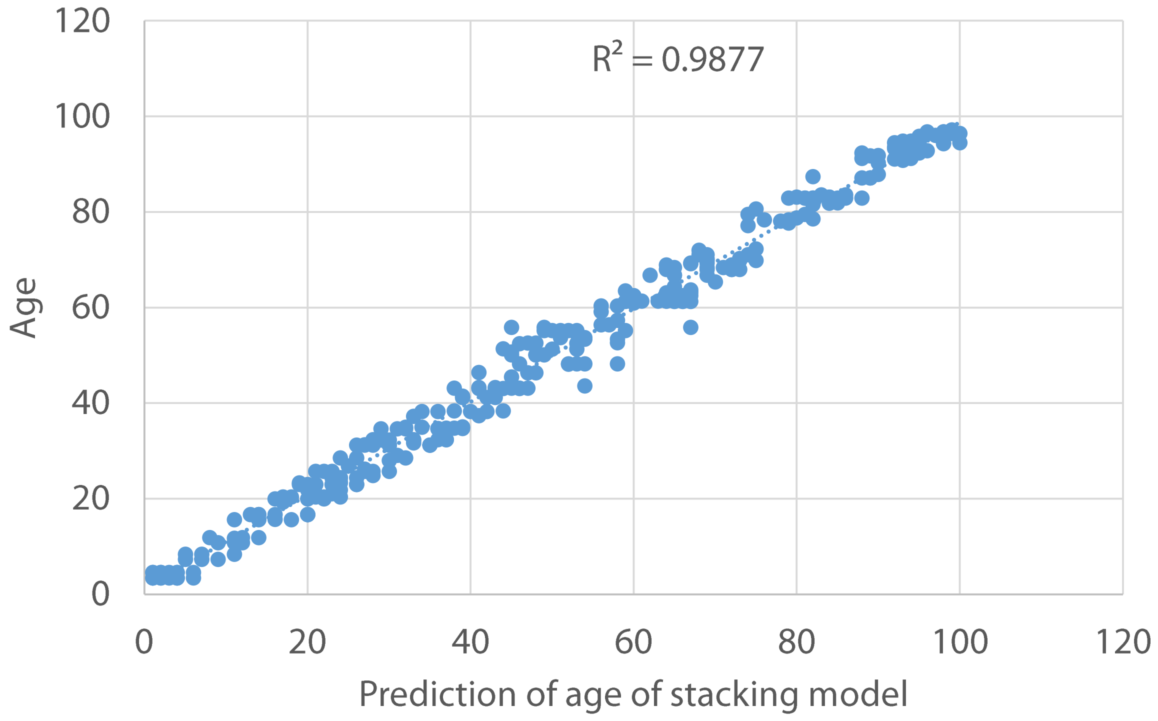


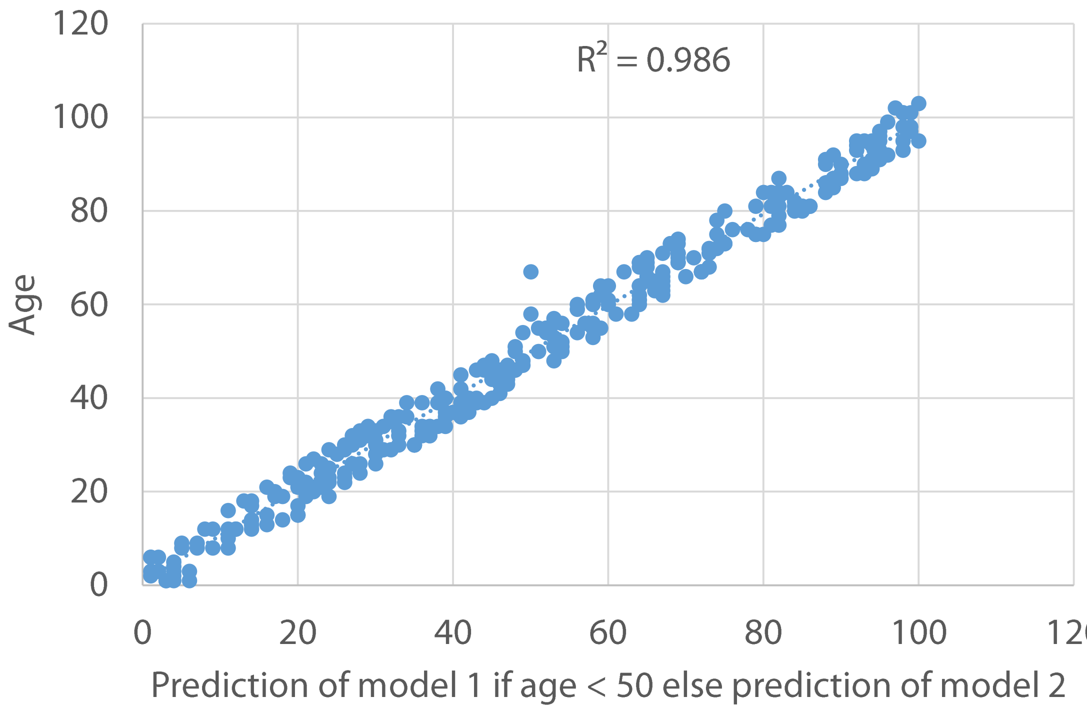


Stacking (past) example

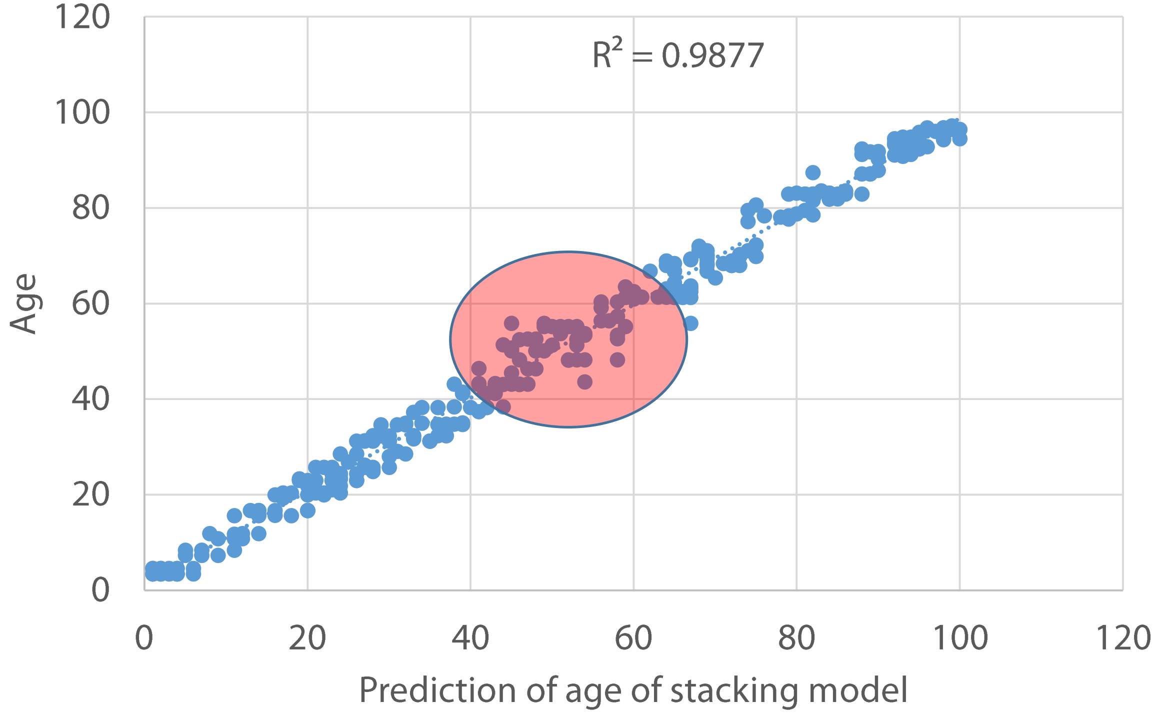


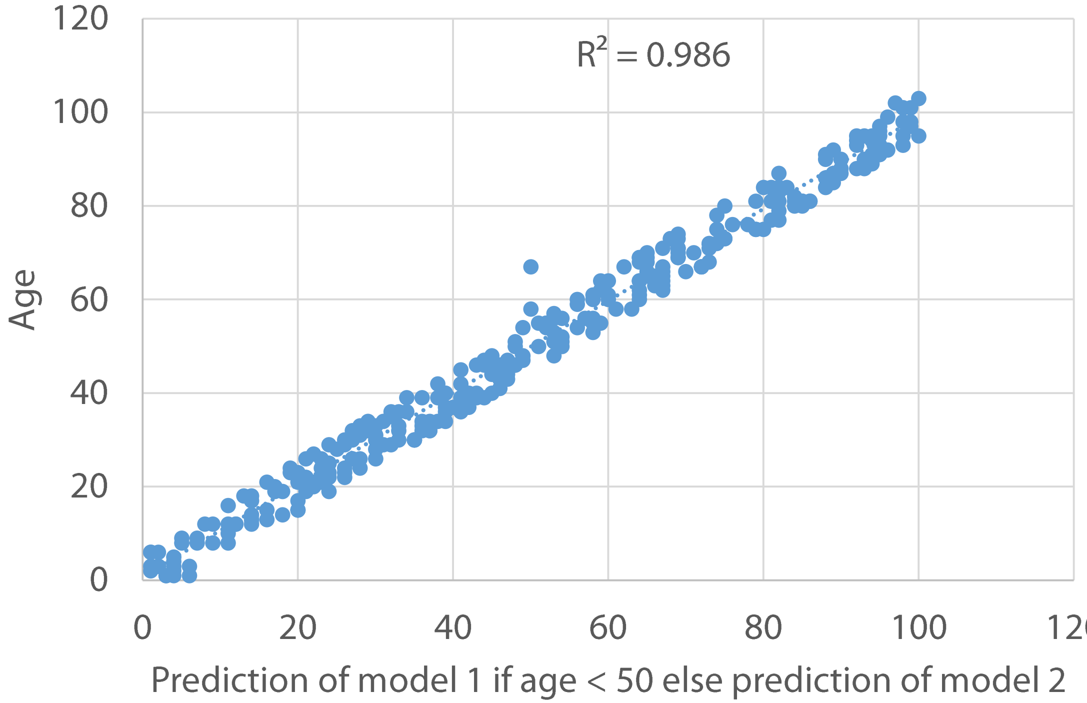
Stacking (past) example





Stacking (past) example





Things to be mindful of

- With time sensitive data – respect time
- Diversity as important as performance
- Diversity may come from:
    - Different algorithms
    - Different input features
- Performance plateauing after N models
- Meta model is normally modest

Ensemble methods: StackNet

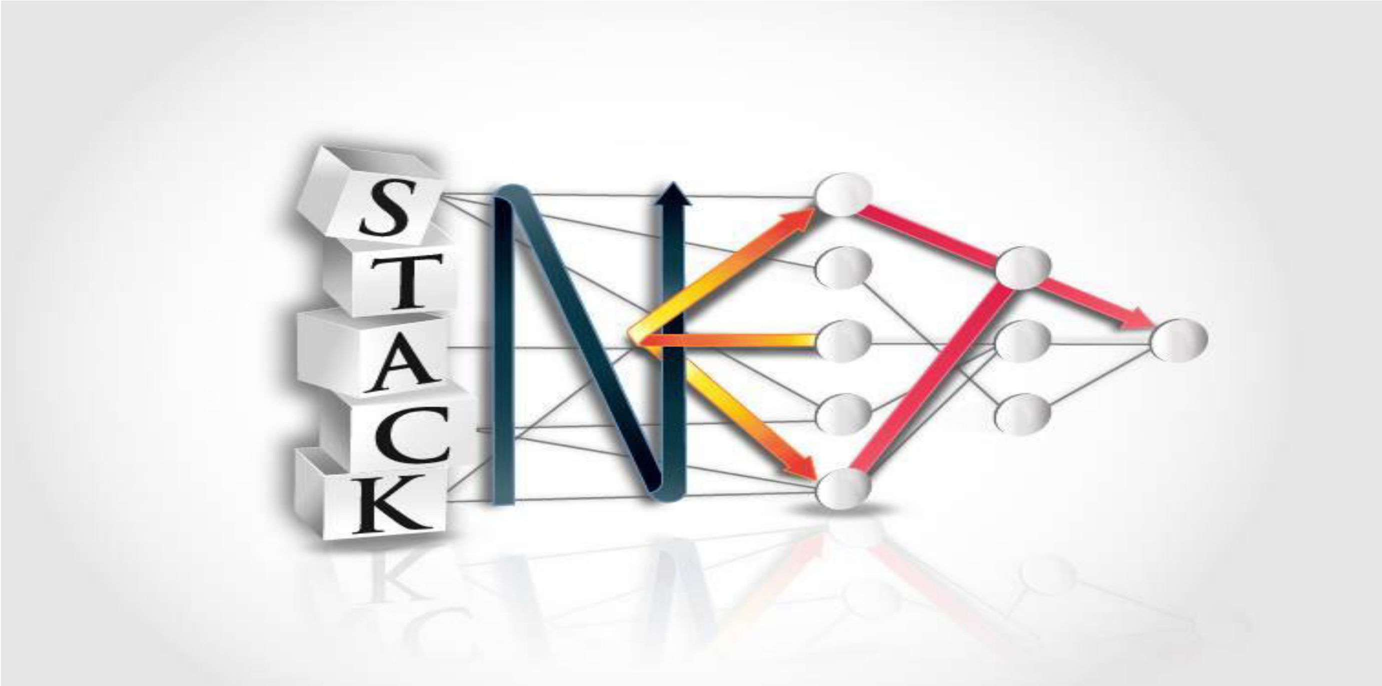
By Marios Michailidis

Examined ensemble methods

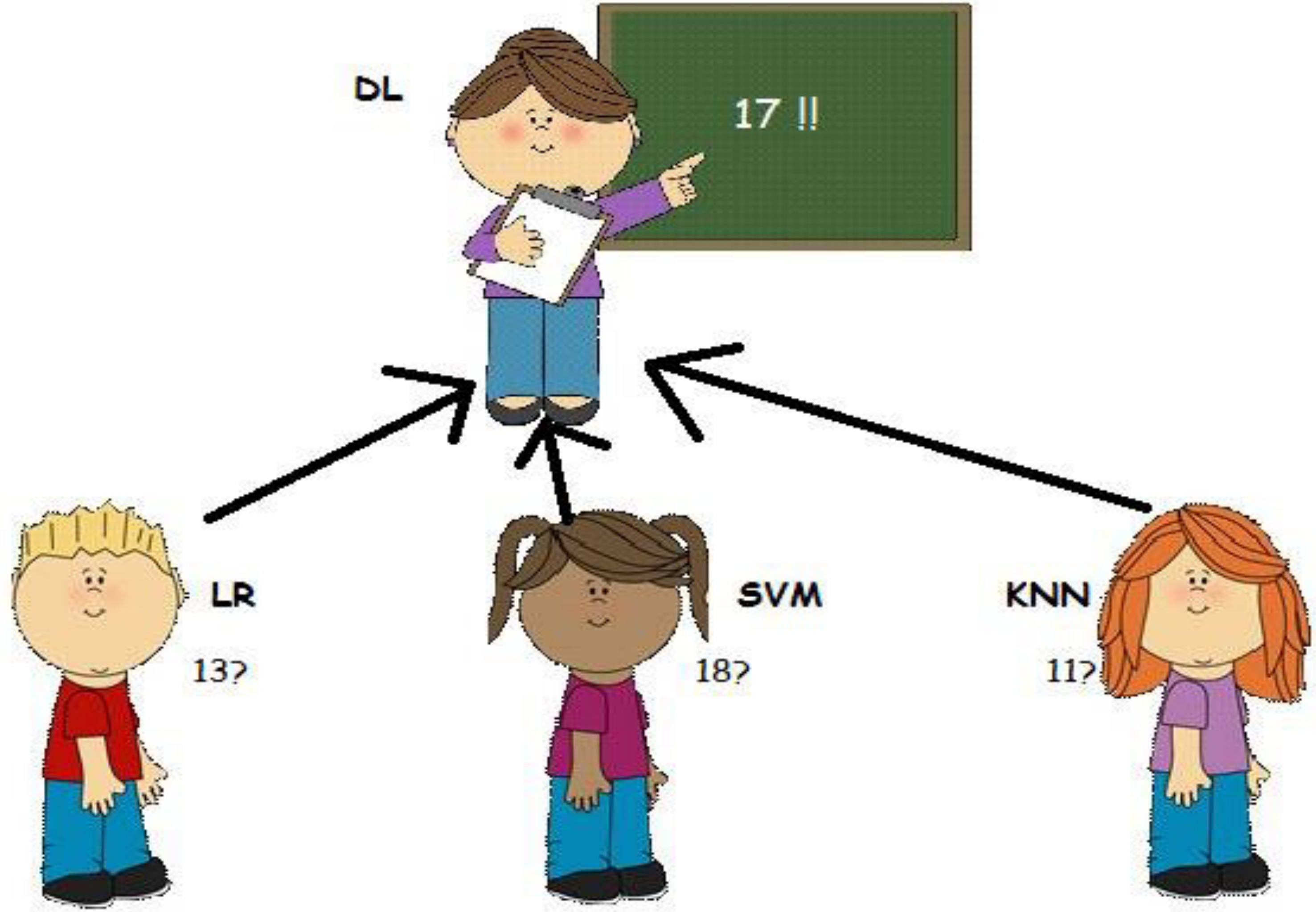
- Averaging (or blending)
- Weighted averaging
- Conditional averaging
- Bagging
- Boosting
- Stacking
- StackNet

What is StackNet

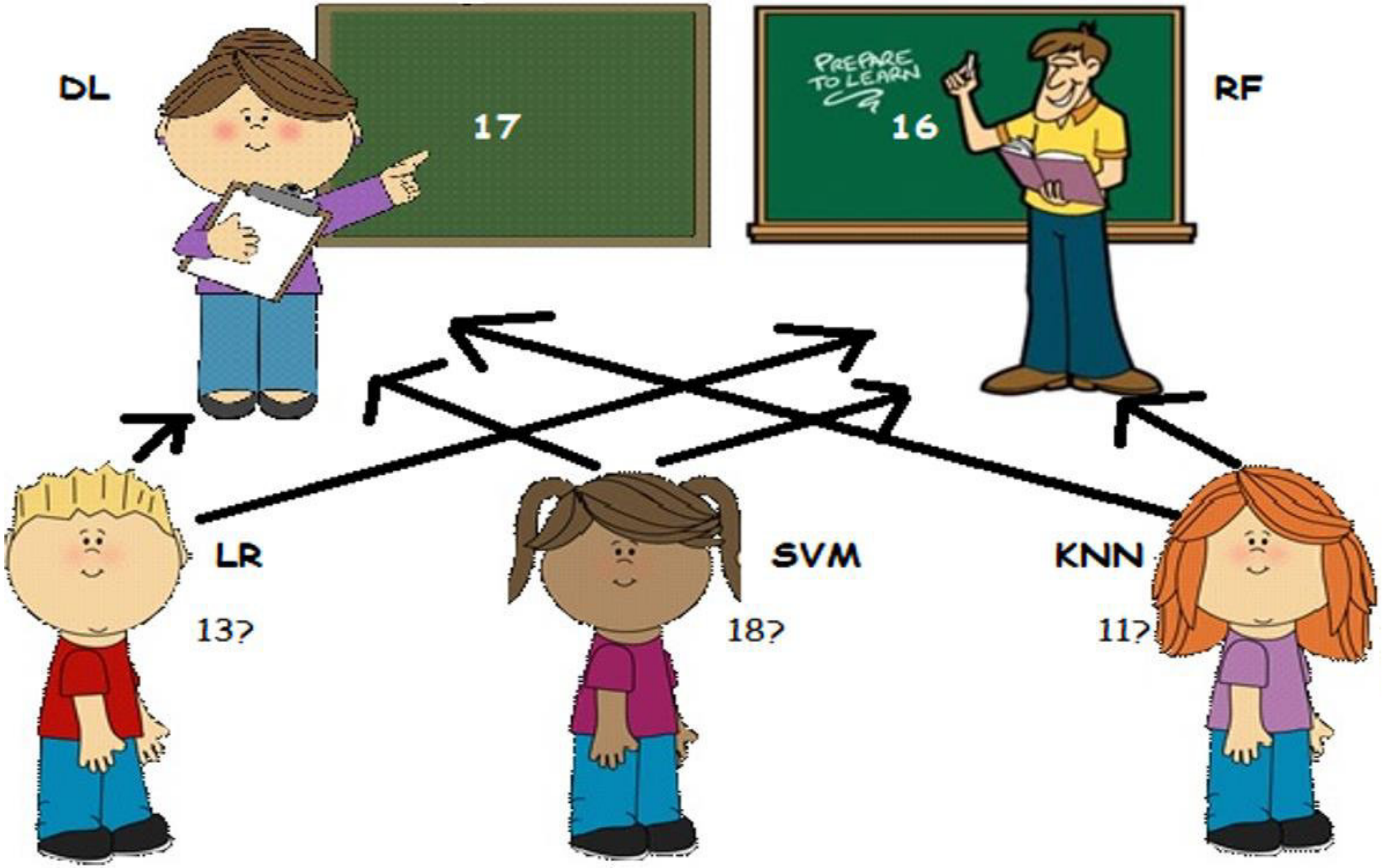
A scalable meta modelling methodology that utilizes stacking to combine multiple models in a neural network architecture of multiple levels.



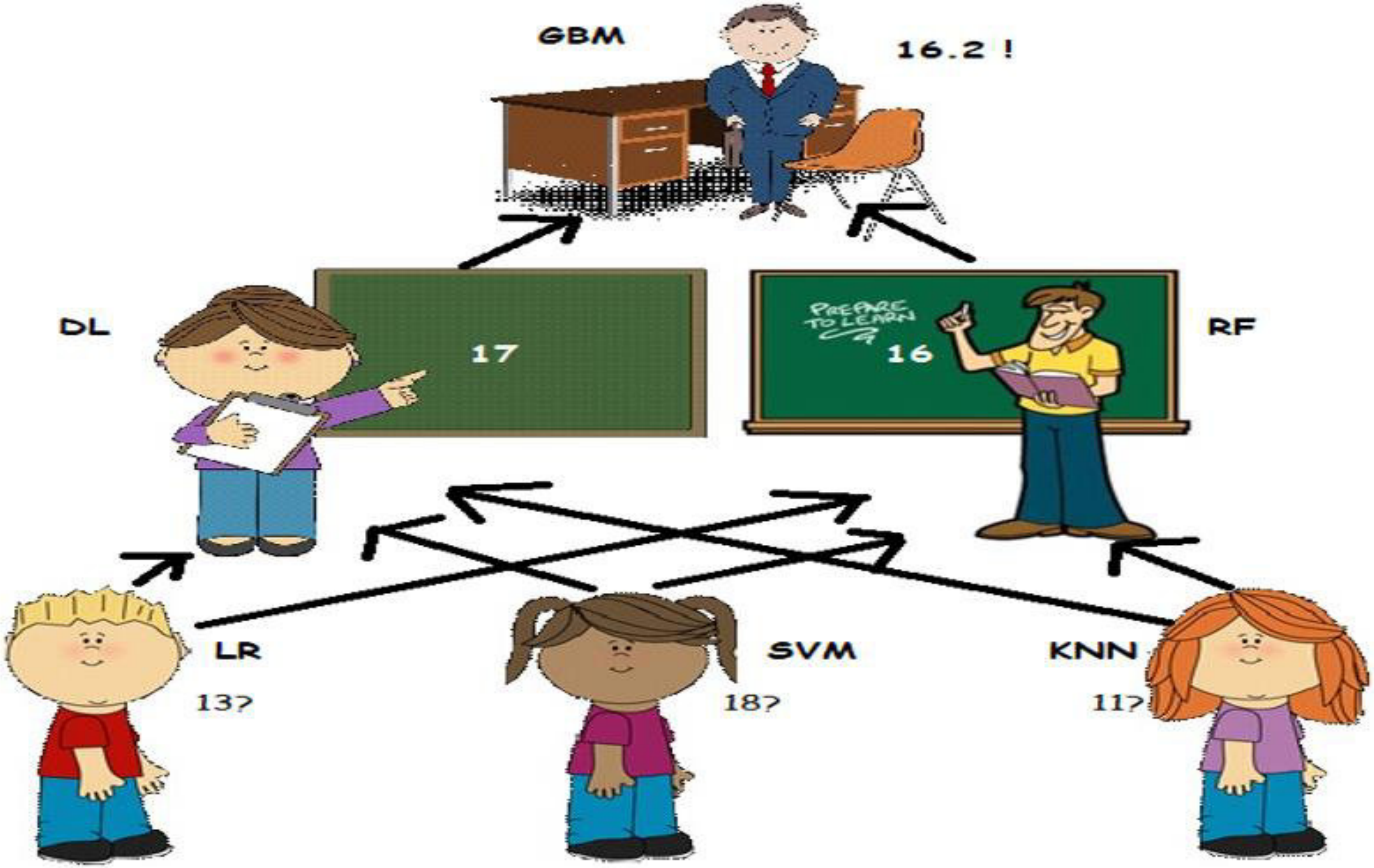
(Continuing) Naïve example



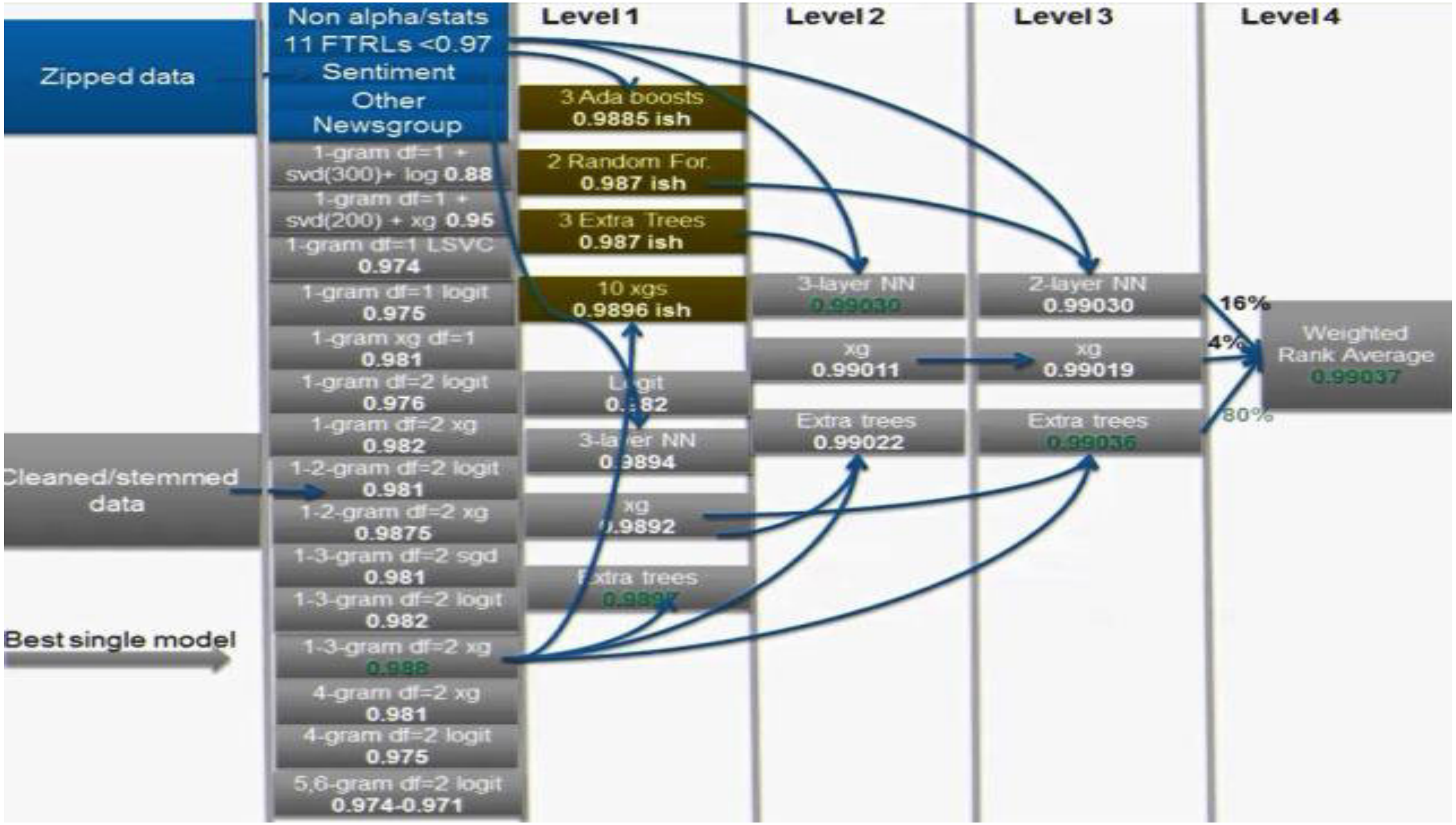
(Continuing) Naïve example



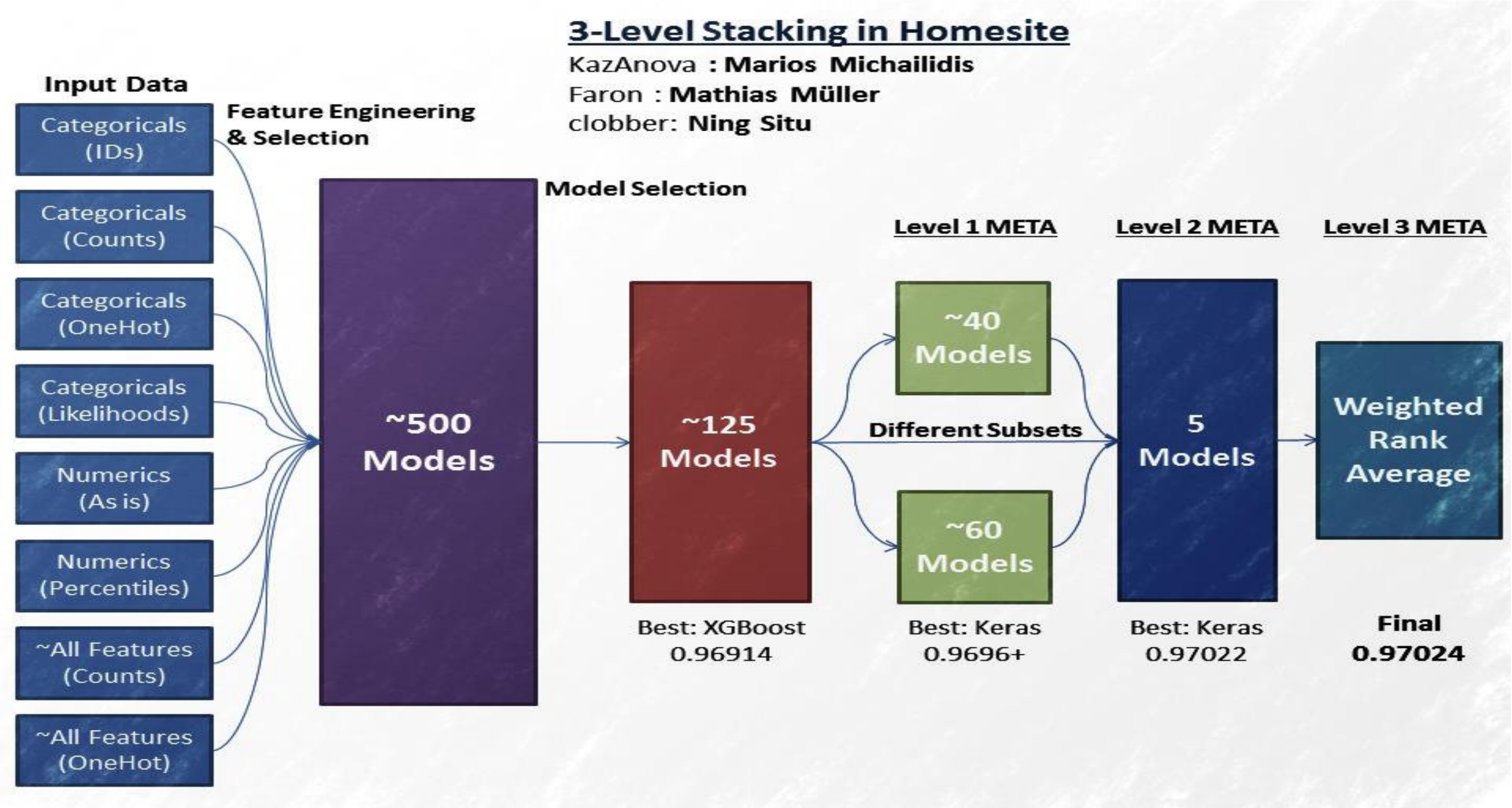
(Continuing) Naïve example



Why would this be of any use

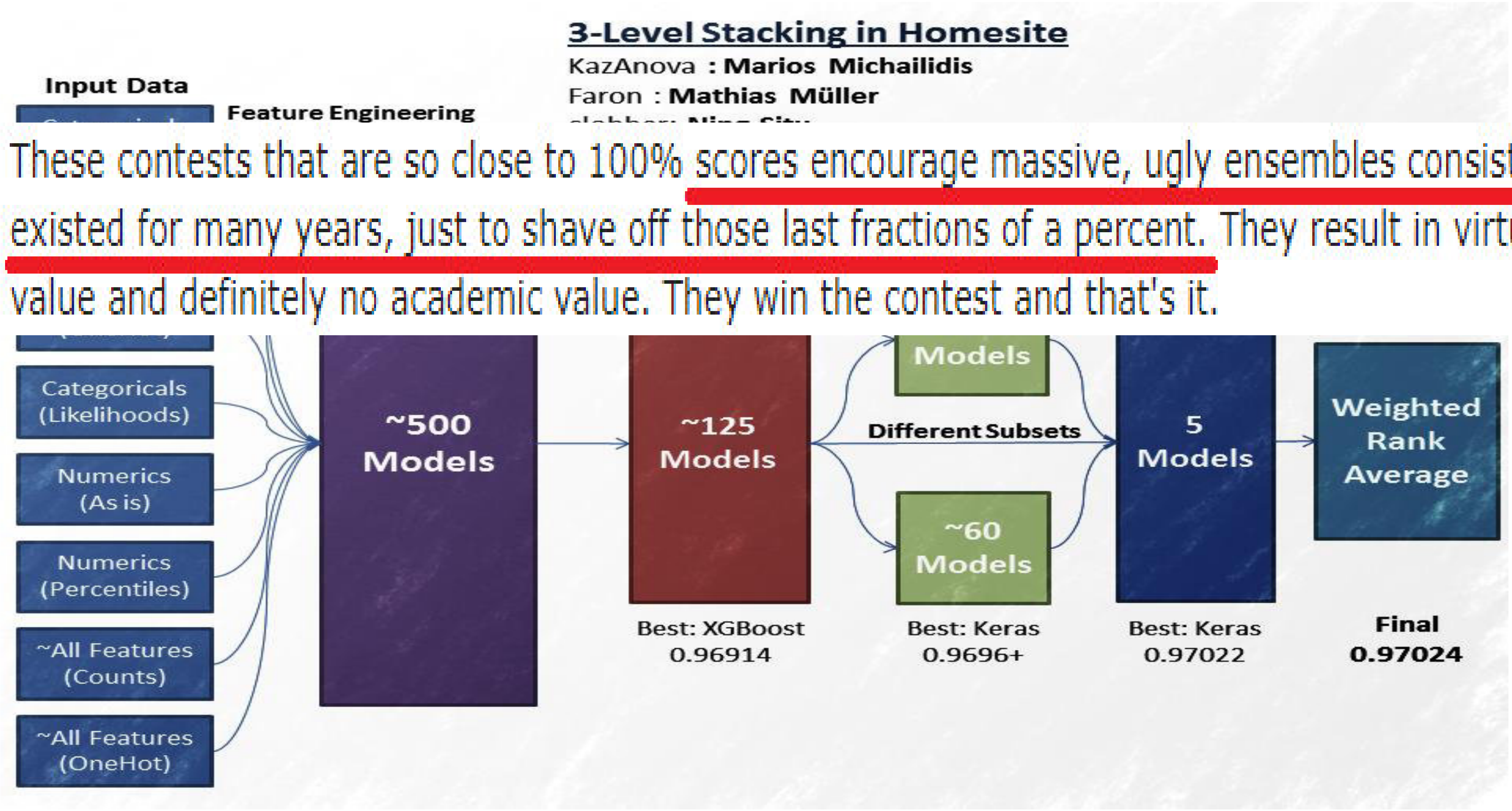


Why would this be of any use



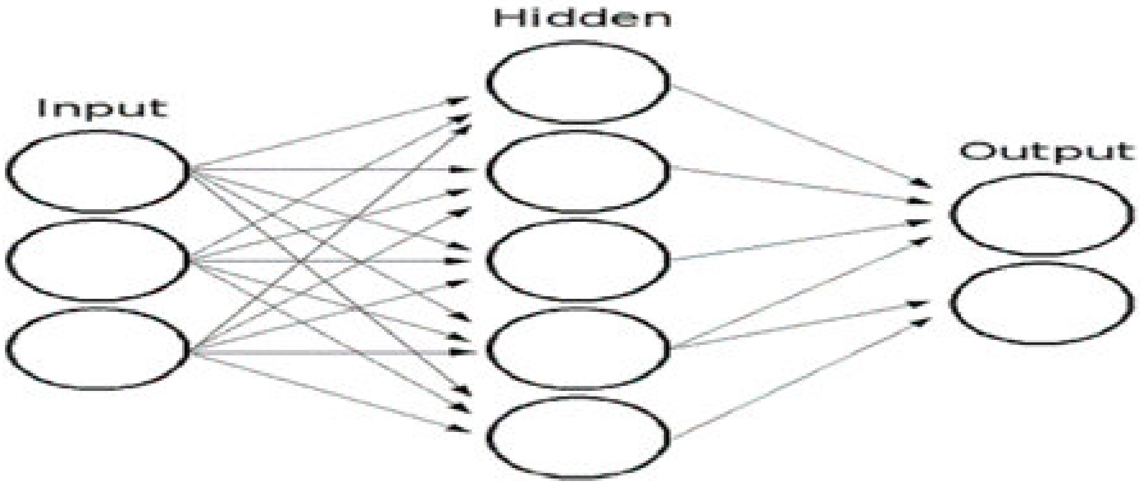
Why would this be of any use

These contests that are so close to 100% scores encourage massive, ugly ensembles consisting of old tech that's existed for many years, just to shave off those last fractions of a percent. They result in virtually no commercial value and definitely no academic value. They win the contest and that's it.



StackNet as a neural network

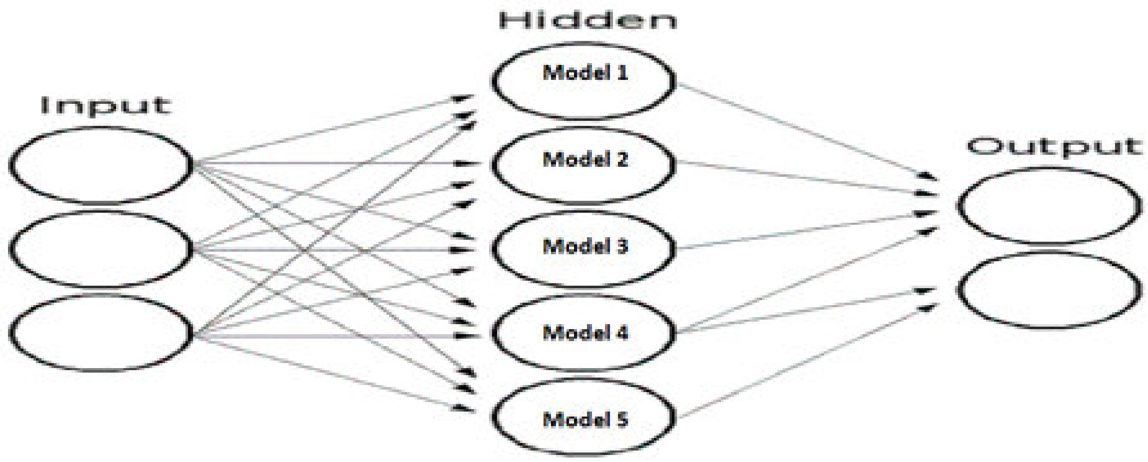
- In a neural network , every node is a **simple linear model** (like linear regression) with some non linear transformation.
- Instead of a linear model we could use **any model.**



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

StackNet as a neural network

- In a neural network , every node is a **simple linear model** (like linear regression) with some non linear transformation.
- Instead of a linear model we could use **any model.**



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

$$f_1(x_i, s) = g_1(x_i) s$$

How to train

- We cannot use **BP** (not all models are differentiable)
- We use **stacking** to link each model/node with target

How to train

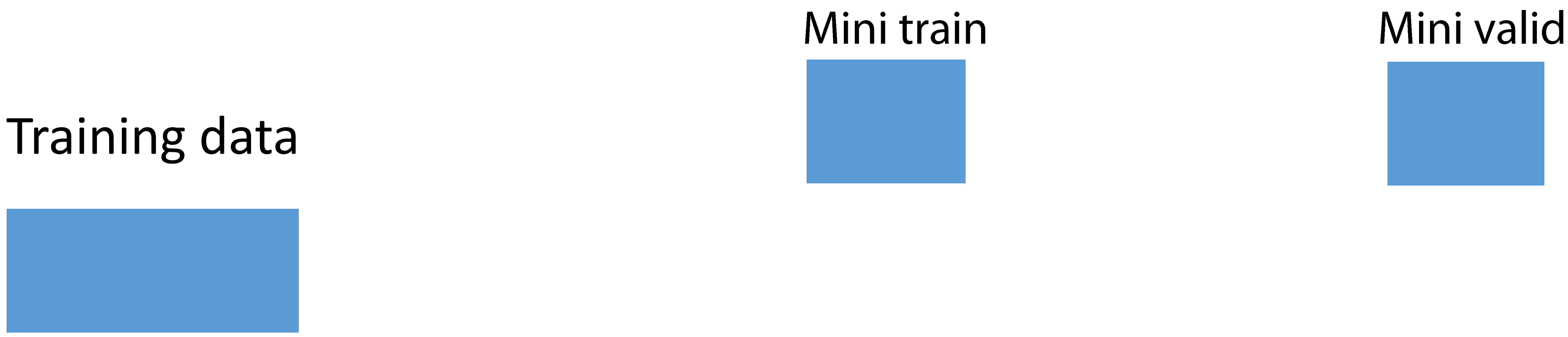
Train data

How to train

Training data

Valid data

How to train



How to train

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

How to train

K=4

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

How to train

K=4

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

pred
0.00
0.00
0.00
0.00
0.00
0.00
0.00
0.00

How to train

Fold : 1

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

pred
0.00
0.00
0.00
0.00
0.00
0.00
0.00
0.00

How to train

Fold : 1

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Train

pred
0.00
0.00
0.00
0.00
0.00
0.00
0.00
0.00

How to train

Fold : 1

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Predict

0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0

Train

0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

pred
0.00
0.00
0.00
0.00
0.00
0.00
0.00
0.00

How to train

Fold : 1

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Predict

0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0

Train

0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

pred
0.96
0.03
0.00
0.00
0.00
0.00
0.00
0.00

How to train

Fold : 2

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Predict

0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0

Train

0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

pred
0.96
0.03
0.00
0.00
0.00
0.00
0.00
0.00

How to train

Fold : 2

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Predict

0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0

Train

0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

pred
0.96
0.03
0.90
0.12
0.00
0.00
0.00
0.00

How to train

Fold : 3

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Predict

0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1

Train

0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

pred
0.96
0.03
0.90
0.12
0.00
0.00
0.00
0.00

How to train

Fold : 3

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Predict

0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1

Train

0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

pred
0.96
0.03
0.90
0.12
0.03
0.77
0.00
0.00

How to train

Fold : 4

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Predict

0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Train

0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1

pred
0.96
0.03
0.90
0.12
0.03
0.77
0.00
0.00

How to train

Fold : 4

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Predict

0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Train

0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1

pred
0.96
0.03
0.90
0.12
0.03
0.77
0.18
0.91

How to train

Fold : 4

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Predict

0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Train

0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1

pred
0.96
0.03
0.90
0.12
0.03
0.77
0.18
0.91

How to train

Fold : 4

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Predict

0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Train

0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1

test
0.43
0.03
0.90
0.12
0.03
0.77
0.18
0.91

pred
0.96
0.03
0.90
0.12
0.03
0.77
0.18
0.91

How to train

Fold : 4

x0	x1	x2	x3	y
0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1
0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Predict

0.08	0.72	0.97	0.04	0
0.84	0.79	0.89	0.05	1

Train

0.94	0.27	0.80	0.34	1
0.02	0.22	0.17	0.84	0
0.83	0.11	0.23	0.42	1
0.74	0.26	0.03	0.41	0
0.08	0.29	0.76	0.37	0
0.71	0.76	0.43	0.95	1

test
0.43
0.03
0.90
0.12
0.03
0.77
0.18
0.91

pred	pred
0.96	0.00
0.03	0.00
0.90	0.00
0.12	0.00
0.03	0.00
0.77	0.00
0.18	0.00
0.91	0.00

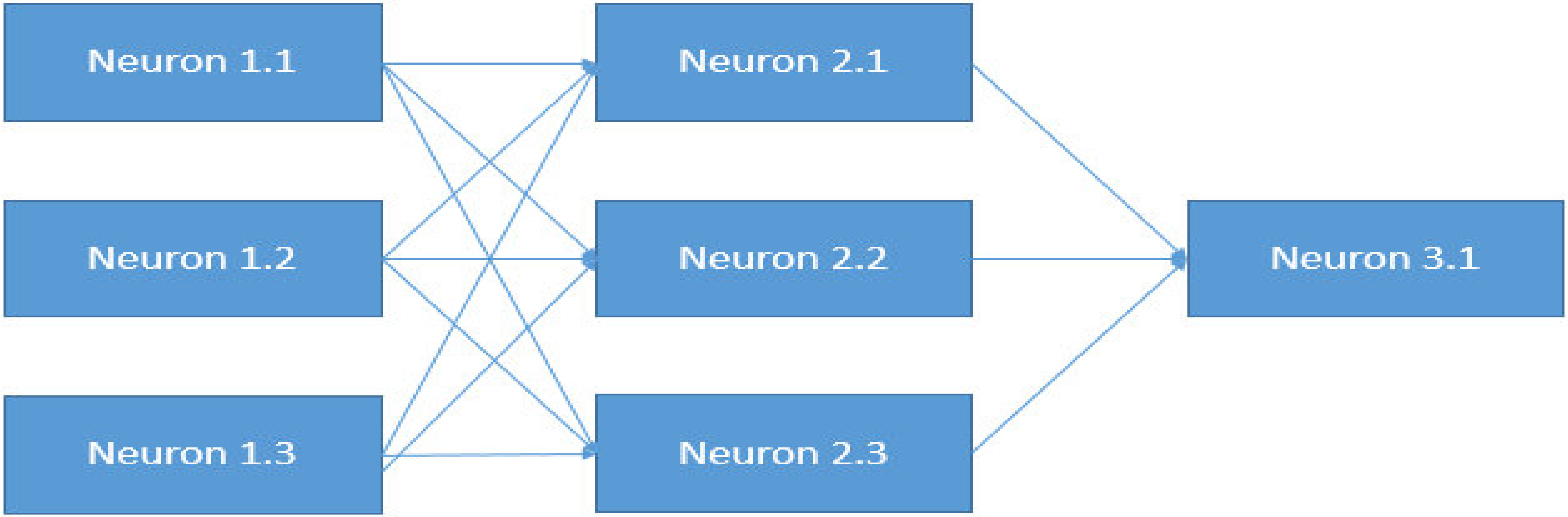
How to train

- We cannot use **BP** (not all models are differentiable)
- We use **stacking** to link each model/node with target
- To extend to many levels, we can use a **Kfold** paradigm

How to train

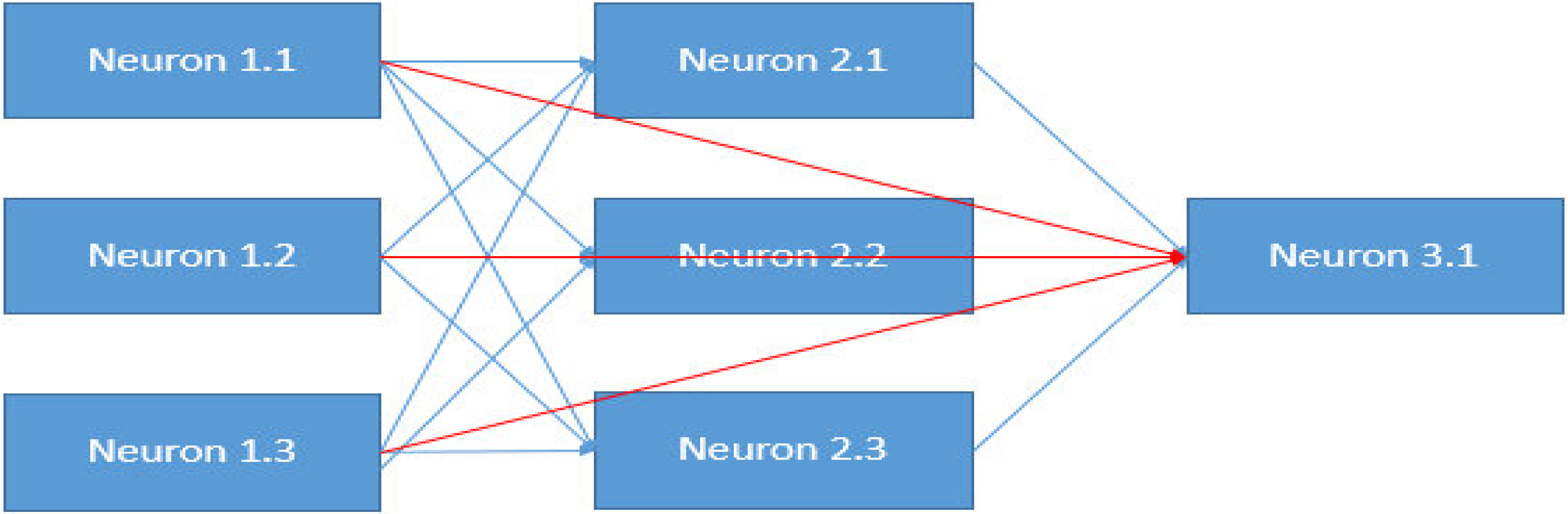
- We cannot use **BP** (not all models are differentiable)
- We use **stacking** to link each model/node with target
- To extend to many levels, we can use a **Kfold** paradigm
- No epochs – different connections instead.

How to train



Neuron 1.1
Neuron 1.2
Neuron 1.3
Neuron 2.1
Neuron 2.2
Neuron 2.3
Neuron 3.1

How to train



Neuron 1.1
Neuron 1.2
Neuron 1.3
Neuron 2.1
Neuron 2.2
Neuron 2.3
Neuron 3.1

1^{st} level tips

- Diversity based on algorithms:
  - ❑ 2-3 gradient boosted trees (lightgb, xgboost, H2O, catboost)
  - ❑2-3 Neural nets (keras, pytorch)
  - ❑1-2 ExtraTrees/Random Forest (sklearn)
  - ❑1-2 linear models as in logistic/ridge regression, linear svm (sklearn)
  - ❑1-2 knn models (sklearn)
  - ❑1 Factorization machine (libfm)
  - ❑1 svm with nonlinear kernel if size/memory allows (sklearn)
- Diversity based on input data:
  - ❑Categorical features: One hot, label encoding, target encoding, frequency
  - ❑Numerical features: outliers, binning, derivatives, percentiles, scaling
  - ❑Interactions : col1*/+-col2, groupby, unsupervised

Subsequent level tips

- Simpler (or shallower) Algorithms:
  - ❑ gradient boosted trees with small depth (like 2 or 3)
  - ❑ Linear models with high regularization
  - ❑ Extra Trees
  - ❑ Shallow networks (as in 1 hidden layer)
  - ❑ knn with BrayCurtis Distance
  - ❑ Brute forcing a search for best linear weights based on cv
- Feature engineering:
  - ❑ pairwise differences between meta features
  - ❑ row-wise statistics like averages or stds
  - ❑Standard feature selection techniques
- For every 7.5 models in previous level we add 1 in meta (empirical)
- Be mindful of target leakage

Software for Stacking

- StackNet (https://github.com/kaz-Anova/StackNet)
- Stacked ensembles from H2O
- Xcessiv (https://github.com/reiinakano/xcessiv)

Tips about StackNet (Software)

- It supports many prominent tools (xgboost, lightgbm, H2O, keras…)
- Can run classifiers in regression and vice versa.
- It has several top 10s in competitions.

Tips about StackNet (Software)

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	☐

#	Δpub	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

Tips about StackNet (Software)

- It supports many prominent tools (xgboost, lightgbm, H2O, keras…)
- Can run classifiers in regression and vice versa.
- It has several top 10s in competitions.
- The parameters' section.

Tips about StackNet (Software)

XgboostClassifier

The original parameters can be found here

```
XgboostClassifier booster:gbtree num_round:1000 eta:0.005 max_leaves:0 gamma:1. max_depth:5 min_child_weight:1.0 subs
```

Parameter	Explanation
scale_pos_weight	used for imbalanced classes(double)
num_round	Number of estimators to build (int) . **This is important.**
max_leaves	Maximum leaves in a tree (int).
eta	Penalty applied to each estimator. Needs to be between 0 and 1 (double). **This is important.**
max_depth	Maximum depth of the tree (int). **This is important.**
subsample	Proportion of observations to consider (double). **This is important.**
colsample_bylevel	Proportion of columns (features) to consider in each level (double).
colsample_bytree	Proportion of columns (features) to consider in each Tree (double) **This is important.**
max_delta_step	controls optimization step (double).

Before we say goodbye…

- Apply what you have learnt (in competitions).
- It takes some time to adjust.
- Always save your code and re-use it
- Seek collaborations
- Read forums/kernels


thank you
danke
рахмат
謝謝
ngiyabonga
teşekkür ederim
Баярлалаа
спасибо
faafetai lava
mersi
kia ora
barka
welalin
tack
spas
dank je
misaotra
matondo
paldies
grazzi
mahalo
tapadh leat
gracias
vinaka
спасибі
blagodaram
kiitos
dankie
nanni
nandri
dhanyavad
mauruuru
hvala
bedankt
bayarlalaa
gràçie
köszönöm
enkosi
dziękuję
dankon
aciu
akun
хвала
asante
manana
obrigada
tenki
murakoze
djiere dieuf
tau
mochchakkeram
mamnun
дякую
chokrane
chnorakaloutioun
gratias ago
gràcies
sulpáy
taiku
go raibh maith agat
sobodi
děkuji
obrigado
sagolun
sukriya
kop khun krap
arigatô
takk
dakujem
trugarez
mèsi
didi madloba
najis tuke
ありがとう
tanemirt
grazie
shukriya
mercé
мерси
kam sah hamnida
rahmat
terima kasih
rahmet
diolch
dhanyavadagalu
তোমাকে ধন্যবাদ
감사합니다
xièxie
ευχαριστώ
merci

Rank	Tier	User		Medals			Points
1		You	joined a year ago	999	0	0	994,882
2		Stanislav Semenov	joined 4 years ago	28	9	0	190,356
3		Μαριος Μιχαηλιδης **KazAnova**	joined 4 years ago	26	23	21	168,976
4		Faron	joined 3 years ago	14	4	3	132,862
5		Eureka	joined 4 years ago	16	13	3	131,759
6		raddar	joined 2 years ago	9	6	3	119,285
7		idle_speculation	joined 4 years ago	7	8	6	116,367
8		weiwei	joined a year ago	5	3	1	108,836
9		bestfitting	joined a year ago	5	3	0	107,497
10		Silogram	joined 5 years ago	10	24	9	97,850
11		utility	joined 3 years ago	13	7	3	95,855