Power of Ensembles

Bargava Subramanian
Data Scientist
Cisco Systems, India

Two huntsmen go bird-hunting. Both huntsmen can hit a target with probability of 0.2.

They see a flock of 150 birds, atop a banyan tree. First huntsman takes aim and fires three continuous shots. A minute after that, the second huntsman fires three shots at the banyan tree.

How many birds did the second huntsman shoot?

How many birds did the second huntsman shoot?

And then, there were none

Your model is only as good as you (and your features)

Feature identification/ creation/generation takes a lot of time

Two different models with same features can result in different outputs

Why?

Two different models with same features can result in different outputs

Searched different regions of the solution space

Some common problems faced by modelers

1. Different models
2. Model parameters
3. Number of features

Possible Solution Approach?

Ensemble models are our friends

What is an ensemble?

A toy example

	Random Forest	Gradient Boosting	Logistic Regression
	0	1	1
	1	0	0
	1	0	1
	1	0	1
	1	1	0
	1	1	1
	0	1	1
	1	1	0
	0	1	1
	1	1	1
Accuracy	**70%**	**70%**	**70%**

Ground Truth: All 1’s

A simple ensemble - `max count`

	Random Forest	Gradient Boosting	Logistic Regression	Ensemble Output
	0	1	1	1
	1	0	0	0
	1	0	1	1
	1	0	1	1
	1	1	0	1
	1	1	1	1
	0	1	1	1
	1	1	0	1
	0	1	1	1
	1	1	1	1
Accuracy	**70%**	**70%**	**70%**	**90%**

Ground Truth: All 1's

CPU as a proxy for human IQ

Clever Algorithmic way to search the solution space

But is it new?

But is it new?

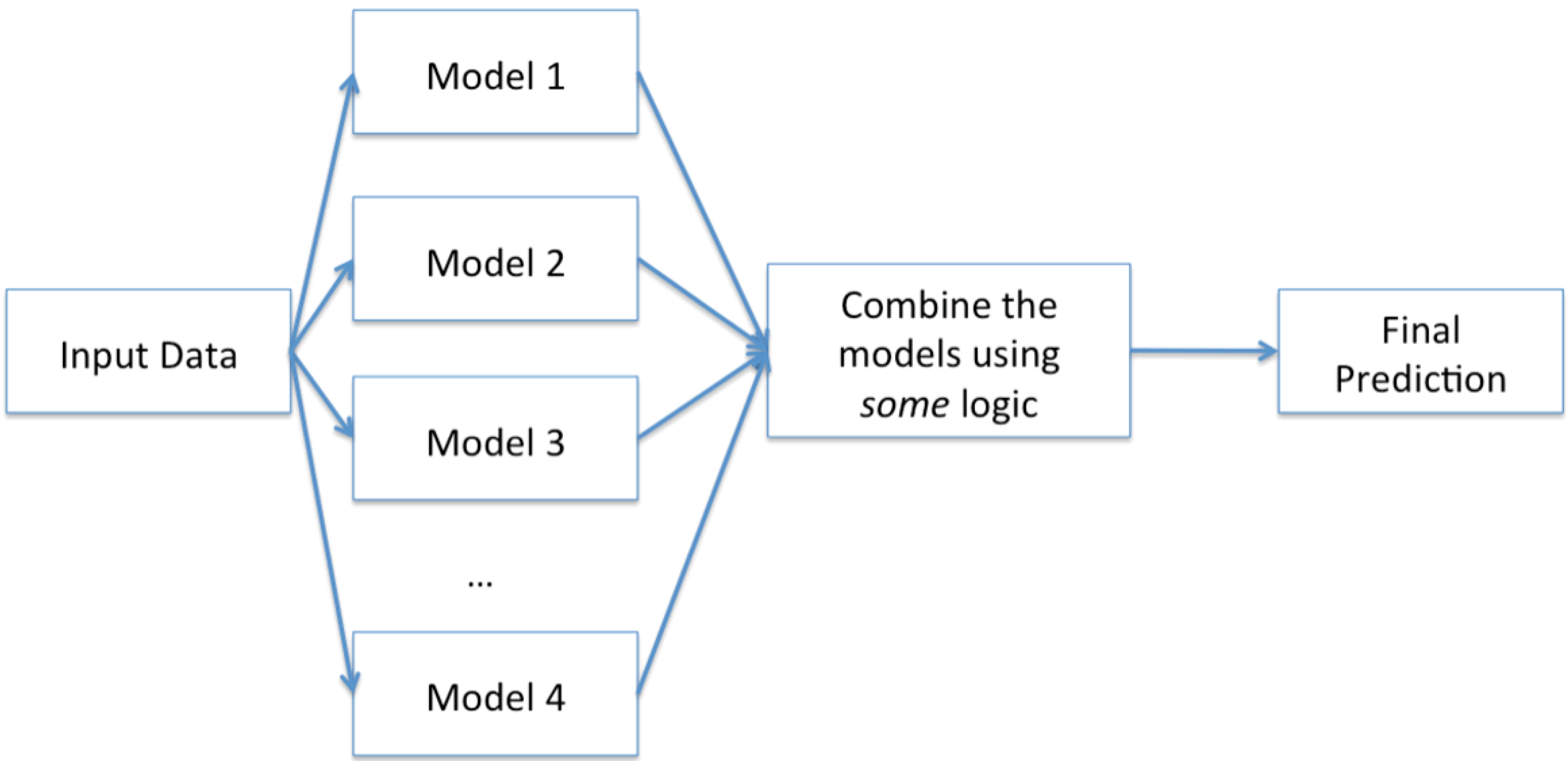
Known to researchers/academia for long.

Wasn't widely used in industry until....

Success Story

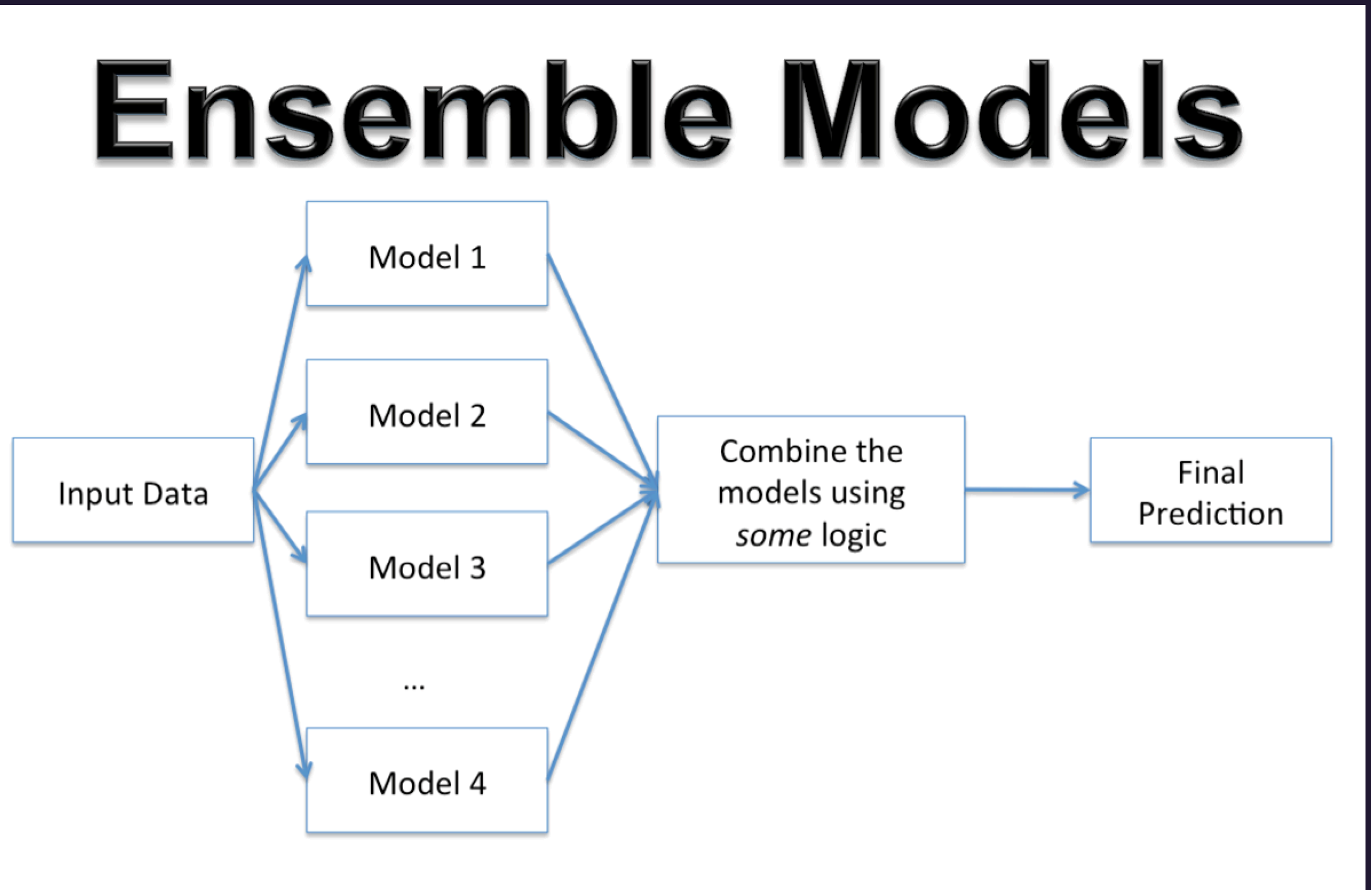
Netflix $ 1 million prize competition

Ensemble Models



Some Advantages

1. Improved accuracy
2. Robustness
3. Parallelization



Base model diversity

Model aggregation

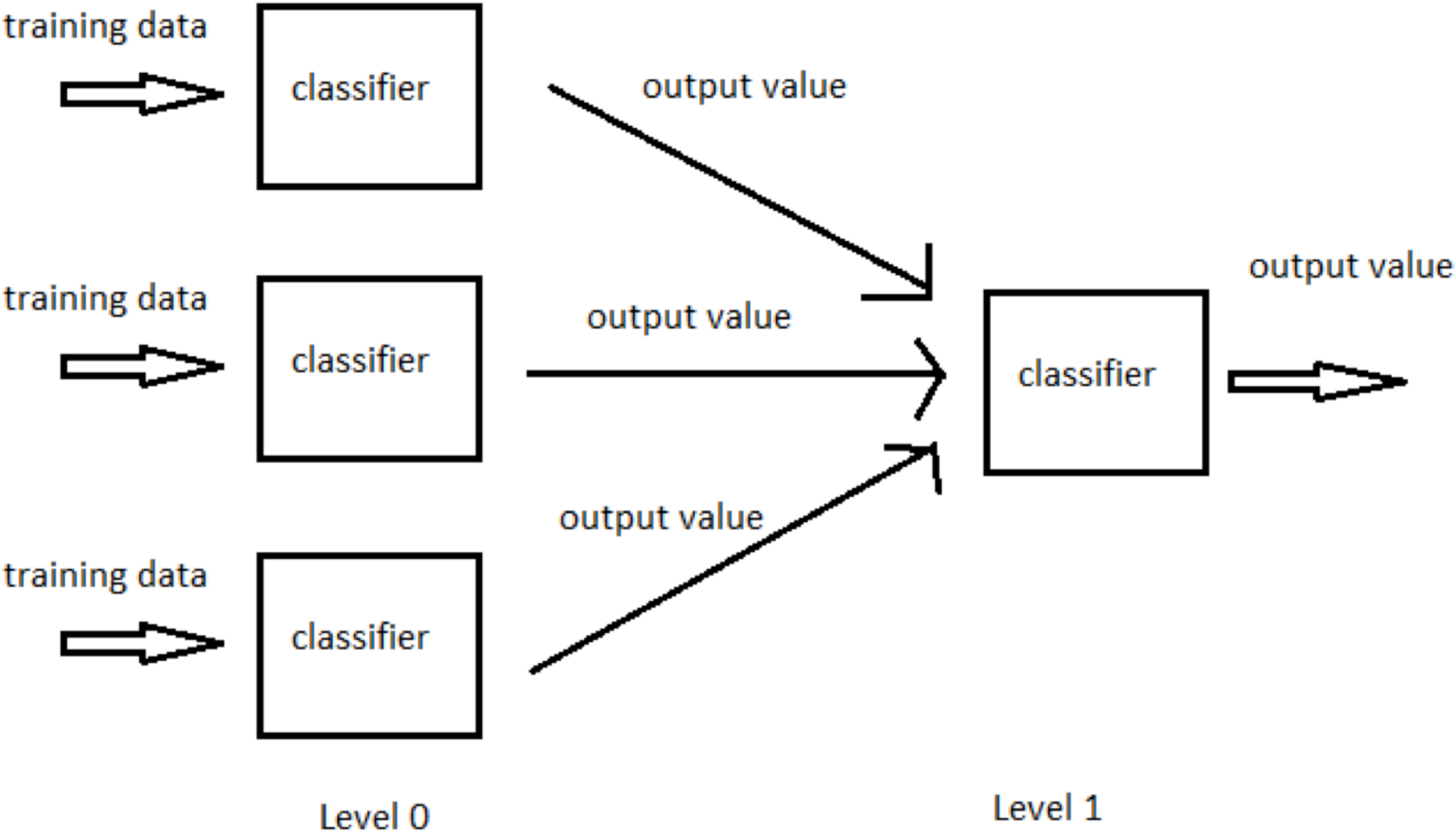
Base Model

1. Different training sets
2. Feature sampling
3. Different algorithms
4. Different Hyperparameters

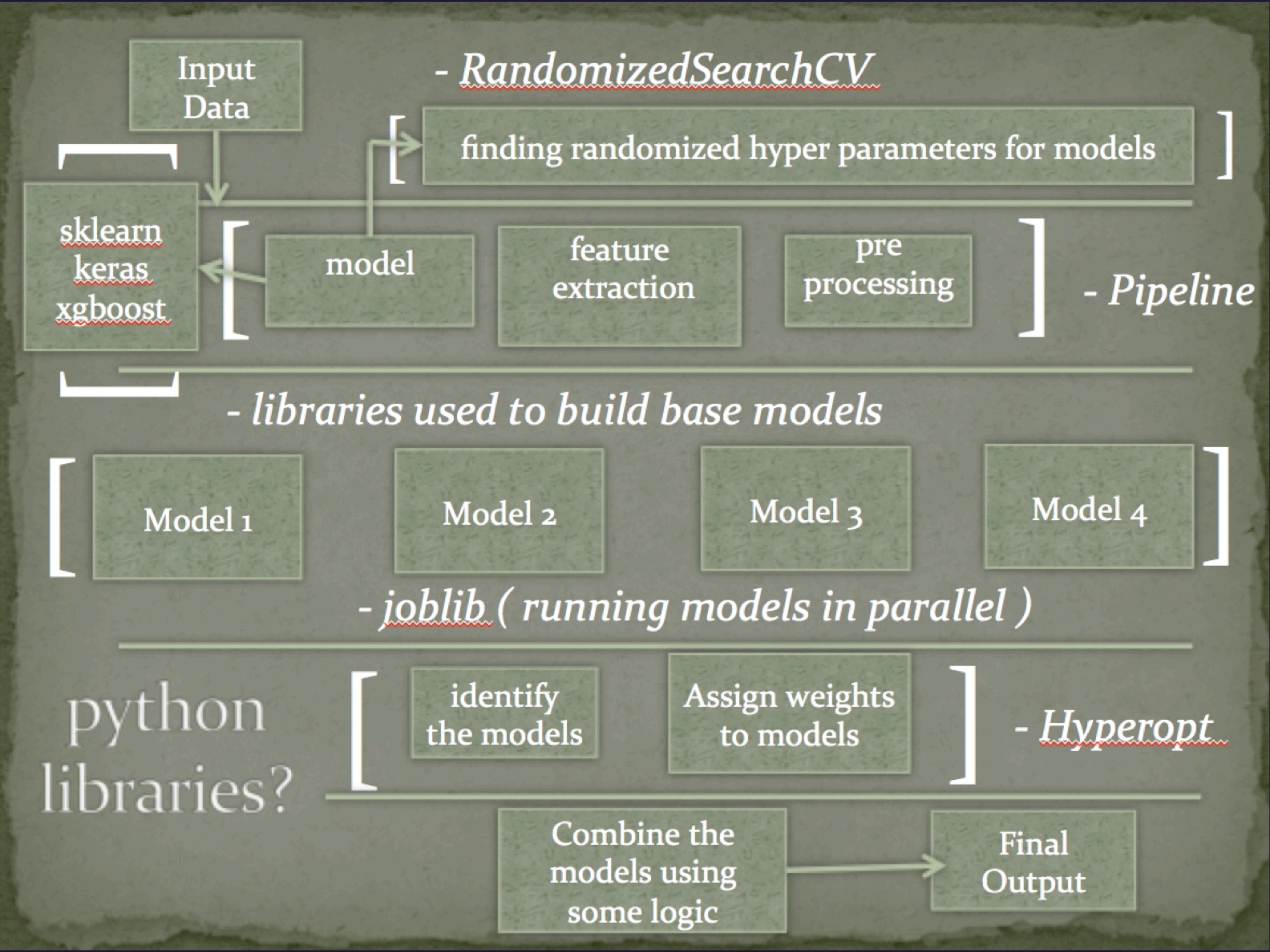
Model Aggregation

1. Voting
2. Averaging
3. Bagging
4. Stacking

Concept Diagram of Stacking



WHERE IS PYTHON ?


Input Data
- RandomizedSearchCV
finding randomized hyper parameters for models
sklearn
keras
xgboost
model
feature extraction
pre processing
- Pipeline
- libraries used to build base models
Model 1
Model 2
Model 3
Model 4
- joblib (running models in parallel)
python libraries?
identify the models
Assign weights to models
- Hyperopt
Combine the models using some logic
Final Output

RandomizedSearchCV

```
from scipy.stats import randint as sp_randint

from sklearn.grid_search import GridSearchCV, RandomizedSearchCV
# build a classifier
clf = RandomForestClassifier(n_estimators=20)
# specify parameters and distributions to sample from
param_dist = {"max_depth": [3, None],
              "max_features": sp_randint(1, 11),
              "min_samples_split": sp_randint(1, 11),
              "min_samples_leaf": sp_randint(1, 11),
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy”]}
# run randomized search
n_iter_search = 20
random_search = RandomizedSearchCV(clf, param_distributions=param_dist,
                                   n_iter=n_iter_search)
```

hyperopt

Python library for serial and parallel optimization over awkward search spaces, which may include real-valued, discrete, and conditional dimensions.

https://github.com/hyperopt/hyperopt

hyperopt

```
# define an objective function
def objective(args):
# Define the objective function here

# define a search space
from hyperopt import hp
space = hp.choice('a',
    [
        ('Model 1', randomForestModel),
        ('Model 2', xgboostModel)
    ])

# minimize the objective over the space
from hyperopt import fmin, tpe
best = fmin(objective, space, algo=tpe.suggest, max_evals=100)
```

joblib

1. transparent disk-caching of the output values and lazy re-evaluation (memoize pattern)
2. easy simple parallel computing
3. logging and tracing of the execution

joblib

```
import pandas as pd
from sklearn.externals import joblib

# build a classifier
train = pd.read_csv('train.csv')
clf = RandomForestClassifier(n_estimators=20)
clf.fit(train)

# once the classifier is built we can store it as a synchronized object
# and can load it later and use it to predict, thereby reducing memory footprint.

joblib.dump(clf, 'randomforest_20estimator.pkl')
clf = joblib.load('randomforest_20estimator.pkl')
```

Disadvantages

1. Model human readability isn't great
2. Time/Effort trade-off to improve accuracy may not make sense

Questions ?