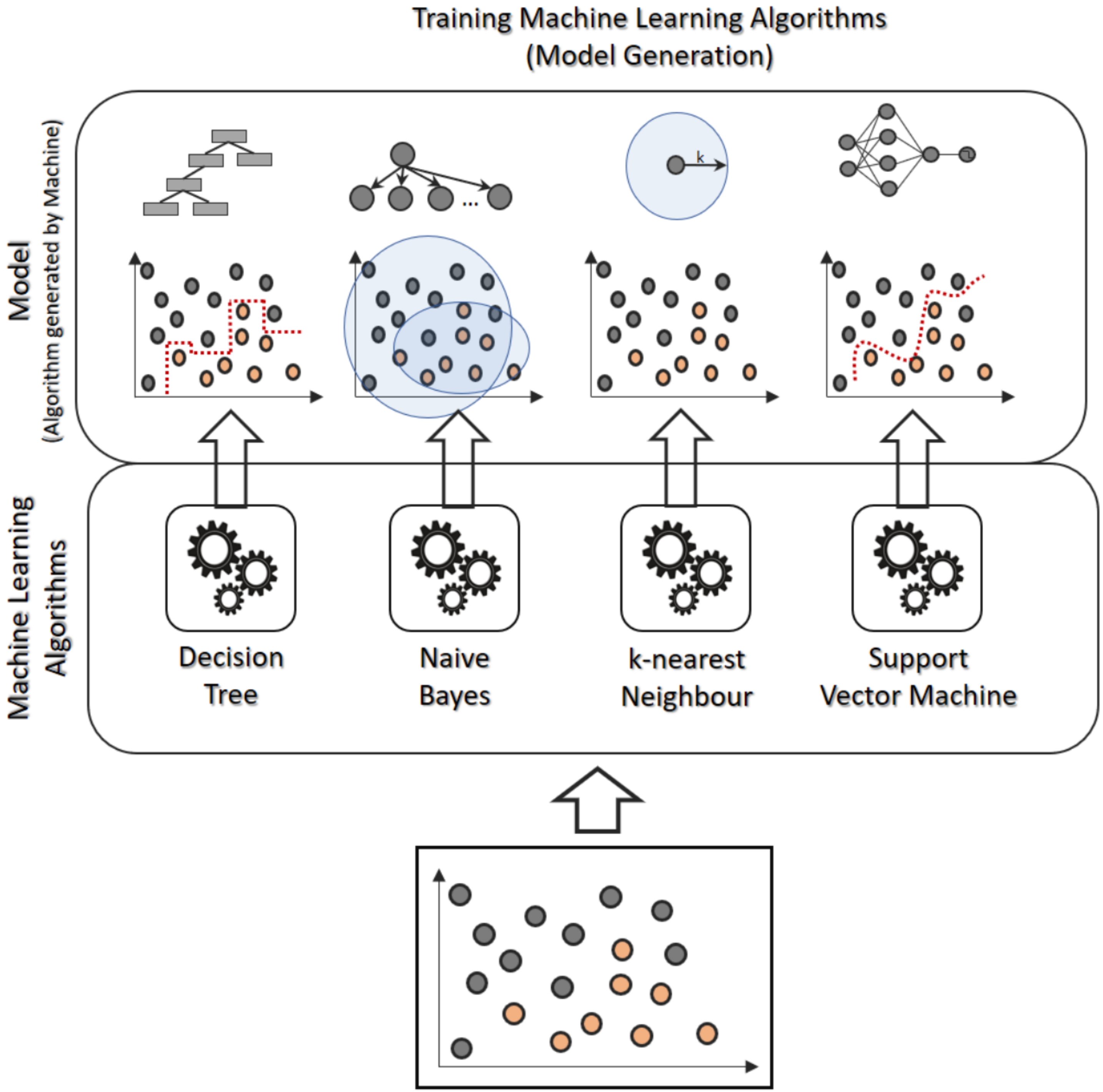
Ensemble Learning via Collaborative Filtering

Barnett Chiu

10.01.19

Ensemble Learning

- Ensemble learning methods combine the decisions from multiple models to improve the overall performance.
  - Key idea: tradeoff between diversity and accuracy
  - Decreases variance and generalization errors
- Homogeneous ensemble
- Heterogeneous ensemble

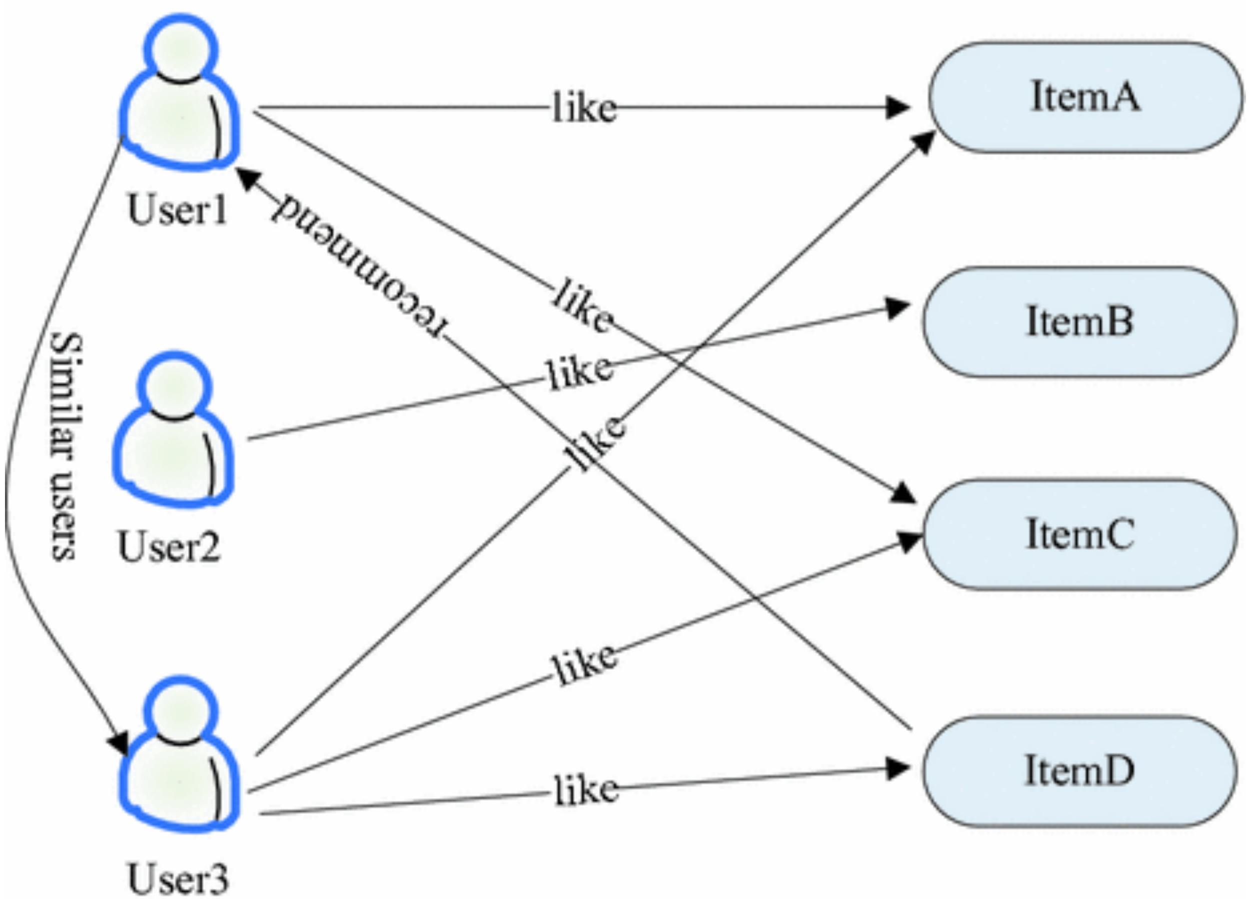

Training Machine Learning Algorithms
(Model Generation)
Model
(Algorithm generated by Machine)
k
Machine Learning
Algorithms
Decision
Tree
Naive
Bayes
k-nearest
Neighbour
Support
Vector Machine

CF Ensemble Learning

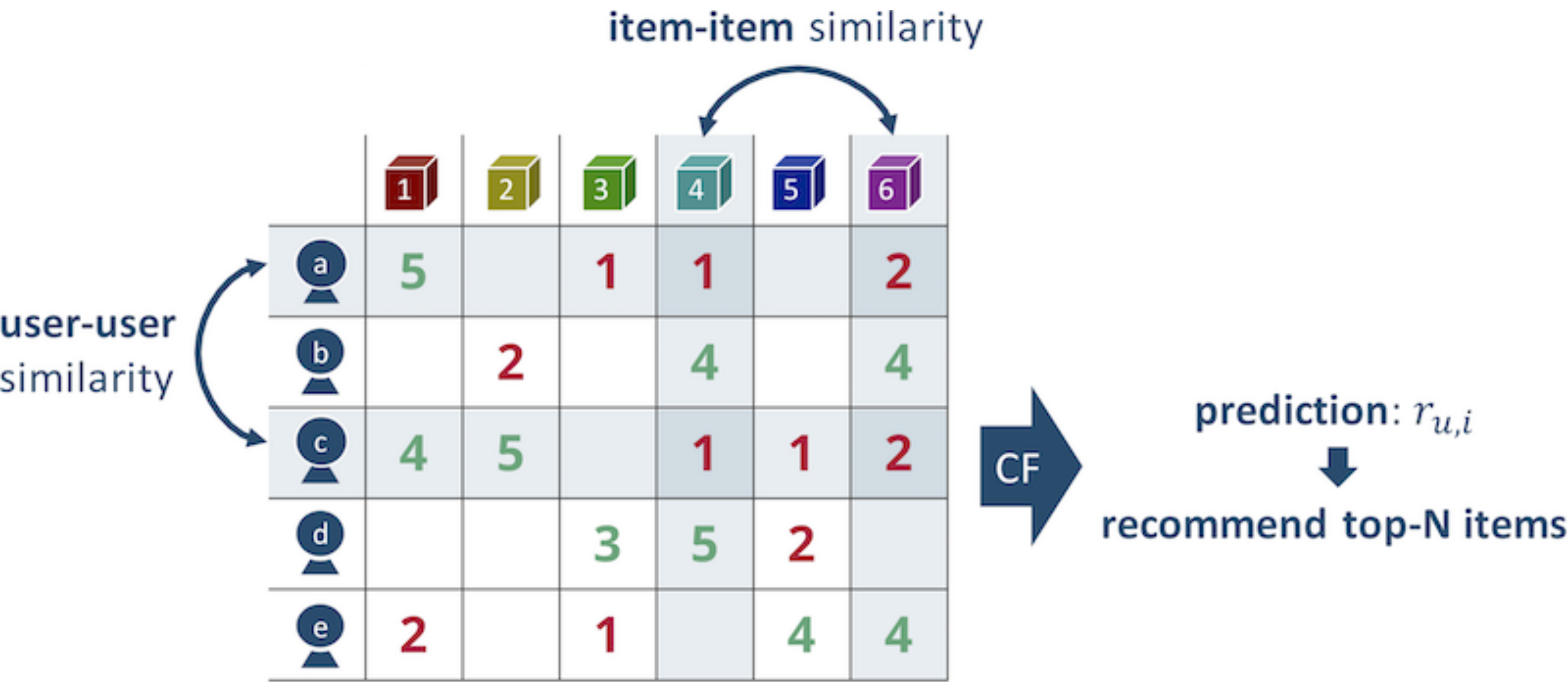
- Collaborative filtering
- Operated mainly in the context of heterogeneous ensemble learning settings, in which base predictors are assumed to be quite different
  - In general, applicable as long as the base predictor outputs are in the form of probability scores
- CF Ensemble deals with probability matrices (i.e. prediction matrices of the base predictors)
  - Probability matrix is analogous to rating matrix in CF settings
  - A method of stacking (or stacked generalization)
  - It attempts to identify unreliable entries in the probability matrix and works its way to re-estimate probability values for these entries

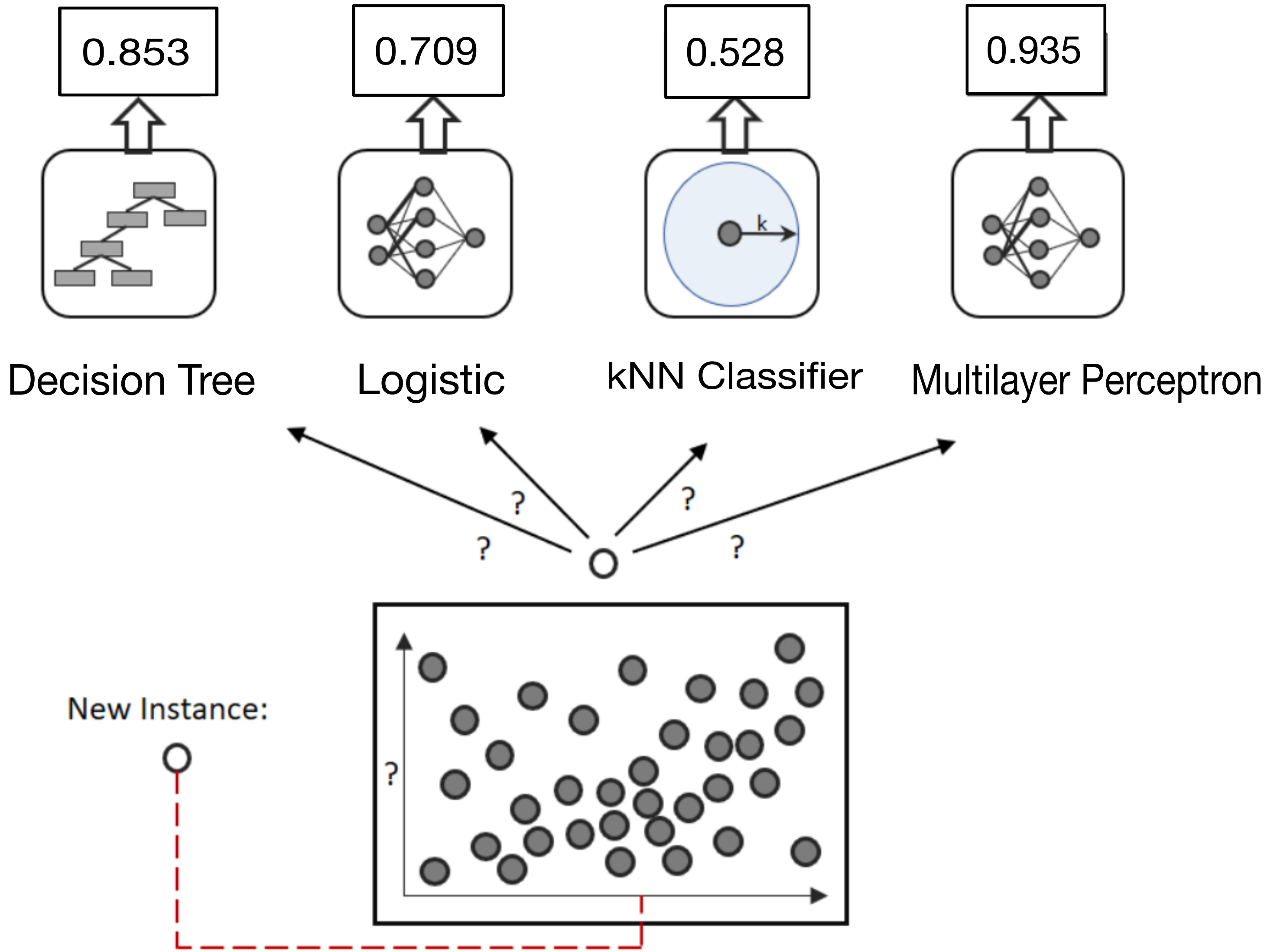
- Collaborative filtering
- Probability matrix / rating matrix

Collaborative Filtering (Basics)

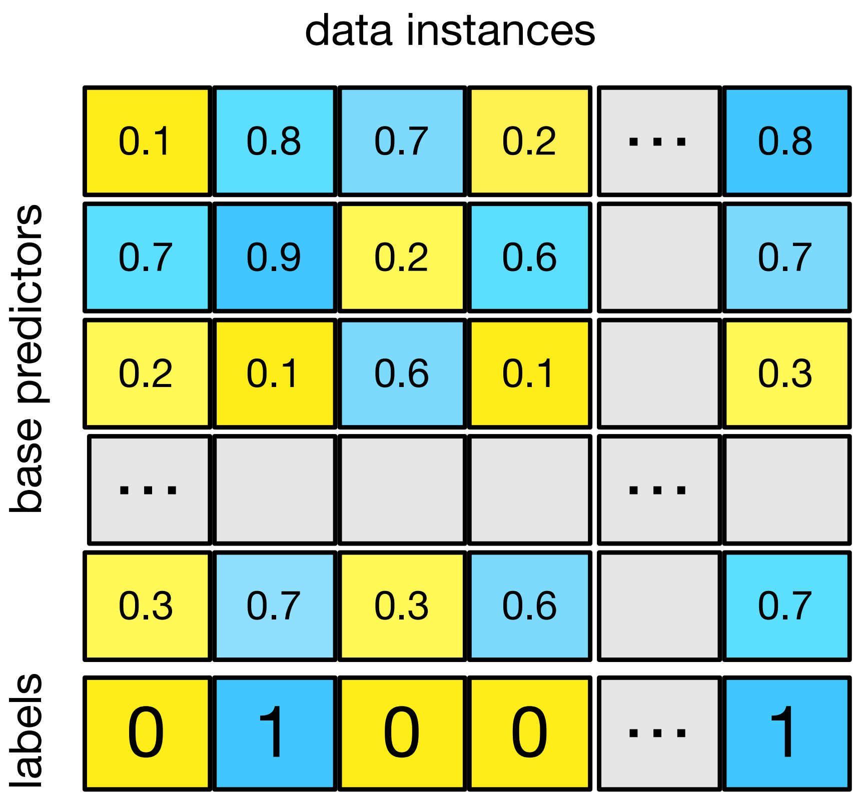


Collaborative Filtering




0.853
0.709
0.528
0.935
k
Decision Tree
Logistic
kNN Classifier
Multilayer Perceptron
?
?
?
?
New Instance:
?

Probability matrix



Majority Votes

0.08	0.34	0.12	0.23	0.71
0.43	0.59	0.05	0.01	0.43
0.33	0.41	0.62	0.34	0.88
0.61	0.28	0.49	0.42	0.92
0.27	0.17	0.55	0.64	0.47

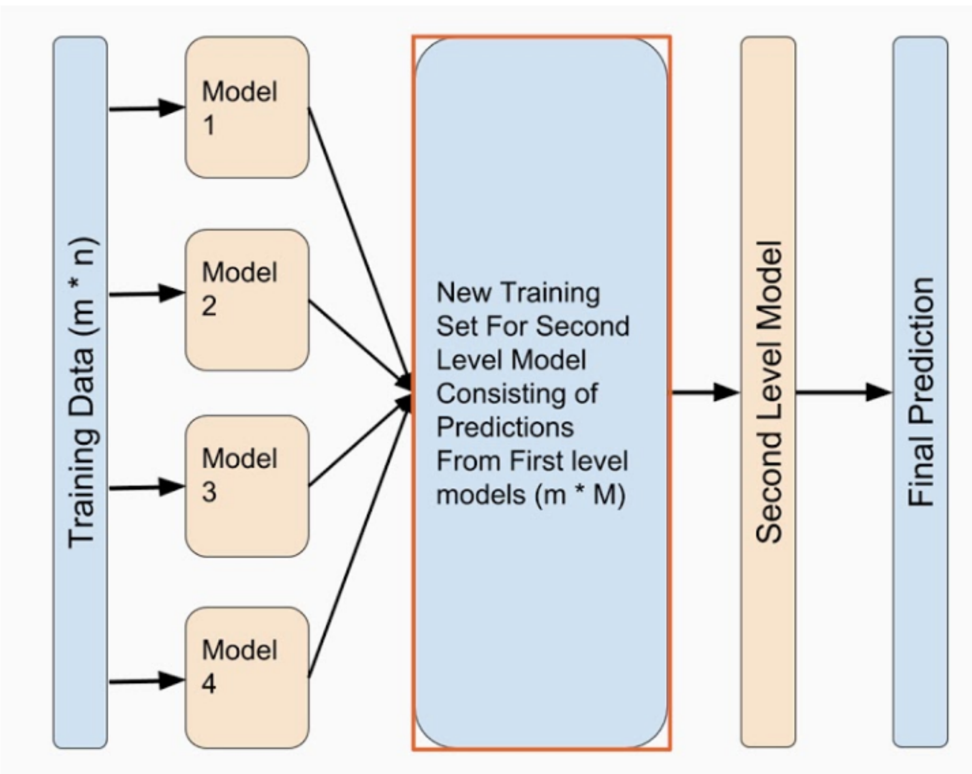
0	0	0	0	1
0	1	0	0	0
0	0	1	0	1
1	0	0	0	1
0	0	1	1	0

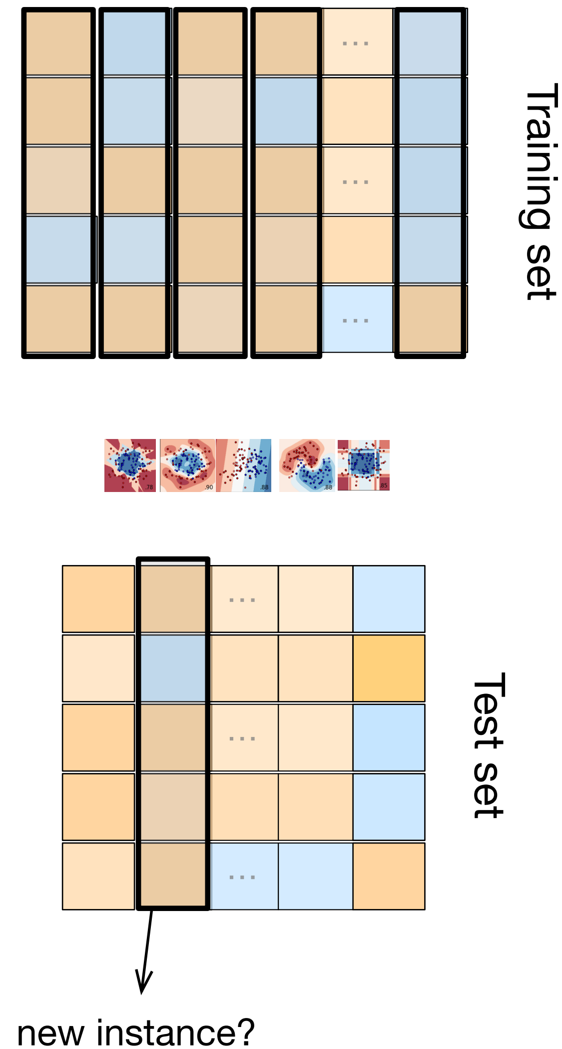
0	0	0	0	1
0	1	0	0	0
0	0	1	0	1
1	0	0	0	1
0	0	1	1	0

0	0	0	0	1

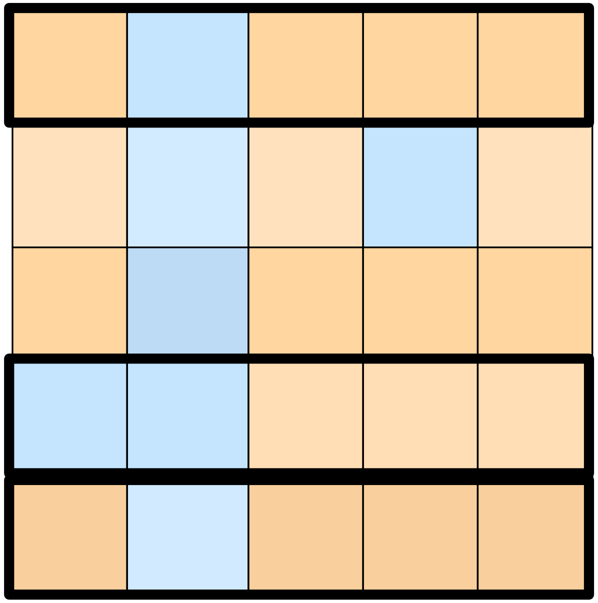
- Labeling matrix (L) as a binary matrix

Stacking


Training Data (m * n)
Model 1
Model 2
Model 3
Model 4
New Training Set For Second Level Model Consisting of Predictions From First level models (m * M)
Second Level Model
Final Prediction


Training set
Test set
new instance?

Ensemble Selection



E.g. CES (Caruana's Ensemble Selection)
… Iteratively grows the ensemble by selecting base predictors that leads to higher gains in chosen performance metric.

CF Ensemble Learning

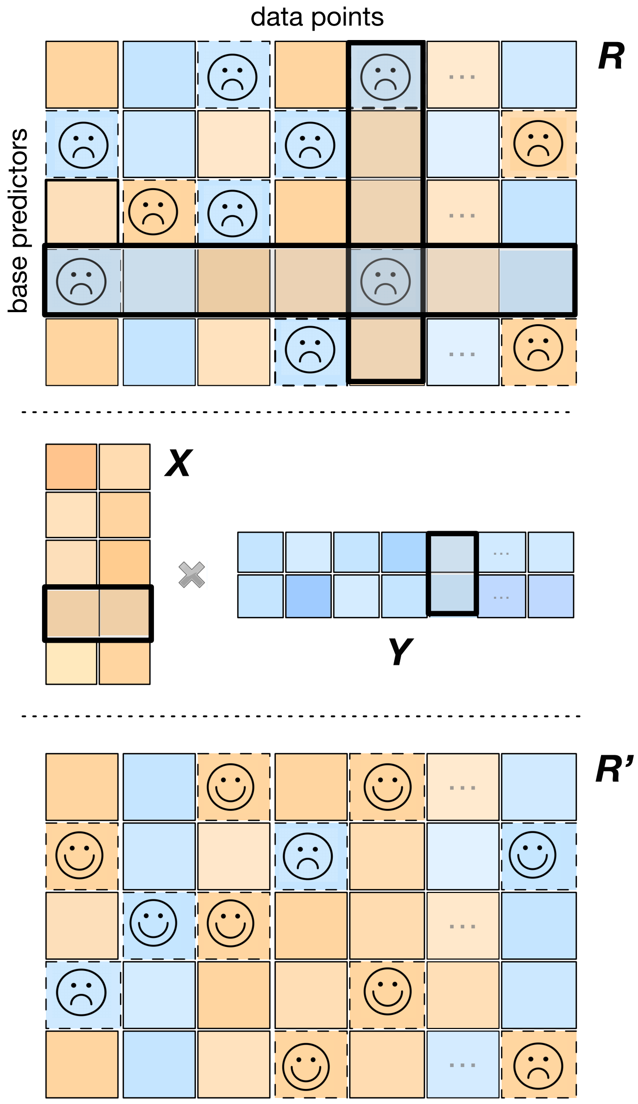
Identify unreliable conditional probabilities

Re-estimate the unreliable entries via reliable entries

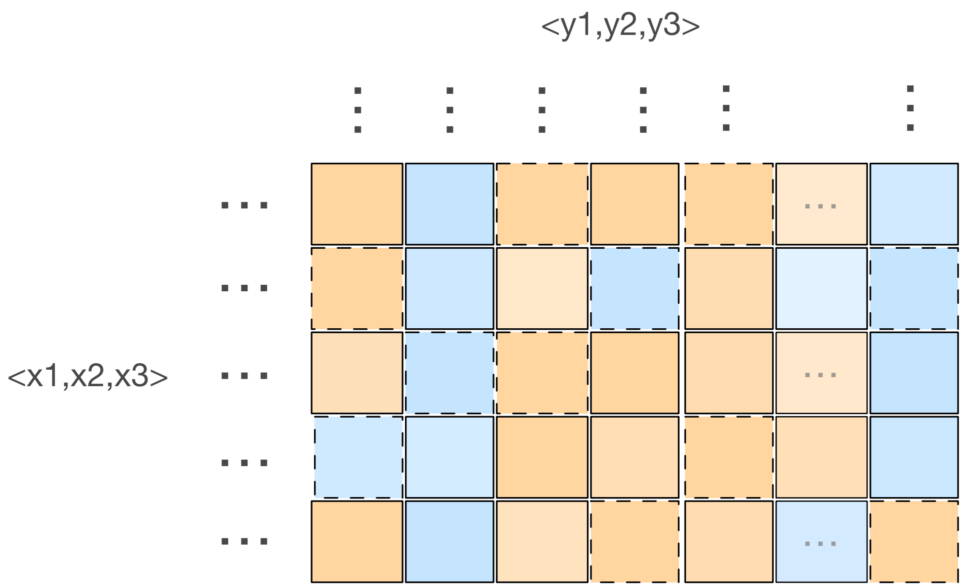
Unreliable entries are analogous to un-rated entries in recommender system

Predict what could have been a better estimate of p(y=1|x) using reliable entries

We need to figure out latent representation of the classifiers an data points in order to define their similarities



Latent Factor Representation



$$x_u^T = (x_u^1, x_u^2, \ldots, x_u^N)$$

$$y_i^T = (y_i^1, y_i^2, \ldots, y_i^N)$$

$$X^T = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ x_{u_1} & x_{u_2} & \cdots & x_{u_{n_{\text{users}}}} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix}$$

$$Y^T = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ y_{i_1} & y_{i_2} & \cdots & y_{i_{n_{\text{items}}}} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix}$$

$$\widehat{R} = XY^T$$

Optimization Objectives (1)

- Minimizing the reconstruction error

$$\sum_{u,i}(r_{ui} - x_u^T \cdot y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right)$$

- Since every entry of the probability matrix is “observed,” instead let’s think about which should remain in the cost functions
  - Include TPs and TNs and leave out FPs and FNs
- Minimizing the weighted reconstruction error, where Cui: {0, 1}

$$\sum_{u,i} c_{ui}(r_{ui} - x_u^T \cdot y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right)$$

Optimization Objectives (2)

- In classification, the situation is more complex because not TPs and TNs are made equal
  - We have very skewed class distributions (e.g. protein function prediction): very few positive classes
  - Each probability score has an associated confidence measure
- Minimizing the weighted reconstruction error with confidence scores (as continuous quantities rather than discrete values like {0, 1}

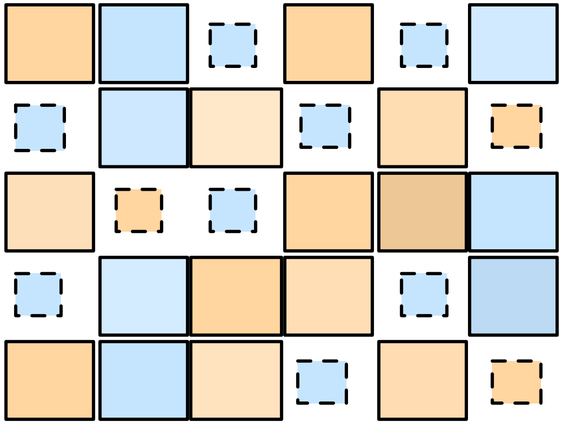
$$\sum_{u,i} c_{ui}(r_{ui} - x_u^T \cdot y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right)$$

CF for Ensemble Learning

- How to determine (degrees) of reliability
- Confidence score
  - Brier score

$$BS = \frac{1}{N}\sum_{i=1}^{N}(p_i - o_i)^2$$

  - Ratio of correct predictions



Optimization Objectives (3)

- Estimate preference score {0, 1}, depending on the "polarity (M)"

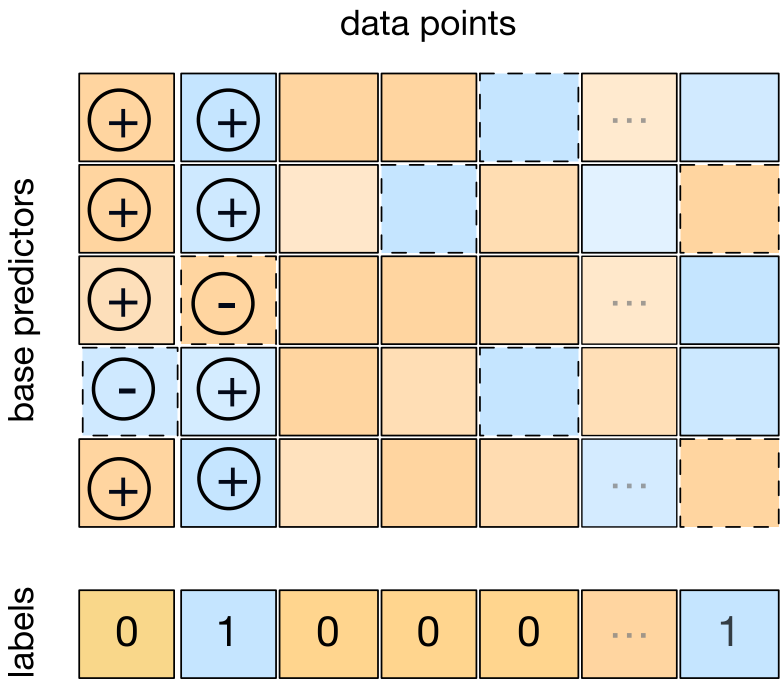
$$\sum_{u,i} c_{ui}(p_{ui} - x_u^T \cdot y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

$$p_{ui} = 1 \; if \; M_{ui} = positive$$

$$p_{ui} = 0 \; if \; M_{ui} = negative$$

- M[u,i] represents a "polarity" of an entry in the matrix
- Positive polarity if the entry (u, i) corresponds to TPs or TNs
- Negative polarity if the entry (u, i) corresponds to FPs or FNs

Polarity Matrix



0.08	0.34	0.12	0.23	0.71
0.43	0.59	0.05	0.01	0.43
0.33	0.41	0.62	0.34	0.88
0.61	0.28	0.49	0.42	0.92
0.27	0.17	0.55	0.64	0.47

0	0	0	0	1
0	1	0	0	0
0	0	1	0	1
1	0	0	0	1
0	0	1	1	0

0	0	0	0	1
0	1	0	0	0
0	0	1	0	1
1	0	0	0	1
0	0	1	1	0

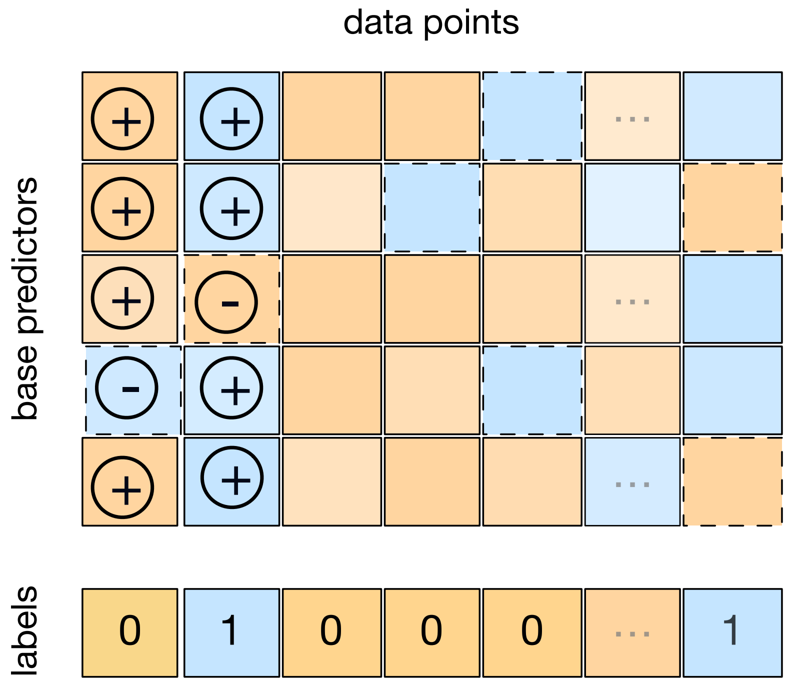
0	0	0	0	1

- If we could identify the polarities reasonably well and **drop the negatives at the prediction time, the predictive performance would be exceptional**
- Why? When there are lots of BPs (e.g. via bagging), chances are that there’s one or more predictors are making correct predictions (for the most part).

Cost Function with Polarities (1)

- When approximating "ratings" ...

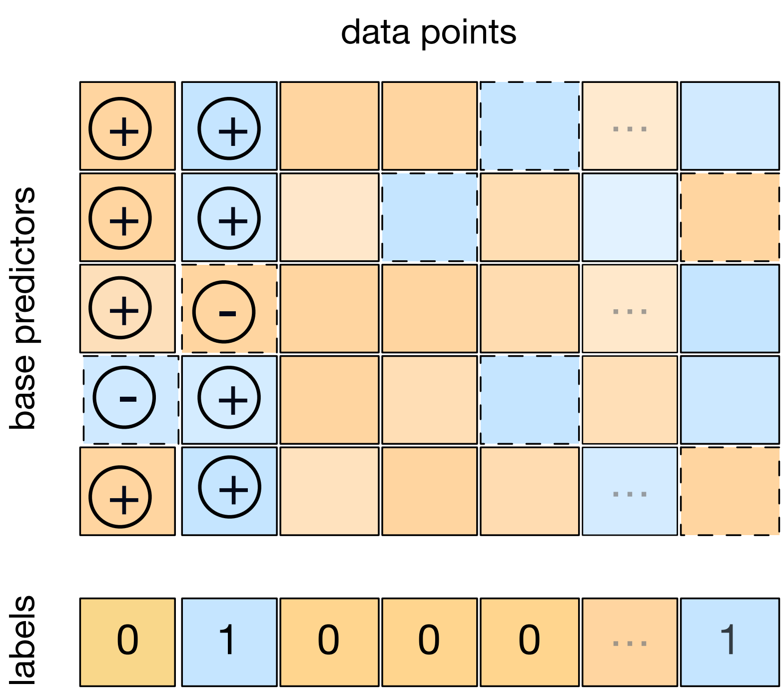
$$\sum_{u,i} c_{ui}(r_{ui} - x_u^T \cdot y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right)$$



Cost Function with Polarities (2)

- When representing preference scores …

$$\sum_{u,i} c_{ui}(p_{ui} - x_u^T \cdot y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right)$$

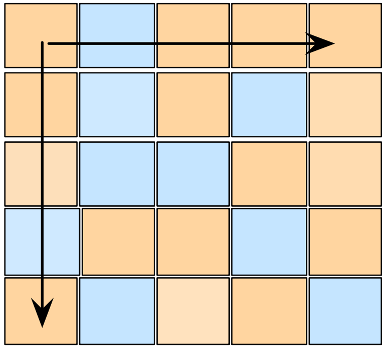


Polarity Modeling

- Can be broken down into a four-class prediction problem: i.e. predicting TPs, TNs, FPs, FNs
- Seems more complex however …
  - In predicting polarities, we actually have more training examples (than when we predict class labels)
    - Each entry in the probability matrix is a training instance
  - We do not need a perfect model

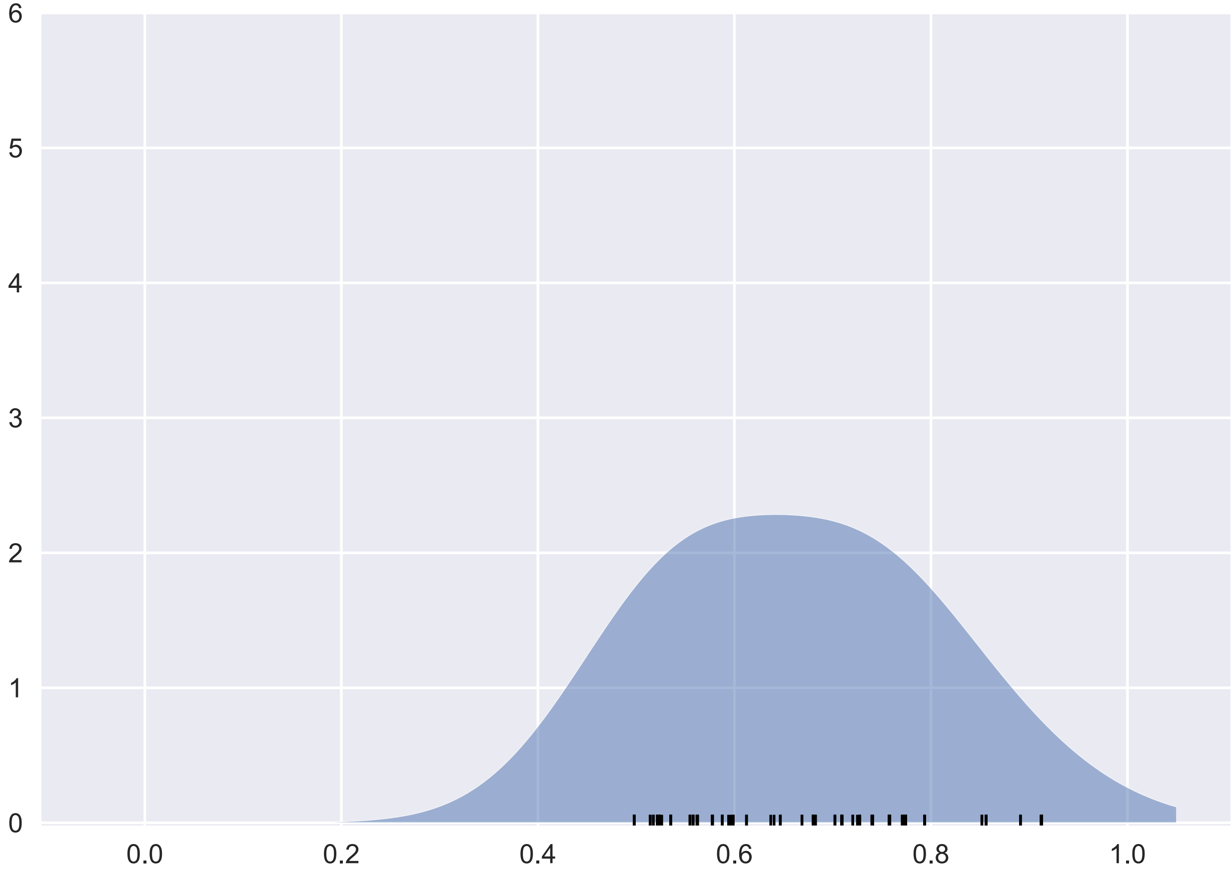
Polarity Modeling

- Baseline model via majority votes
  - Drawback: label dependent
- Identify useful features for predicting polarity
  - Horizontal statistics (foreach BP …)
    - R[u, i] > median of probability scores?
    - **KDE signature** (next slides)
    - How does R[u, i] compare to BP's own probability threshold
  - Vertical statistics (foreach data instance …)
    - Majority votes
    - Rank of R[u, i]
    - KDE? too expensive

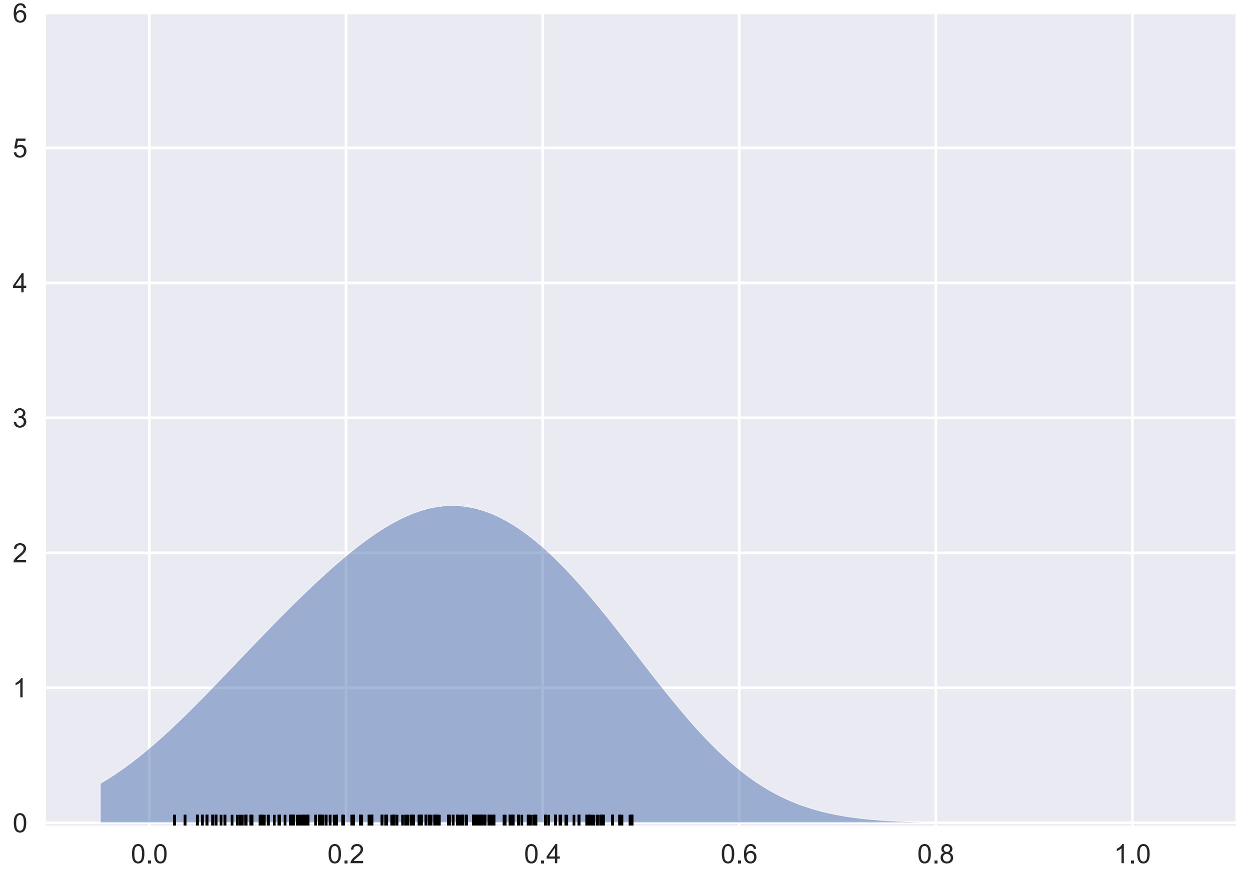


Polarity Modeling: KDEstimates

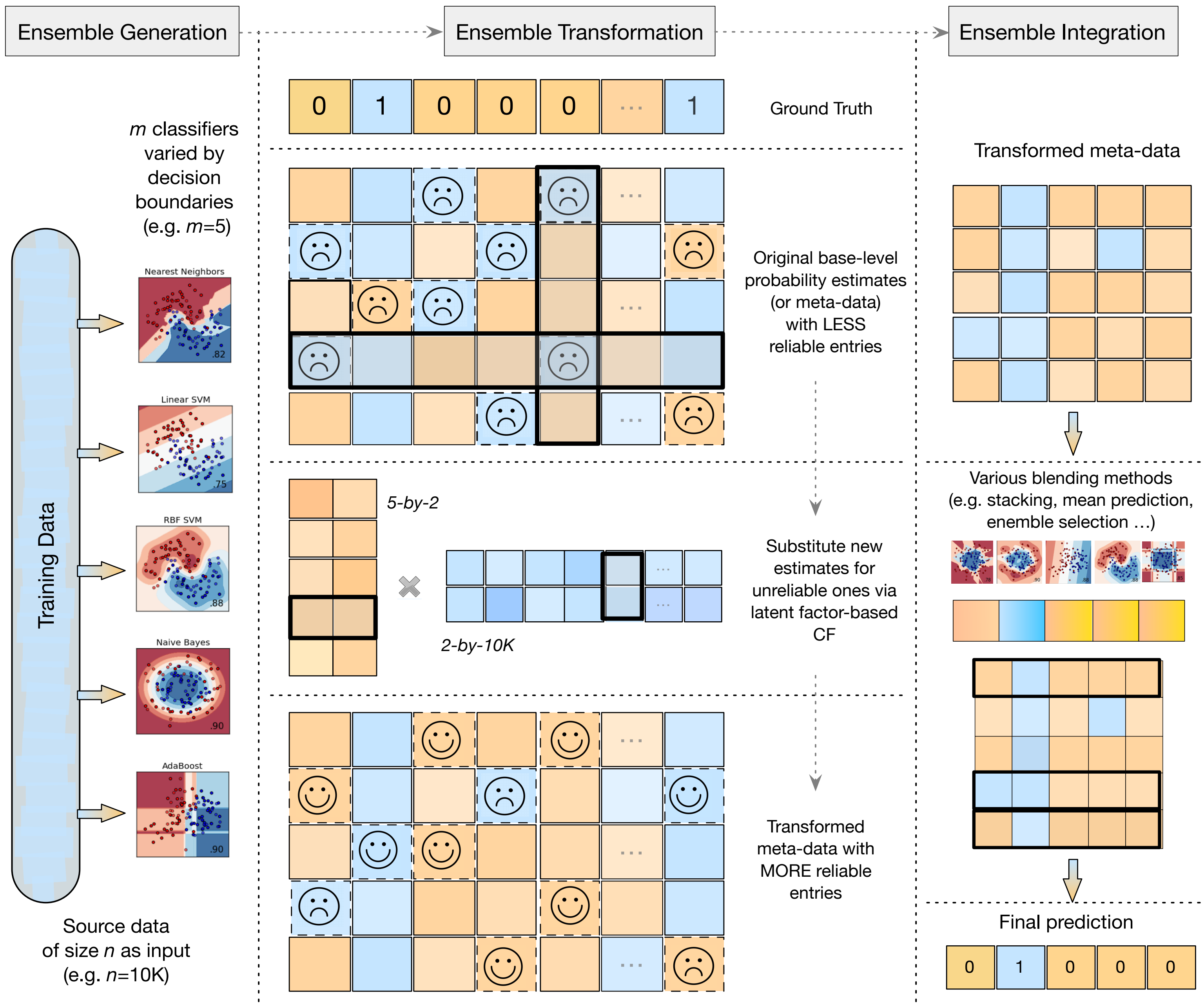
- KDE for the four different flavors of particles: TPs, TNs, FPs, FNs …
- Given a query point R[u, i], get the "amplitudes" for the four flavors of particles
  - If R[u, i] corresponds to TPs, it tends to be large; by contrast, if R[u, i] corresponds to TNs, it tends to be small
  - Perhaps even better, use survival function to find $P(R>=R[u, i])$ given the density estimate



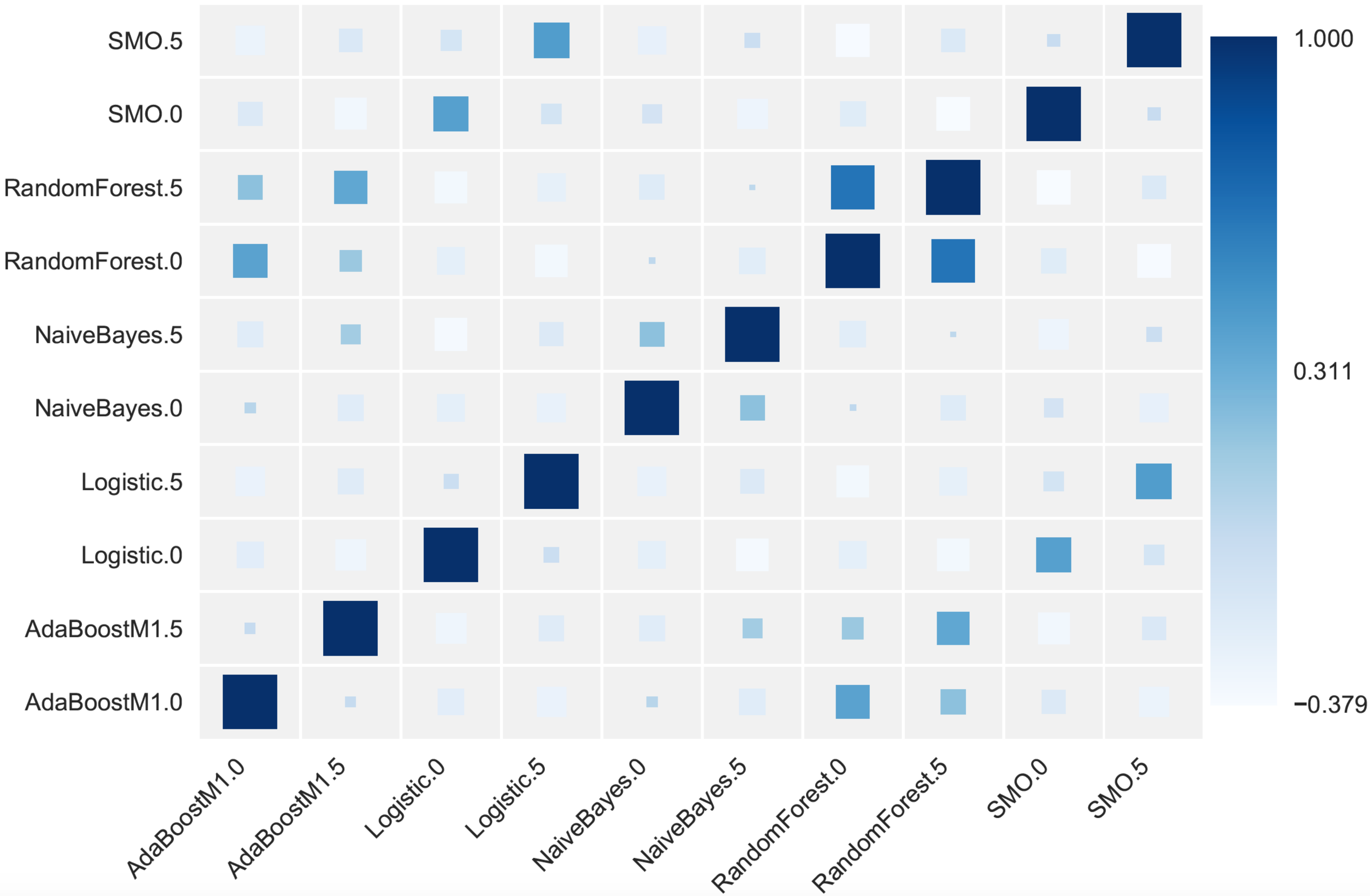
6
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0



6
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0



Ensemble Generation
Ensemble Transformation
Ensemble Integration
m classifiers varied by decision boundaries (e.g. m=5)
Training Data
Nearest Neighbors
Linear SVM
RBF SVM
Naive Bayes
AdaBoost
Source data of size n as input (e.g. n=10K)
Ground Truth
Original base-level probability estimates (or meta-data) with LESS reliable entries
5-by-2
2-by-10K
Substitute new estimates for unreliable ones via latent factor-based CF
Transformed meta-data with MORE reliable entries
Transformed meta-data
Various blending methods (e.g. stacking, mean prediction, enemble selection …)
Final prediction


SMO.5
SMO.0
RandomForest.5
RandomForest.0
NaiveBayes.5
NaiveBayes.0
Logistic.5
Logistic.0
AdaBoostM1.5
AdaBoostM1.0
AdaBoostM1.0
AdaBoostM1.5
Logistic.0
Logistic.5
NaiveBayes.0
NaiveBayes.5
RandomForest.0
RandomForest.5
SMO.0
SMO.5
1.000
0.311
−0.379