New Prediction Methods for Tree Ensembles with Applications in Record Linkage

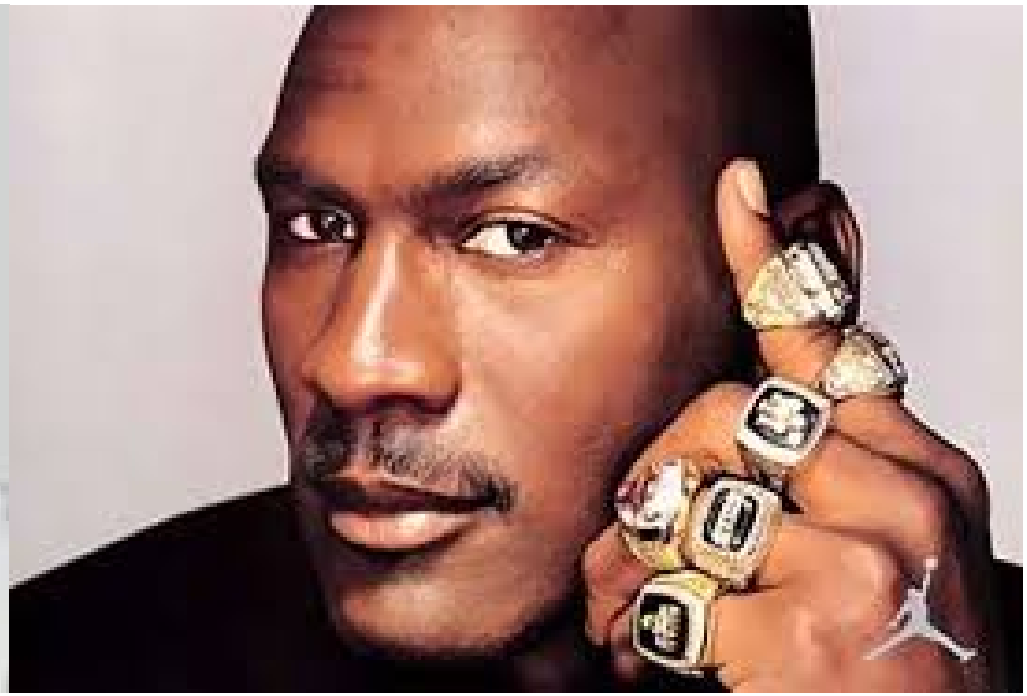
Samuel L. Ventura
Rebecca Nugent

Department of Statistics
Carnegie Mellon University

June 11, 2015

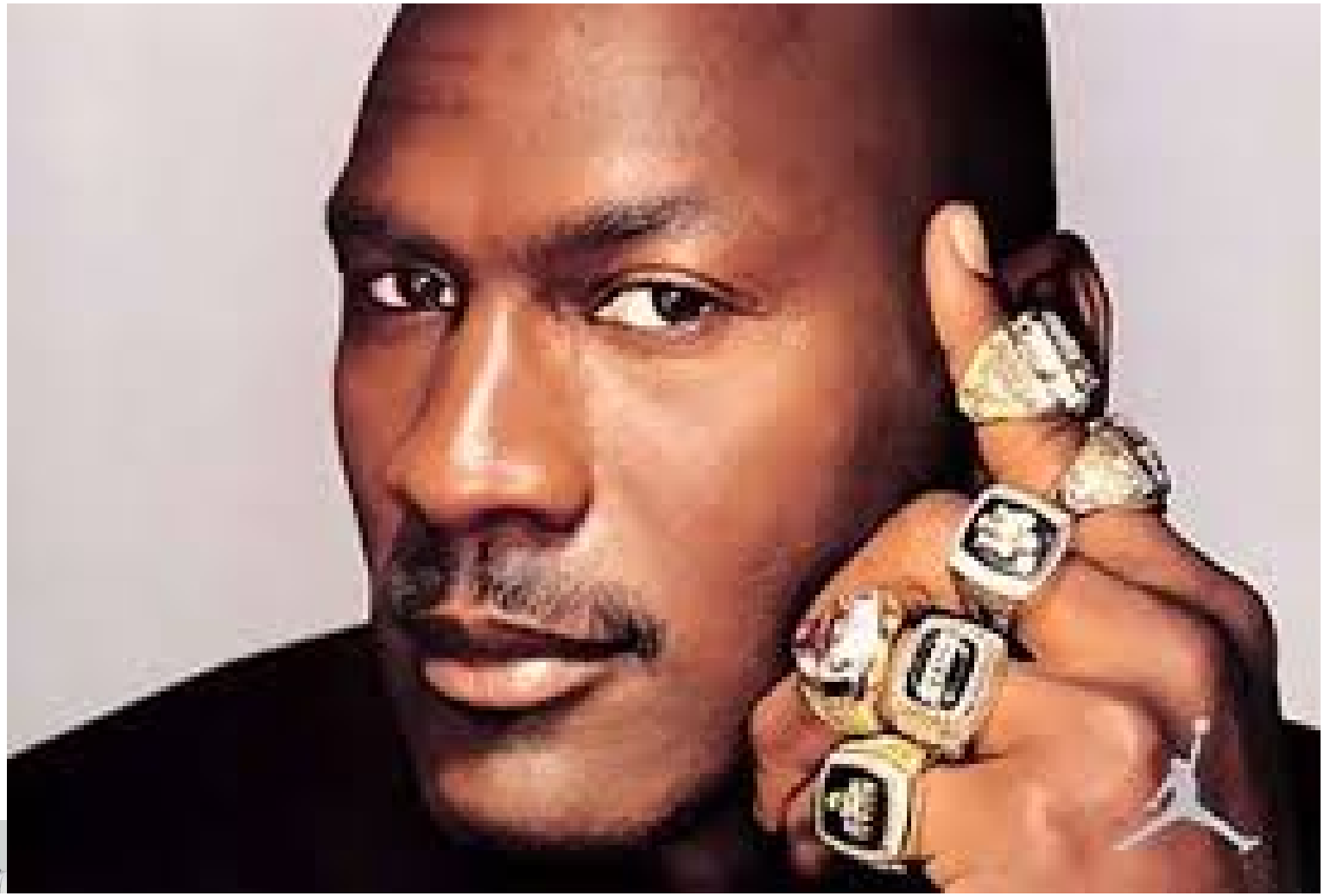
45th Symposium on the Interface
Computing Science and Statistics

Why Do We Need Record Linkage?

What happens if we search "Michael Jordan Statistics" on Google?

Google: "Michael Jordan Statistics"



What is Record Linkage?

Record Linkage: Match records corresponding to unique entities within and across data sources

Fellegi & Sunter (1969) introduced several important early concepts:

- Similarity scores to quantify similarity of names, addresses, etc
- Theoretical framework for estimating probabilities of matching
- Examining effect reducing the comparison space has on error rates

Many extensions and alternatives to the Fellegi-Sunter methodology:

- Larsen & Rubin (2001): Mixture models for automated RL
- Sadinle & Fienberg (2013): Extend Fellegi-Sunter to 3+ files
- Steorts et al (2015): Bayesian approach to graphical RL
- Ventura et al (2014, 2015): **Supervised learning** approaches for RL

Example: United States Patent & Trademark Office

Inventors often have similar identifying information

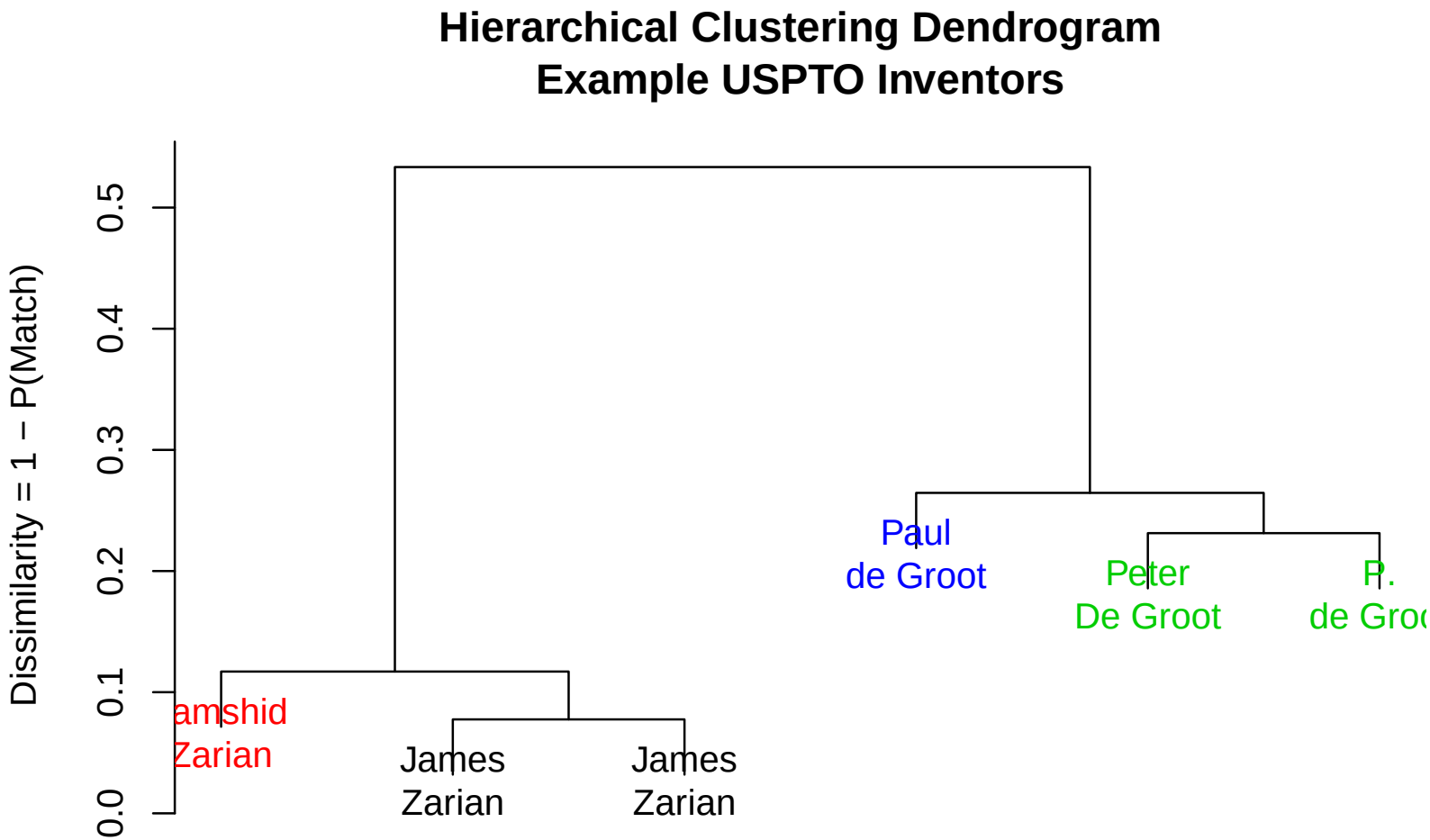
Last	First	Mid	City	St/Co	Assignee
Zarian	James	R.	Corona	CA	Lumentye
Zarian	James	N/A	Corona	CA	Lumentye Corp.
Zarian	Jamshid	C.	Woodland Hills	CA	Lumentye
De Groot	Peter	J.	Middletown	CT	Zygo
de Groot	P.	N/A	Middleton	CT	Boeing
de Groot	Paul	N/A	Grenoble	FR	Thomson CSF

Six records from USPTO database (Akinsanmi et al, 2014; Ventura et al, 2015)

How do we know which records refer to the same person?

How do we compare strings? Allow for typos?

Picture: Our Record Linkage Framework



Within (across) blocks, records are similar (dissimilar)

Outline: Our Record Linkage Framework

1. **Partition the data** into groups of similar records, called "blocks"
   - Reducing the comparison space more efficiently
   - Preserving false negative error rates

2. Within blocks: **Estimate probability** that each record-pair matches
   - Quantify the similarity of record-pairs
   - Classifier ensembles when training data is prohibitively large
   - Improving predictions for classifier ensembles with distributions of estimated probabilities

3. Within blocks: **Identify unique entities**
   - Convert estimated probabilities to dissimilarities
   - Hierarchical clustering to link groups of records

Quantify the Similarity of each Record-Pair: γ_{ij}

Let $\gamma_{ij} = \langle \gamma_{ij1}, ..., \gamma_{ijM} \rangle$ be the similarity profile for records x_i, x_j

Calculate similarity scores γ_{ijm} for each field $m = 1, ..., M$

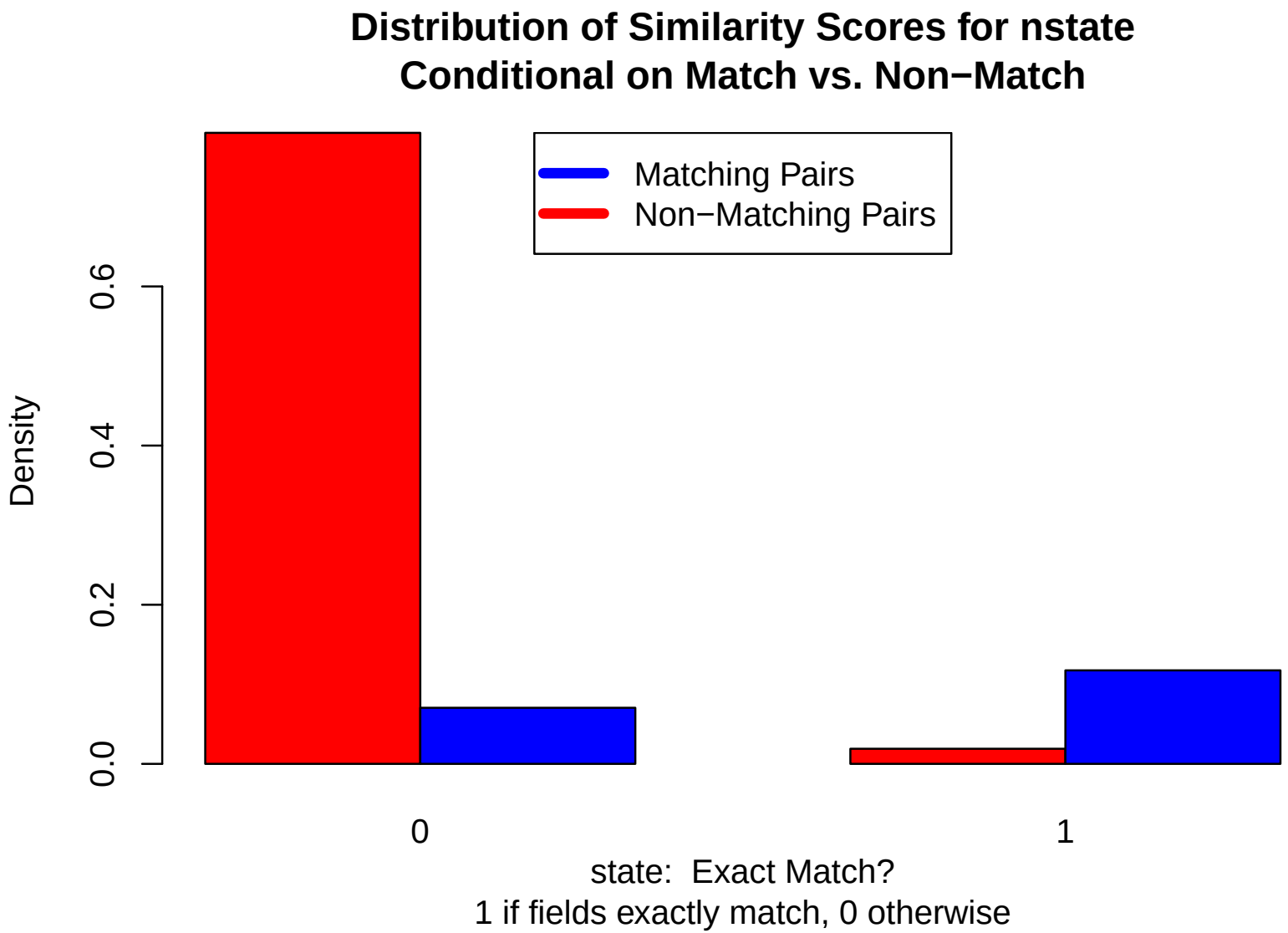
i	j	Last	First	Mid	City	St	Assignee
1	4	0.93	1.00	0.75	1.00	1	0.50
1	7	0.93	1.00	0.00	0.42	0	0.50

Need to calculate $\binom{n}{2}$ pairwise comparisons for n records

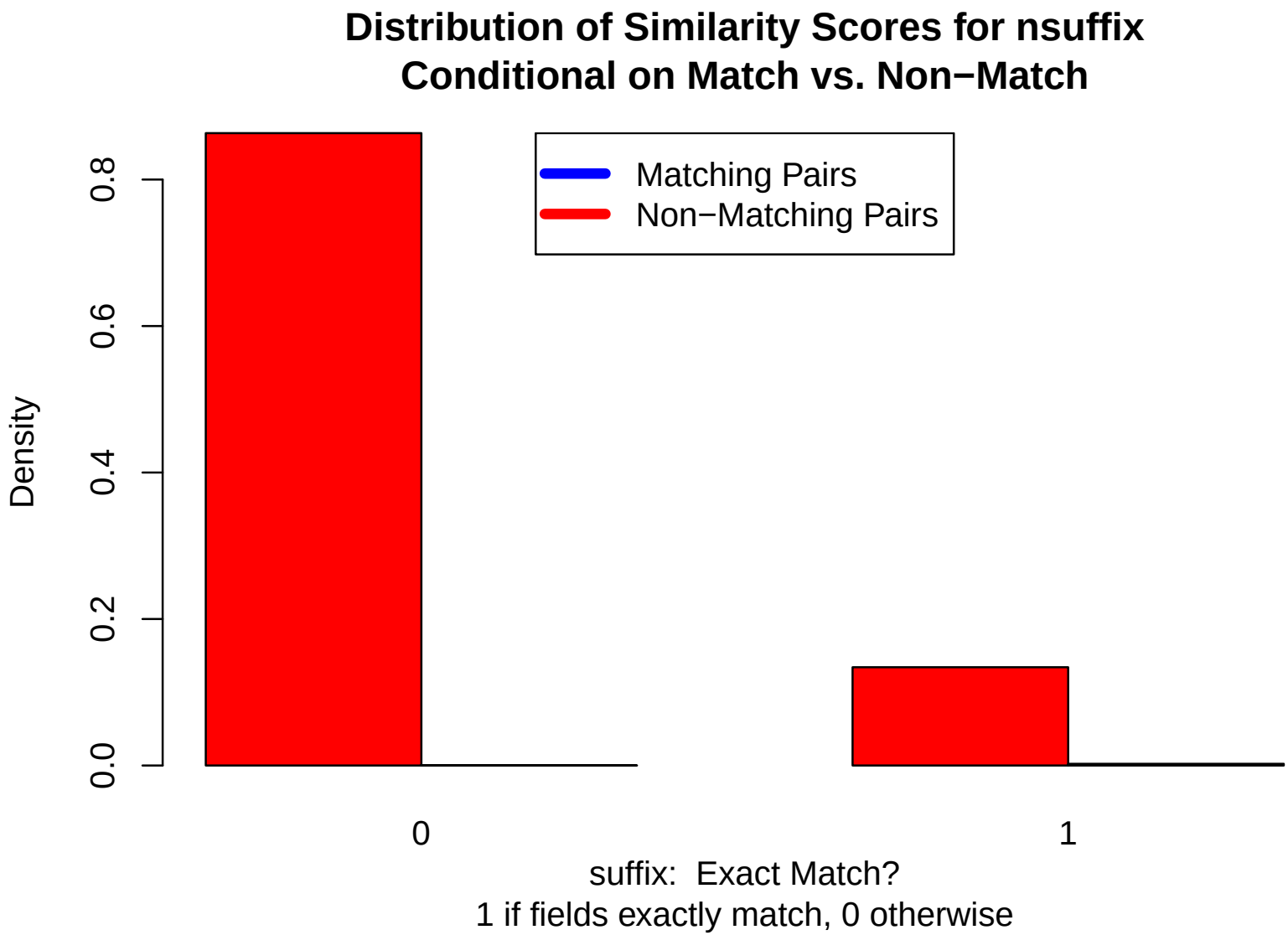
- 1 million records $\approx$ 500 billion comparisons
- Computational tools in place to reduce complexity

Does γ_{ij} help us separate matching and non-matching pairs?

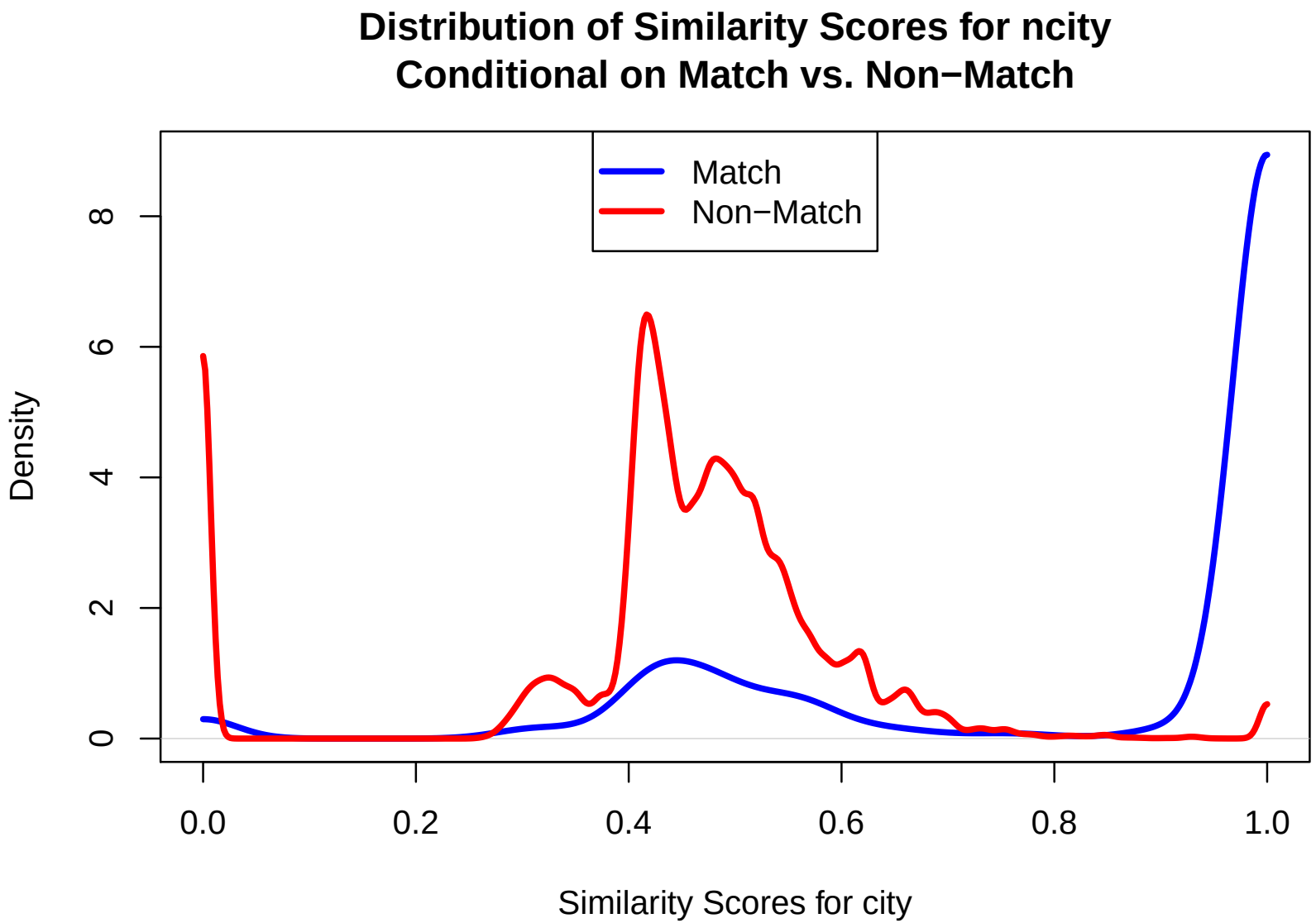
Distributions of Similarity Scores given Match/Non-Match



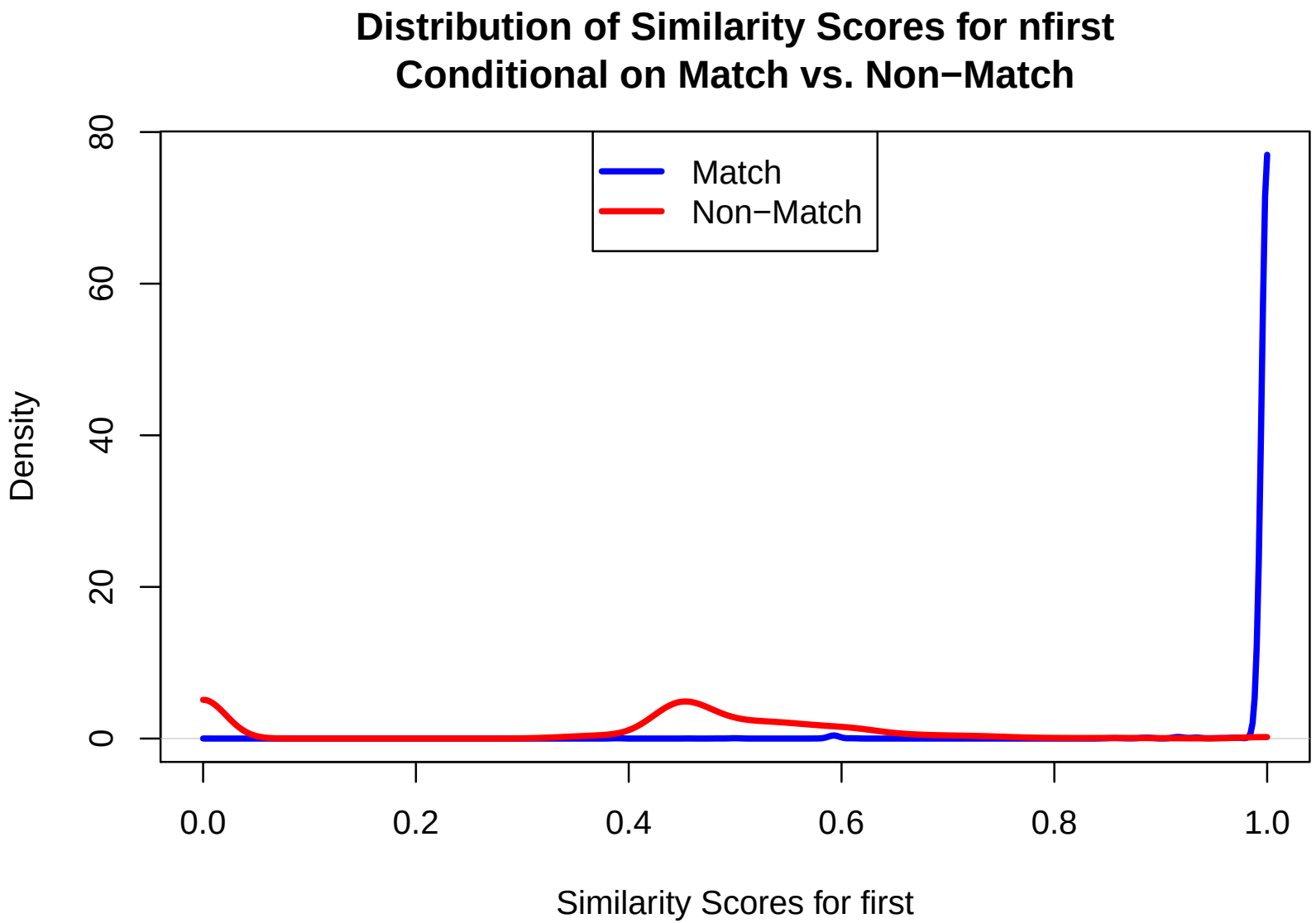
Distributions of Similarity Scores given Match/Non-Match



Distributions of Similarity Scores given Match/Non-Match



Distributions of Similarity Scores given Match/Non-Match



Find the Probability that Record-Pairs Match: $\hat{p}_{ij}$

Our approach: supervised learning to estimate $P(x_i = x_j)$

- Train a **classifier** on comparisons of labeled records
- Use the classifier to **predict** whether record-pairs match
- Result: $\hat{p}_{ij}$ for any record-pair

i	j	Last	First	Mid	City	St	Assignee	Match?
1	4	0.93	1.00	0.75	1.00	1	0.50	Yes
1	7	0.93	1.00	0.00	0.42	0	0.50	No

Given a classifier m, find $P(x_i = x_j) = \hat{p}_{ij} = m(\gamma_{ij})$

Outline: Our Record Linkage Framework

1. **Partition the data** into groups of similar records, called "blocks"
   - Reducing the comparison space more efficiently
   - Preserving false negative error rates

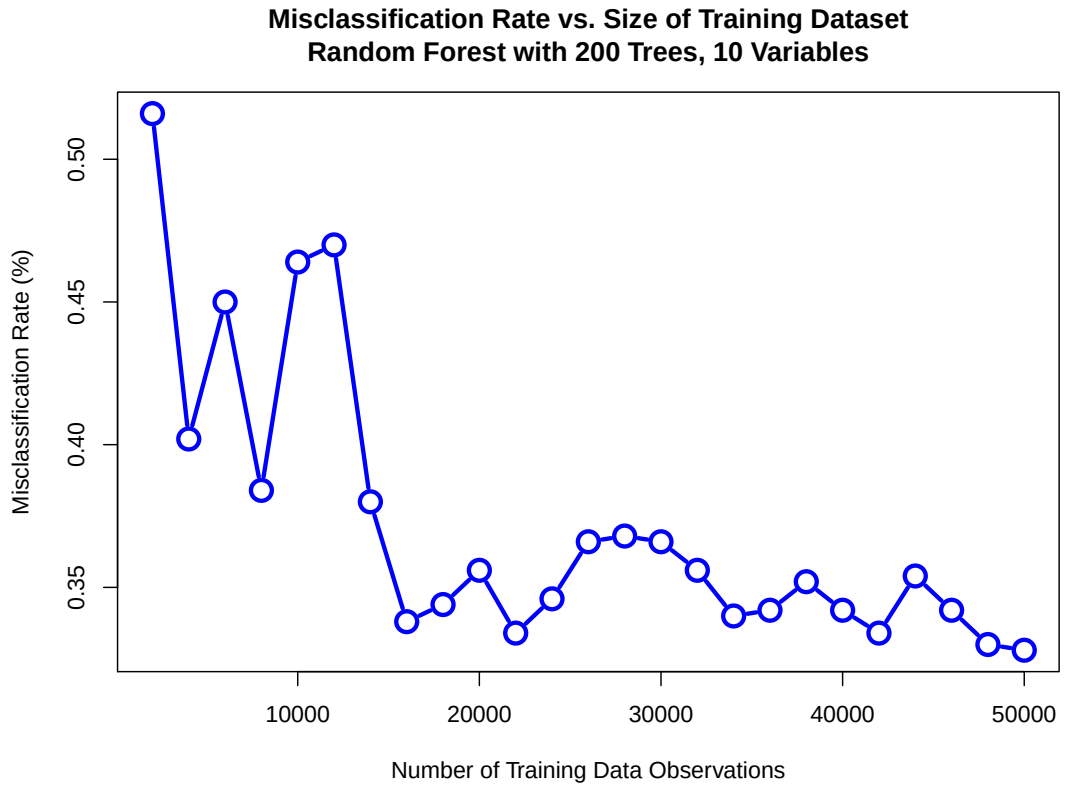
2. Within blocks: **Estimate probability** that each record-pair matches
   - Quantify the similarity of record-pairs
   - Classifier ensembles when training data is prohibitively large
   - Improving predictions for classifier ensembles with distributions of estimated probabilities

3. Within blocks: **Identify unique entities**
   - Convert estimated probabilities to dissimilarities
   - Hierarchical clustering to link groups of records

Ensembles: Why do we have multiple estimates of $\hat{p}_{ij}$?

Often computationally infeasible to train single classifier

USPTO RL application: over 20 million training data observations



Error rates stabilize as number of training data observations increase

Ensembles: Why do we have multiple estimates of $\hat{p}_{ij}$?

Some classifiers are, by definition, ensembles (e.g. random forests)

Majority Vote of trees (Breiman, 2001)

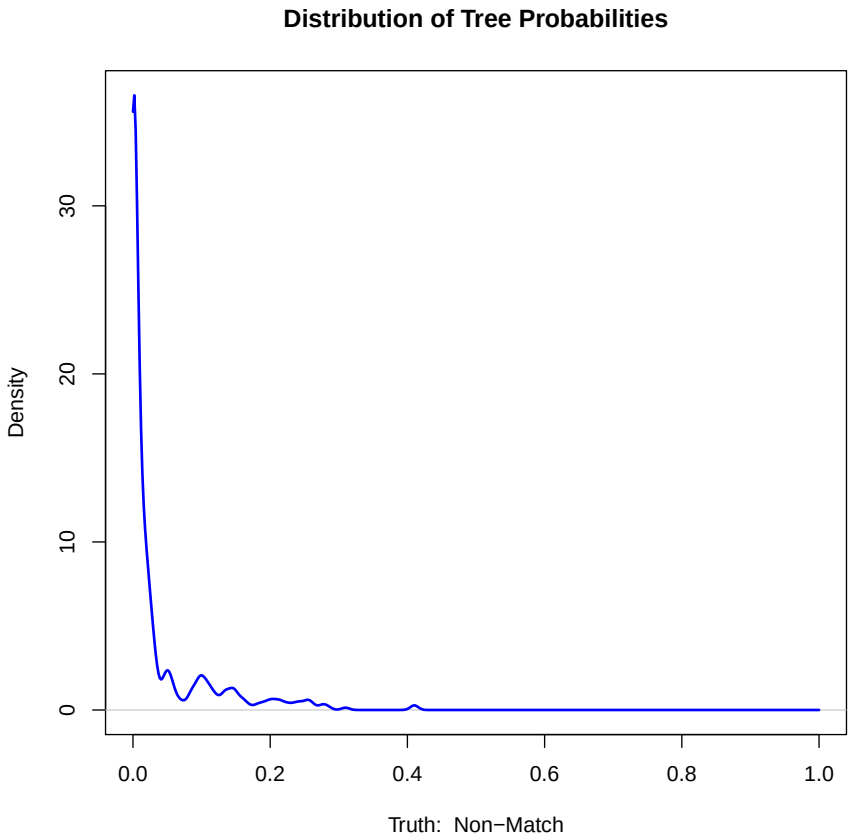
- Predicted probability = Proportion of ensemble's votes for each class
- Predicted class = Majority vote of classifiers in the ensemble

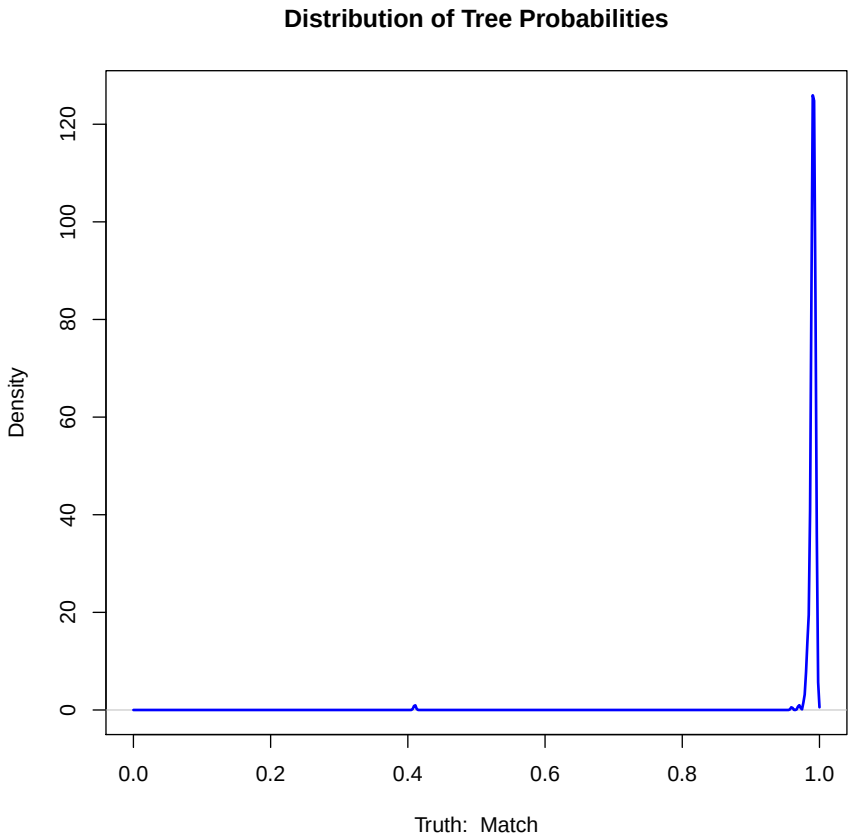
Mean Probability (Bauer & Kohavi, 1999)

- Predicted probability = Mean of all tree probabilities
- Predicted class = 1 if predicted probability ≥ 0.5; 0 otherwise

Distribution of *R* Predicted Probabilities

The good:

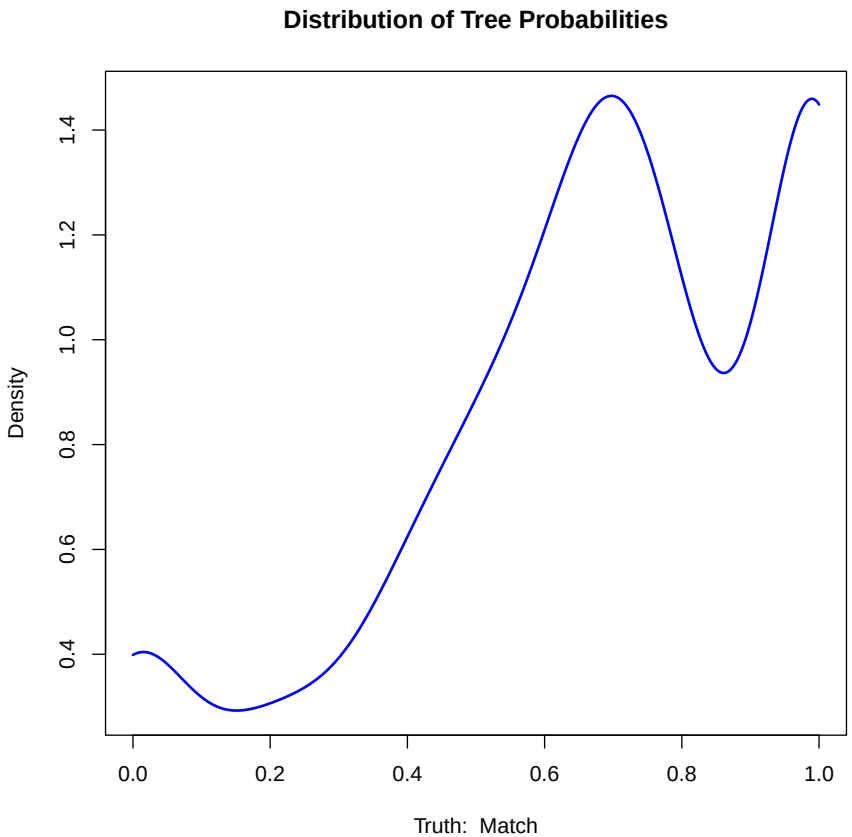


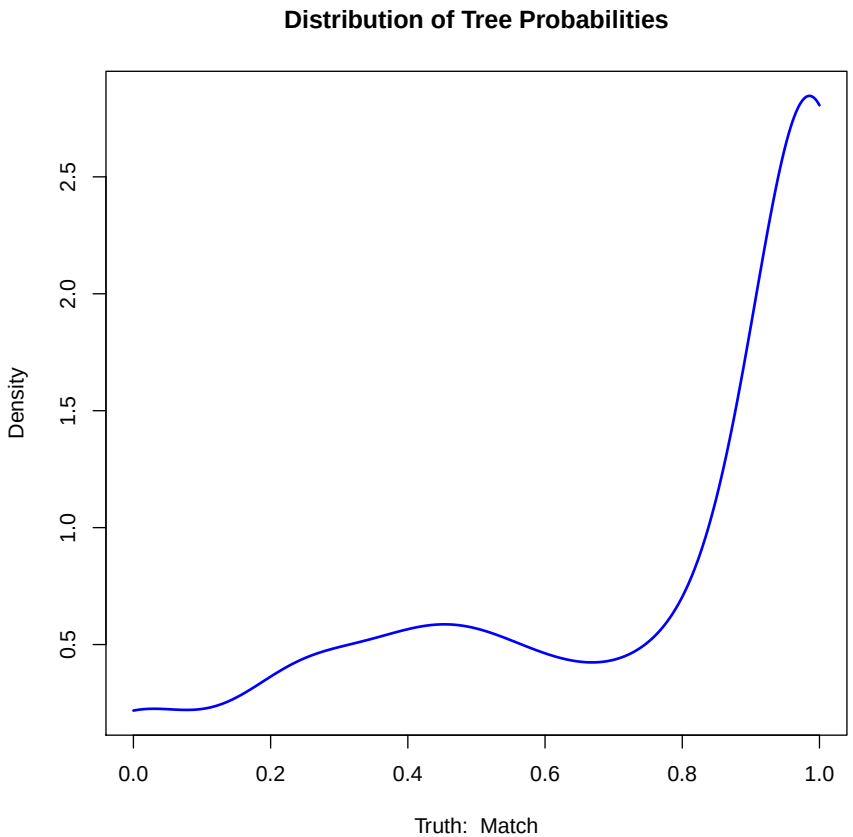


(random forest with 500 underlying classification trees)

Distribution of *R* Predicted Probabilities

The bad:

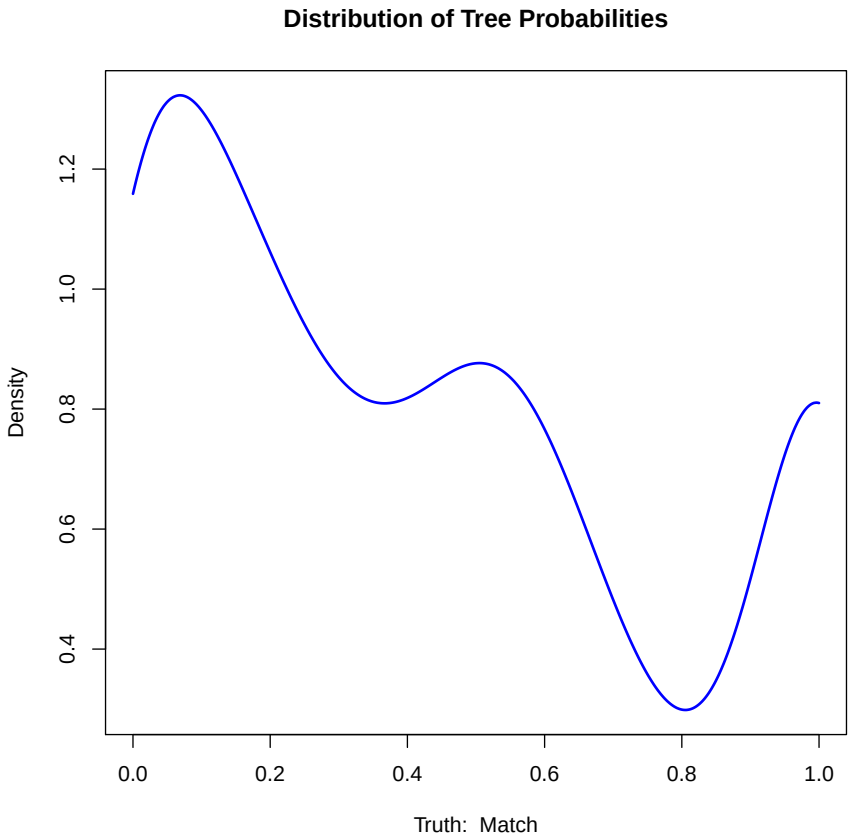


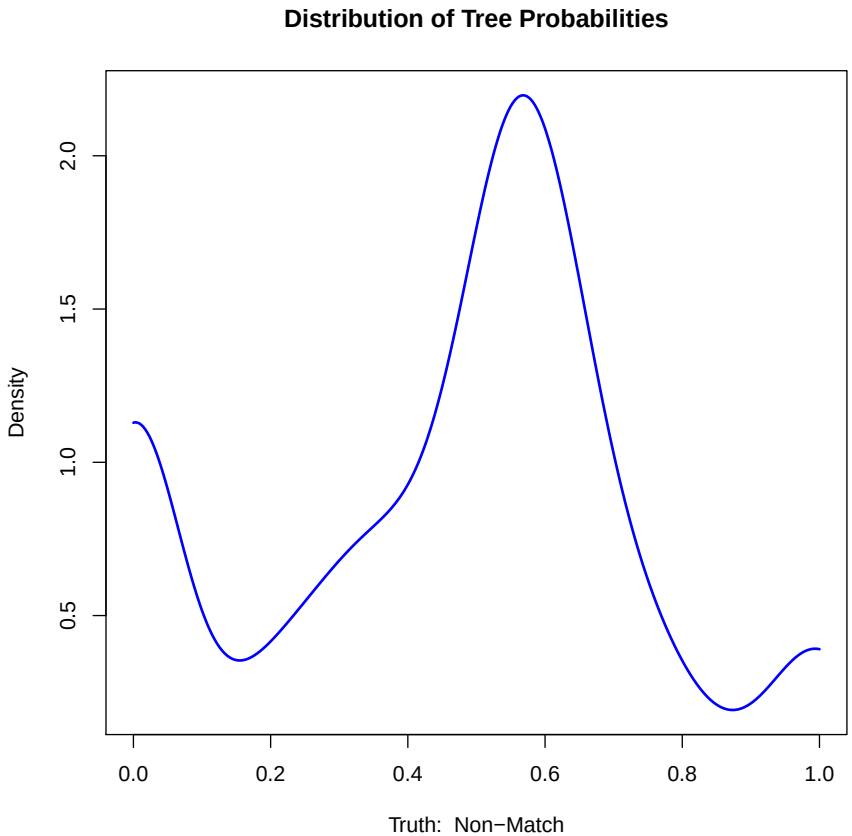


(random forest with 500 underlying classification trees)

Distribution of *R* Predicted Probabilities

The ugly:





(random forest with 500 underlying classification trees)

Idea: Set Aside Some Training Data

Remember: Our training datasets are large!

- E.g., USPTO RL dataset has over 20 million training observations
- We don't need *all* of the training data to build an ensemble
- **Set aside some training data** and use it later...

Use approach similar to stacked generalization/stacking (Wolpert, 1992):

- Split training data into two pieces
- On first piece, build the classifier ensemble
- In second piece, treat each model/predictor as a covariate
- Build logistic regression model to weight predictions from ensemble

Use stacking for random forests?

- Issue for stacking with RF: Logistic regression with 500+ covariates?
- Alternative: **Use distribution summary statistics** as covariates

PRediction with Ensembles using Distribution Summaries

PREDS: Use both pieces of training data (X_1, X_2) for better prediction

1. Build a classifier ensemble, $\{F_{1,r}\}_{r=1}^{R}$, on X_1
2. Apply $\{F_{1,r}\}_{r=1}^{R}$ to X_2

   This yields a distribution of predictions for each observation in X_2
3. Train a new model, F_2, using:
   - Covariates: Features of the distribution of predictions for X_2
   - Response: The actual 0-1 response in X_2

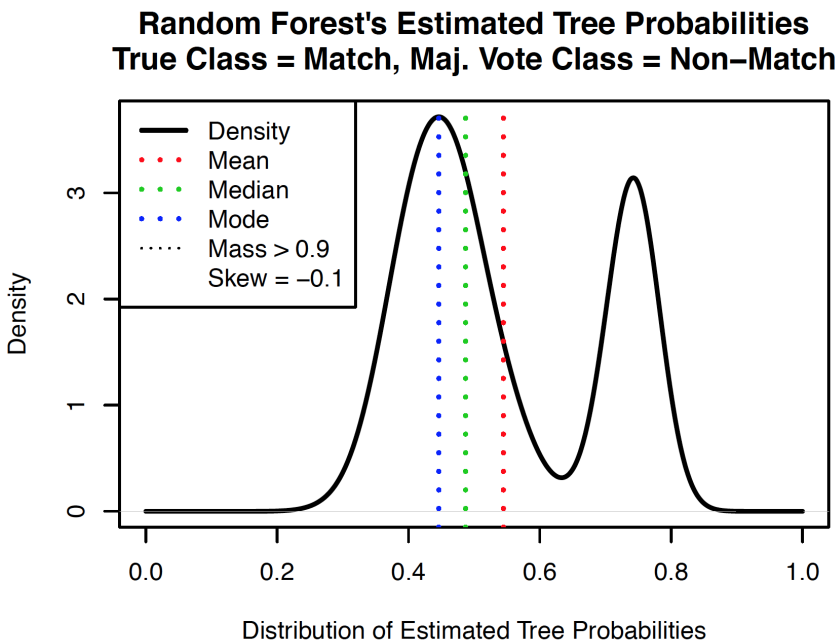
PREDS: When estimating the probability for a new observation

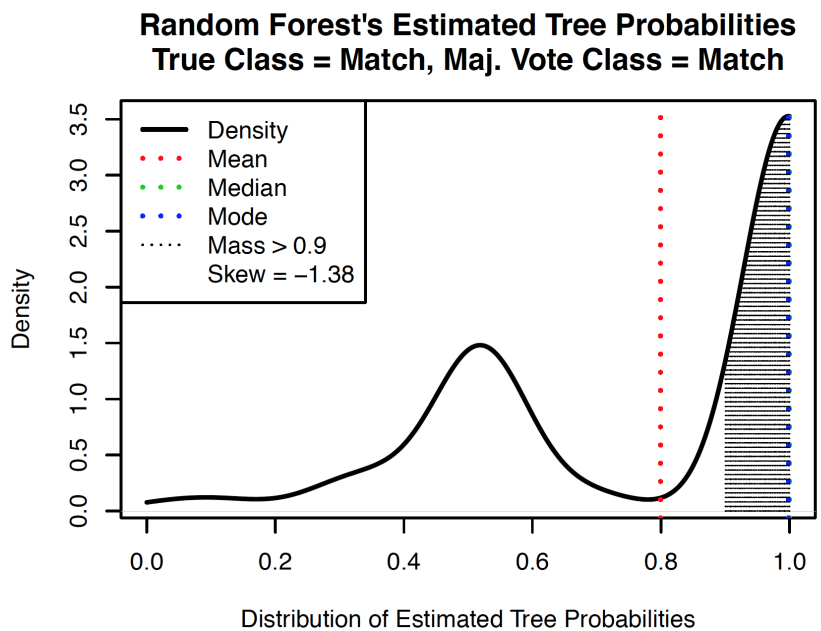
1. Apply $\{F_{1,r}\}_{r=1}^{R}$ to the test data
2. Apply F_2 to the resulting distribution of predictions
3. Use F_2's resulting estimated probability as the final estimate

Distribution Summary Statistics

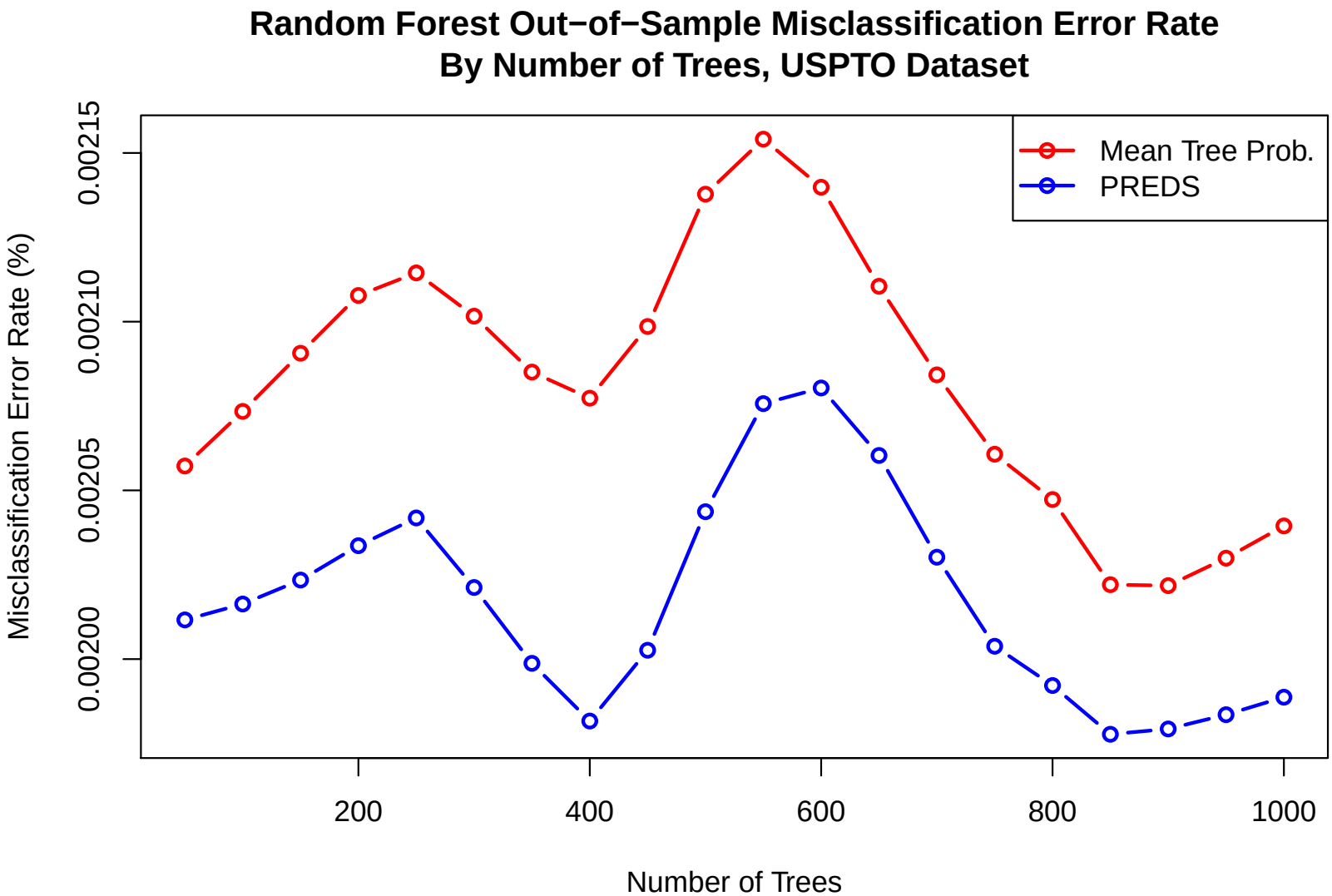
Flexible method: Can use any approach for summarizing the distribution

- Mean of the distribution
- "Majority vote" of the distribution
- Location of the largest mode(s)
- Skew of the distribution
- Mass of the distribution above/below a threshold





PREDS on USPTO Dataset, Varying Number of Classifiers



PREDS lowers error rates vs. mean tree probability and majority vote

Linking Death Records from the Syrian Civil War

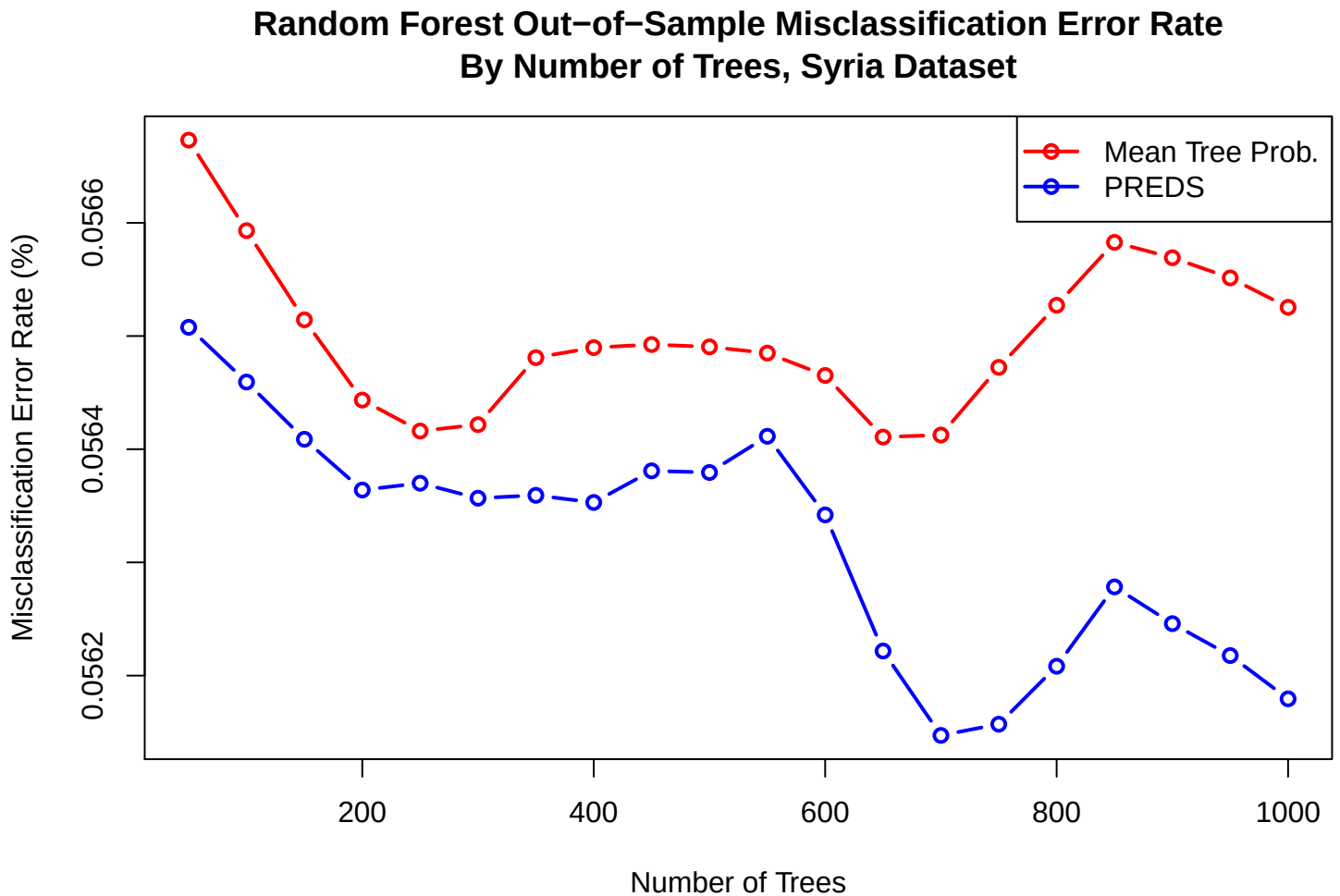
Human Rights Data Analysis Group
everybody counts.

HRDAG compiles datasets of death records from the Syrian Civil War

- 6+ organizations creating death records on the ground in Syria
- Over 200,000 total records
- Some lists have large overlap
- Fields: name, date of death, gender, governorate
- Field information can be missing, subject to error
- Some record-pairs labeled as match/non-match

Goal: Identify duplicate records

PREDS on Syria Dataset, Varying Number of Classifiers



For all values of R, PREDS yields lower misclassification rates than majority vote and mean of tree probabilities

Conclusions & Future Work

In large-scale training data scenarios, improvement in out-of-sample misclassification rates given extra training data is minimal

How can we use the "extra" training data more efficiently?

- Partition training data into two pieces
- Train classifier ensemble on one piece, predict on the other
- Summarize the distribution of predicted probabilities
- Build a new model: match vs. non-match | distribution summaries

Future work:

- Size of training datasets: depend on application, # of covariates?
- Compare to other stacking approaches for ensembles
- Study error properties associated stacking for record linkage