An Empirical Comparison of Learning Methods++

Rich Caruana
Cornell University

joint work with Alex Niculescu, Cristi Bucila, Art Munson

Sad State of Affairs: Supervised Learning

- Linear/polynomial Regression
- Logistic/Ridge regression
- K-nearest neighbor
- Linear perceptron
- Decision trees
- SVMs
- Neural nets
- Naïve Bayes
- Bayesian Neural nets
- Bayes Nets (Graphical Models)
- Bagging (bagged decision trees)
- Random Forests
- Boosting (boosted decision trees)
- Boosted Stumps
- ILP (Inductive Logic Programming)
- Rule learners (C2, …)
- Ripper
- Gaussian Processes
- …

Each algorithm has many variations and free parameters:
- SVM: margin parameter, kernel, kernel parameters (e.g. gamma), …
- ANN: # hidden units, # hidden layers, learning rate, momentum, …
- DT: splitting criterion, pruning options, smoothing options, …
- KNN: K, distance metric, distance weighted averaging, …

Must optimize to each problem:
- failure to optimize makes superior algorithm inferior
- optimization depends on criterion
  - e.g., for kNN: $k_{\text{optimal}} \neq k_{\text{random}} \neq k_{\text{large}}$
- optimization depends on size of train set

Questions

- Is one algorithm “better” than the others?
- Are some learning methods best for certain loss functions?
  - SVMs for classification?
  - ANNs for regression or predicting probabilities?
- If no method(s) dominate, can we at least ignore some algs?
- Why are some methods good on loss X, but poor on loss Y?
- How do different losses relate to each other?
- Are some losses “better” than others?
- …
- What should you use ???
Data Sets

- 8 binary classification data sets (now 10 sets)
  - Adult
  - Cover Type
  - Letter.p1
  - Letter.p2
  - Pneumonia
  - Hyper Spectral
  - SLAC Particle Physics
  - Mg

- 4 k train sets
- 1 k validation sets
- Large final test sets (usually 20k)

Binary Classification Performance Metrics

- **Threshold Metrics:**
  - Accuracy
  - F-Score
  - Lift

- **Ordering/Ranking Metrics:**
  - ROC Area
  - Average Precision
  - Precision/Recall Break-Even Point

- **Probability Metrics:**
  - Root-Mean-Squared-Error
  - Cross-Entropy
  - Probability Calibration

Normalized Scores

- **Small Difficulty:**
  - some metrics, 1.00 is best (e.g. ACC)
  - some metrics, 0.00 is best (e.g. RMS)
  - some metrics, baseline is 0.50 (e.g. AUC)
  - some metrics, best depends on data (e.g. Lift)
  - some problems/metrics, 0.60 is excellent performance
  - some problems/metrics, 0.99 is poor performance

- **Solution: Normalized Scores:**
  - baseline performance => 0.00
  - best observed performance => 1.00 (proxy for Bayes optimal)
  - puts all metrics/problems on equal footing

Massive Empirical Comparison

10 learning methods

100's of parameter settings per method

5-fold cross validation

10,000+ models trained per problem

10 Boolean classification test problems

100,000+ models

9 performance metrics

900,000+ model evaluations
Look at Predicting Probabilities First

- Why?
  - don’t want to hit you with results for nine metrics all at once
  - if you can predict correct conditional probabilities, you’re done—all reasonable performance metrics are optimized by predicting true probabilities
  - results for probabilities are interesting by themselves*

* Alex Niculescu won a best student paper award at ICML05 for this work on predicting probabilities

Results on Test Sets (Normalized Scores)

- Best probabilities overall:
  - Neural Nets
  - Bagged Decision Trees
  - Random Forests

- Not competitive:
  - Boosted decision trees and stumps (exponential loss)
  - SVMs (standard hinge-loss)

- SVMs scaled to [0,1] via simple min/max scaling

Bagged Decision Trees

- Draw 100 bootstrap samples of data
- Train trees on each sample -> 100 trees
- Un-weighted average prediction of trees

Bagging Results

- Highly under-rated!
Random Forests (Bagged Trees++)

- Draw 1000+ bootstrap samples of data
- Draw sample of available attributes at each split
- Train trees on each sample/attribute set -> 1000+ trees
- Un-weighted average prediction of trees

<table>
<thead>
<tr>
<th>Model</th>
<th>Probability Metrics</th>
<th>Cross-Entropy</th>
<th>Squared Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>0.872 0.878 0.326 0.859</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAG-RF</td>
<td>0.875 0.901 0.677 0.884</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-1FR</td>
<td>0.882 0.899 0.917 0.783</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>0.783 0.799 0.884 0.745</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOG-REG</td>
<td>0.814 0.830 0.878 0.852</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0.583 0.616 0.512 0.576</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RFR</td>
<td>0.096 0.090 0.045 0.091</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVRMIX</td>
<td>0.484 0.447 0.080 0.310</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUS-RELMP</td>
<td>0.355 0.320 0.320 0.272</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAIVE-R</td>
<td>0.271 0.300 0.100 0.100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Back to SVMs: Results on Test Sets

- Best probabilities overall:
  - Neural Nets
  - Bagged Probabilistic Trees
  - Random Forests

- Not competitive:
  - Boosted decision trees and stumps (with exponential loss)
  - SVMs (with standard loss)

SVM Reliability Plots

- Calibration & Reliability Diagrams
Platt Scaling by Fitting a Sigmoid

- Linear scaling of SVM \([-\infty, +\infty]\) predictions to \([0,1]\) is bad.
- Platt’s Method [Platt 1999]:
  - scale predictions by fitting sigmoid on a validation set using 3-fold CV and Bayes-motivated smoothing to avoid overfitting.

![Graph showing Platt Scaling by Fitting a Sigmoid]

Results After Platt Scaling SVMs

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>F-Score</th>
<th>Lift</th>
<th>ROC Area</th>
<th>Average Precision</th>
<th>Break Even Point</th>
<th>Squared Error</th>
<th>Cross-Entropy</th>
<th>Calibration</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>0.817</td>
<td>0.875</td>
<td>0.947</td>
<td>0.963</td>
<td>0.956</td>
<td>0.829</td>
<td>0.872</td>
<td>0.878</td>
<td>0.826</td>
<td>0.892</td>
</tr>
<tr>
<td>SVM-PLT</td>
<td>0.823</td>
<td>0.851</td>
<td>0.958</td>
<td>0.961</td>
<td>0.951</td>
<td>0.829</td>
<td>0.882</td>
<td>0.880</td>
<td>0.769</td>
<td>0.844</td>
</tr>
<tr>
<td>RF</td>
<td>0.836</td>
<td>0.889</td>
<td>0.933</td>
<td>0.972</td>
<td>0.950</td>
<td>0.828</td>
<td>0.875</td>
<td>0.901</td>
<td>0.817</td>
<td>0.878</td>
</tr>
<tr>
<td>BAG-DT</td>
<td>0.844</td>
<td>0.845</td>
<td>0.918</td>
<td>0.977</td>
<td>0.957</td>
<td>0.904</td>
<td>0.882</td>
<td>0.899</td>
<td>0.904</td>
<td>0.907</td>
</tr>
<tr>
<td>KNN</td>
<td>0.759</td>
<td>0.830</td>
<td>0.914</td>
<td>0.957</td>
<td>0.983</td>
<td>0.888</td>
<td>0.783</td>
<td>0.789</td>
<td>0.664</td>
<td>0.831</td>
</tr>
<tr>
<td>RST-DT</td>
<td>0.861</td>
<td>0.885</td>
<td>0.956</td>
<td>0.977</td>
<td>0.958</td>
<td>0.952</td>
<td>0.996</td>
<td>0.998</td>
<td>0.145</td>
<td>0.758</td>
</tr>
<tr>
<td>DT</td>
<td>0.612</td>
<td>0.789</td>
<td>0.856</td>
<td>0.781</td>
<td>0.848</td>
<td>0.808</td>
<td>0.583</td>
<td>0.638</td>
<td>0.512</td>
<td>0.717</td>
</tr>
<tr>
<td>BAG-STMP</td>
<td>0.701</td>
<td>0.782</td>
<td>0.808</td>
<td>0.826</td>
<td>0.871</td>
<td>0.854</td>
<td>0.355</td>
<td>0.399</td>
<td>0.125</td>
<td>0.670</td>
</tr>
<tr>
<td>LOG-REG</td>
<td>0.802</td>
<td>0.823</td>
<td>0.829</td>
<td>0.849</td>
<td>0.752</td>
<td>0.714</td>
<td>0.614</td>
<td>0.620</td>
<td>0.678</td>
<td>0.696</td>
</tr>
<tr>
<td>NAVS-R</td>
<td>0.414</td>
<td>0.627</td>
<td>0.746</td>
<td>0.767</td>
<td>0.698</td>
<td>0.689</td>
<td>0.271</td>
<td>0.200</td>
<td>0.00</td>
<td>0.409</td>
</tr>
</tbody>
</table>

* Boosted trees outperform everything else on 5 of 6 non-probability metrics.
* But boosting predicts poor probabilities.

Results After Platt Scaling SVMs

<table>
<thead>
<tr>
<th>Model</th>
<th>Squared Error</th>
<th>Cross-Entropy</th>
<th>Calibration</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>0.872</td>
<td>0.878</td>
<td>0.826</td>
<td>0.859</td>
</tr>
<tr>
<td>SVM-PLT</td>
<td>0.882</td>
<td>0.880</td>
<td>0.769</td>
<td>0.844</td>
</tr>
<tr>
<td>BAG-DT</td>
<td>0.875</td>
<td>0.901</td>
<td>0.637</td>
<td>0.904</td>
</tr>
<tr>
<td>BAG-FOB</td>
<td>0.882</td>
<td>0.899</td>
<td>0.507</td>
<td>0.783</td>
</tr>
<tr>
<td>KNN</td>
<td>0.783</td>
<td>0.769</td>
<td>0.684</td>
<td>0.745</td>
</tr>
<tr>
<td>LOG-REG</td>
<td>0.614</td>
<td>0.620</td>
<td>0.678</td>
<td>0.637</td>
</tr>
<tr>
<td>DT</td>
<td>0.783</td>
<td>0.636</td>
<td>0.512</td>
<td>0.578</td>
</tr>
<tr>
<td>RST-DT</td>
<td>0.596</td>
<td>0.596</td>
<td>0.045</td>
<td>0.413</td>
</tr>
<tr>
<td>SVM-AKA</td>
<td>0.484</td>
<td>0.447</td>
<td>0.000</td>
<td>0.510</td>
</tr>
<tr>
<td>RST-STMP</td>
<td>0.355</td>
<td>0.330</td>
<td>0.123</td>
<td>0.272</td>
</tr>
<tr>
<td>NAVS-R</td>
<td>0.271</td>
<td>0.000</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

* Platt’s Method (Platt 1999) for obtaining posterior probabilities from SVMs by fitting a sigmoid.
* SVM probabilities as good as Neural Net probabilities after scaling with Platt’s Method.
* SVMs slightly better than Neural Nets on 2 of 3 metrics!
* Would other learning methods benefit from calibration with Platt’s Method?
### Summary of Model Performances

<table>
<thead>
<tr>
<th>Model</th>
<th>Best Count</th>
<th>Mean NS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>17</td>
<td>0.892</td>
</tr>
<tr>
<td>SVM-PLT</td>
<td>2</td>
<td>0.884</td>
</tr>
<tr>
<td>BAG-DT</td>
<td>13</td>
<td>0.878</td>
</tr>
<tr>
<td>RND-FOR</td>
<td>4</td>
<td>0.875</td>
</tr>
<tr>
<td>KNN</td>
<td>6</td>
<td>0.831</td>
</tr>
<tr>
<td>BST-DT</td>
<td>19</td>
<td>0.738</td>
</tr>
<tr>
<td>DT</td>
<td>2</td>
<td>0.717</td>
</tr>
<tr>
<td>BST-STMP</td>
<td>7</td>
<td>0.650</td>
</tr>
<tr>
<td>LOG-REG</td>
<td>1</td>
<td>0.696</td>
</tr>
<tr>
<td>NAIVE-B</td>
<td>1</td>
<td>0.460</td>
</tr>
</tbody>
</table>

### Good Model ≠ Good Probs

- Model can be accurate, but be poorly calibrated
  - Only sensitive to side of threshold case falls on
  - Use threshold ≠ 0.5 if poorly calibrated

- Model can have good ROC (Google-like ordering), but predict poor probabilities
  - ROC insensitive to scaling/stretching
  - Only ordering has to be correct, not probabilities

### Ada Boosting

- **Initialization:**
  - Weight all training samples equally

- **Iteration (typically requires 100’s to 1000’s of iterations):**
  - Train model on (weighted) train set
  - Compute error of model on train set
  - Increase weights on cases model gets wrong

- **Return final model:**
  - Carefully weighted prediction of each model

### Why Boosting is Not Well Calibrated

- Predicted values pushed away from 0 and 1
- Calibration becomes increasingly worse
- Shape of the reliability plot becomes sigmoidal
- Looks a lot like SVM predictions
Consistent With Interpretations of Boosting

- Boosting is a maximum-margin method (Schapire et al. 1998, Rosset et al. 2004)
  - Trades lower margin on easy cases for higher margin on harder cases

- Boosting is an additive logistic regression model (Friedman, Hastie and Tibshirani 2000)
  - Tries to fit the logit of the true conditional probabilities

- Boosting is an equalizer (Breiman 1998) (Friedman, Hastie, Tibshirani 2000)
  - Weighted proportion of times example is misclassified by base learners tends to be the same for all training cases

Results After Platt Scaling All Models

- Models that benefit from calibration:
  - SVMs
  - Boosted decision trees
  - Boosted stumps
  - Random forests
  - Naive Bayes
  - Vanilla decision trees

- Do not benefit from calibration:
  - Neural nets
  - Bagged trees
  - Logistic regression
  - MTL-KNN

- Boosting full trees dominates

Platt Scaling of Boosted Trees (7 problems)

Before (Ada Boost with exponential loss): P1 P2 P3 P4 P5 P6 P7

After Platt Scaling: P1 P2 P3 P4 P5 P6 P7

Models that benefit from calibration:
- SVMs
- Boosted decision trees
- Boosted stumps
- Random forests
- Naive Bayes
- Vanilla decision trees

Do not benefit from calibration:
- Neural nets
- Bagged trees
- Logistic regression
- MTL-KNN

Boosting full trees dominates

Return of the Decision Tree!

- After Platt Scaling, boosted trees are best models overall across all metrics
- Neural nets are best models overall if no calibration is applied post-training
Methods for Achieving Calibration

- **Optimize directly to appropriate criterion:**
  - Boosting with log-loss (Collins, Schapire & Singer 2001)
  - SVM to maximize likelihood (e.g. Wahba 1999)
  - Performance comparable to Platt Scaling (Platt 1999)
  - Yields non-sparse solution
  - No need for post-training calibration with these approaches

- **Train models with “usual” criterion and post-calibrate:**
  - Logistic Correction
    - Analytic method justified by the Friedman et al.’s analysis
  - Platt Scaling
  - Method used by Platt to calibrate SVMs by fitting a sigmoid
  - Is sigmoid right calibration function for most learning methods?
  - Isotonic Regression
    - Very general calibration method used by Zadrozny & Elkan (2001)
    - PAV (Pair Adjacent Violators) algorithm (Ayer et al. 1955)
  - Efficient linear-time algorithm

Boosting with Log-Loss

- Log-loss does improve calibration of boosting, but
- Most effective with weak models such as 1-level stumps
- Less effective with more complex models such as full decision trees (or even 2-level stumps)
- Post-calibration with Platt’s Method far more effective, particularly with complex models such as full trees
- Best probabilities come from full trees boosted with exponential loss then calibrated with Platt’s Method

Isotonic Regression

- Basic assumption - there exists an isotonic (monotonically increasing) function \( m \) s.t.:
  \[
  y_i = m(f_i) + \epsilon_i
  \]
- We want to find an isotonic function \( m \) s.t.:
  \[
  \hat{m} = \arg\min_{z} \sum (y_i - z(f_i))^2
  \]
- Bianca Zadrozny and Charles Elkan (2001) first to use isotonic regression for calibration in ML community
**PAV Algorithm**

1. **Algorithm 1.** PAV algorithm for estimating posterior probabilities from uncalibrated model predictions.
   - Input: training set \((f_i, y_i)\) sorted according to \(f_i\)
   - Initialize \(m_{i,1} = y_i, \ w_{i,1} = 1\)
   - While \(\exists i \text{ s.t. } \hat{m}_{k,i} \geq \hat{m}_{i,l}\)
     - Set \(w_{k,i} = w_{k,i-1} + w_{i,l}\)
     - Set \(\hat{m}_{k,i} = (w_{k,i-1} \hat{m}_{k,i-1} + w_{i,l} \hat{m}_{i,l})/w_{k,i}\)
     - Replace \(\hat{m}_{k,i-1}\) and \(\hat{m}_{i,l}\) with \(\hat{m}_{k,i}\)
   - Output the stepwise const. function generated by \(\hat{m}\)

**Isotonic Regression**

- Before:
- After:

**Platt Scaling**

- Before:
- After:

**Platt Scaling vs. Isotonic Regression**

- Platt Scaling:
- Isotonic Regression:
**Summary**

- Boosting full trees outperforms boosting weaker models
- Calibration via Platt Scaling or Isotonic Regression more effective than boosting log-loss when boosting trees
- Platt Scaling better with small data (< 1000 points)
- Isotonic Regression better with large data (> 1000 points)
- Not all learning methods benefit from calibration
- Before calibration, well-tuned neural nets predict the best probabilities
- After calibration, boosted probabilistic decision trees predict best probabilities

**Where Does That Leave Us?**

- Calibration via Platt Scaling or Isotonic Regression improves probs from max-margin methods such as Boosted Trees and SVMs
- Boosted Trees + Calibration best overall
- Are we done?
- No!
### Best of the Best of the Best

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>F-Score</th>
<th>MAP</th>
<th>ROC Area</th>
<th>Average Precision</th>
<th>Break Even Point</th>
<th>Squared Error</th>
<th>Cross-Entropy</th>
<th>Calibration</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>BST</td>
<td>0.928</td>
<td>0.918</td>
<td>0.975</td>
<td>0.907</td>
<td>0.908</td>
<td>0.919</td>
<td>0.944</td>
<td>0.989</td>
<td>0.9333</td>
<td></td>
</tr>
<tr>
<td>ANN</td>
<td>0.860</td>
<td>0.884</td>
<td>0.956</td>
<td>0.977</td>
<td>0.958</td>
<td>0.952</td>
<td>0.932</td>
<td>0.880</td>
<td>0.914</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.817</td>
<td>0.917</td>
<td>0.947</td>
<td>0.963</td>
<td>0.928</td>
<td>0.929</td>
<td>0.884</td>
<td>0.886</td>
<td>0.826</td>
<td></td>
</tr>
<tr>
<td>RF</td>
<td>0.833</td>
<td>0.833</td>
<td>0.953</td>
<td>0.972</td>
<td>0.950</td>
<td>0.928</td>
<td>0.875</td>
<td>0.901</td>
<td>0.878</td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>0.799</td>
<td>0.820</td>
<td>0.914</td>
<td>0.890</td>
<td>0.898</td>
<td>0.786</td>
<td>0.805</td>
<td>0.706</td>
<td>0.835</td>
<td></td>
</tr>
<tr>
<td>BST-RF</td>
<td>0.818</td>
<td>0.849</td>
<td>0.953</td>
<td>0.972</td>
<td>0.950</td>
<td>0.928</td>
<td>0.875</td>
<td>0.901</td>
<td>0.878</td>
<td></td>
</tr>
<tr>
<td>ANN-RF</td>
<td>0.823</td>
<td>0.831</td>
<td>0.953</td>
<td>0.972</td>
<td>0.950</td>
<td>0.928</td>
<td>0.875</td>
<td>0.901</td>
<td>0.878</td>
<td></td>
</tr>
<tr>
<td>SVM-RF</td>
<td>0.833</td>
<td>0.833</td>
<td>0.953</td>
<td>0.972</td>
<td>0.950</td>
<td>0.928</td>
<td>0.875</td>
<td>0.901</td>
<td>0.878</td>
<td></td>
</tr>
<tr>
<td>RF-KNN</td>
<td>0.611</td>
<td>0.771</td>
<td>0.856</td>
<td>0.871</td>
<td>0.888</td>
<td>0.856</td>
<td>0.825</td>
<td>0.608</td>
<td>0.734</td>
<td></td>
</tr>
<tr>
<td>NFC</td>
<td>0.802</td>
<td>0.823</td>
<td>0.839</td>
<td>0.849</td>
<td>0.752</td>
<td>0.714</td>
<td>0.604</td>
<td>0.630</td>
<td>0.678</td>
<td></td>
</tr>
<tr>
<td>NAIVE-B</td>
<td>0.536</td>
<td>0.625</td>
<td>0.766</td>
<td>0.633</td>
<td>0.733</td>
<td>0.730</td>
<td>0.539</td>
<td>0.565</td>
<td>0.161</td>
<td></td>
</tr>
</tbody>
</table>

### Current Ensemble Methods

- Bagging
- Boosting
- Random Forests
- Error Correcting Output Codes (ECOC) …

- Average of multiple models
- Bayesian Model Averaging
- Stacking …

- Ensemble methods differ in:
  - how models are generated
  - how models are combined

---

**If we need to train all models and pick best, can we do better than picking best?**

"A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse."

-- Tom Dietterich (2000)
Normalized Scores of Ensembles

<table>
<thead>
<tr>
<th>Model</th>
<th>Threshold Metrics</th>
<th>Rank-Ordering Metrics</th>
<th>Probability Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Precision</td>
<td>Score</td>
</tr>
<tr>
<td>BAYESAVG</td>
<td>0.9566</td>
<td>0.9504</td>
<td>0.9557</td>
</tr>
<tr>
<td>BAYESAVG</td>
<td>0.9871</td>
<td>0.9851</td>
<td>0.9851</td>
</tr>
<tr>
<td>STACK_LR</td>
<td>0.8906</td>
<td>0.9258</td>
<td>0.9258</td>
</tr>
<tr>
<td>BEST</td>
<td>0.9067</td>
<td>0.9188</td>
<td>0.9188</td>
</tr>
</tbody>
</table>

New Ensemble Method: ES

- Train many different models:
  - different algorithms
  - different parameter settings
  - all trained on same train set
  - all trained to “natural” optimization criterion
- Add all models to library:
  - no model selection
  - no validation set
  - some models bad, some models good, a few models excellent
  - yields diverse set of models, some of which are good on almost any metric
- Forward stepwise model selection from library:
  - start with empty ensemble
  - try adding each model one-at-a-time to ensemble
  - commit model that maximizes performance on metric on a test set
  - repeat until performance stops getting better

Basic Ensemble Selection Algorithm

<table>
<thead>
<tr>
<th>Model Library</th>
<th>Ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td></td>
</tr>
<tr>
<td>Model 2</td>
<td></td>
</tr>
<tr>
<td>Model 3</td>
<td></td>
</tr>
<tr>
<td>Model 4</td>
<td></td>
</tr>
<tr>
<td>Model 5</td>
<td></td>
</tr>
<tr>
<td>Model 6</td>
<td></td>
</tr>
<tr>
<td>Model 7</td>
<td></td>
</tr>
<tr>
<td>Model 8</td>
<td></td>
</tr>
<tr>
<td>Model 9</td>
<td></td>
</tr>
</tbody>
</table>

Basic Ensemble Selection Algorithm

<table>
<thead>
<tr>
<th>Model Library</th>
<th>AUC Score on the 1k validation set</th>
<th>Ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.8453</td>
<td></td>
</tr>
<tr>
<td>Model 2</td>
<td>0.8726</td>
<td></td>
</tr>
<tr>
<td>Model 3</td>
<td>0.9164</td>
<td></td>
</tr>
<tr>
<td>Model 4</td>
<td>0.8142</td>
<td></td>
</tr>
<tr>
<td>Model 5</td>
<td>0.8453</td>
<td></td>
</tr>
<tr>
<td>Model 6</td>
<td>0.8745</td>
<td></td>
</tr>
<tr>
<td>Model 7</td>
<td>0.9024</td>
<td></td>
</tr>
<tr>
<td>Model 8</td>
<td>0.7034</td>
<td></td>
</tr>
<tr>
<td>Model 9</td>
<td>0.8342</td>
<td></td>
</tr>
</tbody>
</table>
### Basic Ensemble Selection Algorithm

<table>
<thead>
<tr>
<th>Model Library</th>
<th>AUC Score on the 1k validation set</th>
<th>Ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.8453</td>
<td></td>
</tr>
<tr>
<td>Model 2</td>
<td>0.8726</td>
<td></td>
</tr>
<tr>
<td><strong>Model 3</strong></td>
<td><strong>0.9164</strong></td>
<td><strong>0.9164</strong></td>
</tr>
<tr>
<td>Model 4</td>
<td>0.8142</td>
<td></td>
</tr>
<tr>
<td>Model 5</td>
<td>0.8453</td>
<td></td>
</tr>
<tr>
<td>Model 6</td>
<td>0.8745</td>
<td></td>
</tr>
<tr>
<td>Model 7</td>
<td>0.9024</td>
<td></td>
</tr>
<tr>
<td>Model 8</td>
<td>0.7034</td>
<td></td>
</tr>
<tr>
<td>Model 9</td>
<td>0.8342</td>
<td></td>
</tr>
</tbody>
</table>

AUC Score on the 1k validation set

\[ + \text{ Ensemble } = \]

<table>
<thead>
<tr>
<th>Model Library</th>
<th>AUC Score on the 1k validation set</th>
<th>Ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.8327</td>
<td></td>
</tr>
<tr>
<td>Model 2</td>
<td>0.8702</td>
<td></td>
</tr>
<tr>
<td>Model 3</td>
<td><strong>0.9164</strong></td>
<td><strong>0.9164</strong></td>
</tr>
<tr>
<td>Model 4</td>
<td>0.9284</td>
<td></td>
</tr>
<tr>
<td>Model 5</td>
<td>0.9047</td>
<td></td>
</tr>
<tr>
<td>Model 6</td>
<td>0.8832</td>
<td></td>
</tr>
<tr>
<td>Model 7</td>
<td>0.9126</td>
<td></td>
</tr>
<tr>
<td>Model 8</td>
<td>0.8245</td>
<td></td>
</tr>
<tr>
<td>Model 9</td>
<td>0.9384</td>
<td></td>
</tr>
</tbody>
</table>

AUC Score on the 1k validation set

\[ + \text{ Ensemble } = \]
Basic Ensemble Selection Algorithm

<table>
<thead>
<tr>
<th>Model Library</th>
<th>Ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Model 3</td>
</tr>
<tr>
<td>Model 2</td>
<td>Model 9</td>
</tr>
<tr>
<td>Model 3</td>
<td>0.9164</td>
</tr>
<tr>
<td>Model 4</td>
<td>0.9384</td>
</tr>
</tbody>
</table>

AUC Score on the 1k validation set + Ensemble =

<table>
<thead>
<tr>
<th>Model Library</th>
<th>AUC Score on the 1k validation set</th>
<th>Ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.8502 0.8327</td>
<td>Model 3</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.9243 0.8702</td>
<td>Model 9</td>
</tr>
<tr>
<td>Model 4</td>
<td>0.8992 0.9284</td>
<td></td>
</tr>
<tr>
<td>Model 5</td>
<td>0.8090 0.9047</td>
<td></td>
</tr>
<tr>
<td>Model 6</td>
<td>0.9424 0.8832</td>
<td></td>
</tr>
<tr>
<td>Model 7</td>
<td>0.9045 0.9126</td>
<td></td>
</tr>
<tr>
<td>Model 8</td>
<td>0.9243 0.8245</td>
<td></td>
</tr>
</tbody>
</table>

Basic Ensemble Selection Algorithm

<table>
<thead>
<tr>
<th>Model Library</th>
<th>AUC Score on the 1k validation set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.9164</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.9384</td>
</tr>
<tr>
<td>Model 3</td>
<td>0.9164</td>
</tr>
<tr>
<td>Model 4</td>
<td>0.9384</td>
</tr>
<tr>
<td>Model 5</td>
<td>0.9424</td>
</tr>
<tr>
<td>Model 6</td>
<td>0.9424</td>
</tr>
<tr>
<td>Model 7</td>
<td>0.9424</td>
</tr>
<tr>
<td>Model 8</td>
<td>0.9424</td>
</tr>
</tbody>
</table>

Basic Ensemble Selection Algorithm

<table>
<thead>
<tr>
<th>Model Library</th>
<th>AUC Score on the 1k validation set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.8502 0.8327</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.9243 0.8702</td>
</tr>
<tr>
<td>Model 3</td>
<td>0.9164 0.9384</td>
</tr>
<tr>
<td>Model 4</td>
<td>0.8992 0.9284</td>
</tr>
<tr>
<td>Model 5</td>
<td>0.8090 0.9047</td>
</tr>
<tr>
<td>Model 6</td>
<td>0.9424 0.8832</td>
</tr>
<tr>
<td>Model 7</td>
<td>0.9045 0.9126</td>
</tr>
<tr>
<td>Model 8</td>
<td>0.9243 0.8245</td>
</tr>
</tbody>
</table>
Big Problem: Overfitting

- More models => better chance of finding combination with good performance on any given problem and metric,
- but …
- also better chance of overfitting to the hillclimb set

- Tricks to Reduce Overfitting:
  - Eliminate Inferior Models: prevents mistakes
  - Ensemble Initialization: give “inertia” to initial ensemble
  - Stepwise Selection with Replacement: stopping point less critical
  - Calibrate Models in Ensemble: all models speak same language
  - Bagged Ensemble Selection: reduces variance

- Critical to take steps to reduce overfitting

1st Trick: Ensemble Initialization

Instead of starting with empty ensemble, initialize with best N models

2nd Trick: Selection with Replacement

- After initializing ensemble with best N models

- Forward Selection with Replacement:
  - add each model one-at-a-time to ensemble
  - models added by averaging predictions
  - calculate performance metric
  - commit model that improves performance most
  - repeat until ensemble too large (we typically use ~ 250 steps)
  - return ensemble with best performance on validation set

- models added 3 times have 3X weight of models added once
- simple form of model weighting is less prone to overfitting
3rd Trick: Bagged Selection

- Draw a sample of models from library (we use $p = 0.5$)
- Do ensemble selection from this sample of models
- Repeat $N$ times (we use $N=20$)
- Final model is average of the $N$ ensembles
  - each ensemble is simple weighted average of base-level models
  - average of $N$ such ensembles also is a simple weighted average of the base-level models

Normalized Scores for ES

<table>
<thead>
<tr>
<th>Model</th>
<th>Threshold Metrics</th>
<th>Rank/Ordering Metrics</th>
<th>Probability Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>F-Score</td>
<td>AUC Area</td>
</tr>
<tr>
<td>ES</td>
<td>0.9506</td>
<td>0.9442</td>
<td>0.9916</td>
</tr>
<tr>
<td>MAE</td>
<td>0.9283</td>
<td>0.9188</td>
<td>0.9733</td>
</tr>
<tr>
<td>MSE</td>
<td>0.9363</td>
<td>0.9116</td>
<td>0.9886</td>
</tr>
<tr>
<td>RNN</td>
<td>0.8763</td>
<td>0.8807</td>
<td>0.9721</td>
</tr>
<tr>
<td>STACK</td>
<td>0.8751</td>
<td>0.7772</td>
<td>0.7992</td>
</tr>
<tr>
<td>BAG-DT</td>
<td>0.8600</td>
<td>0.8540</td>
<td>0.9556</td>
</tr>
<tr>
<td>BAYESAVG</td>
<td>0.9283</td>
<td>0.9096</td>
<td>0.9708</td>
</tr>
<tr>
<td>BEST</td>
<td>0.8978</td>
<td>0.7028</td>
<td>0.8981</td>
</tr>
<tr>
<td>AVG-ALL</td>
<td>0.9067</td>
<td>0.8540</td>
<td>0.9556</td>
</tr>
</tbody>
</table>
Ensemble Selection vs Best: 3 NLP Problems

Ensemble Selection Works, But Is It Worth It?

Computational Cost
- Have to train multiple models anyway
  - models can be trained in parallel
    - different packages, different machines, at different times, by different people
  - just generate and collect (no optimization necessary, no test sets)
  - saves human effort - no need to examine/optimize models
  - model library can be built before optimization metric is known
  - anytime selection - no need to wait for all models
- Ensemble Selection is cheap:
  - each iteration, consider adding 2000+ models to ensemble
  - adding model is simple unweighted averaging of predictions
  - caching makes this very efficient
  - compute performance metric when each model is added
  - for 250 iterations, evaluate 250*2000 = 500,000 ensembles
  - ~ 1 minute on workstation if metric is not expensive

What Models are Used in Ensembles?

<table>
<thead>
<tr>
<th>What Models are Used in Ensembles?</th>
<th>Acc</th>
<th>Fsc</th>
<th>Lft</th>
<th>Rec</th>
<th>Apr</th>
<th>Bep</th>
<th>Rms</th>
<th>Mse</th>
<th>Sur</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADULT</td>
<td>0.71</td>
<td>0.132</td>
<td>0.101</td>
<td>0.360</td>
<td>0.430</td>
<td>0.243</td>
<td>0.167</td>
<td>0.944</td>
<td>0.573</td>
<td>0.272</td>
</tr>
<tr>
<td>ANN</td>
<td>0.020</td>
<td>0.015</td>
<td>0.586</td>
<td>0.271</td>
<td>0.029</td>
<td>0.049</td>
<td>0.000</td>
<td>0.000</td>
<td>0.009</td>
<td>0.009</td>
</tr>
<tr>
<td>SVM</td>
<td>0.001</td>
<td>0.000</td>
<td>0.110</td>
<td>0.246</td>
<td>0.274</td>
<td>0.092</td>
<td>0.057</td>
<td>0.000</td>
<td>0.022</td>
<td>0.105</td>
</tr>
<tr>
<td>DT</td>
<td>0.020</td>
<td>0.035</td>
<td>0.007</td>
<td>0.055</td>
<td>0.049</td>
<td>0.199</td>
<td>0.746</td>
<td>0.867</td>
<td>0.234</td>
<td>0.258</td>
</tr>
<tr>
<td>BAG_DT</td>
<td>0.002</td>
<td>0.001</td>
<td>0.000</td>
<td>0.004</td>
<td>0.005</td>
<td>0.002</td>
<td>0.006</td>
<td>0.100</td>
<td>0.014</td>
<td>0.005</td>
</tr>
<tr>
<td>BST_DT</td>
<td>0.101</td>
<td>0.152</td>
<td>0.255</td>
<td>0.057</td>
<td>0.032</td>
<td>0.047</td>
<td>0.024</td>
<td>0.028</td>
<td>0.075</td>
<td>0.069</td>
</tr>
<tr>
<td>BST_STMP</td>
<td>0.776</td>
<td>0.666</td>
<td>0.171</td>
<td>0.166</td>
<td>0.181</td>
<td>0.548</td>
<td>0.000</td>
<td>0.000</td>
<td>0.032</td>
<td>0.317</td>
</tr>
<tr>
<td>CVD_TYPE</td>
<td>0.010</td>
<td>0.010</td>
<td>0.001</td>
<td>0.038</td>
<td>0.023</td>
<td>0.009</td>
<td>0.087</td>
<td>0.097</td>
<td>0.052</td>
<td>0.056</td>
</tr>
<tr>
<td>ANN</td>
<td>0.179</td>
<td>0.166</td>
<td>0.576</td>
<td>0.232</td>
<td>0.295</td>
<td>0.202</td>
<td>0.436</td>
<td>0.427</td>
<td>0.364</td>
<td>0.362</td>
</tr>
<tr>
<td>KNN</td>
<td>0.021</td>
<td>0.016</td>
<td>0.087</td>
<td>0.056</td>
<td>0.106</td>
<td>0.051</td>
<td>0.010</td>
<td>0.013</td>
<td>0.038</td>
<td>0.056</td>
</tr>
<tr>
<td>SVM</td>
<td>0.061</td>
<td>0.054</td>
<td>0.012</td>
<td>0.030</td>
<td>0.242</td>
<td>0.029</td>
<td>0.088</td>
<td>0.368</td>
<td>0.200</td>
<td>0.202</td>
</tr>
<tr>
<td>DT</td>
<td>0.005</td>
<td>0.006</td>
<td>0.002</td>
<td>0.010</td>
<td>0.015</td>
<td>0.006</td>
<td>0.016</td>
<td>0.022</td>
<td>0.044</td>
<td>0.036</td>
</tr>
<tr>
<td>BAG_DT</td>
<td>0.553</td>
<td>0.613</td>
<td>0.130</td>
<td>0.279</td>
<td>0.240</td>
<td>0.644</td>
<td>0.442</td>
<td>0.073</td>
<td>0.292</td>
<td>0.358</td>
</tr>
<tr>
<td>BST_DT</td>
<td>0.170</td>
<td>0.134</td>
<td>0.194</td>
<td>0.030</td>
<td>0.080</td>
<td>0.059</td>
<td>0.000</td>
<td>0.000</td>
<td>0.009</td>
<td>0.091</td>
</tr>
</tbody>
</table>

[Art Munson, Claire Cardie, Rich Caruana. EMNLP/HLDT 2005]
### What Models are Used in Ensembles?

<table>
<thead>
<tr>
<th>COV-TYPE</th>
<th>Acc</th>
<th>Fsc</th>
<th>Lift</th>
<th>Roc</th>
<th>Apr</th>
<th>Bep</th>
<th>Rms</th>
<th>Mse</th>
<th>Sat</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>.071</td>
<td>.132</td>
<td>.101</td>
<td>.365</td>
<td>.430</td>
<td>.243</td>
<td>.167</td>
<td>.094</td>
<td>.573</td>
<td>.272</td>
</tr>
<tr>
<td>KNN</td>
<td>.020</td>
<td>.015</td>
<td>.586</td>
<td>.037</td>
<td>.029</td>
<td>.049</td>
<td>.000</td>
<td>.000</td>
<td>.049</td>
<td>.093</td>
</tr>
<tr>
<td>SVM</td>
<td>.001</td>
<td>.000</td>
<td>.110</td>
<td>.284</td>
<td>.274</td>
<td>.192</td>
<td>.057</td>
<td>.000</td>
<td>.022</td>
<td>.105</td>
</tr>
<tr>
<td>DT</td>
<td>.020</td>
<td>.035</td>
<td>.007</td>
<td>.088</td>
<td>.049</td>
<td>.019</td>
<td>.746</td>
<td>.867</td>
<td>.234</td>
<td>.258</td>
</tr>
<tr>
<td>BAG_DT</td>
<td>.002</td>
<td>.001</td>
<td>.000</td>
<td>.004</td>
<td>.005</td>
<td>.002</td>
<td>.006</td>
<td>.010</td>
<td>.014</td>
<td>.005</td>
</tr>
<tr>
<td>BST_DT</td>
<td>.110</td>
<td>.152</td>
<td>.025</td>
<td>.057</td>
<td>.032</td>
<td>.047</td>
<td>.024</td>
<td>.028</td>
<td>.075</td>
<td>.069</td>
</tr>
<tr>
<td>BST_STMP</td>
<td>.776</td>
<td>.666</td>
<td>.171</td>
<td>.166</td>
<td>.181</td>
<td>.548</td>
<td>.000</td>
<td>.000</td>
<td>.032</td>
<td>.112</td>
</tr>
<tr>
<td>COV-TYPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANN</td>
<td>.011</td>
<td>.010</td>
<td>.001</td>
<td>.038</td>
<td>.023</td>
<td>.009</td>
<td>.087</td>
<td>.097</td>
<td>.052</td>
<td>.041</td>
</tr>
<tr>
<td>SVM</td>
<td>.021</td>
<td>.016</td>
<td>.087</td>
<td>.104</td>
<td>.106</td>
<td>.051</td>
<td>.010</td>
<td>.013</td>
<td>.038</td>
<td>.056</td>
</tr>
<tr>
<td>DT</td>
<td>.061</td>
<td>.054</td>
<td>.012</td>
<td>.238</td>
<td>.242</td>
<td>.029</td>
<td>.408</td>
<td>.368</td>
<td>.200</td>
<td>.202</td>
</tr>
<tr>
<td>BAG_DT</td>
<td>.005</td>
<td>.006</td>
<td>.002</td>
<td>.010</td>
<td>.015</td>
<td>.006</td>
<td>.016</td>
<td>.022</td>
<td>.044</td>
<td>.016</td>
</tr>
<tr>
<td>BST_DT</td>
<td>.553</td>
<td>.613</td>
<td>.130</td>
<td>.278</td>
<td>.240</td>
<td>.644</td>
<td>.042</td>
<td>.073</td>
<td>.292</td>
<td>.358</td>
</tr>
<tr>
<td>BST_STMP</td>
<td>.700</td>
<td>.134</td>
<td>.194</td>
<td>.080</td>
<td>.080</td>
<td>.059</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
<td>.091</td>
</tr>
</tbody>
</table>

### What Models are Used by ES?

- Most ensembles use 10-100 of the 2000 models
- Different models are selected for different problems
- Different models are selected for different metrics
- Most ensembles use a diversity of model types
- Most ensembles use different parameter settings
- Selected Models often make sense:
  - Neural nets for RMS, Cross-Entropy
  - Max-margin methods for Accuracy
  - Large $k$ in k nn for AUC

### ES Pros & Cons

**Disadvantages:**
- Have to train many models
  - If you want the best, you were going to do it anyway
  - Packages such as WEKA and MLC++ make it easier
- Loss of intelligibility
- No cool theory!

**Advantages:**
- Can optimize to almost any performance metric
- Better performance than anything else we compared to

### Ensemble Selection

**Good news:**
- A carefully selected ensemble that combines many models outperforms boosting, bagging, random forests, SVMs, and neural nets, … (just because it builds on top of them)

**Bad news:**
- The ensembles are too big, too slow, too cumbersome to use for most applications
Best Ensembles are Big and Ugly!

- Best ensemble for one problem/metric has 422 models:
  - 72 boosted trees (28,642 individual decision trees!)
  - 1 random forest (1024 decision trees)
  - 5 bagged trees (100 decision trees in each model)
  - 44 neural nets (2,200 hidden units, total, >100,000 weights)
  - 115 knn models (both large and expensive!)
  - 38 SVMs (100’s of support vectors in each model)
  - 26 boosted stump models (36,184 stumps total -- could compress)
  - 122 individual decision trees
  - ...
- Best ensemble:
  - takes more than 1GB to store
  - takes ~5 seconds to execute per test case!

Solution: Model Compression

- Pass large amounts of unlabeled data (synthetic data points or real unlabeled data) through ensemble and collect predictions
  - 100,000 to 10,000,000 synthetic training points
  - Extensional representation of the ensemble model
- Train copycat model on this large synthetic train set to mimic the high-performance ensemble
  - Train neural net to mimic ensemble
  - Potential to not only perform as well as target ensemble, but possibly outperform it

Work In Progress (Cristi Bucila)

Results

- Neural nets trained to mimic high performing bagged tree models
  - perform better than the target models on eight test problems and three test metrics
  - perform much better than any ANN we could train on the original data
- Massive experiment using ensemble selection predictions and nine performance metrics currently underway
  - getting ensemble predictions is much more expensive
  - willing to trade off cost at train-time for speed and compactness at run-time
Why Mimic with Neural Nets?

- Decision trees do not work well
  - Synthetic data must be very large because of recursive partitioning
  - Mimic decision trees are enormous (depth > 1000 and > $10^6$ nodes)
    making them expensive to store and compute
  - Single tree does not seem to model ensemble accurately enough

- SVMs
  - Number of support vectors increases quickly with complexity

- Artificial Neural nets
  - Can model complex functions with modest # of hidden units
  - Can compress millions of training cases into thousands of weights
  - Expense to train, but execution cost is low (just matrix multiplies)
  - Models with few thousand weights have small footprint

---

How Important is it to Optimize to the Correct Performance Metric?

---

RMS Loss for Simple 2-Param Model

Loss on Six Metrics for 2-Param Model
Scaling, Ranking, and Normalizing

- Problem:
  - some metrics, 1.00 is best (e.g., ACC)
  - some metrics, 0.00 is best (e.g., RMS)
  - some metrics, baseline is 0.50 (e.g., AUC)
  - some problems/metrics, 0.60 is excellent performance
  - some problems/metrics, 0.99 is poor performance

- Solution 1: Normalized Scores:
  - baseline performance => 0.00
  - best observed performance => 1.00 (proxy for Bayes optimal)
  - puts all metrics on equal footing

- Solution 2: Scale by Standard Deviation
- Solution 3: Rank Correlation

- The 10 metrics span a 2-5 dimension subspace
2-D Multi-Dimensional Scaling

Normalized Scores Scaling
Rank-Correlation Distance

Correlation Analysis

- 2000 performances for each metric on each problem
- Correlation between all pairs of metrics
  - 10 metrics
  - 45 pairwise correlations
- Average of correlations over 7 test problems
- Standard correlation
- Rank correlation
- Present rank correlation here
Rank Correlations

<table>
<thead>
<tr>
<th>Metric</th>
<th>ACC</th>
<th>FSC</th>
<th>LFT</th>
<th>AUC</th>
<th>APR</th>
<th>BEP</th>
<th>RMS</th>
<th>MXE</th>
<th>CAL</th>
<th>SAR</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>1.00</td>
<td>0.87</td>
<td>0.85</td>
<td>0.88</td>
<td>0.89</td>
<td>0.93</td>
<td>0.87</td>
<td>0.75</td>
<td>0.56</td>
<td>0.92</td>
<td>0.852</td>
</tr>
<tr>
<td>FSC</td>
<td>0.87</td>
<td>1.00</td>
<td>0.77</td>
<td>0.81</td>
<td>0.82</td>
<td>0.87</td>
<td>0.79</td>
<td>0.69</td>
<td>0.50</td>
<td>0.84</td>
<td>0.796</td>
</tr>
<tr>
<td>LFT</td>
<td>0.85</td>
<td>0.77</td>
<td>1.00</td>
<td>0.96</td>
<td>0.91</td>
<td>0.89</td>
<td>0.82</td>
<td>0.73</td>
<td>0.47</td>
<td>0.92</td>
<td>0.832</td>
</tr>
<tr>
<td>AUC</td>
<td>0.88</td>
<td>0.81</td>
<td>0.96</td>
<td>1.00</td>
<td>0.95</td>
<td>0.92</td>
<td>0.85</td>
<td>0.77</td>
<td>0.51</td>
<td>0.96</td>
<td>0.861</td>
</tr>
<tr>
<td>APR</td>
<td>0.89</td>
<td>0.82</td>
<td>0.91</td>
<td>0.95</td>
<td>1.00</td>
<td>0.92</td>
<td>0.86</td>
<td>0.75</td>
<td>0.50</td>
<td>0.93</td>
<td>0.853</td>
</tr>
<tr>
<td>BEP</td>
<td>0.93</td>
<td>0.87</td>
<td>0.89</td>
<td>0.92</td>
<td>0.92</td>
<td>1.00</td>
<td>0.87</td>
<td>0.75</td>
<td>0.52</td>
<td>0.93</td>
<td>0.860</td>
</tr>
<tr>
<td>RMS</td>
<td>0.87</td>
<td>0.79</td>
<td>0.82</td>
<td>0.85</td>
<td>0.86</td>
<td>0.87</td>
<td>1.00</td>
<td>0.92</td>
<td>0.79</td>
<td>0.95</td>
<td>0.872</td>
</tr>
<tr>
<td>MXE</td>
<td>0.75</td>
<td>0.69</td>
<td>0.73</td>
<td>0.77</td>
<td>0.75</td>
<td>0.75</td>
<td>0.92</td>
<td>1.00</td>
<td>0.81</td>
<td>0.86</td>
<td>0.803</td>
</tr>
<tr>
<td>CAL</td>
<td>0.56</td>
<td>0.50</td>
<td>0.47</td>
<td>0.51</td>
<td>0.50</td>
<td>0.52</td>
<td>0.79</td>
<td>0.81</td>
<td>1.00</td>
<td>0.65</td>
<td>0.631</td>
</tr>
<tr>
<td>SAR</td>
<td>0.92</td>
<td>0.84</td>
<td>0.92</td>
<td>0.96</td>
<td>0.93</td>
<td>0.93</td>
<td>0.95</td>
<td>0.86</td>
<td>0.65</td>
<td>1.00</td>
<td>0.896</td>
</tr>
</tbody>
</table>

- Correlation analysis consistent with MDS analysis
- Ordering metrics have high correlations to each other
- ACC, AUC, RMS have best correlations of metrics in each metric class
- RMS has good correlation to other metrics
- SAR has best correlation to other metrics

Summary

- Predicting Probabilities:
  - Neural Nets, Bagged Trees, Random Forests best models overall right out of box
  - Calibration with Platt Scaling or Isotonic Regression yields better probabilities for
    Boosting, SVMs, Random Forests, Decision Trees, and Naïve Bayes
  - Where sigmoid is appropriate, Platt Scaling is more effective with little data
  - Isotonic Regression more powerful, use when data is plentiful
- Empirical Comparison:
  - Calibrated Boosted Trees best performance overall (win 7/9 metrics!)
  - Even after calibration, no one learning method does it all
  - Best method depends on problem, metric, and train set size
  - Picking best model yields much better performance than any one method
- Ensemble Selection:
  - Carefully selected ensemble of models yields further improvements
  - Can optimize ensemble to any performance metric
- Performance Metrics:
  - 9 metrics span 2-4 Dim subspace
  - Ordering Metrics Tightly Cluster: AUC ~ APR ~ BEP
  - RMS ~ MXE, but RMS more centrally located. RMS is king!

Thank You!