Predicting Code Efficiency Automatically on the Google Code Jam Dataset

Daniel Graf and Jijiang Yan
Department of Computer Science & Engineering, University of Washington

CSE515 Statistical Methods in Computer Science – Spring 2013

Background

- Largest online coding competition with over 10,000 participants every year
- Set of Tasks, each having:
  - small input: brute force
  - large input: clever algorithm
- Popularity of automatic grading systems for programming classes in large scale (coursera)

Motivation

- Organizers can not review all submissions in a timely manner
- Interest in detecting outliers, new types of solutions and attacks
- Compiling and running submitted code requires a lot of resources and a trusted environment
- Deeper insights from automatic classification of code efficiency

Goals

- Collect and prepare all the submissions from several tasks
- Extract static features of the submitted code sources
- Train and evaluate classifiers
  - Naive Bayes and logistic regression for single tasks
  - Multi-task logistic regression classifier for new tasks

Data Collection and Feature Extraction

- collected correct programs in C++ for 6 different tasks
  - 18606 submissions with 1.275 million lines of code
- extracted 35 features using only static string search, like: counts of keywords (defines, includes, loops, conditionals, STL-classes), lengths of comments, depth of branching, biggest integer constant
- converted to binary features by comparing with quantiles (i.e. median only or 3-quantiles) and others

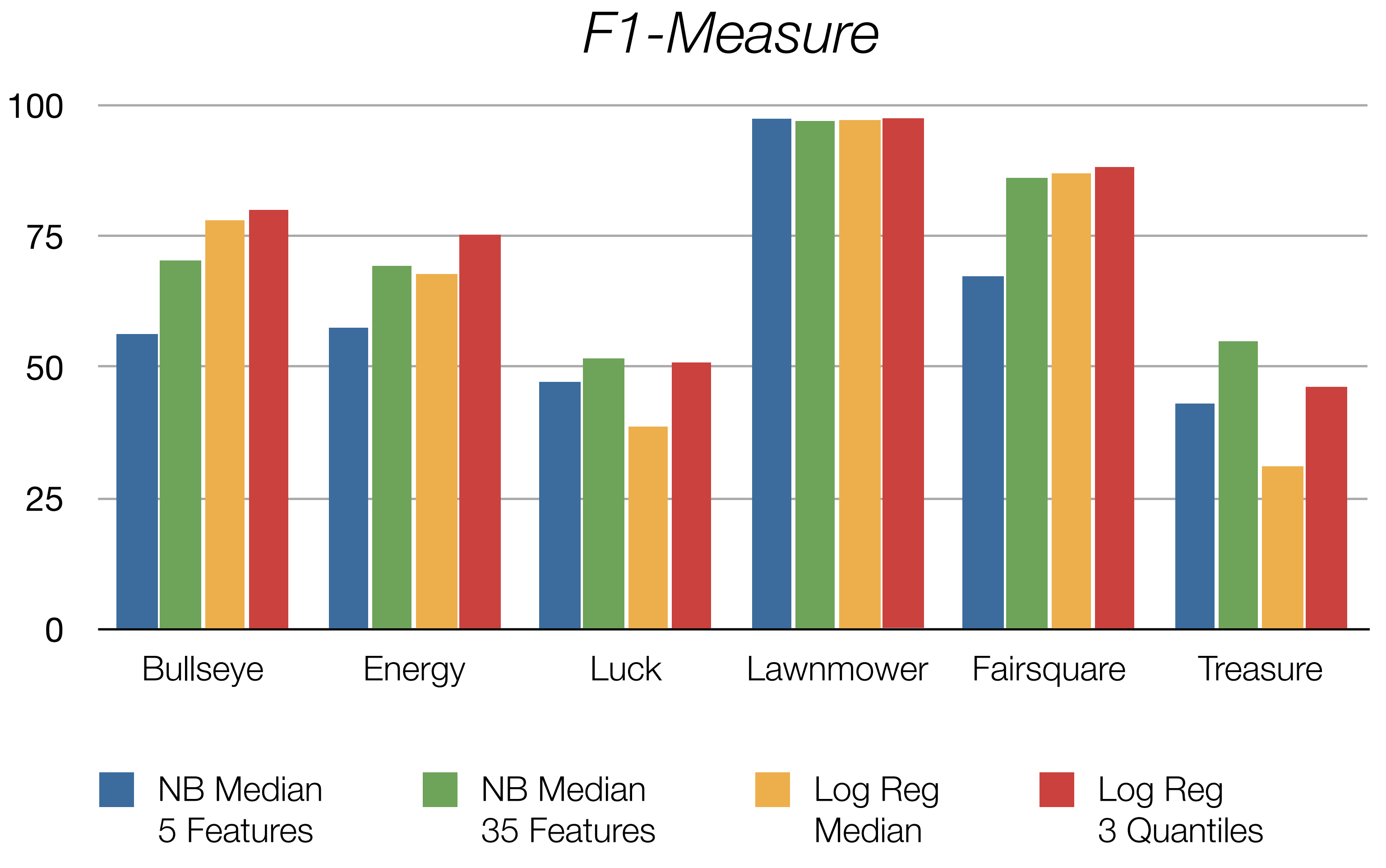
Multi-Task Logistic Regression

by À. Lapedriza,, D. Masip, and J. Vitrià, 2007
Pattern Recognition and Image Analysis, Springer

- train models for multiple tasks $T_1, \ldots, T_M$ as weight matrix $W = (\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(M)})$ simultaneously
- regularize feature weights by penalizing deviations from the mean weight vector $\bar{\mathbf{w}}$ resulting in the loss $G(W) = L(D,W) + \frac{1}{\sigma_1^2}\|\bar{\mathbf{w}}\|_2 + \frac{1}{\sigma_2^2}\sum_{i=1}^{M}\|\mathbf{w}^{(i)} - \bar{\mathbf{w}}\|_2$ where $L(D,W)$ is the negated log-likelihood estimator

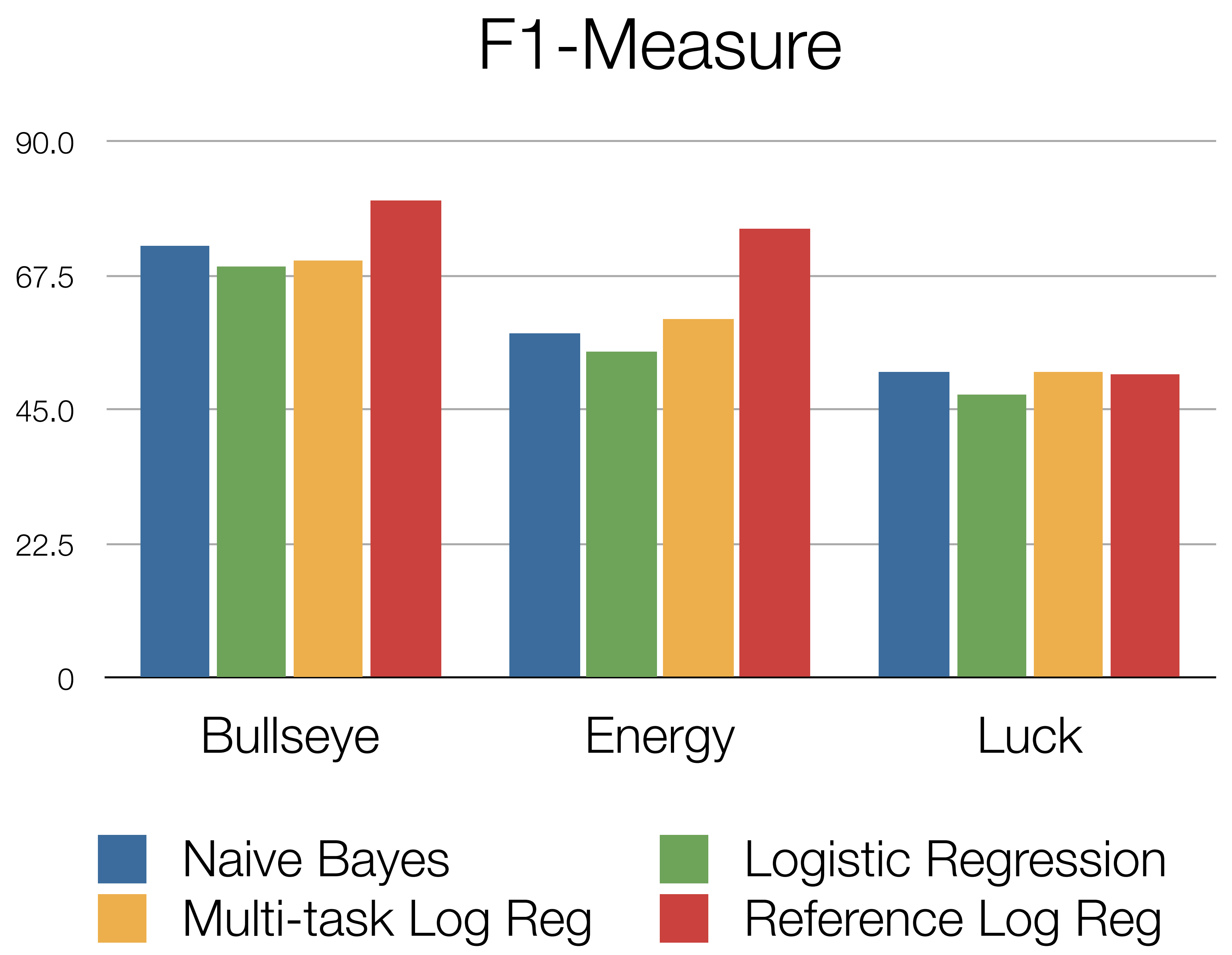
Single-Task Classifier Results

- Naive Bayes gives best results using median-threshold
- Logistic regression outperforms Naive Bayes
- Logistic regression regularization parameter has only for the 2 more difficult tasks a significant effect



Multi-Task Classifier Results

- RESULTS TODO


F1-Measure
Bullseye
Energy
Luck
Naive Bayes
Logistic Regression
Multi-task Log Reg
Reference Log Reg