EECS-349 Final Project
Phoenix: An Article Recommendation Engine

Avy Faingezicht
Leon Sasson
2014-03-20

Task

Through our work, we tried to predict whether a user would like an online article based on previous reading behavior. In today's information age, we are all constantly barraged with content, from short tweets, to Buzzfeed-style listicles, longform journalism and other time consuming material. The rapid increase of the rate at which this content is generated makes it impossible for anyone to actually consume all the content published, and we as users must rely on recommendations to select how to spend our time when reading online. Specifically, we wanted to predict whether a user would mark an article as a favorite or not.

Data

We used data retrieved from Pocket's API. Pocket is an online application that allows its users to save online content in order to access it later. By using their developer API, we were able to extract our personal archives, lists of read articles and related data, such as favorited articles. We retrieved data from one user's archive and built a training set containing 910 examples, while our test set has 102 observations, a 90:10 ratio, with about 11% of observations in either set tagged as favorite articles by the user.

Method

Scikit-learn's CountVectorizer, a widely used frequency table extractor, was tweaked to our needs in order to extract useful tokens and produce a more compact dataset from the retrieved articles. The result was a sparse matrix of the token counts. The derived feature set comprised on content information such as author, domain, article length (word count), and stemmed word frequencies, yielding around 50,000 unique features.

By using scikit-learn's modeling tools we were able to benchmark the performance on the task with multiple classifying algorithms studied in class. We attempted to use perceptron, logistic regression, Naive Bayes, nearest neighbors, and SVM classifiers, among others. We decided against utilizing a decision tree model, given that the latest version of scikit-learn does not yet support sparsity when running the models its tree library.

For every algorithm, we performed dimensionality reduction to avoid overfitting by means of chi squared statistic testing. For each model, we selected the k-percentile features with the highest χ^2 significance. Our models were run for multiple values of k, and the various models' performance are shown in the figures 1 and 2, showing the change in performance as we increased the number of features included.

The models' validity was ensured by using 10-fold CV. There was a negative effect on the overall values of precision, recall and F1 scores, as the cross validated models avoid overfitting and theoretically should be more effective at predicting observations that deviate from the training data set. The performance of the cross validated models can be seen on figures 3 and 4.

Results

From the chart in figure 2, we can see that logistic regression models with low values of k have the highest possible F1 score, with ridge regression classifier and multinomial Naive Bayes as close runner-ups. Once we cross-validated our results, we found that multinomial Naive Bayes with 30% k was the most effective model. With our best cross-validated Naive Bayes model in the 25-30% F1-score range, and at around 25% precision, our learners are clearly performing at a significant level above the baseline (11%). We can see how logistic regression was suffering from overfitting, and while it is still the model with best precision scores (Figure 3), the low sensitivity in its recall hurts its F1 score (Figure 5). As we increase the number of features used, we can see how our best model — Naive Bayes — improves its precision, while the recall decreases (Figure 6), so there is a clear tradeoff in the selection of k.

Given the nature of our dataset, we suffered from bias towards false negatives. We believe that active learning methods could solve this issue, as our learners are currently being overwhelmed by uninformative examples. By interactively querying the user for his opinion on observations in which our learner obtained ambiguous results, we could provide our model with the most efficient observations for classification. This remains an unsolved problem, and we encourage future machine learners to tackle it in future iterations of this course.

Figures

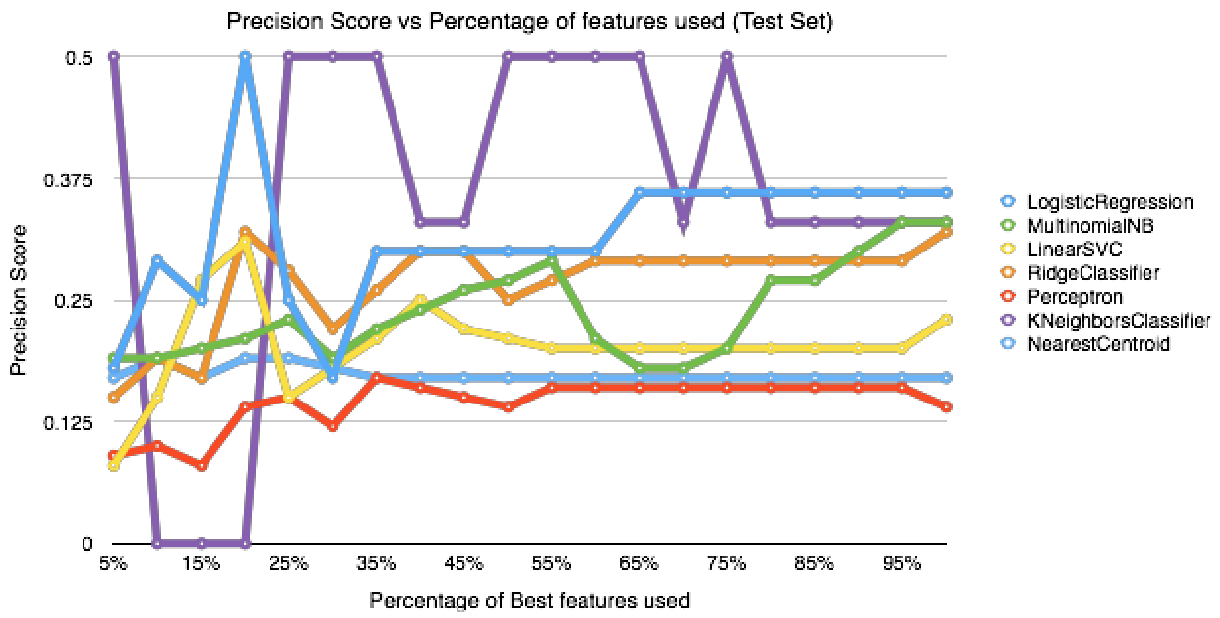

Figure 1. Precision score vs. the K-Percentile best features, without CV

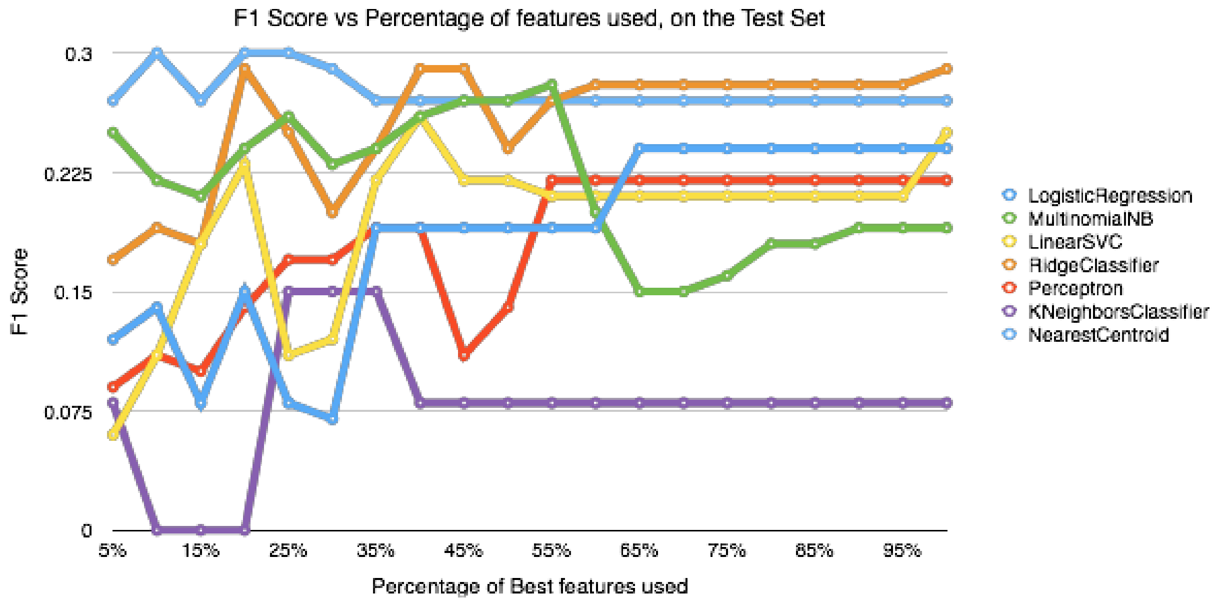

Figure 2. F1 score vs. the K-Percentile best features, without CV

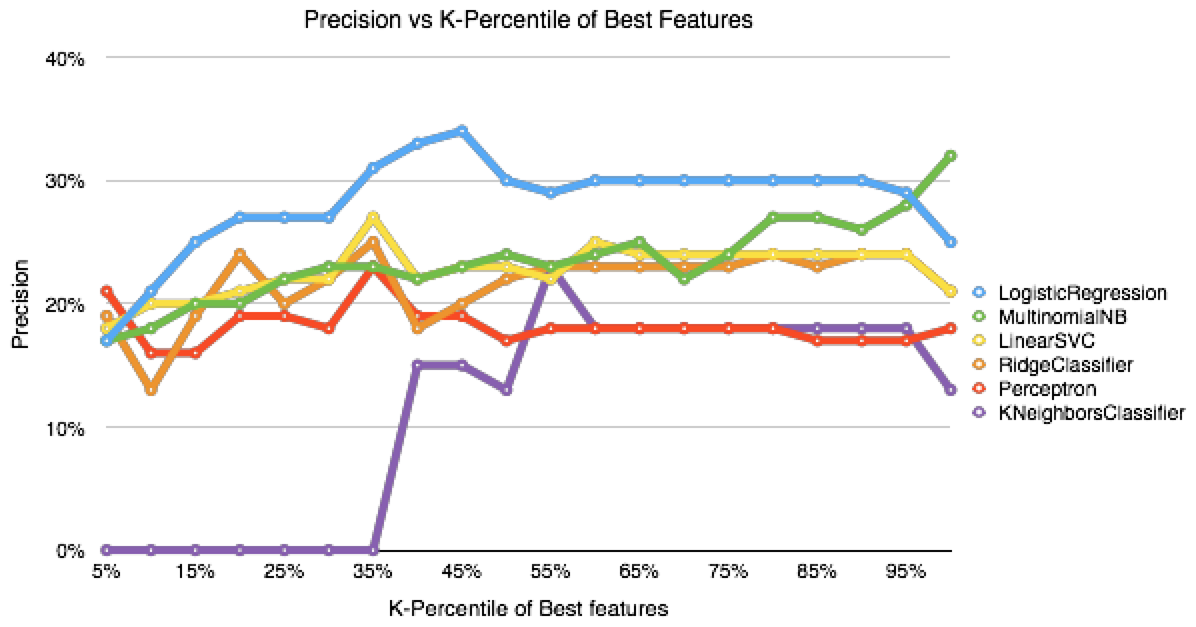

Figure 3. Precision score vs. the K-Percentile best features, using 10-fold CV

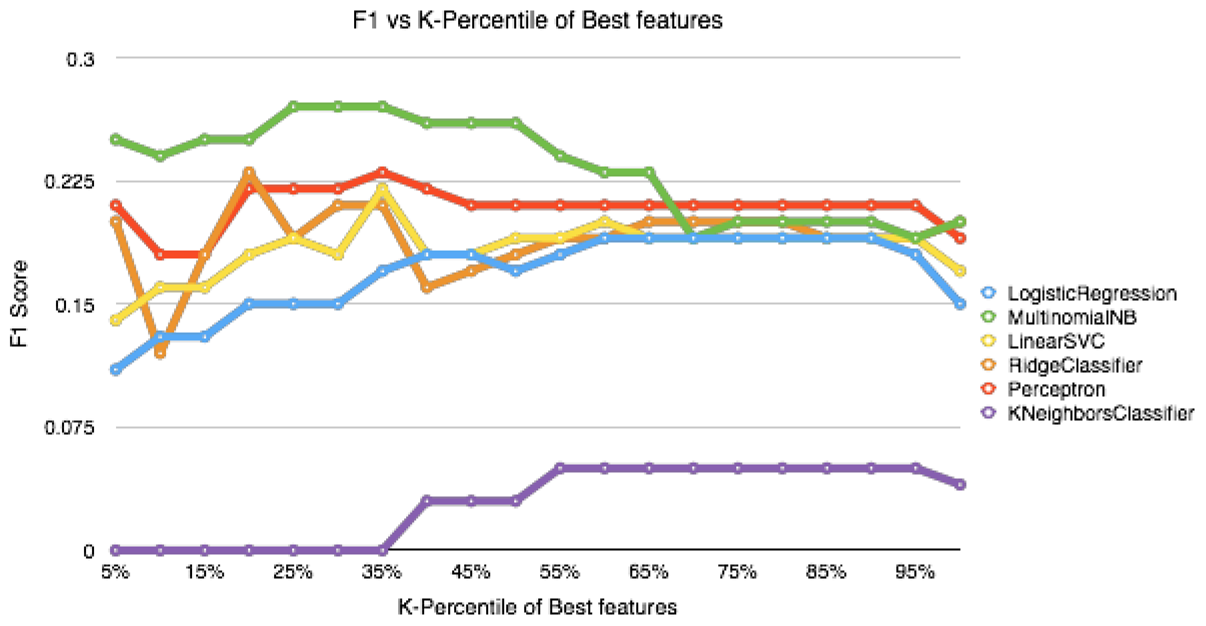

Figure 4. F1 score vs. the K-Percentile best features, using 10-fold CV

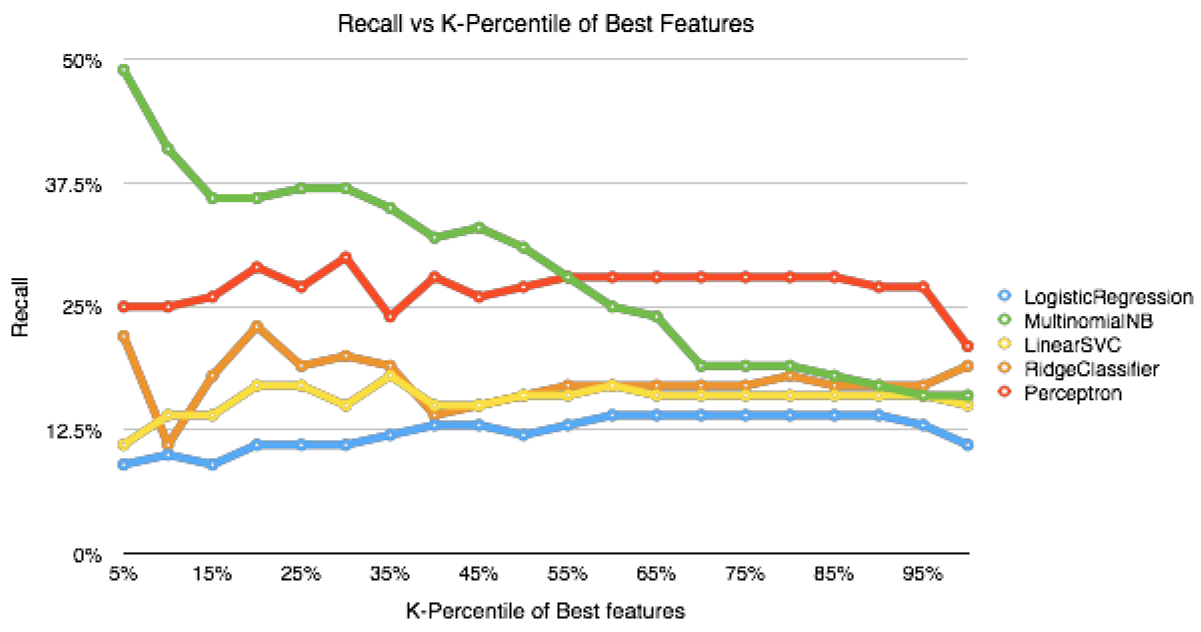

Figure 5. Recall score vs. the K-Percentile best features, using 10-fold CV

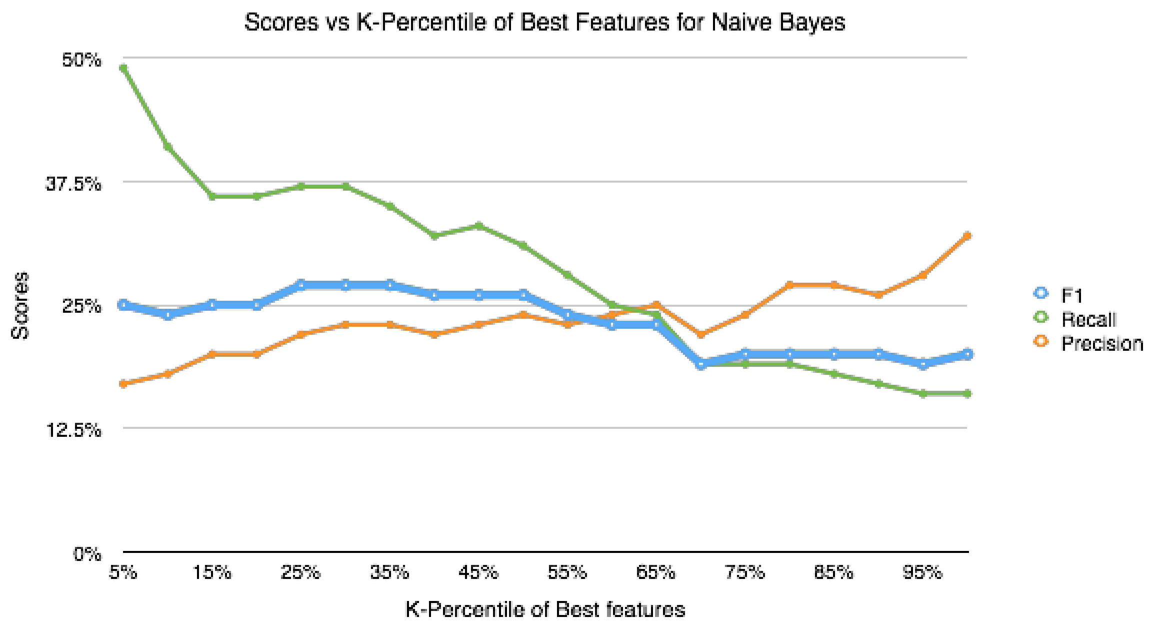

Figure 6. Performance Scores vs. the K-Percentile best features for multinomial Naive Bayes, using 10-fold CV.