COS 424 Homework 1: Classifying Negative vs. Positive Sentiments Using Bag-of-Words Representation

Lili Cai
Neuroscience
lxcai@princeton.edu

Christopher Criscitiello
Math
cc26@princeton.edu

Abstract

We use bag-of-words representation to classify text into positive or negative sentiment. We use variants of Naive Bayes, logistic regression, discriminant analysis, support vector machines, k-nearest neighbors and tree-based classifiers. We find that linear classifiers (those with linear decision boundary) perform best in cross-validation accuracy, test accuracy and AUC. This is either because the data is best classified by a linear decision boundary, or because the training sample size was too small for the nonlinear classifiers to detect nonlinearities in the decision boundary.

1 Introduction and Data Processing

The task is to classify a series of reviews across different categories as either positive or negative sentiment. For example,"the food was tasty" is a positive sentiment, while "I was bored during the movie" is negative sentiment.

We use bag-of-words to represent each review (code provided by COS 424 staff). Words that occur more than $f_{thres} = 5$ times in the whole training set $\mathfrak{T} = \{T_1, ..., T_n\}$ are used as features of the training data. Thus, each training example T_i is reduced to a vector x_i of d feature words (dimension d), where $x_{i,j}$ is the frequency of each word $j \in \{1, 2, ..., d\}$ in T_i. Every T_i also has an associated label $y_i \in \{0, 1\}$, where zero means the example has negative sentiment, and one means positive sentiment. We then feed $\{(x_i, y_i)\}_{i=1}^n$ into each classifier. We had $n = 2400$ training examples and 600 test examples, both with equal positive and negative sentiment, and $d = 541$ feature words.

2 Data Exploration

We anticipate any classification methods to be on par with human ability. Thus, we first approximate how good humans are at classifying bag-of-words representations. During human 'training,' we spent 1-2 min skimming through training examples to get a sense of which bag of words lead to either positive or negative sentiment. During 'testing', we randomly chose 20 bag-of-words to score. Human classifiers 1 and 2 achieved a mean accuracy of 77.5 %, with 17.5 % false negative and 5 % false positive rate.

3 Classification Methods

We implement the following classifiers using python's scikit learn toolbox [3] [1]:

1. Naive Bayes classifier (NB): using Gaussian implementation

2. Logistic reg. with ℓ_2 penalty (LR2): using a variant of stochastic average gradient descent
3. Logistic reg. with $\ell_!$ penalty (LR1): using a variant of stochastic average gradient descent
4. Linear discriminant analysis (LDA): using singular value decomposition (SVD)
5. Support vector machine with linear kernel (SVML1): using parameter $C = 1$
6. Support vector machine with radial basis function kernel (SVMR): $C = 1, \gamma = 0.0208$
7. Support vector machine with linear kernel (SVML2): using parameter $C = 100$
8. Nu Support vector machine with linear kernel (NuSVM): using $\nu = 0.5$
9. K-nearest neighbors (KNN): using three nearest neighbors
10. AdaBoost (AB): using a decision tree base learner and 200 estimators
11. Gradient Boost (GB): using 100 estimators and binomial deviance loss function
12. Decision tree (DT): using Gini impurity scores
13. Random forest (RF): using Gini impurity scores and 100 estimators

3.1 Evaluation Methods

We use a variety of evaluation metrics on training and test data to determine which classifier performs the best. Although these are reviews about non-life threatening things, we reason there's a preference for decreasing false positives. It would be best to know which things to avoid, but if something is falsely classified as negative, there are many positive alternatives to chose from.

On **training data**, we use 10-fold cross validation. This separates the training data into 10 subdivisions, using 9 for training and 1 for testing. Each classifier is trained and tested ten times using different combinations of the 9 to 1 division. Smaller variance across the ten validation results corresponds to a more stable classifer, and we are likely to trust its results when used on the actual test dataset. Moreover, these variances allow one to perform significance testing and inference.

On the **test data**, we use a confusion matrix to look at basic accuracy and error rate, which can be further broken down into false positive (fp), true positive (tp), false negative (fn) and true negative (tn). We next use these measures to evaluate more complex metrics and compare relatively between models:

- False positive rate: fp / (fp + tn)
- True positive rate: tp / (tp + fn) (also known as sensitivity, recall or hit rate)
- Receiver Operating Characteristics (ROC) curve: plots the true positive rate versus the false positive rate over the threshold based on model threshold values. A positive $45°$ line suggests the classifier is random, any line that falls above it with steeper slope is better than random, and a line that falls below it is worse than random.
- Area Under the Curve (AUC): This is the area under the ROC curve, and ranges from 0 to 1. Higher AUC means classifier is more accurate relative to another, and AUC under .5 for binary classifiers suggests it is worse than random. Two ROC curves might be different but have the same AUC.

Both test accuracy and AUC are convenient metrics since they are only a single number, as opposed to an entire curve (e.g., ROC). AUC is more informative if the test set has unequal positive vs. negative labels. Suppose test set is 80% positive labels. If a classifier outputs 1 with probability $p = 1$, regardless of input x, then it already has 80% test accuracy. However, the AUC for this classifier will be around 0.5, making it a better metric.

3.2 Spotlight Classifier: SVM

The support vector classifier (SVC) is a supervised learning method for classifying data with binary labels. The SVC trains on labeled data $\{(x_i, y_i)\}_{i=1}^n, x_i \in \mathbb{R}^d$, where, without loss of generality, we assume $y_i \in \{-1, 1\}$. The SVC finds the hyperplane $w^T x + b = 0, \|w\| = 1$ which "best divides" the data. A prediction $\hat{y}$ on new datum x is determined by which side of the hyperplane x is on: $\hat{y} = \text{sign}(w^T x + b)$. (If we want the labels to be in $\{0, 1\}$ we can simply do the transformation $\hat{y} = (1 + \text{sign}(w^T x + b))/2$.)

Since there are many ways to divide linearly separable data, the SVC chooses the "best" hyperplane by maximizing the minimum distance M between the separating hyperplane and the data points. The margin is the region of points whose distance from the separating hyperplane is less than M (see Fig. 1a).

Since $y_i \in \{-1, 1\}$ and $\|w\| = 1$, the signed distance between the hyperplane $w^T x + b = 0$ and data point x_i is $y_i(w^T x_i + b)$ (negative distance indicates the data point is on the wrong side of the hyperplane) [2]. Thus, we seek to solve:

$$\max_{w,b} M, \text{subj. to constraints: } \|w\| = 1, y_i(w^T x_i + b) \geq M, \text{ for } i = 1, ..., n. \tag{1}$$

Complex data is usually not linearly separable, so SVC seeks to maximize M while allowing some data points to be on the wrong side of the margin. To formalize, we use a standard optimization trick of inserting nonnegative slack variables $s_1, ..., s_n$. The goal is now to solve

$$\max_{w,b} M, \text{subj. to constraints: } \|w\| = 1, y_i(w^T x_i + b) \geq M(1 - s_i), s_i \geq 0, \text{ for } i = 1, ..., n, \sum_{i=1}^{n} s_i \leq K. \tag{2}$$

If $s_i = 0$, then x_i is outside of the margin and on the correct side of the hyperplane. If $0 < s_i < 1$, x_i is inside the margin and on the correct side of the hyperplane. If $s_i > 1$, x_i is on the wrong side of the hyperplane. Since $\sum_{i=1}^{n} s_i \leq K$, the number of incorrectly classified data points is controlled by the parameter K of the model (see Fig. 1b).

The data points for which $s_i > 0$ are called the support vectors. The support vectors are those data points which are inside the margin or on the wrong side of the boundary. If we omit data points x_i which are not support vectors ($s_i = 0$), the solution of the SVC will not change. Thus, SVC cross validation is faster because it depends only on the number of support vectors, which is generally less than the size of the training set [2].

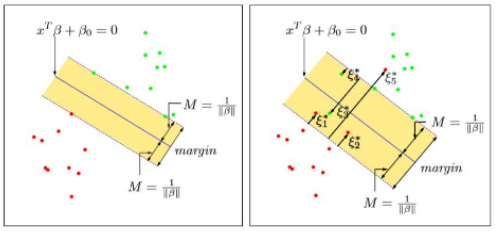

Figure 1. a) Left: Linearly separable data with a hyperplane that maximizes M. b) Right: Optimal hyperplane for nonlinearly separable data. Note some data points are inside the margin, and some are on the wrong side of the hyperplane. Here, the slack variables are xi. Figure copied from Hastie et al. 2001.

Using support vectors, the optimization problem in (2) becomes:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} s_i, \tag{3}$$
$$\text{subject to the constraints: } s_i \geq 0, y_i(w^T x_i + b) \geq 1 - s_i, \text{ for } i = 1, ..., n,$$

where the parameter $C > 0$ replaces the parameter K (i.e., K is function of C and $K \to 0$ as $C \to \infty$). The parameter C penalizes error. When C is small, the margin is larger and contains data points (support vectors) which are far away from the hyperplane. When C is large, the margin is smaller, and the support vectors are only those data points which are either close to the hyperplane or classified incorrectly. When $C = \infty$, we minimize (3) by setting $\sum_{i=1}^{n} s_i = 0$, i.e., there are no support vectors, and (3) becomes the problem for linearly separable data (1).

As with regression, we can extend the feature space by choosing $D > d$ basis functions $h_1, ..., h_D$, where $h_k : \mathbb{R}^d \to \mathbb{R}$. The new classifier input features are $\{h(x_i) = (h_1(x_i), ..., h_D(x_i))\}_{i=1}^{n}$, maintaining the labels y_i. The SVC fits an optimal hyperplane in the extended feature space, which translates to a nonlinear decision boundary in the original feature space. The numerical solution to optimizing this hyperplane uses h only through inner products $\langle h(x), h(x')\rangle$. Thus, it suffices to describe h with a kernel K and set $\langle h(x), h(x_i)\rangle = K(x, x_i)$ (this is called the "kernel property"). A common kernel is the radial basis function (rbf) $K(x, x') = e^{-\gamma\|x-x'\|^2}$. Small γ creates a smooth decision boundary in the original feature space, and permits data points to be classified incorrectly. Large γ allows the decision boundary to correctly classify almost all training data points, resulting in overfitting.

We implemented our SVMs with the svc package of scikit learn [3] and used cross-validation to pseudo-optimize C and γ. Since $C = 1$ linear kernel performed better than rbf or $C = 100$, we didn't try other additional parameters.

The main source for this section is Chapter 12 of [2].

4 Results

	Cross-Val Accuracy (%)	Cross-Val Stdev (%)	Test Accuracy (%)	Test FPR (%)	Test TPR (%)	AUC	Training Accuracy (%)
NB	70.3	2.13	70.5	21	66	0.8	77.9
LR2	77.8	2.34	80.8	23	86	0.88	86.7
LR1	78.2	2.65	81.8	22	87	0.89	85.8
LDA	75.2	3.41	78.0	25	81	0.85	87.4
SVML1	77.8	3.01	78.3	25	82	0.86	87.6
SVML2	72.7	3.91	75.5	26	78	0.83	91.3
NuSVM	77.5	2.82	79.5	24	84	0.87	87.9
SVMR	59.4	4.46	58.2	45	92	0.81	77.9
KNN	67.8	4.73	66.2	38	74	0.74	82.5
AB	70.0	4.86	71.5	35	91	0.78	71.9
GB	74.1	4.22	76.3	31	91	0.85	79.2
RF	70.5	4.03	71.5	35	91	0.83	72.8
DT	69.4	3.69	74.3	27	75	0.77	97.8

Figure 2. Except for AUC, all metrics are percents (%)

Logistic regression (LR2, LR1) yields the highest cross validation accuracy, test accuracy and AUC, and lowest cross-validation variance. **LDA and SVM** with linear kernel (SVML1, NuSVM) also perform very well. (See Fig.2, 3 and Appendix Supplemental Fig. 2) Except for SVML2, these are the only classifiers which find linear decision boundaries. It seems that the data is best divided by a linear decision boundary. This might be because the data is inherently linear, or this might be because the training-set size to dimension ratio is quite small ($2400/541 \approx 4$). If the training set were much larger, the nonlinear classifiers (e.g., SVMR, AB, GB, RF) might have detected non-linearities in the decision boundary should they exist.

Why does logistic regression with ℓ_1 penalty (LR1) perform slightly better than logistic regression with ℓ_2 penalty (LR2)? We suspect this is because the ℓ_1 penalty (also known as the LASSO) favors sparsity [2]. Indeed, our data is very sparse: most of the components in the d-tuple are zero because only a few words appear in each review.

Tree-based ensemble classifiers AdaBoost (AB), Gradient Boost (GB) and Random Forest (RF) did not improve cross validation or test accuracy of the decision tree classifier. However, they did improve the AUC of the decision tree, and they greatly reduced the overfitting of the decision tree. For example, for AB and RF the difference between test accuracy and training accuracy is about 2%; for DT this difference is about 23%.

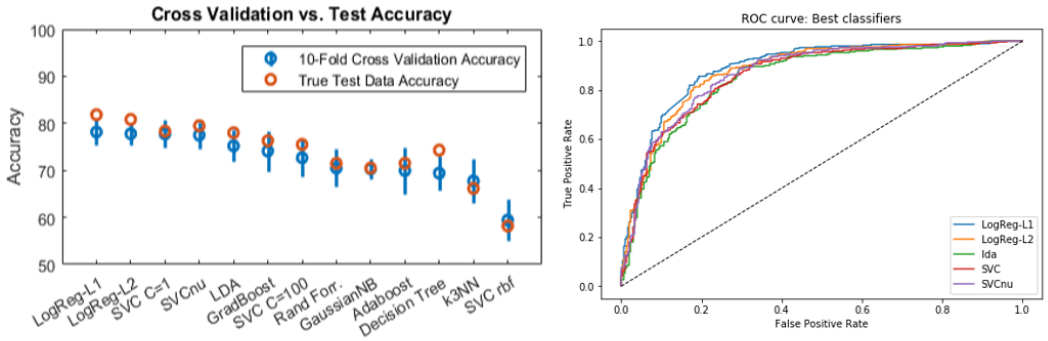

Figure 3. (a) Left: Cross validation and true test accuracy of 5 best classifiers. Error bars are standard deviation (b) Right: ROC curves of the 5 best classifiers on test data, correlating with 5 best on the training and test accuracy.

Linear discriminant analysis (LDA) makes strong assumptions about the data. Namely that the class densities $f_y(x) = Pr(X = x|Y = y)$ for $y = 0, 1$ are Gaussians in x with common covariance matrices [2]. These assumptions are not satisfied for this data set, yet LDA still produces good

performance. This is in agreement with Hastie, et al.'s observation that LDA usually gives similar performance as logistic regression [2] (p. 128).

Naive Bayes (NB) assumes the features in each class are independent. This is not true for our data, because we expect certain combinations of words to appear more frequently than others. This is why the n-gram model is so useful for natural language processing. Nonetheless, Naive Bayes performs just as well as more complex models like the decision tree classifier (DT) and AdaBoost (AB). This might be because our training set size is quite small compared to the dimension. If we had much more training data (e.g., about 1 million examples), we hypothesize AdaBoost would significantly outperform Naive Bayes.

For **training accuracy**, the decision tree classifier (DT) overfit the data (Fig. 3a). DT had a training accuracy of 97.8%, and a test accuracy of only 74.3%. On the other hand, logistic regression, LDA and linear SVMs had a training accuracy of 86% to 88%, and a test accuracy of 78% to 80%. Similarly, SVM with large C (SVML2) also overfit the data, which we expected from our discussion of SVMs.

For **test accuracy**, we can plot each classifier's performance using a **confusion matrix** (see Appendix Supp Fig. 1). Interestingly, both high performing linear regression and SVMs tend to have higher false positives than false negatives. However, overall error is sufficiently distributed across false positives and false negatives. The Naive Bayes classifier most resembles human classifier based on false positive and false negatives, but the true positive and true negative rates are switched.

4.1 Feature Selection

For logistic regression (LR2, LR1) we performed recursive feature elimination to determine the ten most significant features and words (see Fig. 4a). The coefficient sign indicates direction of labeling (+ vs. -). Interestingly, the majority of the most significant words have positive sentiment. (implemented as feature_selection.RFE in scikit learn).

LR2		LR1	
Word	**Coefficient**	**Word**	**Coefficient**
bad	-2.57	awesom	2.70
delici	2.18	bad	-3.00
excel	2.58	beauti	2.78
fantast	1.98	delici	3.22
great	2.69	excel	3.19
love	2.53	fantast	2.96
nice	2.20	great	2.90
poor	-2.24	love	2.96
terribl	-1.96	poor	-2.90
worst	-2.24	worst	-3.01

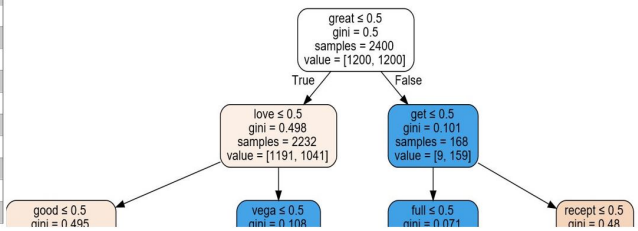

Figure 4. (a) Left: Ten most significant words with corresponding coefficients in logistic reg. (b) Right: Top of the decision tree produced by DT, tree is cut off on bottom on purpose.

The decision tree classifier (DT) produces a decision tree which we can interpret. We can identify the most significant words with those that are closest to the root; however, we should be skeptical of this identification because decision trees are nonrobust [2]. Fig. 4b shows the top of the decision tree produced.

5 Discussion and Conclusion

Linear regression, linear SVM and LDA had the best performance classifying positive vs. negative sentiments, with an accuracy of around 77 % – which matched human classification. This relatively low accuracy could be because not enough information was retained in the bag-of-words representation with a threshold of 5 words. For example, sentences that had "not great" would retain "great" but not "not." Future classifiers could use more sophisticated information representation. Examples include: (1) weight negation words (2) use n-grams to examine n consecutive words, (3) bag of concepts [4]. It would also be interesting to use unsupervised learning methods (e.g., k-means clustering, spectral clustering, or diffusion maps) on this data set. Perhaps some combination of supervised and unsupervised learning procedures would produce a better model.

References

[1] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2 edition, 2017.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] Magnus Sahlgren and Rickard Coster. Using bag-of-concepts to improve the performance of support vector machines in text categorization. *Coling*, ,:,, 2004.

6 Resources used for Coding

[1] Fraj, Mohtadi Ben. "In Depth: Parameter Tuning for Gradient Boosting - All Things AI. Medium.com, Medium, 24 Dec. 2017, `medium.com/all-things-ai/in-depth-parameter-tuning-for-gradient-boosting-3363992e9bae`.

[2] "Graphviz - Graph Visualization Software." Graphviz - Graph Visualization Software, `graphviz.gitlab.io/`.

[3] PyGAM Documentation, `pygam.readthedocs.io/en/latest/`.

[4] "Why Is Pydot Unable to Find GraphViz's Executables in Windows 8?" Stack Overflow, `stackoverflow.com/questions/18438997/why-is-pydot-unable-to-find-graphvizs-executables-in-windows-8`.

[5] "SyntaxError: Non-Default Argument Follows Default Argument." Stack Overflow, `stackoverflow.com/questions/24719368/syntaxerror-non-default-argument-follows-default-argument/39942121`.

[6] Two-Class AdaBoost. Scikit-Learn 0.19.2 Documentation, `scikit-learn.org`.

[7] 1.10. Decision Trees. Scikit-Learn 0.19.2 Documentation, `scikit-learn.org`

[8] sklearn.feature_selection.RFE. Scikit-Learn 0.19.2 Documentation, `scikit-learn.org`

[9] sklearn.svm.NuSVC. Scikit-Learn 0.19.2 Documentation, `scikit-learn.org`

[10] sklearn.svm.LinearSVC. Scikit-Learn 0.19.2 Documentation, `scikit-learn.org`

[11] Logistic Regression 3-class Classifier. Scikit-Learn 0.19.2 Documentation, `scikit-learn.org`

[12] 1.6. Nearest Neighbors. Scikit-Learn 0.19.2 Documentation, `scikit-learn.org`

[13] sklearn.tree.DecisionTreeClassifier. Scikit-Learn 0.19.2 Documentation, `scikit-learn.org`

[14] Receiver Operating Characteristic (ROC) with cross validation. Scikit-Learn 0.19.2 Documentation, `scikit-learn.org`

[15] Linear and Quadratic Discriminant Analysis with covariance ellipsoid. Scikit-Learn 0.19.2 Documentation, `scikit-learn.org`

[16] sklearn.svm.SVC. Scikit-Learn 0.19.2 Documentation, `scikit-learn.org`

7 Appendix

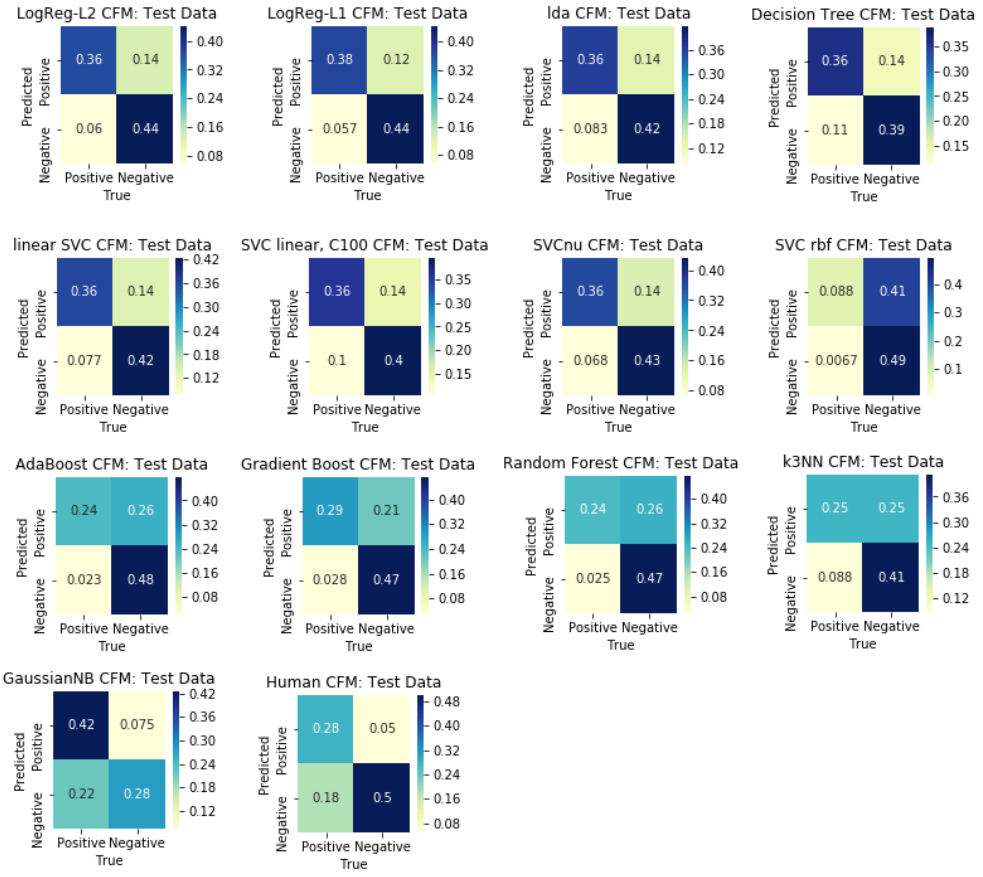

Figure 1: *

Supp Figure 1. Confusion matrices for the classifiers on the test data.

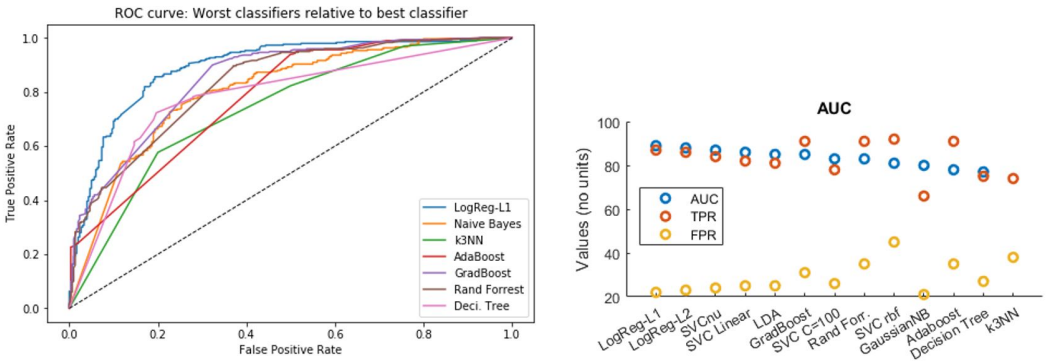

Figure 2: *

Supp Figure 2. (a) Left: ROC curves of worse classifiers, as well as the ROC curve of the best classifier. (b) Right: In terms of AUC, the linear classifiers perform best. This is consistent with both test and cross validation accuracy.