CS 175, Project in Artificial Intelligence

Lecture 7: Evaluation Methods

Padhraic Smyth
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

Announcements

- Project Proposals
  - Due Friday 6pm
  - Detailed information on the class Website
  - If you are not on a team: please visit my office after lecture

- Office Hours:
  - Eric: Thursday 1 to 3
  - Me: Today after lecture, Friday 9:30 to 11:30

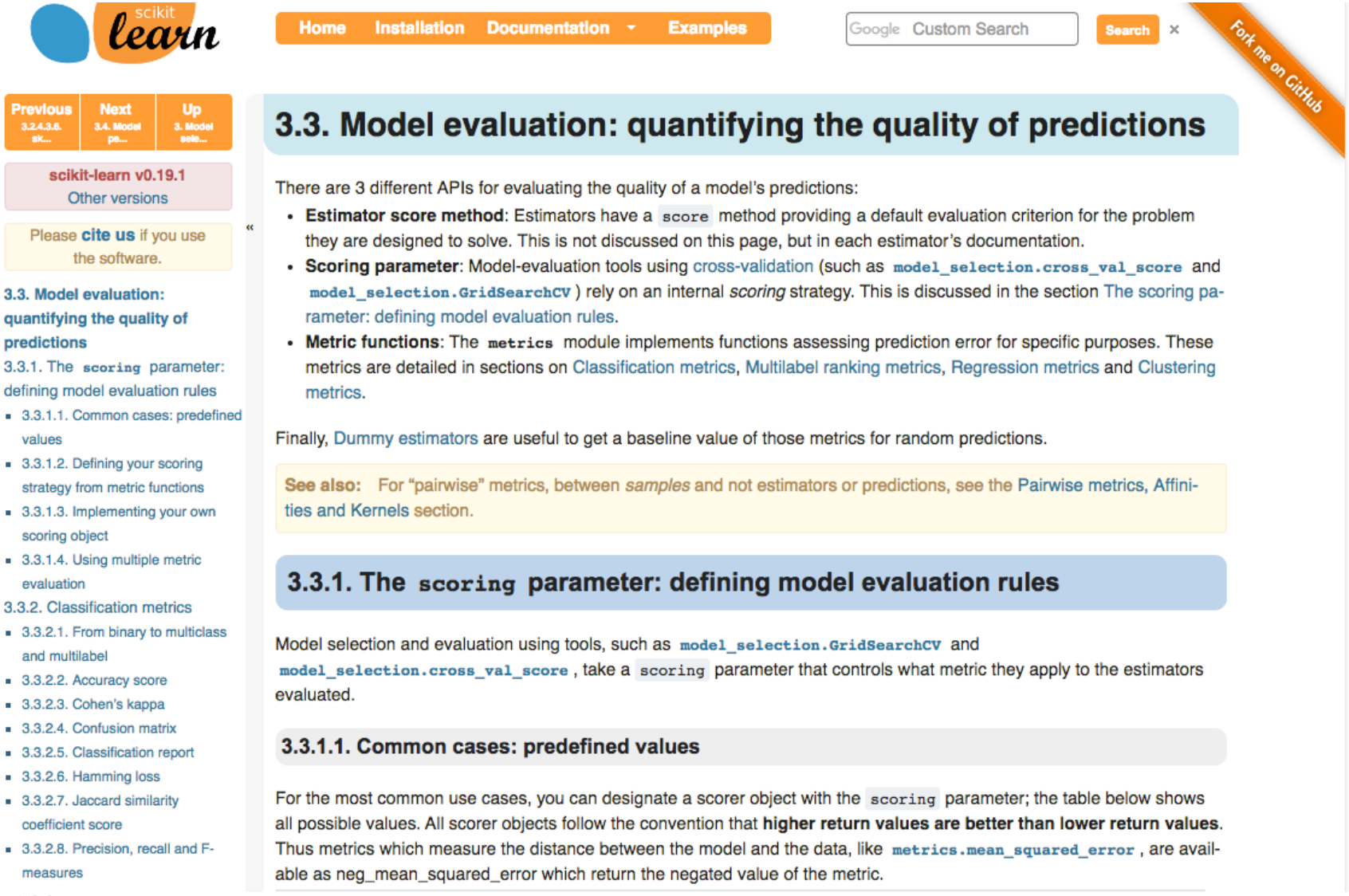
- Today's Lecture:
  - Evaluation methods

Weekly Schedule (Updated)

Week	Monday	Wednesday
Jan 8	Lecture: Introduction and course outline	Lecture: Basic concepts in text analysis
Jan 15	No class (university holiday)	Lecture: Text classification, part 1 **Assignment 1 due, 5pm**
Jan 22	Text classification, part 2	Lecture: Neural networks for text **Assignment 2 due, 5pm**
Jan 29	Lecture: Project Proposal Discussion	Lecture: Evaluation Methods **Project proposal due, Friday 6pm**
Feb 5	Office hours (no lecture)	Lecture: Unsupervised learning algorithms
Feb 12	Office hours (no lecture)	Office hours (no lecture)
Feb 19	No class (university holiday)	Lecture: Discussion of progress reports **Progress report due, Friday 6pm**
Feb 26	Office hours (no lecture)	Office hours (no lecture)
Mar 5	Office hours (no lecture)	Lecture: Discussion of final reports
Mar 12	Project Presentations (in class) Upload slides by 4pm	Project Presentations (in class) Upload slides by 4pm
Mar 19	**Final project reports due (day/time TBD)**	

Evaluating Classification Algorithms

See http://scikit-learn.org/stable/modules/model_evaluation.html for lots of useful information on evaluation



scikit learn

Home Installation Documentation Examples

Google Custom Search Search

Fork me on GitHub

Previous 3.2.4.3.6. sk... Next 3.4. Model pe... Up 3. Model sele...

scikit-learn v0.19.1
Other versions

Please cite us if you use the software.

3.3. Model evaluation: quantifying the quality of predictions
3.3.1. The scoring parameter: defining model evaluation rules
- 3.3.1.1. Common cases: predefined values
- 3.3.1.2. Defining your scoring strategy from metric functions
- 3.3.1.3. Implementing your own scoring object
- 3.3.1.4. Using multiple metric evaluation

3.3.2. Classification metrics
- 3.3.2.1. From binary to multiclass and multilabel
- 3.3.2.2. Accuracy score
- 3.3.2.3. Cohen's kappa
- 3.3.2.4. Confusion matrix
- 3.3.2.5. Classification report
- 3.3.2.6. Hamming loss
- 3.3.2.7. Jaccard similarity coefficient score
- 3.3.2.8. Precision, recall and F-measures

3.3. Model evaluation: quantifying the quality of predictions

There are 3 different APIs for evaluating the quality of a model's predictions:

- **Estimator score method**: Estimators have a `score` method providing a default evaluation criterion for the problem they are designed to solve. This is not discussed on this page, but in each estimator's documentation.
- **Scoring parameter**: Model-evaluation tools using cross-validation (such as `model_selection.cross_val_score` and `model_selection.GridSearchCV`) rely on an internal *scoring* strategy. This is discussed in the section The scoring parameter: defining model evaluation rules.
- **Metric functions**: The `metrics` module implements functions assessing prediction error for specific purposes. These metrics are detailed in sections on Classification metrics, Multilabel ranking metrics, Regression metrics and Clustering metrics.

Finally, Dummy estimators are useful to get a baseline value of those metrics for random predictions.

See also: For "pairwise" metrics, between *samples* and not estimators or predictions, see the Pairwise metrics, Affinities and Kernels section.

3.3.1. The `scoring` parameter: defining model evaluation rules

Model selection and evaluation using tools, such as `model_selection.GridSearchCV` and `model_selection.cross_val_score`, take a `scoring` parameter that controls what metric they apply to the estimators evaluated.

3.3.1.1. Common cases: predefined values

For the most common use cases, you can designate a scorer object with the `scoring` parameter; the table below shows all possible values. All scorer objects follow the convention that **higher return values are better than lower return values**. Thus metrics which measure the distance between the model and the data, like `metrics.mean_squared_error`, are available as neg_mean_squared_error which return the negated value of the metric.

The Simple Majority Classifier

- Say we have K classes, and class k occurs with probability p(k), k = 1,…K
- Let k* be the most likely class, i.e., highest value of p(k)
- The "majority classifier" always predicts class k*
  - What is the accuracy of this classifier?

    Accuracy = p(k*)

  - This is a "dumb" classifier, but its always good to report this number when reporting classification results
  - On some problems, if p(k*) = 0.99, it may be hard to beat the majority classifier
  - Or if there are more than 2 classes, p(k*) may be quite small (e.g., 0.3) and an accuracy of 0.5 (say) tells you that you are picking up predictive signal

Illustration of 5-fold Cross-Validation

Document	Feature Vector	Class Label
d1	x	1
d2	x	2
d3	x	1
d4	x	2
d5	x	2
d6	x	1
d7	x	1
d8	x	2
d9	x	1
d10	x	2

Illustration of 5-fold Cross-Validation

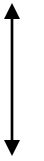
Document	Feature Vector	Class Label
d1	x	1
d2	x	2
d3	x	1
d4	x	2
d5	x	2
d6	x	1
d7	x	1
d8	x	2
d9	x	1
d10	x	2

Fold 1: Training Data

Fold 1: Test Data

Illustration of 5-fold Cross-Validation

Document	Feature Vector	Class Label
d1	x	1
d2	x	2
d3	x	1
d4	x	2
d5	x	2
d6	x	1
d7	x	1
d8	x	2
d9	x	1
d10	x	2



Fold 2: Test Data

Illustration of 5-fold Cross-Validation

Document	Feature Vector	Class Label
d1	x	1
d2	x	2
d3	x	1
d4	x	2
d5	x	2
d6	x	1
d7	x	1
d8	x	2
d9	x	1
d10	x	2

Fold 5:

Fold 5: Training Data

The classifier is trained 5 times, once on each training set, accuracy computed on the test set, and the average accuracy across the 5 test sets is computed. This is the cross-validated estimate of the classifier accuracy

Cross-Validation for Development, with a Test Set

Alternatively, use cross-validation on a development set and use a separate test data set for more realistic assessment of error erate

Document	Feature Vector	Class Label
d1	x	1
d2	x	2
d3	x	1
d4	x	2
d5	x	2
d6	x	1
d7	x	1
d8	x	2
d9	x	1
d10	x	2

Development Set

Document	Feature Vector	Class Label
d1	x	1
d2	x	2
d3	x	1
d4	x	2
d5	x	2
d6	x	1
d7	x	1
d8	x	2
d9	x	1
d10	x	2

Hold Out Test Set

Comparing Multiple Algorithms with Cross-Validation

- Important that they are evaluated on exactly the same data

- Correct approach:
  - In cross-validation, outer loop is over the train-test folds, and inner loop is over the different methods

- Incorrect approach:
  - Outer loop is over the different methods, and inner loop performs CV (with different random train/test folds for each method)

- The 2nd approach is incorrect because it introduced additional (unnecessary) variability or noise into the process of measuring the difference between the algorithms or methods

Cross-Validation for Development, with a Test Set

Alternatively, with large data sets we can use a single data set for validation and model development

Training Set

Document	Feature Vector	Class Label
d1	x	1
d2	x	2
d3	x	1
d4	x	2
d5	x	2
d6	x	1
d7	x	1
d8	x	2
d9	x	1
d10	x	2

Validation Set

Document	Feature Vector	Class Label
d1	x	1
d2	x	2
d3	x	1
d4	x	2
d5	x	2
d6	x	1
d7	x	1
d8	x	2
d9	x	1
d10	x	2

Test Set

Document	Feature Vector	Class Label
d1	x	1
d2	x	2
d3	x	1
d4	x	2
d5	x	2
d6	x	1
d7	x	1
d8	x	2
d9	x	1
d10	x	2

Model Development

Statistical Significance

- Say we compare Algorithm A versus B on 20 data sets/trials
  - A does better 12 times and B does better 8 times
  - What should we conclude?

- We can use statistical hypothesis testing to help us decide
  - Null hypothesis: the two methods are equally accurate (and any variation we see is just random noise)

- Can use different types of hypothesis tests, eg,.
  - Sign test: just looks at number of times A or B "won" on a validation set
    - Is the different statistically significant from random chance (binomial model)
  - Wilcoxon test: looks at magnitude of differences on validation sets
  - t-test on the difference in accuracies across the 20 tests: mean is zero or not

- Can use same idea for user studies: how often does the user select A over B

Going Beyond Accuracy

- To get insight into what a classifier is doing, its often useful to dig deeper beyond accuracy numbers

- Examples:
  - Look at random examples of the documents it got right and got wrong and see if there are any obvious patterns (e.g., negation in sentiment analysis)
  - If you have a classifier that is producing probabilities, you could look at the test examples with highest predicted probability of being in class 1 but that are actually class 0 (and vice-versa)
  - With 2 or more classifiers, you could see if they tend to make errors on the same examples (look at 2 x 2 tables for 2 classifiers)

Confusion Matrices

TRUE LABELS

PREDICTED LABELS	Class 1	Class 2	Class 3
Class 1	24	15	0
Class 2	5	5	0
Class 3	0	1	50

Confusion Matrices

TRUE LABELS

PREDICTED LABELS	Class 1	Class 2	Class 3
Class 1	24	15	0
Class 2	5	5	0
Class 3	0	1	50

Accuracy = (24 + 5 + 50)/100 = 79%

Confusion Matrices

TRUE LABELS

PREDICTED LABELS	Class 1	Class 2	Class 3
Class 1	24	15	0
Class 2	5	5	0
Class 3	0	1	50

Confusion Matrices

TRUE LABELS

PREDICTED LABELS

	Class 1	Class 2	Class 3	
Class 1	24	15	0	39
Class 2	5	5	0	10
Class 3	0	1	50	51
	29	21	50	

Predicted Number per Class

Actual Number per Class

Ranking-based Metrics

- These include precision, recall, F1, ROC/AUC, and more

- Can provide additional useful information about the performance of a prediction model

- Precision Example:
  - We want an algorithm to return the 10 most likely documents for class k
  - We rank the test documents by P(class = k | features) produced by the model
  - Select the test documents with the 10 highest probabilities
  - Say 8 of the 10 have true labels that are class k
  - Precision = 8/10

  - An algorithm might have high precision even if overall its classification accuracy is not that high

Using Class Probabilities

- Can look at how confident a classifier is
  - E.g., which test examples for each class is it most confident on?
- Can help with analyzing what errors a classifier is making
  - E.g., find all test examples that were incorrectly classified and look at those examples that have the highest probability of the incorrect class
- Can compare classifiers
  - E.g., using scatter plots, as with the examples on the previous slides
- Can rank the test examples and analyze rankings

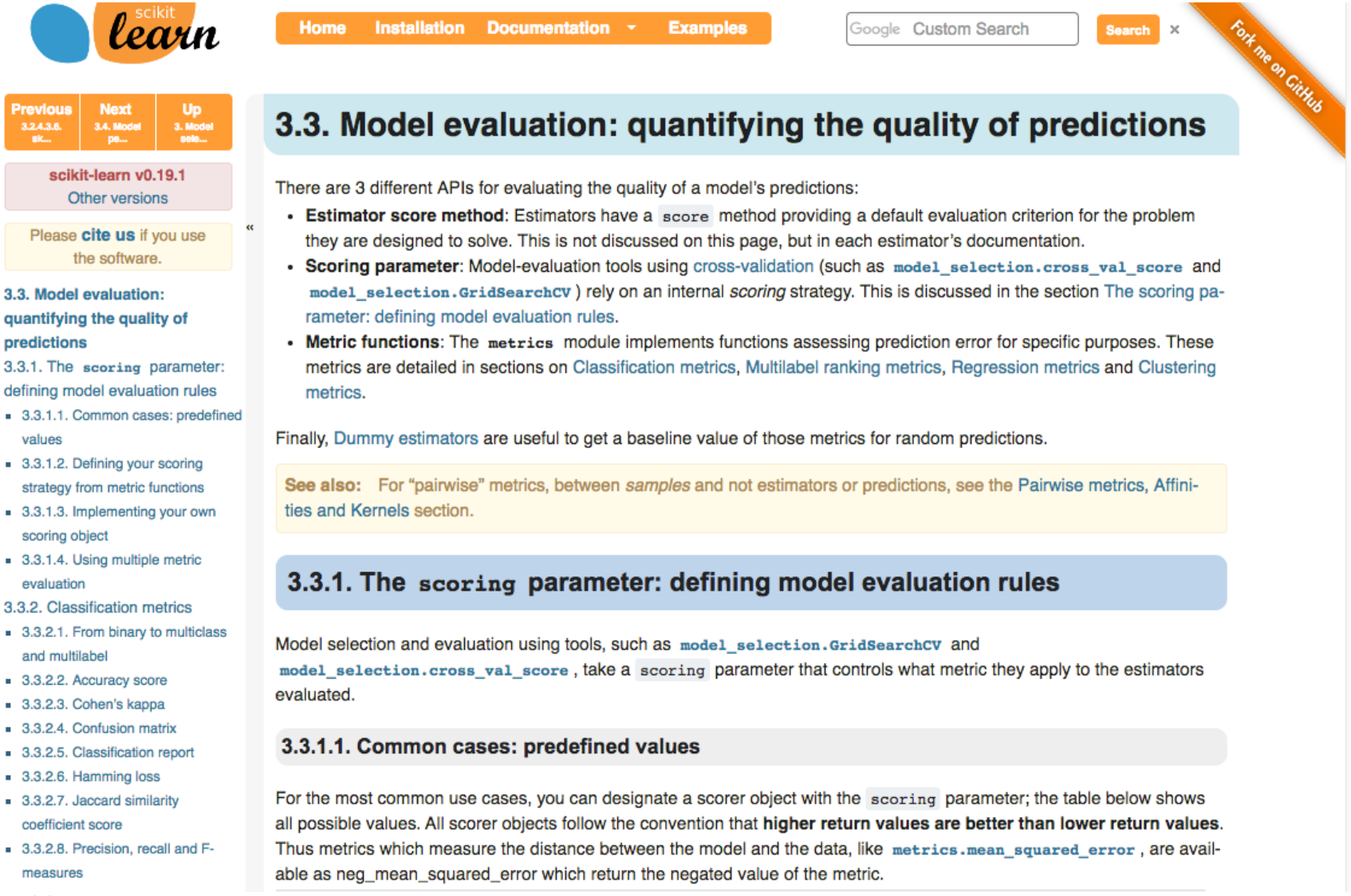
Advice: Its not always the about the algorithm…

- It is tempting in a predictive modeling project to spend a lot of time experimenting with the algorithms
  - E.g., different neural network architectures and learning heuristics

- But in practice which variables are used may be much more important than which algorithm

- So a useful strategy may be to rethink what the variables/attributes are. There might be another variable or two you can add to your problem that will increase classification accuracy by a large margin.

See http://scikit-learn.org/stable/modules/model_evaluation.html for lots of useful information on evaluation



scikit learn

Home Installation Documentation Examples

Google Custom Search Search

Fork me on GitHub

Previous 3.2.4.3.6. sk... Next 3.4. Model pe... Up 3. Model sele...

scikit-learn v0.19.1
Other versions

Please cite us if you use the software.

3.3. Model evaluation: quantifying the quality of predictions
3.3.1. The scoring parameter: defining model evaluation rules
- 3.3.1.1. Common cases: predefined values
- 3.3.1.2. Defining your scoring strategy from metric functions
- 3.3.1.3. Implementing your own scoring object
- 3.3.1.4. Using multiple metric evaluation

3.3.2. Classification metrics
- 3.3.2.1. From binary to multiclass and multilabel
- 3.3.2.2. Accuracy score
- 3.3.2.3. Cohen's kappa
- 3.3.2.4. Confusion matrix
- 3.3.2.5. Classification report
- 3.3.2.6. Hamming loss
- 3.3.2.7. Jaccard similarity coefficient score
- 3.3.2.8. Precision, recall and F-measures

3.3. Model evaluation: quantifying the quality of predictions

There are 3 different APIs for evaluating the quality of a model's predictions:

- **Estimator score method**: Estimators have a `score` method providing a default evaluation criterion for the problem they are designed to solve. This is not discussed on this page, but in each estimator's documentation.
- **Scoring parameter**: Model-evaluation tools using cross-validation (such as `model_selection.cross_val_score` and `model_selection.GridSearchCV`) rely on an internal *scoring* strategy. This is discussed in the section The scoring parameter: defining model evaluation rules.
- **Metric functions**: The `metrics` module implements functions assessing prediction error for specific purposes. These metrics are detailed in sections on Classification metrics, Multilabel ranking metrics, Regression metrics and Clustering metrics.

Finally, Dummy estimators are useful to get a baseline value of those metrics for random predictions.

See also: For "pairwise" metrics, between *samples* and not estimators or predictions, see the Pairwise metrics, Affinities and Kernels section.

3.3.1. The `scoring` parameter: defining model evaluation rules

Model selection and evaluation using tools, such as `model_selection.GridSearchCV` and `model_selection.cross_val_score`, take a `scoring` parameter that controls what metric they apply to the estimators evaluated.

3.3.1.1. Common cases: predefined values

For the most common use cases, you can designate a scorer object with the `scoring` parameter; the table below shows all possible values. All scorer objects follow the convention that **higher return values are better than lower return values**. Thus metrics which measure the distance between the model and the data, like `metrics.mean_squared_error`, are available as neg_mean_squared_error which return the negated value of the metric.

Announcements

- Project Proposals
  - Due Friday 6pm
  - Detailed information on the class Website
  - If you are not on a team: please visit my office after lecture

- Office Hours:
  - Eric: Thursday 1 to 3
  - Me: Today after lecture, Friday 9:30 to 11:30

Weekly Schedule (Updated)

Week	Monday	Wednesday
Jan 8	Lecture: Introduction and course outline	Lecture: Basic concepts in text analysis
Jan 15	No class (university holiday)	Lecture: Text classification, part 1 **Assignment 1 due, 5pm**
Jan 22	Text classification, part 2	Lecture: Neural networks for text **Assignment 2 due, 5pm**
Jan 29	Lecture: Project Proposal Discussion	Lecture: Evaluation Methods **Project proposal due, Friday 6pm**
Feb 5	Office hours (no lecture)	Lecture: Unsupervised learning algorithms
Feb 12	Office hours (no lecture)	Office hours (no lecture)
Feb 19	No class (university holiday)	Lecture: Discussion of progress reports **Progress report due, Friday 6pm**
Feb 26	Office hours (no lecture)	Office hours (no lecture)
Mar 5	Office hours (no lecture)	Lecture: Discussion of final reports
Mar 12	Project Presentations (in class) Upload slides by 4pm	Project Presentations (in class) Upload slides by 4pm
Mar 19	**Final project reports due (day/time TBD)**	

Additional (Optional) Slides

Class Probabilities produced by Classifiers

Class Probabilities

- Many classifiers produce class probabilities as outputs
  - E.g., logistic regression, neural networks, naïve Bayes
  - For each training or test example the classifier produces P(c | x) where x is the input vector
  - For K classes there are K class probabilities that sum to 1

Methods for Logistic Regression Classifier in scikit-learn

From http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Methods

`decision_function` (X)	Predict confidence scores for samples.
`densify` ()	Convert coefficient matrix to dense array format.
`fit` (X, y[, sample_weight])	Fit the model according to the given training data.
`fit_transform` (X[, y])	Fit to data, then transform it.
`get_params` ([deep])	Get parameters for this estimator.
`predict` (X)	Predict class labels for samples in X.
`predict_log_proba` (X)	Log of probability estimates.
`predict_proba` (X)	Probability estimates.
`score` (X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
`set_params` (**params)	Set the parameters of this estimator.
`sparsify` ()	Convert coefficient matrix to sparse format.
`transform` (*args, **kwargs)	DEPRECATED: Support to use estimators as feature selectors will be removed in version 0.19.

Class Probabilities in scikit-learn

From http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

predict_proba (*X*) [source]

Probability estimates.

The returned estimates for all classes are ordered by the label of classes.

For a multi_class problem, if multi_class is set to be "multinomial" the softmax function is used to find the predicted probability of each class. Else use a one-vs-rest approach, i.e calculate the probability of each class assuming it to be positive using the logistic function. and normalize these values across all the classes.

Parameters:	**X** : array-like, shape = [n_samples, n_features]
Returns:	**T** : array-like, shape = [n_samples, n_classes] Returns the probability of the sample for each class in the model, where classes are ordered as they are in `self.classes_`.

Setting up a 2-Class Problem (from 20 Newsgroups Data)

```
n_categories = 2
twenty_train, twenty_test = load_news_dataset(n_categories)

The names of the 2 most common labels in the data set are:
['rec.sport.hockey', 'soc.religion.christian']

Dimensions of X_train_counts are (1199, 21891)
Number of non-zero elements in X_train_counts: 157195
Percentage of non-zero elements in X_train_counts: 0.60
Average number of word tokens per document: 196.54
Average number of documents per word token: 10.76
```

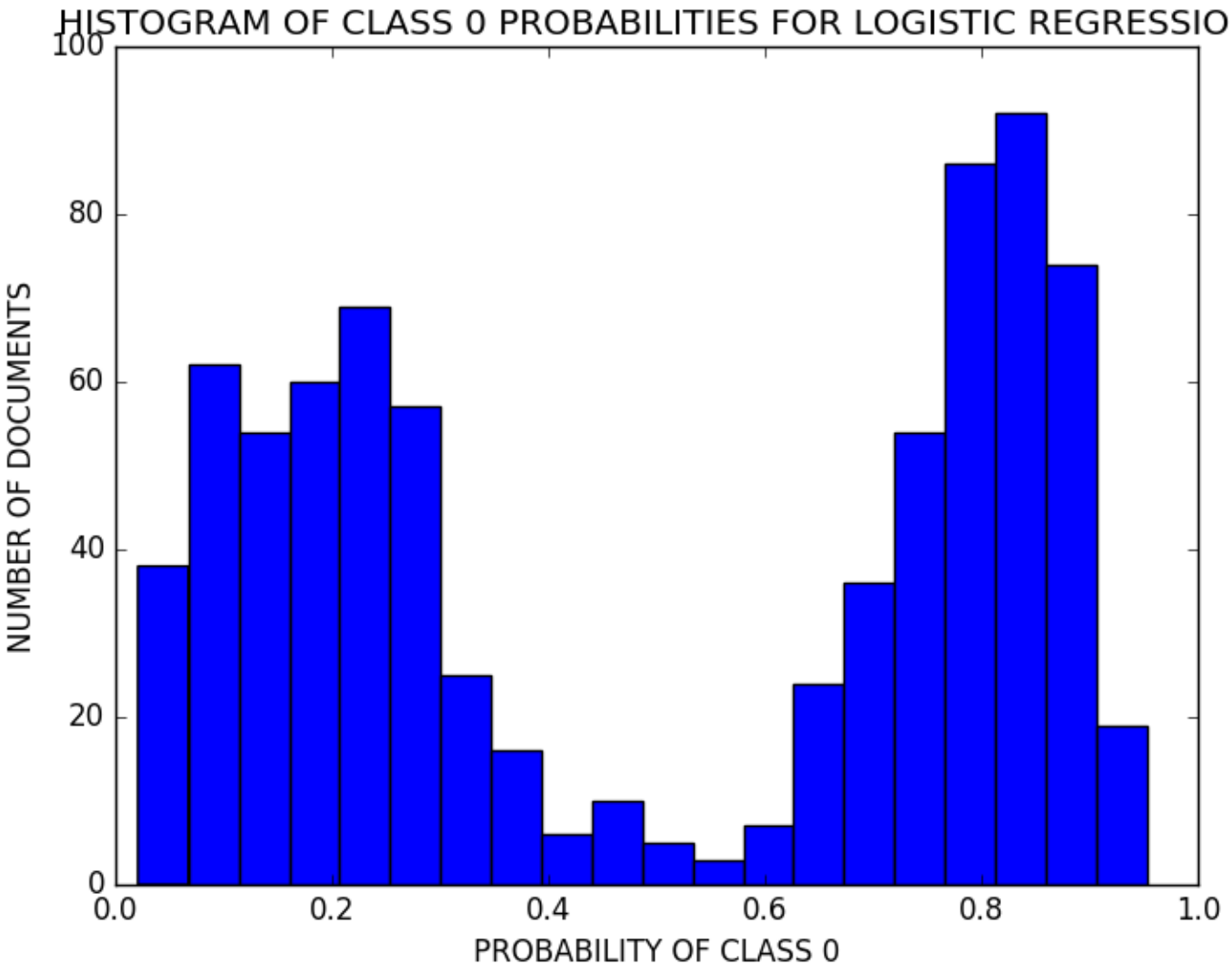
Class Probabilities from a Logistic Regression Model

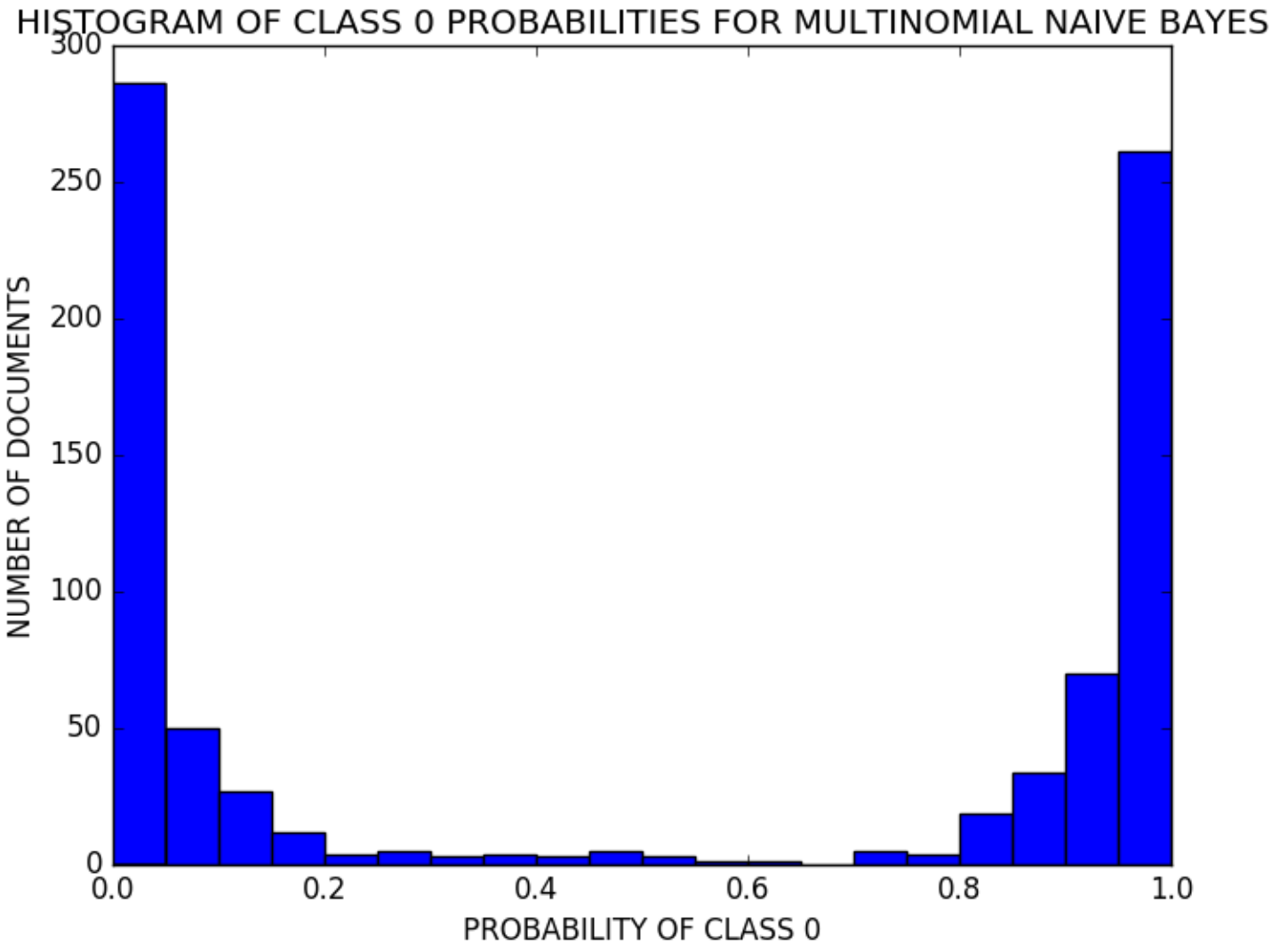
```
# fit a logistic regression model and generate the probabilities
clfLR = LogisticRegression().fit(X_train_tfidf, twenty_train.target)

# generate predictions on the test data
predicted_LR = clfLR.predict(X_test_tfidf);

# generate class probabilities on the test data
prob_test_LR = clfLR.predict_proba(X_test_tfidf);
```

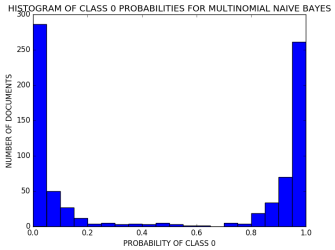
Class Probabilities from a Logistic Regression Model

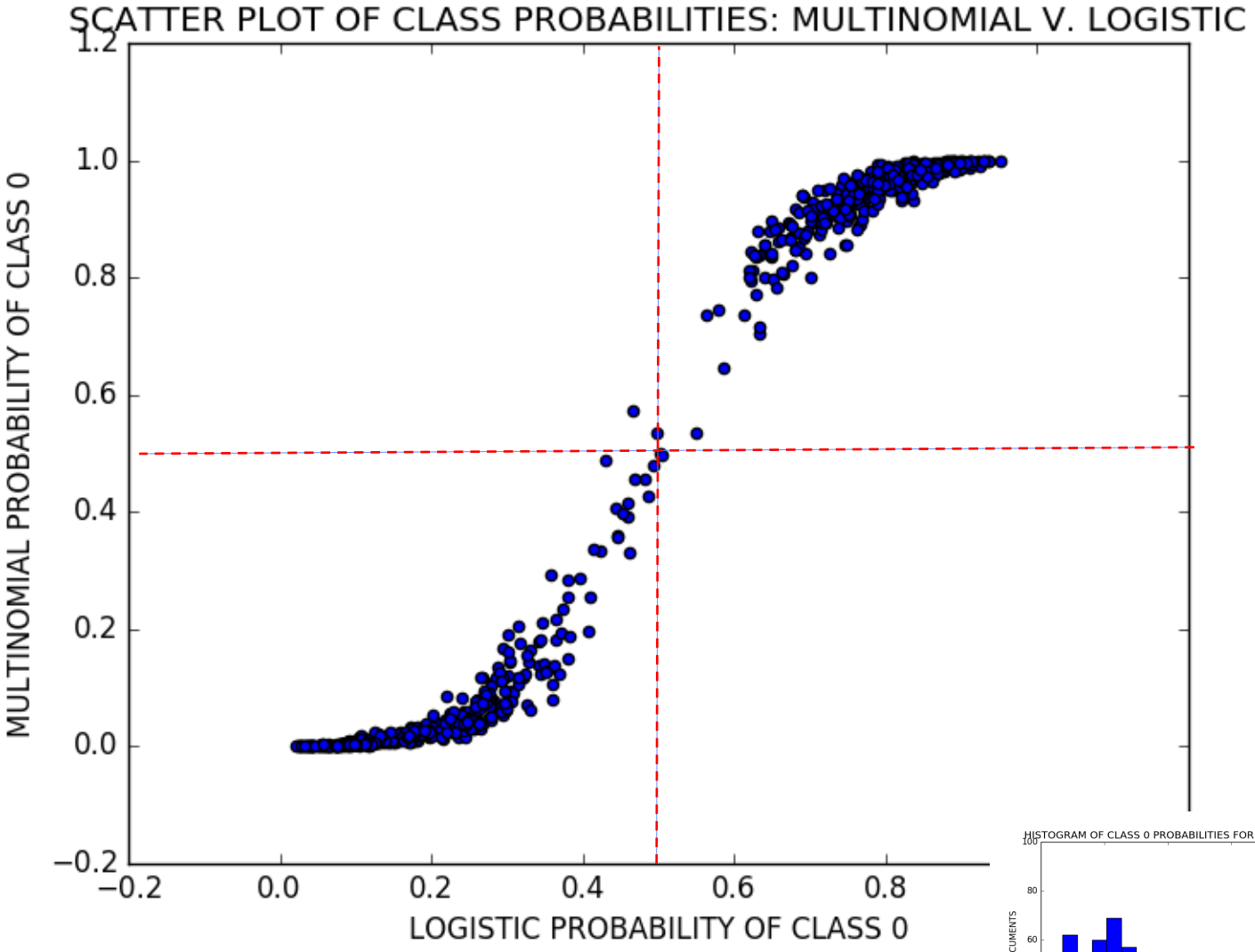
```
prob_test_LR

array([   [ 0.7041694 ,  0.2958306 ],
          [ 0.50150781,  0.49849219],
          [ 0.74678626,  0.25321374],
          ...,
          [ 0.45904319,  0.54095681],
          [ 0.09707758,  0.90292242],
          [ 0.86623216,  0.13376784]] )
```

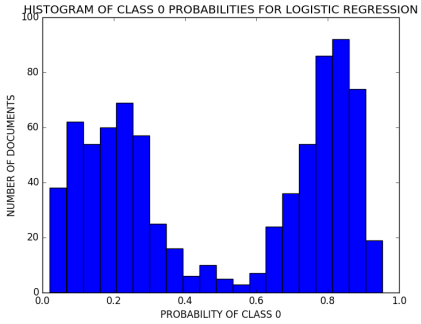
2 class probabilities that sum to 1 for every example in the test data array

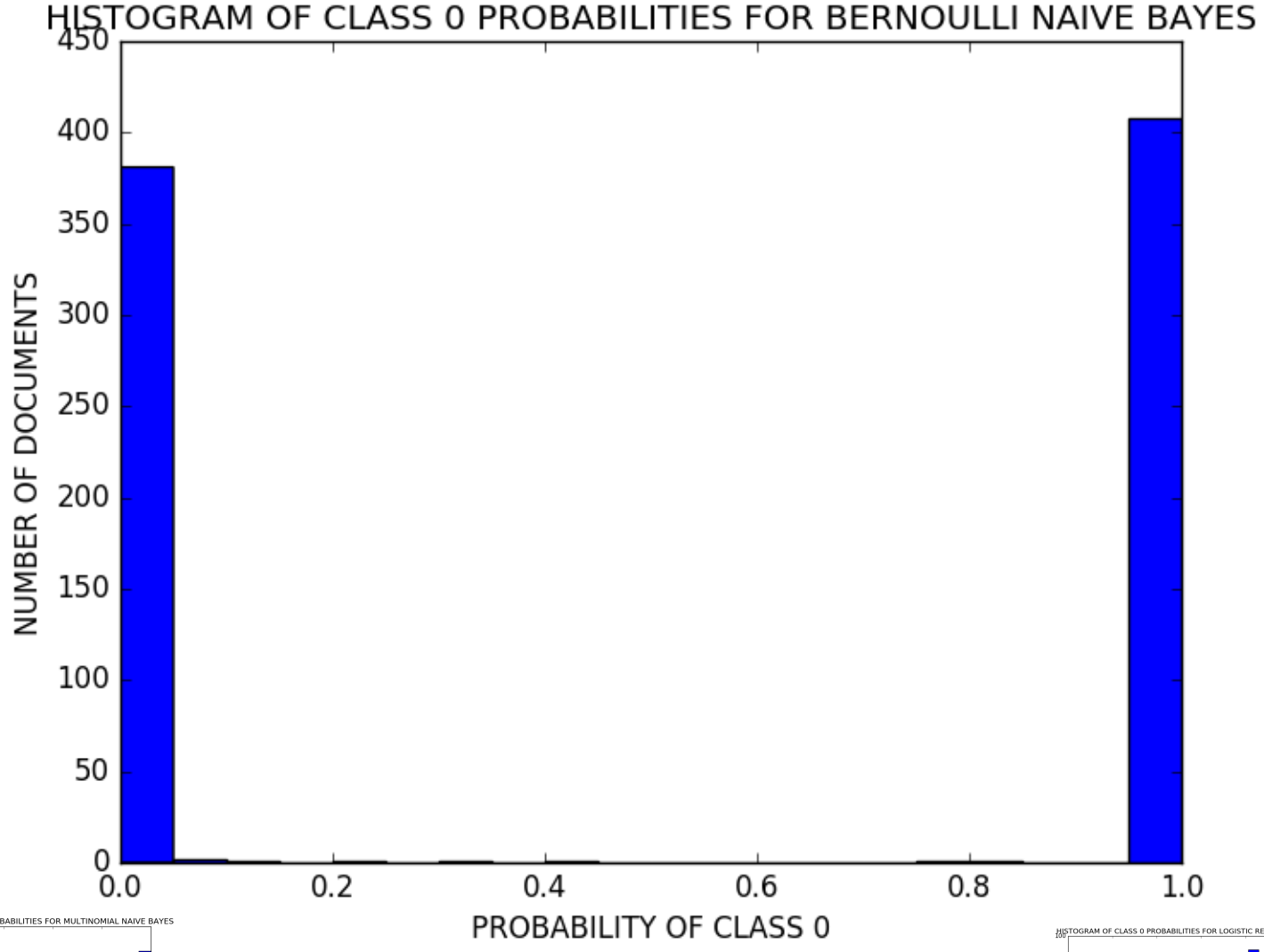

HISTOGRAM OF CLASS 0 PROBABILITIES FOR LOGISTIC REGRESSION
NUMBER OF DOCUMENTS
PROBABILITY OF CLASS 0
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

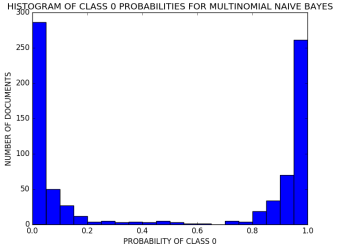

HISTOGRAM OF CLASS 0 PROBABILITIES FOR MULTINOMIAL NAIVE BAYES
NUMBER OF DOCUMENTS
PROBABILITY OF CLASS 0
0
50
100
150
200
250
300
0.0
0.2
0.4
0.6
0.8
1.0

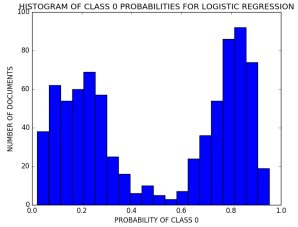

HISTOGRAM OF CLASS 0 PROBABILITIES FOR MULTINOMIAL NAIVE BAYES
NUMBER OF DOCUMENTS
PROBABILITY OF CLASS 0

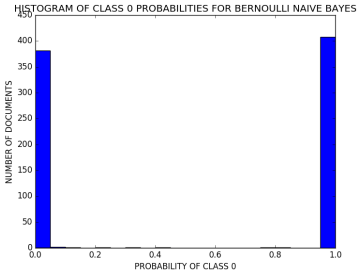

SCATTER PLOT OF CLASS PROBABILITIES: MULTINOMIAL V. LOGISTIC
MULTINOMIAL PROBABILITY OF CLASS 0
LOGISTIC PROBABILITY OF CLASS 0

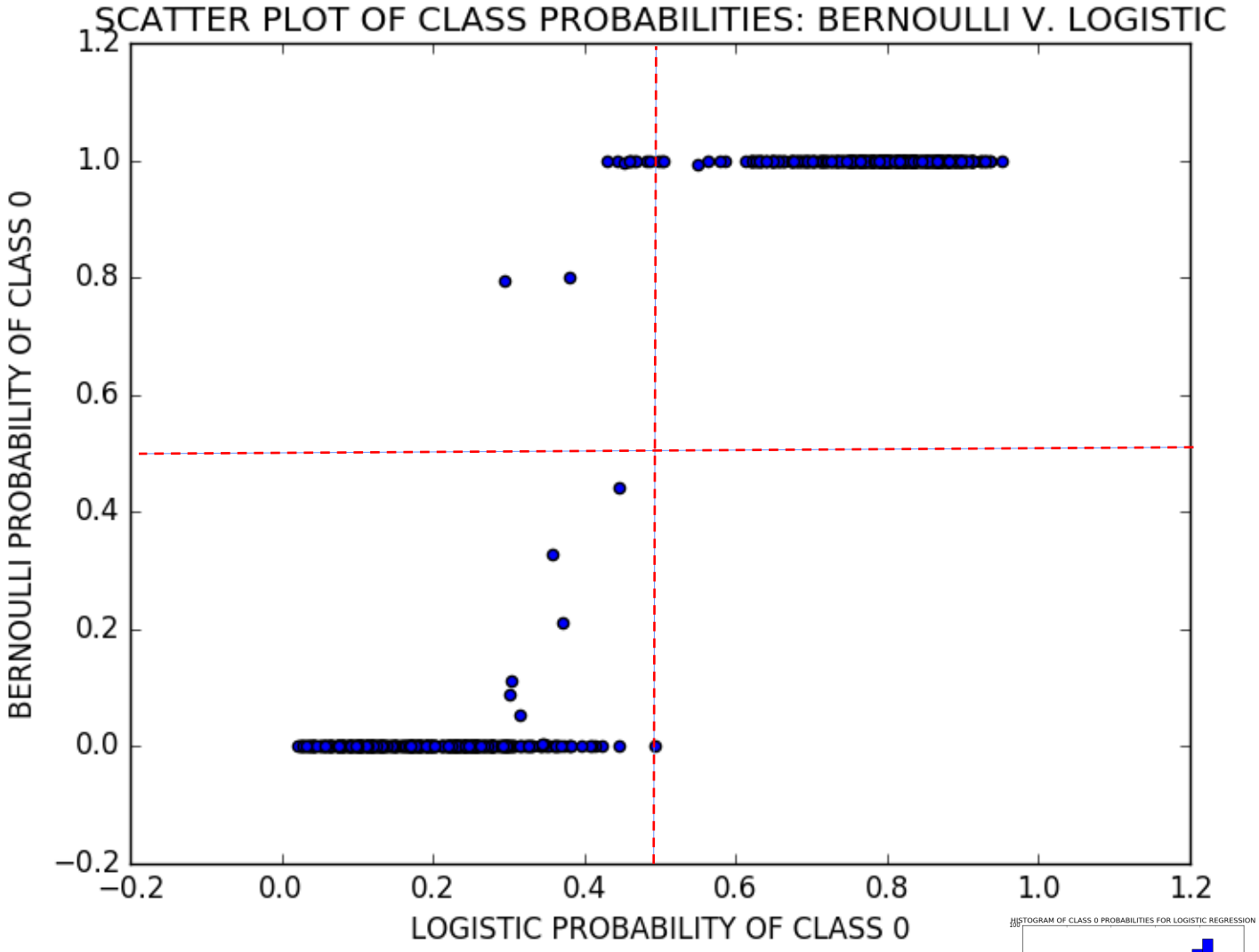

HISTOGRAM OF CLASS 0 PROBABILITIES FOR LOGISTIC REGRESSION
NUMBER OF DOCUMENTS
PROBABILITY OF CLASS 0

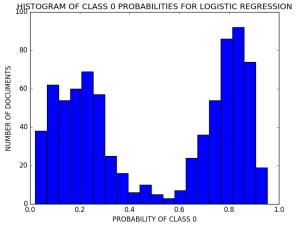

HISTOGRAM OF CLASS 0 PROBABILITIES FOR BERNOULLI NAIVE BAYES
NUMBER OF DOCUMENTS
PROBABILITY OF CLASS 0
0
50
100
150
200
250
300
350
400
450
0.0
0.2
0.4
0.6
0.8
1.0


HISTOGRAM OF CLASS 0 PROBABILITIES FOR MULTINOMIAL NAIVE BAYES
NUMBER OF DOCUMENTS
PROBABILITY OF CLASS 0


HISTOGRAM OF CLASS 0 PROBABILITIES FOR LOGISTIC REGRESSION
NUMBER OF DOCUMENTS
PROBABILITY OF CLASS 0


HISTOGRAM OF CLASS 0 PROBABILITIES FOR BERNOULLI NAIVE BAYES
NUMBER OF DOCUMENTS
PROBABILITY OF CLASS 0


SCATTER PLOT OF CLASS PROBABILITIES: BERNOULLI V. LOGISTIC
BERNOULLI PROBABILITY OF CLASS 0
LOGISTIC PROBABILITY OF CLASS 0


HISTOGRAM OF CLASS 0 PROBABILITIES FOR LOGISTIC REGRESSION
NUMBER OF DOCUMENTS
PROBABILITY OF CLASS 0