MusicMood

Machine Learning in Automatic Music Mood Prediction Based on Song Lyrics

Sebastian Raschka
December 10, 2014

Music Mood Prediction

- We like to listen to music [1][2]
- Digital music libraries are growing
- Recommendation system for *happy* music (clinics, restaurants ...) & genre selection

[1] Thomas Schaefer, Peter Sedlmeier, Christine Sta¨dtler, and David Huron. The psychological functions of music listening. Frontiers in psychology, 4, 2013.

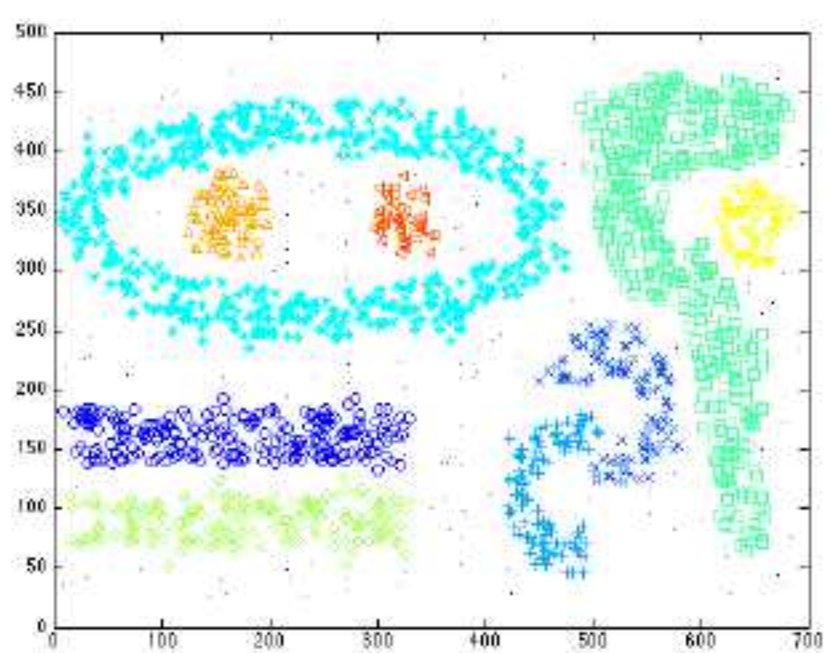
[2] Daniel Vaestfjaell. Emotion induction through music: A review of the musical mood induction procedure. Musicae Scientiae, 5(1 suppl):173–211, 2002.

Predictive Modeling

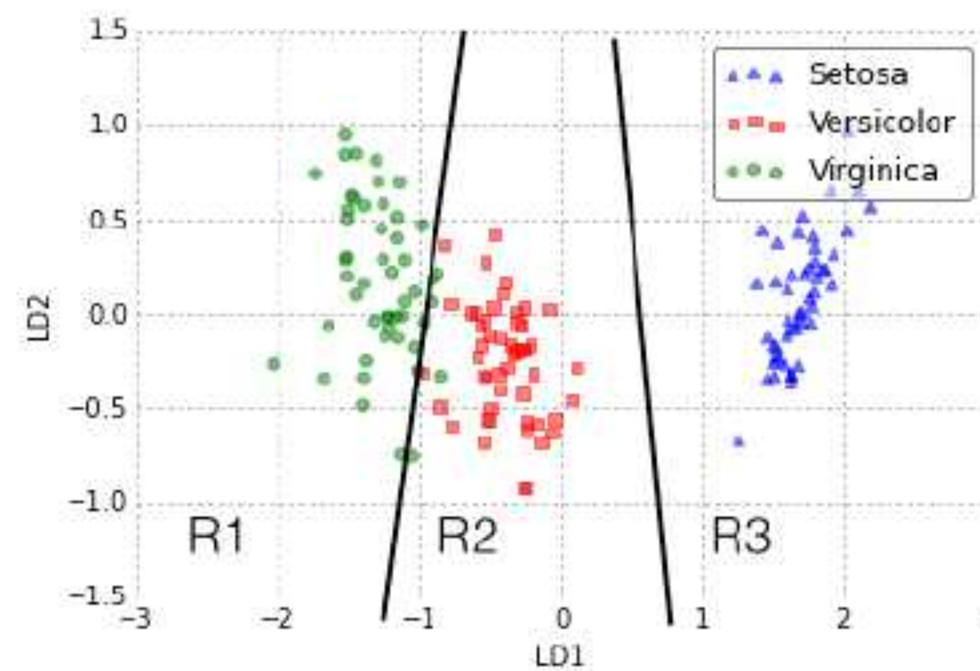
Reinforcement learning

Unsupervised learning
- Hidden Markov models
- Clustering

Supervised learning
- Ranking
- **Classification**
- Regression

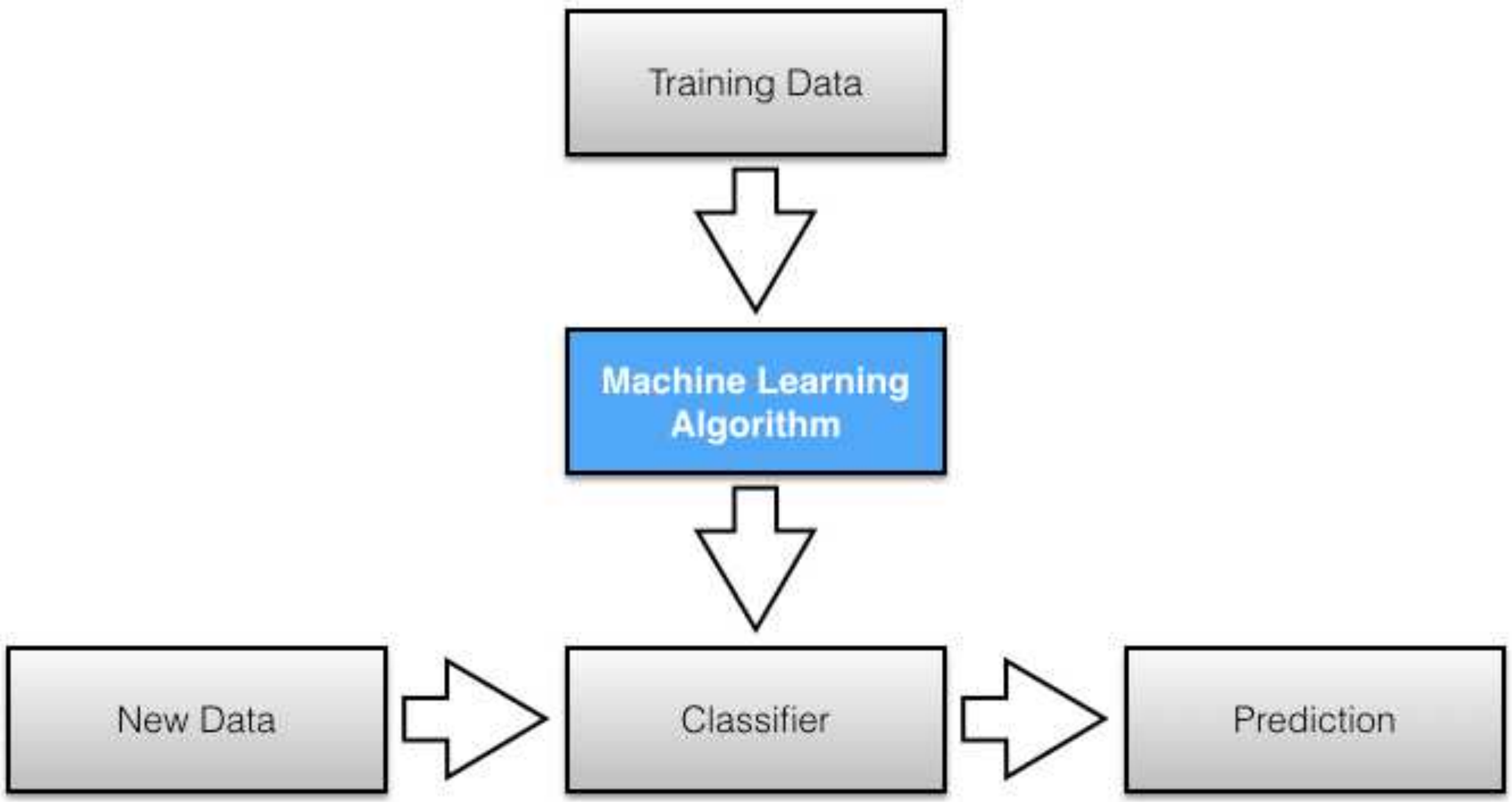


DBSCAN on a toy dataset

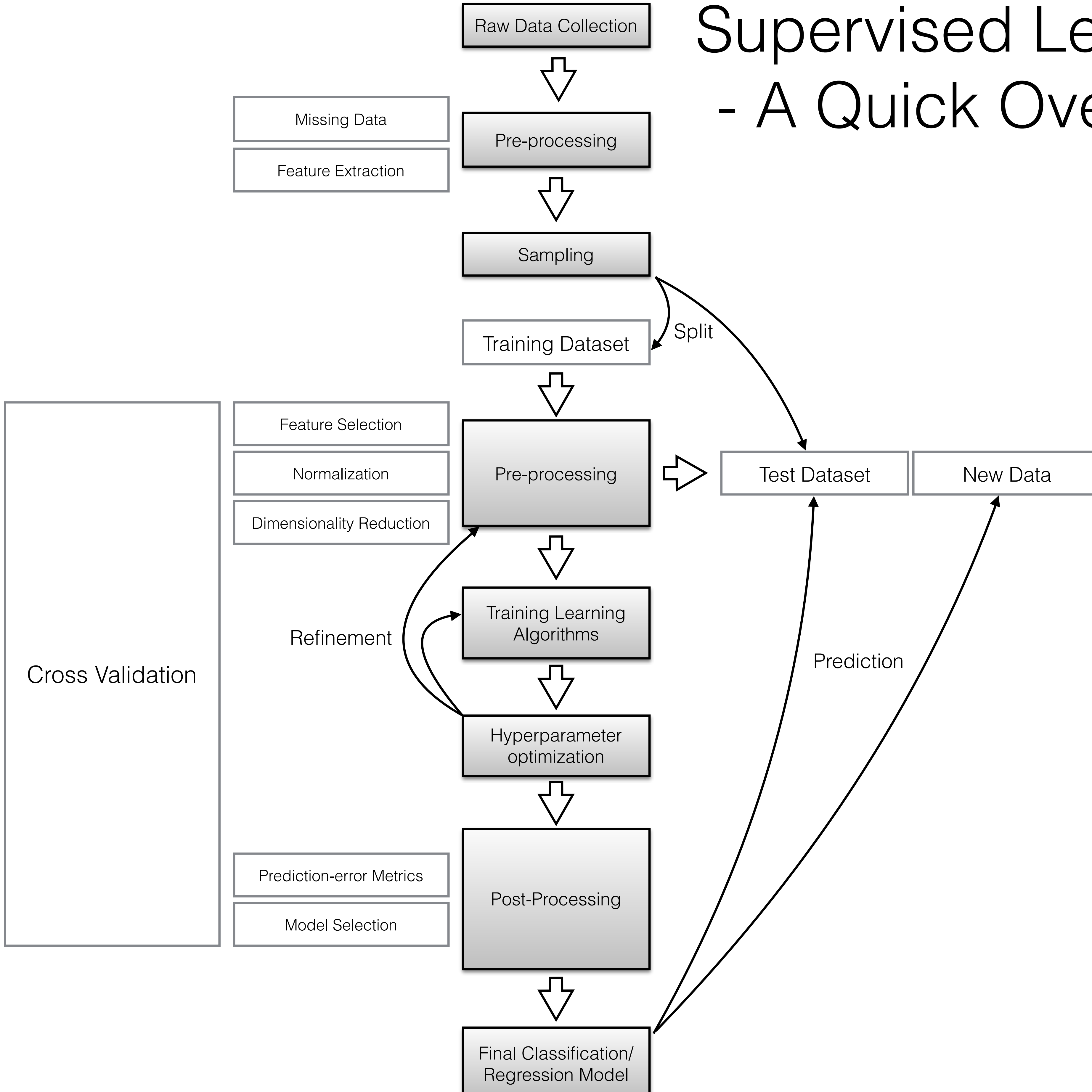


Naive Bayes on Iris (after LDA)

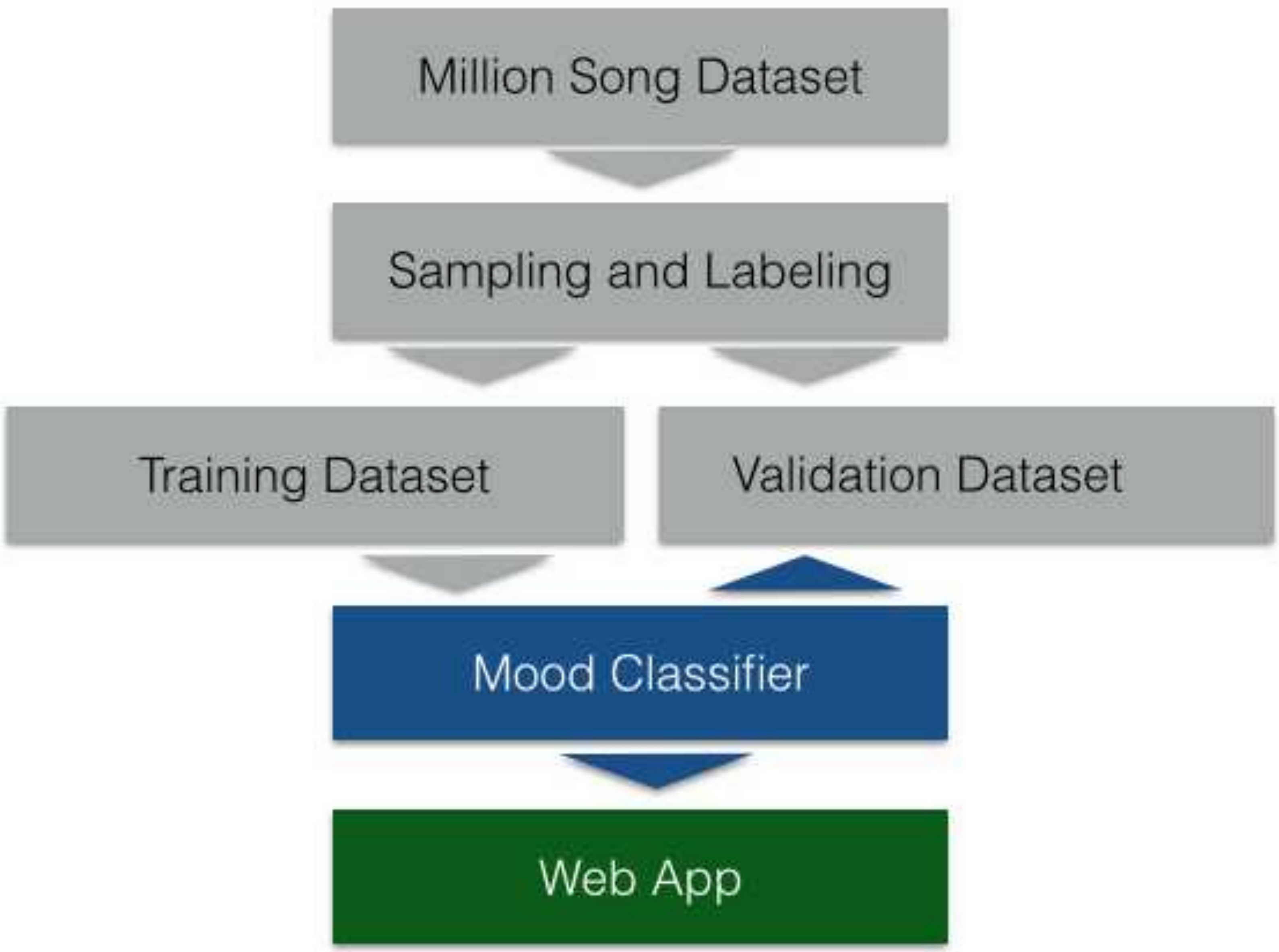
Supervised Learning In a Nutshell



Supervised Learning - A Quick Overview



MusicMood - The Plan



The Dataset

MILLION SONG DATASET

Million Song Dataset

Home | Getting the dataset | Code | Tutorial | Tasks / Demos | More data | Forum

Home » Getting the dataset

Getting the dataset

The logistics of distributing a 300 GB dataset are a little more complicated than for smaller collections. We do, however, provide a directly-downloadable subset for a quick look.

Before you start, you might want to review exactly what the dataset contains. Here is a page showing the contents of a single example file. You can download the corresponding raw HDF5 file here: TRAXLZU12903D05F94.h5.

http://labrosa.ee.columbia.edu/millionsong/

Sampling


1000 songs
for training
Lyrics available?
http://lyrics.wikia.com/Lyrics_Wiki
Lyrics in English?
Python NLTK
200 songs
for validation

Mood Labels

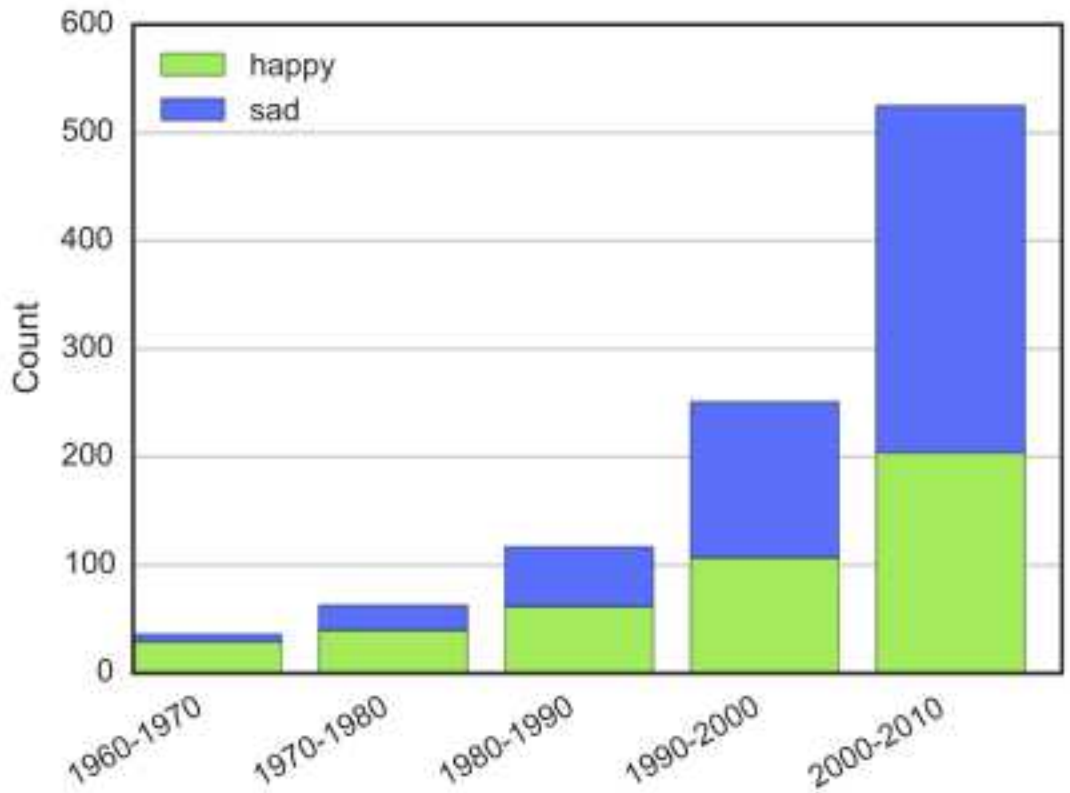
~~Downloading mood labels from Last.fm~~

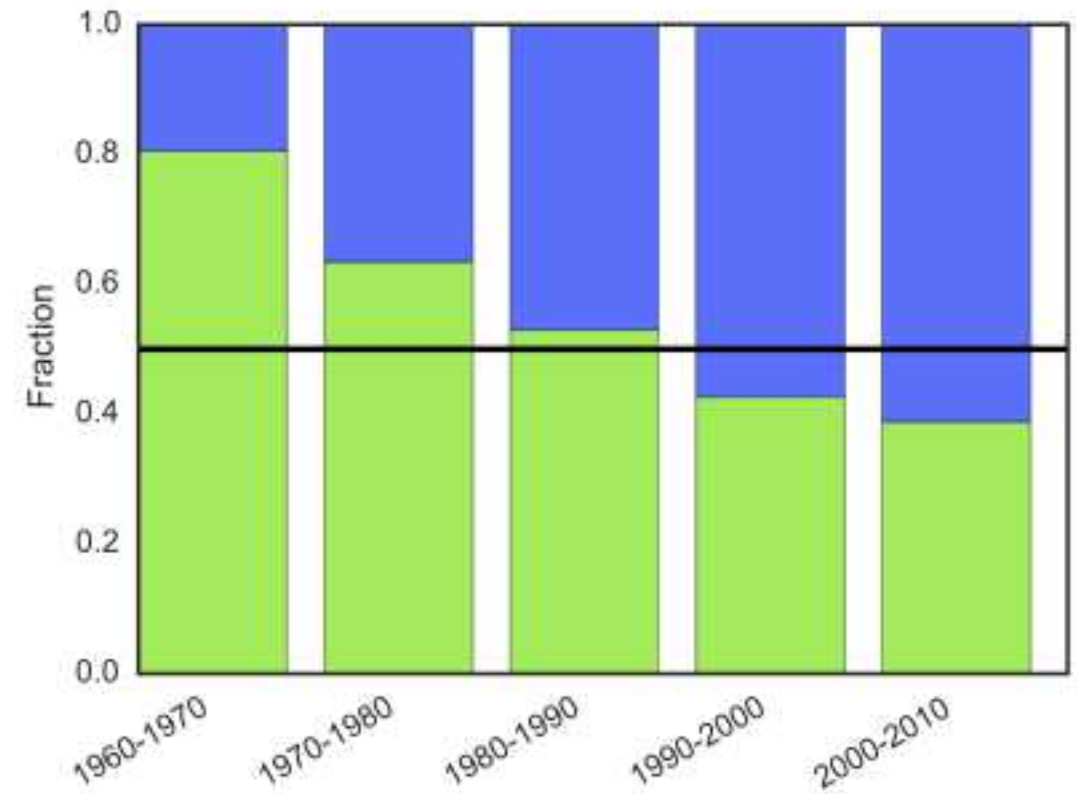
Manual labeling based on lyrics and listening

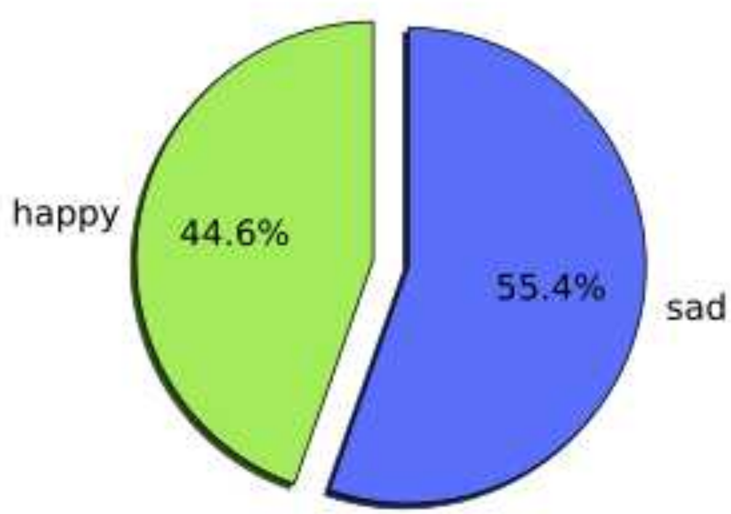
sad if ...

- Dark topic (killing, war, complaints about politics, ...)
- Artist in sorrow (lost love, ...)

Why so sad?
The mood of music over the last 50 years







[based on the 1000-song training dataset]

https://github.com/rasbt/musicmood

Sebastian Raschka 2014

Word Clouds

happy:

sad:

A Short Introduction to Naive Bayes Classification

Naive Bayes - Why?

- Small sample size, can outperform the more powerful alternatives [1]
- "Eager learner" (on-line learning vs. batch learning)
- Fast for classification and re-training
- Success in Spam Filtering [2]
- High accuracy for predicting positive and negative classes in a sentiment analysis of Twitter data [3]

[1] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[2] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105, 1998.

[3] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, pages 1–12, 2009.

Bayes Classifiers
It's All About Posterior Probabilities

$$P\left(\omega_j \mid \mathbf{x}_i\right) = \frac{P\left(\mathbf{x}_i \mid \omega_j\right) \cdot P\left(\omega_j\right)}{P\left(\mathbf{x}_i\right)}$$

$$\text{posterior probability} = \frac{\text{conditional probability} \cdot \text{prior probability}}{\text{evidence}}$$

objective function: maximize the posterior probability

$$\text{predicted class label} \leftarrow \underset{j=1\ldots,m}{\arg\max}\ P\left(\omega_j \mid \mathbf{x}_i\right)$$

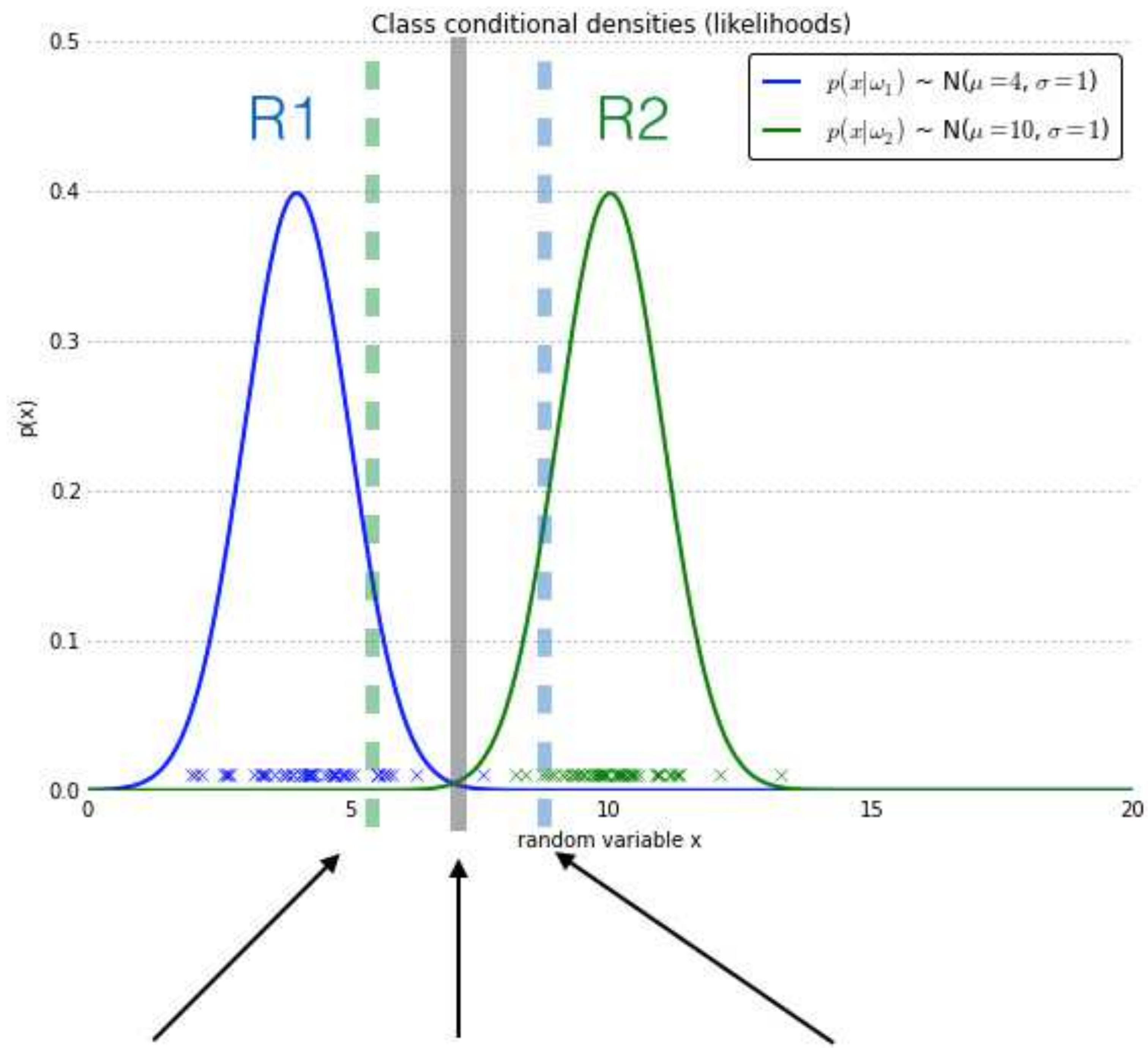
The Prior Probability

Maximum Likelihood Estimate (MLE)

$$\widehat{P}\left(\omega_j\right) = \frac{N_{\omega_j}}{N_c}$$

- N_{ω_j}: Count of samples from class ω_j.
- N_c: Count of all samples.

The Effect of Priors on the Decision Boundary



if $P(\omega_1) < P(\omega_2)$ if $P(\omega_1) = P(\omega_2)$ if $P(\omega_1) > P(\omega_2)$

Class-Conditional Probability

Maximum Likelihood Estimate (MLE)

$$\widehat{P}\left(x_i \mid \omega_j\right) = \frac{N_{i,c}}{N_i} \qquad \left(i = \ (1, \ldots, d)\right)$$

- $N_{i,c}$: Count of feature x_i in class ω_j .
- $N_i c$: Count of feature x_i in all classes.

"chance of observing feature x_i *given that it belongs to class* ω_j *"*

Evidence

$$P(\mathbf{x}_i) = P\left(\mathbf{x}_i \mid \omega_j\right) \cdot P\left(\omega_j\right) + P\left(\mathbf{x}_i \mid \omega_j^C\right) \cdot P\left(\omega_j^C\right)$$

just a normalization factor,
can be omitted in decision rule:

$$\frac{P\left(\mathbf{x}_i \mid \omega_1\right) \cdot P(\omega_1)}{P(\mathbf{x}_i)} > \frac{P\left(\mathbf{x}_i \mid \omega_2\right) \cdot P(\omega_2)}{P(\mathbf{x}_i)}$$

$$\propto P\left(\mathbf{x}_i \mid \omega_1\right) \cdot P(\omega_1) > P\left(\mathbf{x}_i \mid \omega_2\right) \cdot P(\omega_2)$$

Naive Bayes Models

Gaussian Naive Bayes

$$P\left(x_{ik} \mid \omega\right) = \frac{1}{\sqrt{2\pi\sigma_{\omega}^{2}}} \exp\left(-\frac{\left(x_{ik} - \mu_{\omega}\right)^{2}}{2\sigma_{\omega}^{2}}\right),$$

$$P\left(\mathbf{x}_{i} \mid \omega\right) = \prod_{k=1}^{d} P\left(\mathbf{x}_{ik} \mid \omega\right)$$

for continuous variables

Naive Bayes Models

Multi-variate Bernoulli Naive Bayes

$$P\left(\mathbf{x}|\omega_j\right) = \prod_{i=1}^{m} P\left(x_i \mid \omega_j\right)^b \cdot \left(1 - P\left(x_i \mid \omega_j\right)\right)^{(1-b)} \qquad (b \in 0, 1)$$

for binary features

Naive Bayes Models

Multinomial Naive Bayes

$$\widehat{P}\left(x_i \mid \omega_j\right) = \frac{\sum tf\left(x_i, d \in \omega_j\right) + \alpha}{\sum N_{d\in\omega j} + \alpha \cdot V}$$

$$P\left(\mathbf{x}|\omega_j\right) = P\left(x_1 \mid \omega_j\right) \cdot P\left(x_2 \mid \omega_j\right) \cdot \ldots \cdot P\left(x_n \mid \omega_j\right) = \prod_{i=1}^{m} P\left(x_i \mid \omega_j\right)$$

- x_i : A word from the feature vector $\mathbf{x}$ of a particular sample.
- $\sum tf\left(x_i, d \in \omega_j\right)$: The sum of raw term frequencies of word x_i from all documents in the training sample that belong to class ω_j.
- $\sum N_{d\in\omega j}$: The sum of all term frequencies in the training dataset for class ω_j.
- α: An additive smoothing parameter ($\alpha = 1$ for Laplace smoothing).
- V: The size of the vocabulary (number of different words in the training set).

Naive Bayes and Text Classification

Feature Vectors

The Bag of Words Model

- D_1: "Each state has its own laws."
- D_2: "Every country has its own culture."

	each	state	has	its	own	laws	every	country	culture
$\mathbf{x}_{D1}$	1	1	1	1	1	1	0	0	0
$\mathbf{x}_{D2}$	0	0	1	1	1	0	1	1	1
$\sum$	1	1	2	2	2	1	1	1	1

Tokenization and N-grams

"a swimmer likes swimming thus he swims"

- unigram (1-gram):

a	swimmer	likes	swimming	thus	he	swims

- bigram (2-gram):

a swimmer	swimmer likes	likes swimming	swimming thus	...

- trigram (3-gram):

a swimmer likes	swimmer likes swimming	likes swimming thus	...

Stemming and Lemmatization

Porter Stemming

A swimmer likes swimming, thus he swims.

↓

a	swimmer	like	swim	,	thu	he	swim	.

Lemmatization

A swimmer likes swimming, thus he swims.

↓

A	swimmer	like	swimming	,	thus	he	swim	.

Stop Word Removal

A swimmer likes swimming, thus he swims.

↓

swimmer	likes	swimming	,	swims	.

Term and Frequency

$$\text{normalized term frequency} = \frac{tf(t,d)}{n_d}$$

where

- $tf(t,d)$: Raw term frequency (the count of term t in document d).
- n_d: The total number of terms in document d.

Term Frequency - Inverse Document Frequency (Tf-idf)

$$\text{Tf-idf} = tf_n\left(t,d\right) \cdot idf\left(t\right)$$

Let $tf_n\left(d,f\right)$ be the normalized term frequency, and idf, the inverse document frequency, which can be calculated as follows

$$idf\left(t\right) = \log\left(\frac{n_d}{n_d\left(t\right)}\right),$$

where

- n_d: The total number of documents.
- $n_d\left(t\right)$: The number of documents that contain the term t.

Grid Search and 10-fold Cross Validation to Optimize F1

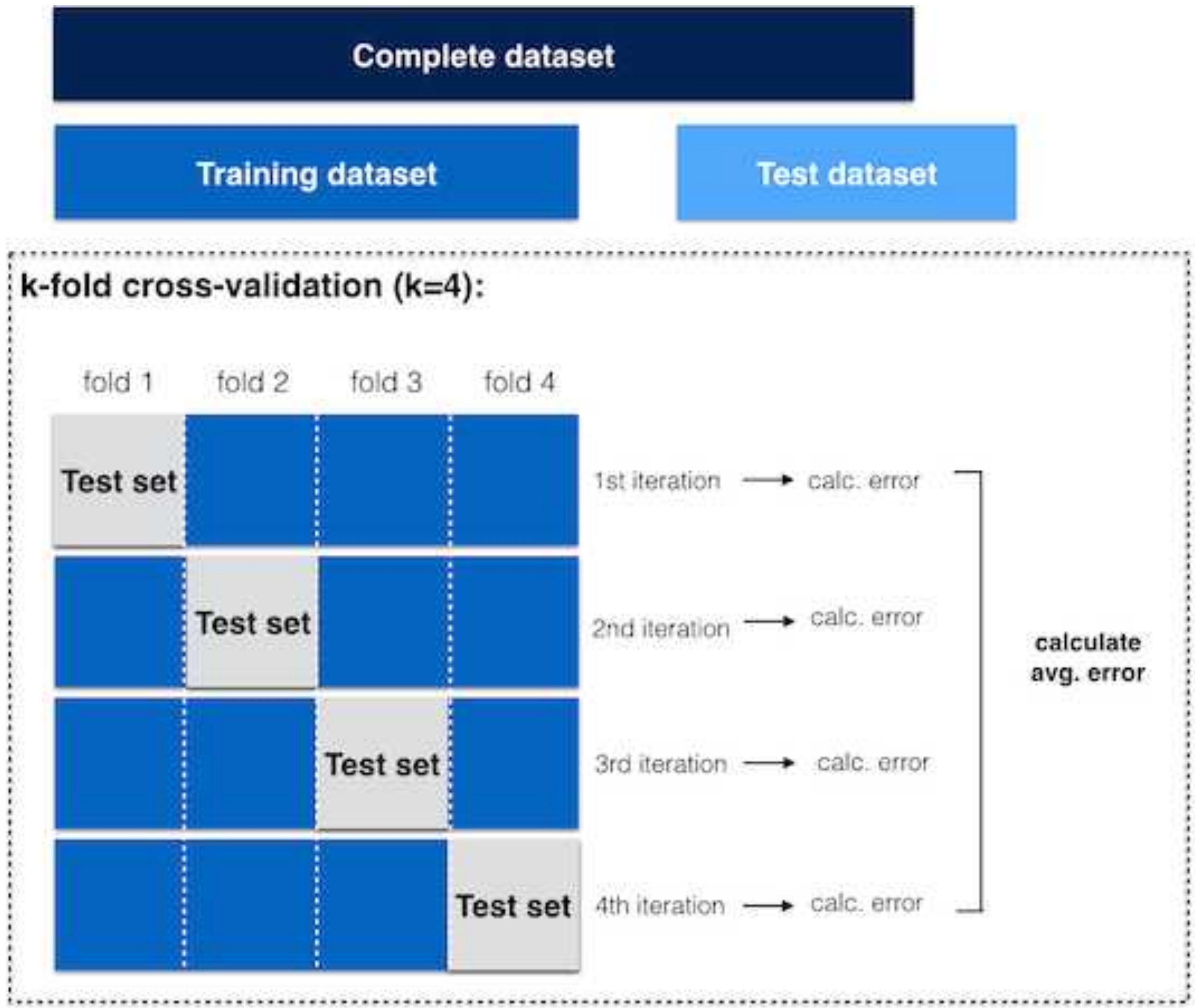
$$F1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

$$\mathrm{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

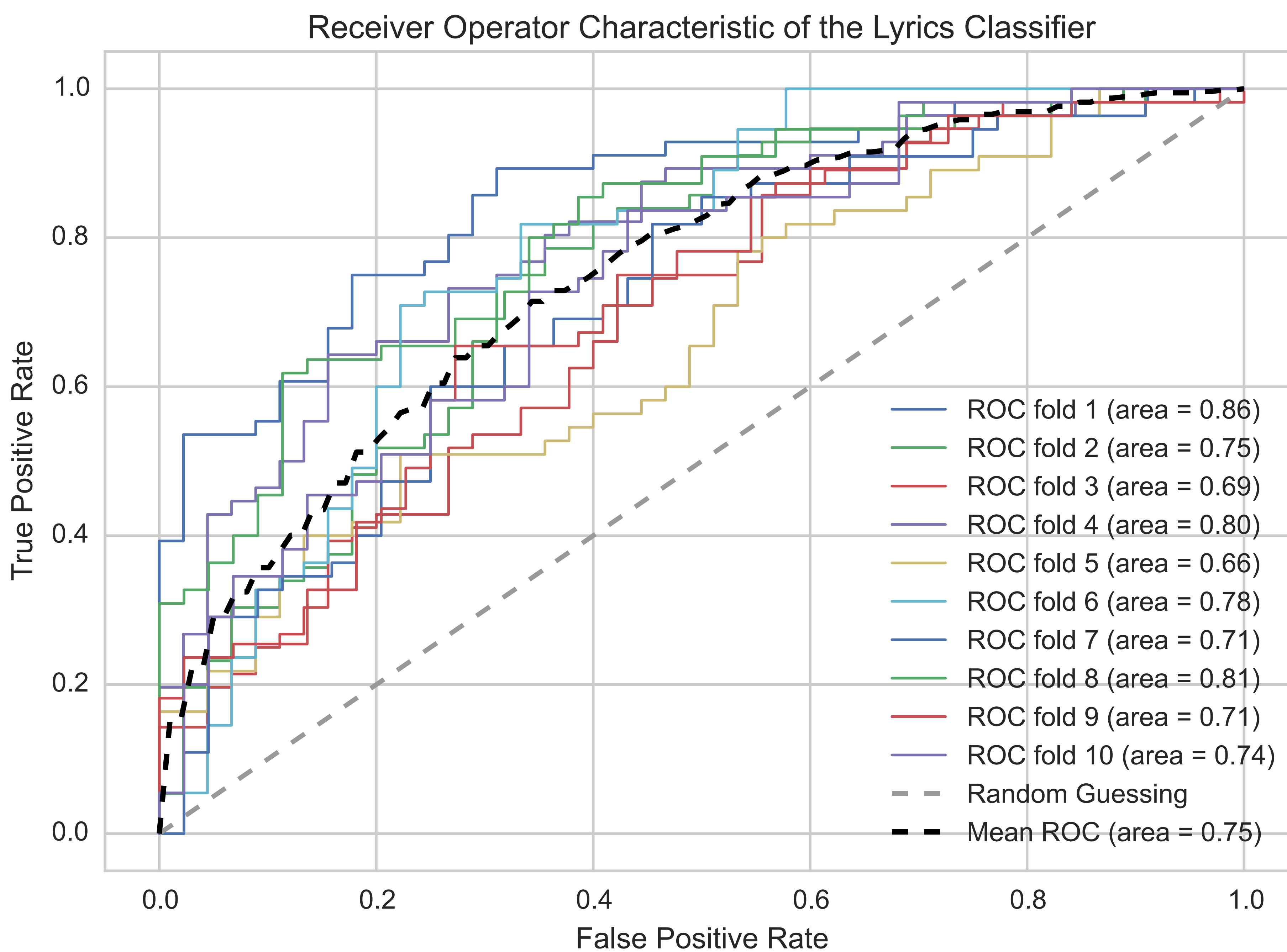
$$\mathrm{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

- TP = true positive (happy predicted as happy)
- FP = false positive (sad predicted as happy)
- FN = false negative (happy predicted as sad)

K-Fold Cross Validation



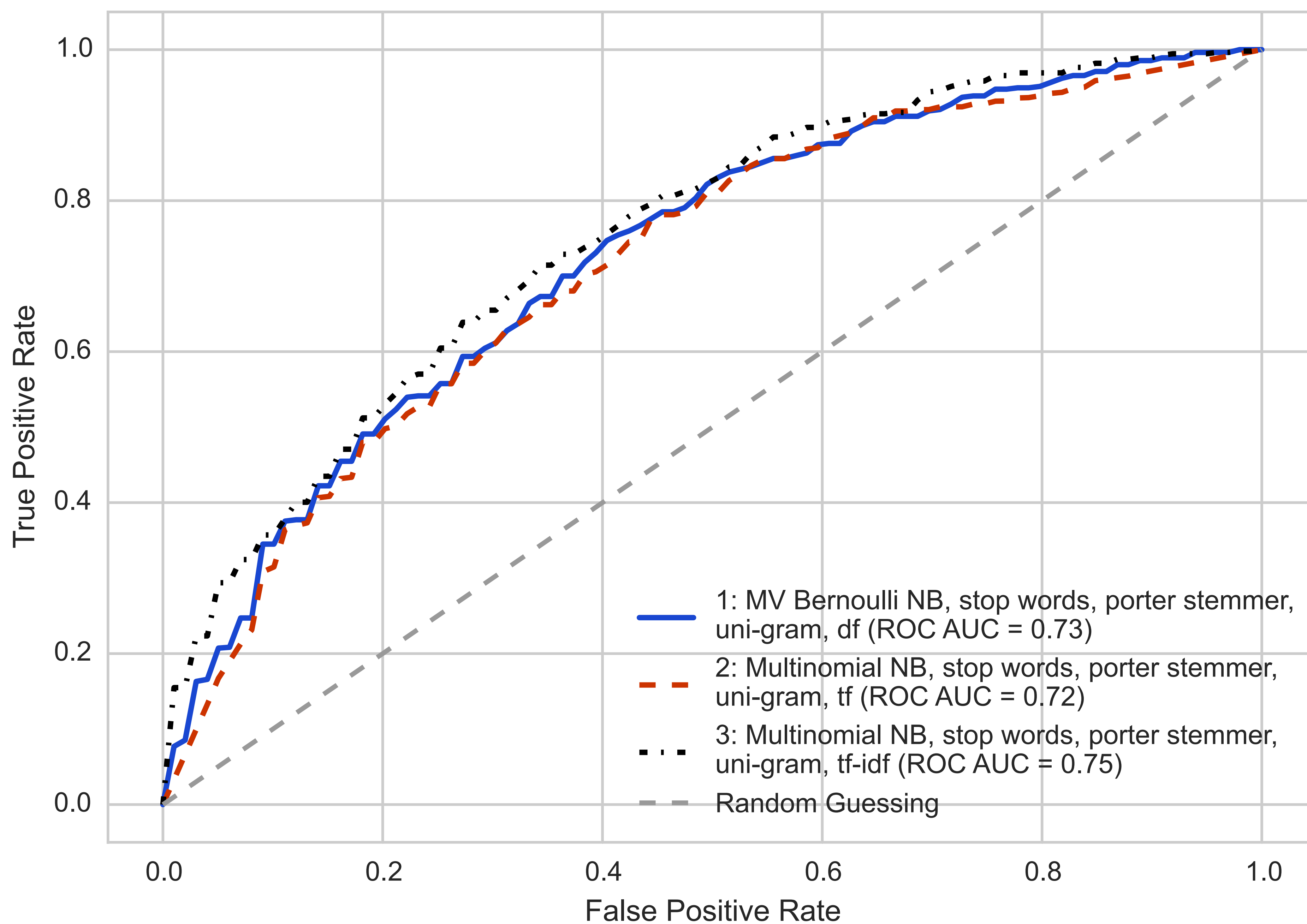
10-Fold Cross Validation After Grid Search



(final model)

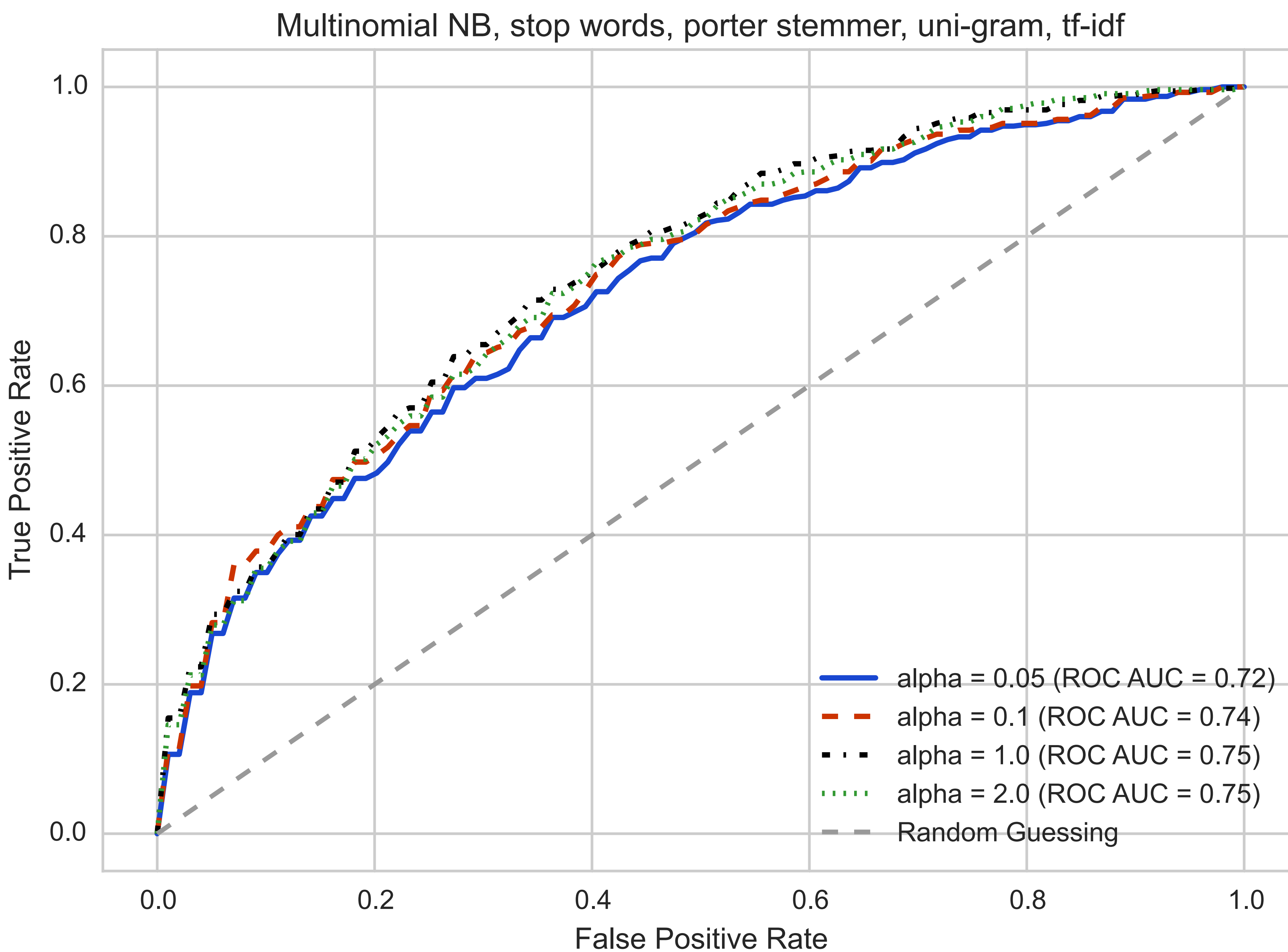
10-fold Cross Validation (mean ROC)

Multinomial vs Multi-variate Bernoulli Naive Bayes



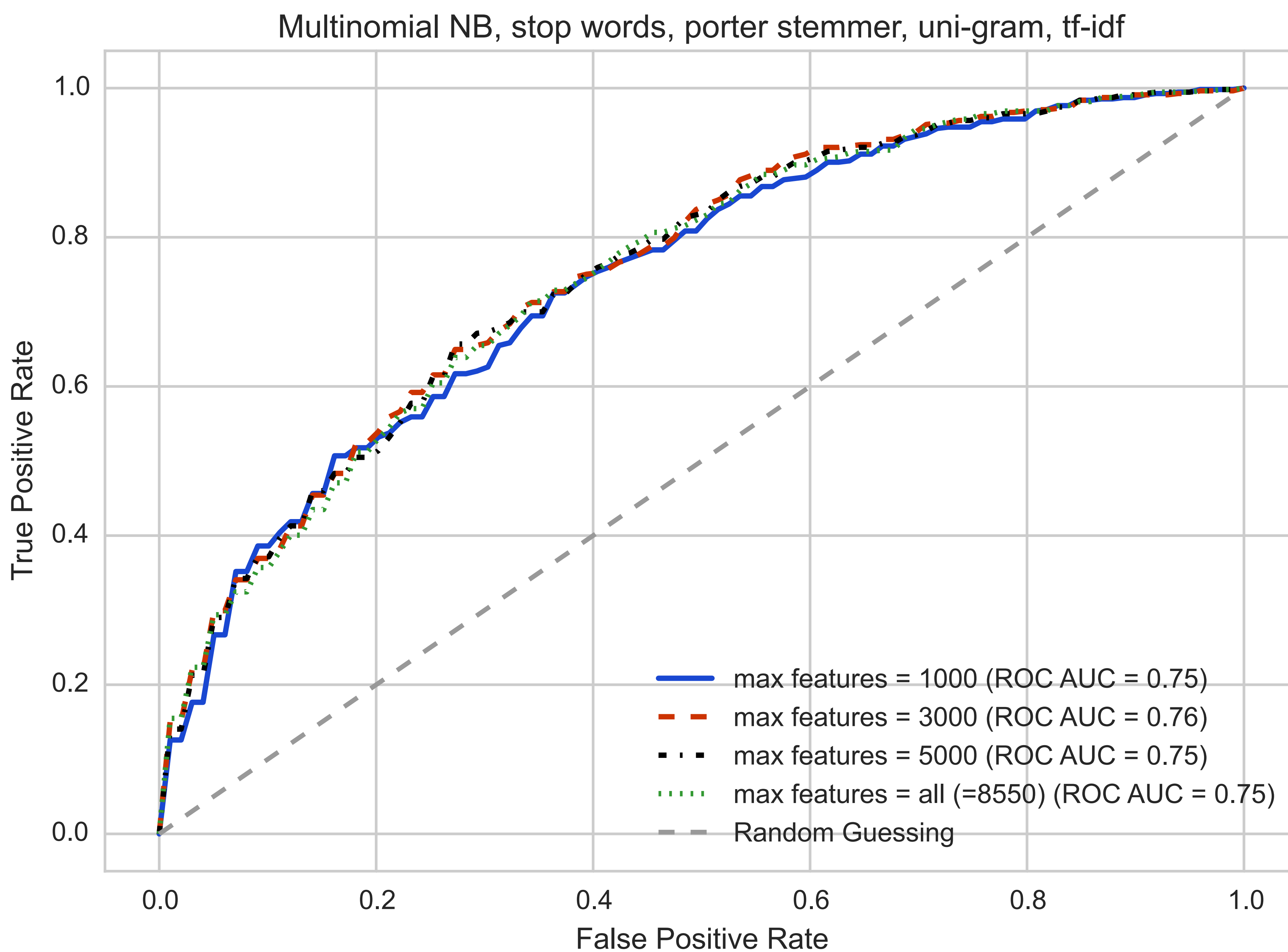
10-fold Cross Validation (mean ROC)

Multinomial Naive Bayes & Hyperparameter Alpha



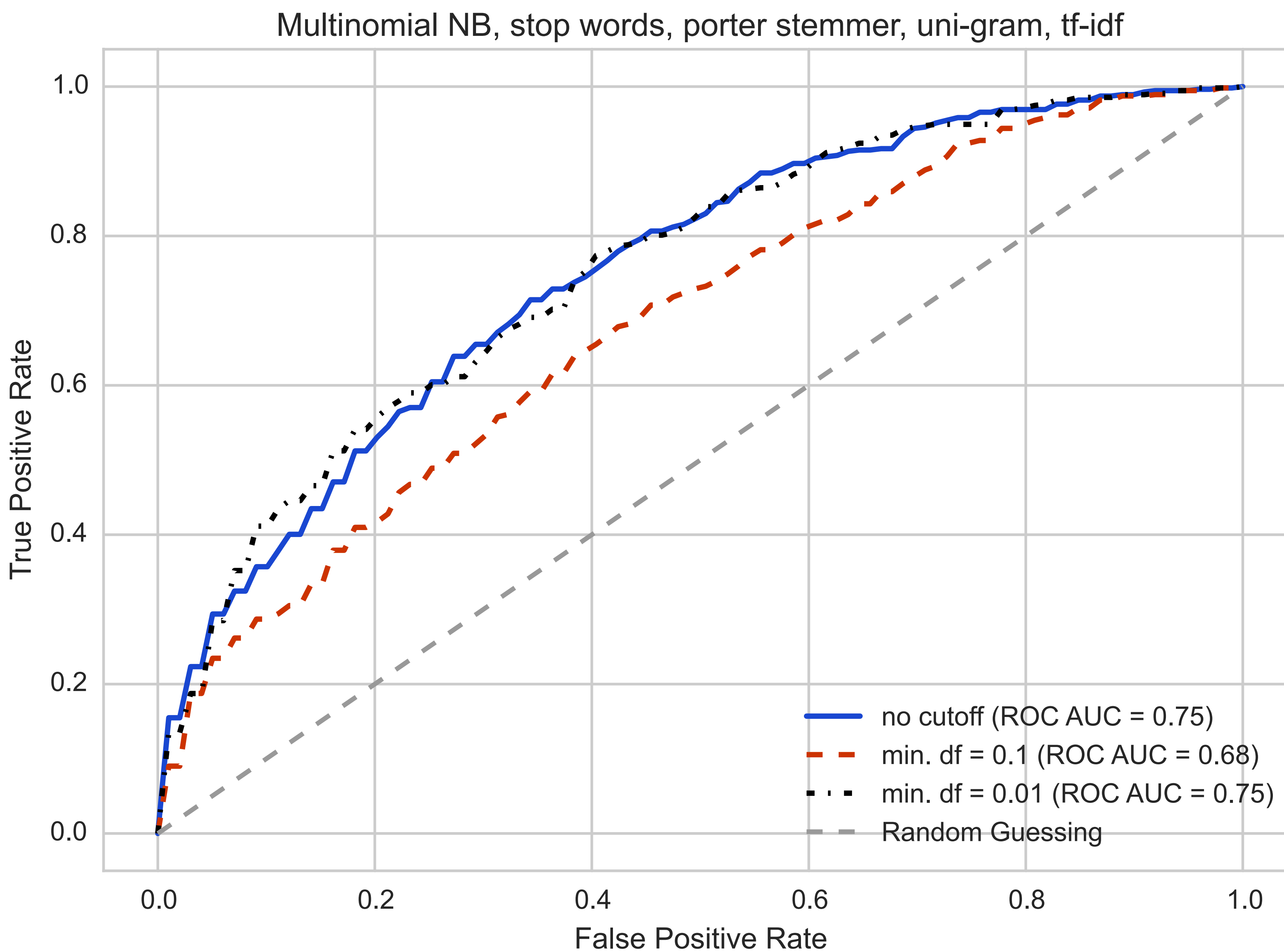
10-fold Cross Validation (mean ROC)

Multinomial Naive Bayes & Vocabulary Size



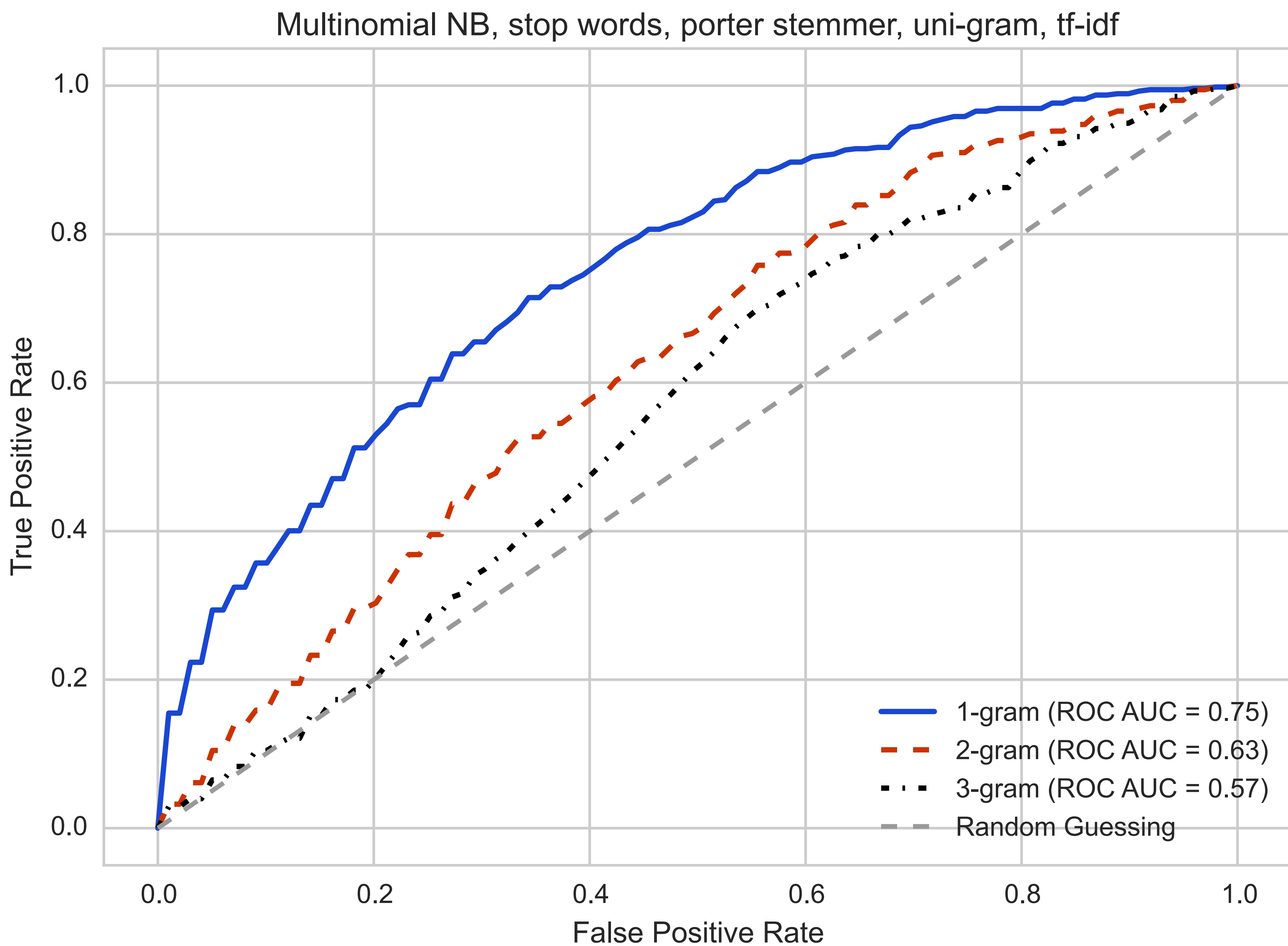
10-fold Cross Validation (mean ROC)

Multinomial Naive Bayes & Document Frequency Cut-off



10-fold Cross Validation (mean ROC)

Multinomial Naive Bayes & N-gram Sequence Length

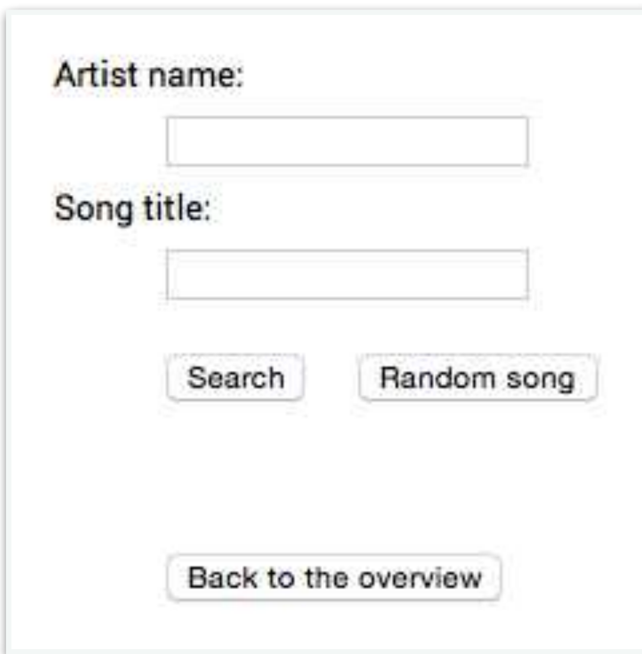


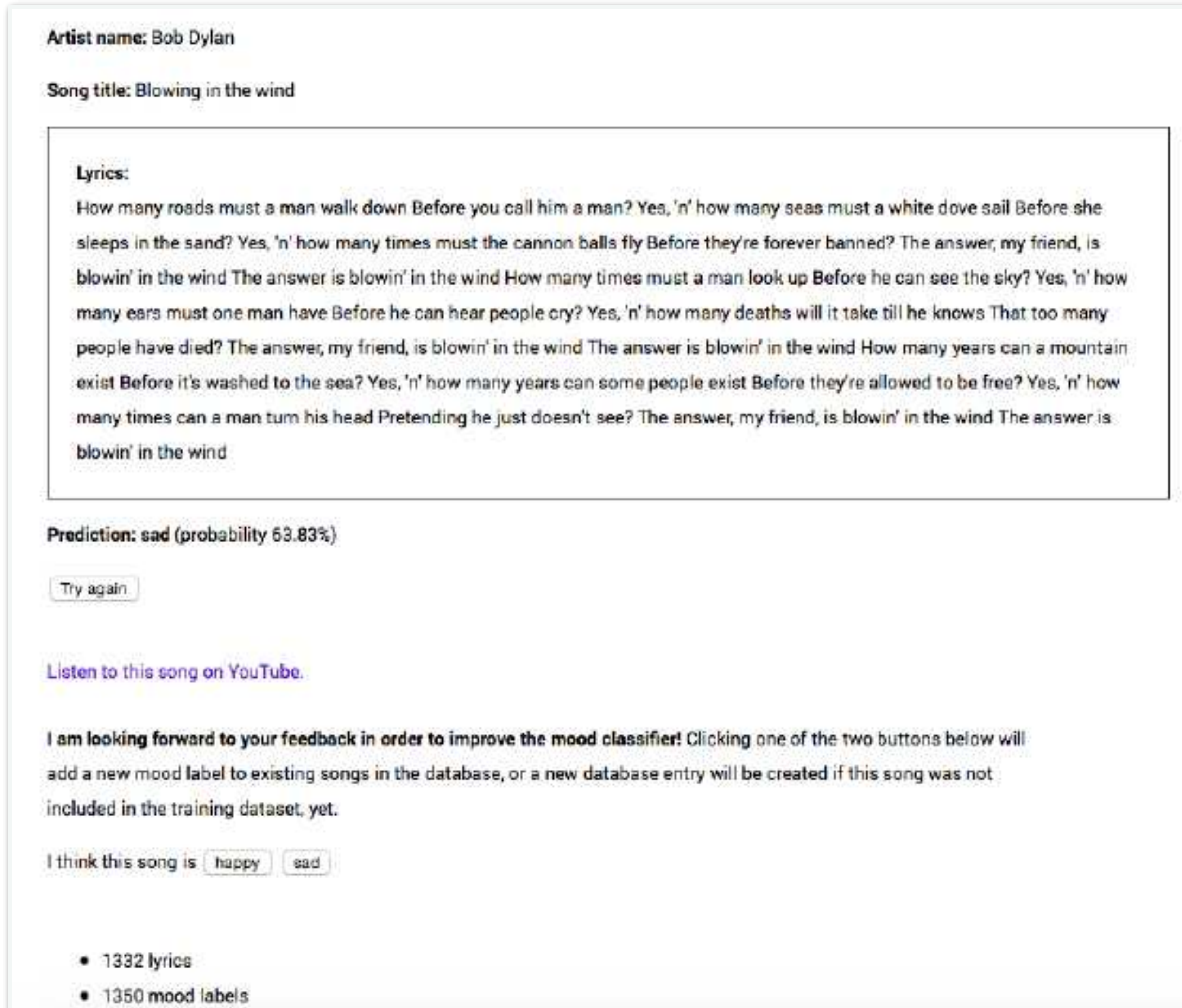
Contingency Tables of the Final Model

Performance metric	Training	Validation
Accuracy	80.00%	54.50%
Precision	99.60%	88.89%
Recall	55.38%	15.24%
F1-score	71.18%	26.02%
ROC auc	77.60%	56.57%

Live Demo

http://sebastianraschka.com/Webapps/musicmood.html





Flask web development, one drop at a time

Future Plans

- Growing a list of mood labels (majority rule).

- Performance comparisons of different machine learning algorithms.

- Genre prediction and selection based on sound.

Thank you!