Naive Bayes and NLP

PIERIAN DATA

Naive Bayes and NLP

- In this section we will begin a discussion on using raw string text for machine learning models.
- This idea in general is known as “Natural Language Processing”.

Naive Bayes and NLP

- Section Overview
  - Naive Bayes Algorithm and NLP
  - Extracting Features from Text Data
  - Text Classification Project
- Keep in mind:
  - *This section focuses on supervised learning text tasks. We will discuss unsupervised text tasks later on.*

Let’s get started!

Naive Bayes

Part One: Bayes' Theorem

Natural Language Processing

- Naive Bayes is the shorthand for a set of algorithms that use Bayes' Theorem for supervised learning classification.
- Bayes' Theorem is a probability formula that leverages previously known probabilities to define probability of related events occuring.

- Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' Theorem.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Natural Language Processing

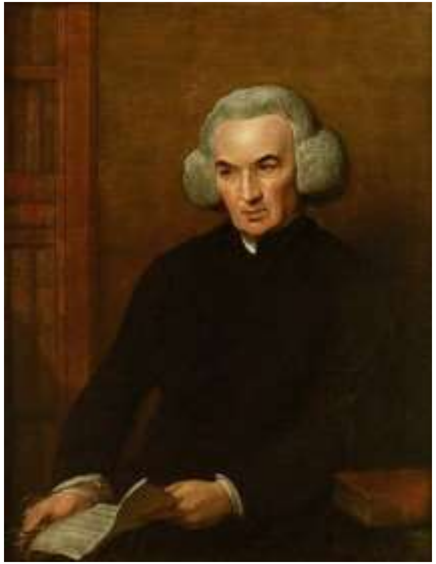
- 1700s: Thomas Bayes was a Presbyterian minister in England who studied theology, statistics, and logic.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Natural Language Processing

- 1700s: Bayes' Theorem was published after his death! Richard Price edited and published his notes.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$



Natural Language Processing

- Bayes' Theorem

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

  - **A** and **B** are events
  - **P(A|B)** is probability of event **A** given that **B** is True.
  - **P(B|A)** is probability of event **B** given that **A** is True.
  - **P(A)** is probability of A occurring.
  - **P(B)** is probability of B occurring.

Natural Language Processing

- Bayes' Theorem
  - Let's walk through a quick example!

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Natural Language Processing

- Imagine the following situation for a city:
  - Every apartment in a building is fit with a fire alarm detection system.
  - However, there are false alarms where smoke is detected but there is not a dangerous fire to put out (e.g. smoke from an oven).

Natural Language Processing

- The associated probabilities:
  - Actual dangerous fires occur only 1% of the time.
  - Smoke alarms are not very good, and go off about 10% of the time.
  - When there is an actual dangerous fire, 95% of the time the smoke alarms go off.

Natural Language Processing

- **Question to answer:**
  - *If you get a smoke alarm detecting a fire, what is the probability that there actually is a dangerous fire?*

Natural Language Processing

- In terms of probability events:
    - Event A: Dangerous Fire
    - Event B: Smoke Alarm Triggered
    - P(A|B):
        - Probability of Fire given Smoke Alarm
    - P(B|A):
        - Probability of Smoke Alarm given a ngerous fire

Natural Language Processing

- **In terms of probability events:**
  - **Event A: Dangerous Fire**
  - **Event B: Smoke Alarm Triggered**
  - *P(A|B):*
    - *Probability of Fire given Smoke Alarm*
  - **P(B|A):**
    - **Probability of Smoke Alarm given a dangerous fire**

Natural Language Processing

- Imagine the following situation for a city:
    - Actual dangerous fires occur only 1% of the time. **P(Fire) = 1/100**
    - Smoke alarms are not good and go off about 10% of the time. **P(Smoke) = 1/10**
    - When there is an actual dangerous fire, 95% of the time the smoke alarms go off.
    - **Smoke|Fire) = 95/100**

Natural Language Processing

- Using Bayes' Theorem: $P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$

P(Fire|Smoke) =
P(Smoke|Fire)*P(Fire)/P(Smoke)

Natural Language Processing

- Using Bayes' Theorem: $P(A \mid B) = \dfrac{P(B \mid A) \cdot P(A)}{P(B)}$

P(Fire|Smoke) =
P(Smoke|Fire)*P(Fire)/P(Smoke)
P(Fire|Smoke) = 0.95 * 0.01 / 0.1
P(Fire|Smoke) = 0.095
P(Fire|Smoke) = 9.5%

Natural Language Processing

- Let’s move on to explore how Bayes’ Theorem can be extended to perform classification.
- Specifically, we’ll focus on using Bayes’ Theorem for Natural Language Processing Classification.

Naive Bayes

Part Two: Naive Bayes

Natural Language Processing

- Let’s move on to explore how Bayes’ Theorem can be extended to perform classification.
- Specifically, we’ll focus on using Bayes’ Theorem for Natural Language Processing Classification.

Natural Language Processing

- Let's first walkthrough the conversion of Bayes' Theorem to a machine learning model!

Natural Language Processing

- We model the probability of belonging to a class given a vector of features.

$$\mathbf{x} = (x_1, \dots, x_n)$$

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} \Rightarrow p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

- The numerator is equivalent to a joint probability model:

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})} \Rightarrow p(C_k \mid \mathbf{x}) = \frac{p(C_k, x_1, \dots, x_n)}{p(\mathbf{x})}$$

Natural Language Processing

- The chain rule can rewrite this numerator as a series of products of conditional probabilities:

$$
\begin{aligned}
p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\
&= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2, \dots, x_n, C_k) \\
&= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2 \mid x_3, \dots, x_n, C_k)\, p(x_3, \dots, x_n, C_k) \\
&= \cdots \\
&= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2 \mid x_3, \dots, x_n, C_k) \cdots p(x_{n-1} \mid x_n, C_k)\, p(x_n \mid C_k)\, p(C_k)
\end{aligned}
$$

Natural Language Processing

- Finally we need to make an assumption, we assume all x features are **mutually independent** of each other.
- Allowing for this conditional probability:

$$p(x_i \mid x_{i+1}, \ldots, x_n, C_k) = p(x_i \mid C_k)$$

Natural Language Processing

- Then the joint model (the full Naive Bayes model) is fully written as:

$$\begin{aligned} p(C_k \mid x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k)\, \cdots \\ &\propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,, \end{aligned}$$

Natural Language Processing

- Where ∝ denotes proportionality:

$$\begin{aligned} p(C_k \mid x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k)\ p(x_1 \mid C_k)\ p(x_2 \mid C_k)\ p(x_3 \mid C_k)\ \cdots \\ &\propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,, \end{aligned}$$

Natural Language Processing

- Let's walk through an example of using this Naive Bayes model.

$$
\begin{aligned}
p(C_k \mid x_1, \ldots, x_n) &\propto p(C_k, x_1, \ldots, x_n) \\
&\propto p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k)\, \cdots \\
&\propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,,
\end{aligned}
$$

Natural Language Processing

- There are many variations of Naive Bayes models, including:
  - Multinomial Naive Bayes
  - Gaussian Naive Bayes
  - Complement Naive Bayes
  - Bernoulli Naive Bayes
  - Categorical Naive Bayes
  - ***Check out the online documentation!***

Natural Language Processing

- We will be focusing on Multinomial Naive Bayes, since its used most often in the context of natural language processing.
- Let’s imagine we want to create a movie review aggregation website where we need to classify movie reviews into two categories: positive or negative.

Natural Language Processing

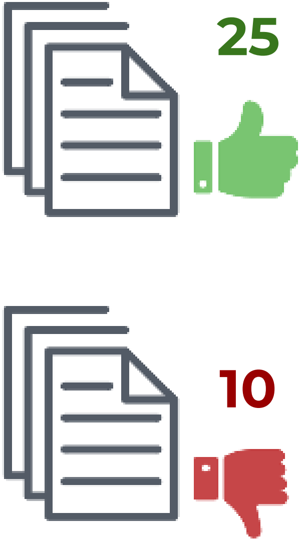
- Using previous reviews, we can have someone manually label them in order to create a labeled data set.
- Then in the future, we could use our machine learning algorithm to automatically classify a new text review for us.

Natural Language Processing

- But how do we actually train on this text data?
- Multinomial Bayes can work quite well with a simple count vectorization model (counting the frequency of each word in each document).

Natural Language Processing

- Begin by separating out document classes:



Natural Language Processing

- Create “prior” probabilities for each class:

PIERIAN DATA

- Create “prior” probabilities for each class:

P(pos) = 25/35

P(neg) = 10/35

Natural Language Processing

- We will use these later!

P(pos) = 25/35

P(neg) = 10/35

- Start with count vectorization on classes:

Natural Language Processing

- Start with count vectorization on classes:

10	2	8	4
movie	actor	great	film

8	10	0	2
movie	actor	great	film

Natural Language Processing

- Calculate conditional probabilities per class and word.

10	2	8	4
movie	actor	great	film

8	10	0	2
movie	actor	great	film

- Calculate conditional probabilities:

10	2	8	4
movie	actor	great	film

P(movie|pos)

8	10	0	2
movie	actor	great	film

Natural Language Processing

- Calculate conditional probabilities:

10	2	8	4
movie	actor	great	film

$P(\text{movie}|\text{pos}) = 10/24 = 0.42$

8	10	0	2
movie	actor	great	film

Natural Language Processing

- Calculate conditional probabilities:

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

8	10	0	2
movie	actor	great	film

P(movie|pos) = 10/24 = 0.42
P(actor|pos) = 2/24 = 0.08
P(great|pos) = 8/24 = 0.33
P(film|pos) = 4/24 = 0.17

- Calculate conditional probabilities:

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

8	10	0	2
movie	actor	great	film

P(movie|neg)= 8/20 = 0.4
P(actor|neg) = 10/20 = 0.5
P(great|neg) = 0/20 = 0
P(film|neg) = 2/20 = 0.1

PIERIAN DATA

Natural Language Processing

- Calculate conditional probabilities:

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

P(movie|neg)= 8/20 = 0.4
P(actor|neg) = 10/20 = 0.5
P(great|neg) = 0/20 = 0
P(film|neg) = 2/20 = 0.1

Natural Language Processing

- Now a new review was created:

0.42	**0.08**	**0.33**	**0.17**
10	2	8	4
movie	actor	great	film

“movie actor”

0.4	**0.5**	**0**	**0.1**
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Now a new review was created:

0.42	**0.08**	**0.33**	**0.17**
10	2	8	4
movie	actor	great	film

0.4	**0.5**	**0**	**0.1**
8	10	0	2
movie	actor	great	film

"movie actor"

P(pos)

Natural Language Processing

- Start with the prior probability

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

“movie actor”

P(pos)

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Start with prior probability

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

“movie actor”

P(pos) = (25/35)

Natural Language Processing

- Continue with conditional probabilities

25

0.42	**0.08**	**0.33**	**0.17**
10	2	8	4
movie	actor	great	film

“movie actor”

P(pos)×P(movie|pos)

10

0.4	**0.5**	**0**	**0.1**
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Continue with conditional probabilities

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

“movie actor”

P(pos)×P(movie|pos)×P(actor|pos)

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Continue with conditional probabilities

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

"movie actor"

(0.71)×(0.42)×(0.08)

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Continue with conditional probabilities

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

“movie actor”

$(0.71) \times (0.42) \times (0.08) = 0.024$

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Calculate this score factor as 0.024

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

“movie actor”

$$(0.71)\times(0.42)\times(0.08) = 0.024$$

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Score is proportional to P(pos|"movie actor")

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

"movie actor"

$$(0.71)\times(0.42)\times(0.08) = 0.024$$

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Score is proportional to P(pos|“movie actor”)

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

“movie actor”

0.024 ∝ P(pos| “movie actor”)

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Repeat same process with negative class

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

"movie actor"

P(neg)×P(movie|neg)×P(actor|neg)

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Repeat same process with negative class

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

“movie actor”

$$(10/35)\times(0.4)\times(0.5) = 0.057$$

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Score is proportional to P(neg|"movie actor")

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

"movie actor"

$0.057 \propto$ P(neg| "movie actor")

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Compare both scores against each other

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

“movie actor”

$0.057 \propto$ P(neg| “movie actor”)
$0.024 \propto$ P(pos| “movie actor”)

Natural Language Processing

- Classify based on highest score:

25

0.42	**0.08**	**0.33**	**0.17**
10	2	8	4
movie	actor	great	film

10

0.4	**0.5**	**0**	**0.1**
8	10	0	2
movie	actor	great	film

“movie actor”

$0.057 \propto$ P(neg| “movie actor”)
$0.024 \propto$ P(pos| “movie actor”)

Natural Language Processing

- This is classified as a **negative** review

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

“movie actor”

0.057 ∝ P(neg| “movie actor”)
0.024 ∝ P(pos| “movie actor”)

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- What about 0 count words?

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

“great movie”

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- What about 0 count words?

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

“great movie”

Natural Language Processing

- Probability is zero! Regardless of text!

25

0.42	0.08	0.33	0.17
10	2	8	4
movie	actor	great	film

“great movie”

$$P(\text{neg}) \times \boxed{P(\text{great}|\text{neg})} \times P(\text{movie}|\text{neg})$$

10

0.4	0.5	0	0.1
8	10	0	2
movie	actor	great	film

Natural Language Processing

- Alpha smoothing parameter to add counts:

25

10+1	2+1	8+1	4+1
movie	actor	great	film

"great movie"

P(neg)×P(great|neg)×P(movie|neg)

10

8+1	10+1	0+1	2+1
movie	actor	great	film

Natural Language Processing

- Recalculate conditional probabilities...

25

10+1	2+1	8+1	4+1
movie	actor	great	film

10

8+1	10+1	0+1	2+1
movie	actor	great	film

Natural Language Processing

- Note how a higher alpha value will be more “smoothing”, giving each word less distinct importance.
- Now let’s move on to focusing on feature extraction in general.
- Are there better ways than just simply word frequency counts to extract features from text?

Extracting Features From Text Data

Theory and Intuition

Natural Language Processing

- Most classic machine learning algorithms can’t take in raw text as data.
- Instead we need to perform a feature “extraction” from the raw text in order to pass numerical features to the machine learning algorithm.

Natural Language Processing

- Main Methods for Feature Extraction:
  - Count Vectorization
  - TF-IDF:
    - Term Frequency - Inverse Document Frequency

Natural Language Processing

- Count Vectorization

You are good
I feel good
I am good

Natural Language Processing

- Create a vocabulary of all possible words

You are good
I feel good
I am good

Natural Language Processing

- Create a vocabulary of all possible words

YOU	ARE	GOOD	I	FEEL	AM

Natural Language Processing

- Create a vector of frequency counts

	YOU	ARE	GOOD	I	FEEL	AM
You are good						
I feel good						
I am good						

Natural Language Processing

- Create a vector of frequency counts

	YOU	ARE	GOOD	I	FEEL	AM
You are good	1	1	1	0	0	0
I feel good	0	0	1	1	1	0
I am good	0	0	1	1	0	1

Natural Language Processing

- Notice most values will be zero!

	YOU	ARE	GOOD	I	FEEL	AM
You are good	1	1	1	0	0	0
I feel good	0	0	1	1	1	0
I am good	0	0	1	1	0	1

Natural Language Processing

- Count Vectorization

```
messages = ["Hey, lets go to the game today!",
            "Call your sister.",
            "Want to go walk your dogs?"]
```

Natural Language Processing

- Document Term Matrix (DTM)

call	dogs	game	go	hey	lets	sister	the	to	today	walk	want	your
0	0	1	1	1	1	0	1	1	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	1
0	1	0	1	0	0	0	0	1	0	1	1	1

Natural Language Processing

- Count Vectorization treats every word as a feature, with the frequency counts acting as a “strength” of the feature/word.
- For larger documents, matrices are stored as a **sparse matrix** to save space, since so many values will be zero.

Natural Language Processing

- Issues to consider:
  - Very common words (e.g. “a” , “the”, “about”).
  - Words common to a particular set of documents (e.g. “run” in a set of different sports articles).

Natural Language Processing

- Stop Words are words common enough throughout a language that its usually safe to remove them and not consider them as important.
- Most NLP libraries have a built-in list of common stop words.

Natural Language Processing

- We can address the issue of document frequency by using a TF-IDF Vectorization process.
- Instead of filling the DTM with word frequency counts it calculates term frequency-inverse document frequency value for each word(TF-IDF).

Natural Language Processing

- Term frequency **tf(t,d)**: is the raw count of a term in a document:
  - The number of times that term **t** occurs in document **d**.

Natural Language Processing

- However, Term Frequency alone isn’t enough for a thorough feature analysis of the text!
- Let’s imagine very common terms, like “a” or “the”...

Natural Language Processing

- Because the term "the" is so common, term frequency will tend to incorrectly emphasize documents which happen to use the word "the" more frequently, without giving enough weight to the more meaningful terms "red" and "dogs".

Natural Language Processing

- We also need to consider a group of documents where non stop words are common throughout all the documents:
    - The word “run” in documents about various sports.

Natural Language Processing

- An inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

Natural Language Processing

- It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient)

Natural Language Processing

- The IDF is how common or rare a word is in the entire document set.
- The closer it is to 0, the more common a word is.
- Calculated by taking the **total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.**

Natural Language Processing

- TF-IDF = term frequency * (1 / document frequency)
- TF-IDF = term frequency * inverse document freq

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Natural Language Processing

- Fortunately Scikit-learn can calculate all these terms for us through the use of its API.
- Notice how similar the syntax is to our previous use of ML models in Scikit-Learn!

Natural Language Processing

```
from sklearn.feature_extraction.text import TfidfVectorizer

vect = TfidfVectorizer()
dtm = vect.fit_transform(messages)
```

call	dogs	game	go	hey	lets	sister	the	to	today	walk	want	your
0.000	0.00	0.403	0.307	0.403	0.403	0.000	0.403	0.307	0.403	0.00	0.00	0.000
0.623	0.00	0.000	0.000	0.000	0.000	0.623	0.000	0.000	0.000	0.00	0.00	0.474
0.000	0.46	0.000	0.349	0.000	0.000	0.000	0.000	0.349	0.000	0.46	0.46	0.349

Natural Language Processing

- TF-IDF allows us to understand the context of words across an entire corpus of documents, instead of just its relative importance in a single document.
- Coming up next we'll explore how to perform these operations with Python and SciKit-Learn!

Extracting Features From Text Data

Understanding Core Concepts

Natural Language Processing

- Let’s begin understanding core concepts by manually creating a “bag of words” model.
- Recall that this is a frequency count of words in the documents.
- Let’s get started!

Extracting Features From Text Data

Utilizing Scikit-Learn

PIERIAN DATA

Classification with Text Data

Part One: Data Analysis and Features

PIERIAN DATA

Classification with Text Data

Part Two: Building Models

Text Classification Project Exercise

Solutions

PIERIAN DATA