COMP47750/COMP47990
Machine Learning with Python

Naïve Bayes Classifier

Pádraig Cunningham
original slides by Derek Greene

School of Computer Science

COMP47750/COMP47990
Machine Learning with Python

Naïve Bayes Classifier

Part I
Basics

Pádraig Cunningham
original slides by Derek Greene

School of Computer Science

UCD DUBLIN

Overview

- Probability-based Learning
- Bayes Theorem
- Naïve Bayes Classifier
- Examples & Exercises
- Handling Numeric Features
- Naïve Bayes in scikit learn

Thomas Bayes

- 1701 - 1761
- Presbyterian Minister - Kent, England
- Main paper published after his death

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

Fun fact: this is almost certainly not a picture of him.

Bayesian probability is the name given to several related interpretations of probability as an amount of epistemic confidence – the strength of beliefs, hypotheses etc. – rather than a frequency.

Conditional Probabilities

- Antibody Tests
  - $P(I)$ ~ 500/10,000 = 0.05 (5% infection rate)
  - $P(Pos|I)$ ~ 490/500 = 0.98
  - What is $P(I|Pos)$? - probability someone with a positive test is infected

	Infected	Not Infected	*Total*
Positive	490	190	***680***
Negative	10	9,310	***9,320***
Total	***500***	***9,500***	***10,000***

$$P(I \mid Pos) = \frac{P(Pos \mid I)P(I)}{P(Pos)}$$

$$P(Pos) = P(Pos|I) \text{ x } P(I) + P(Pos|NI) \text{ x } P(NI)$$
$$= 0.98 \text{ x } 0.05 + 0.02 \text{ x } 0.95$$
$$= 0.068$$

$$P(I \mid Pos) = 0.98 \text{ x } 0.05 / 0.068$$
$$= 0.72$$

28% False Positive rate

Probability-based Learning

- **Key Idea:** Use estimates of likelihoods to determine the most likely prediction which should be made for classification e.g. “email X is more likely to be spam than non-spam”.
- Revise these estimates based on the data we collect.
- Most common probabilistic approach for classification is the Naïve Bayes classifier, an eager learning approach based on Bayes Theorem.
- **Why use a Naïve Bayes classifier?**
  - Intuitive and easy to implement.
  - Fast to train and to use as a classifier.
  - Suitable for moderate or large data sets with many features.
  - Can deal with missing features.

Naive Bayes: Text Classification

06 NaiveBayesTC Notebook

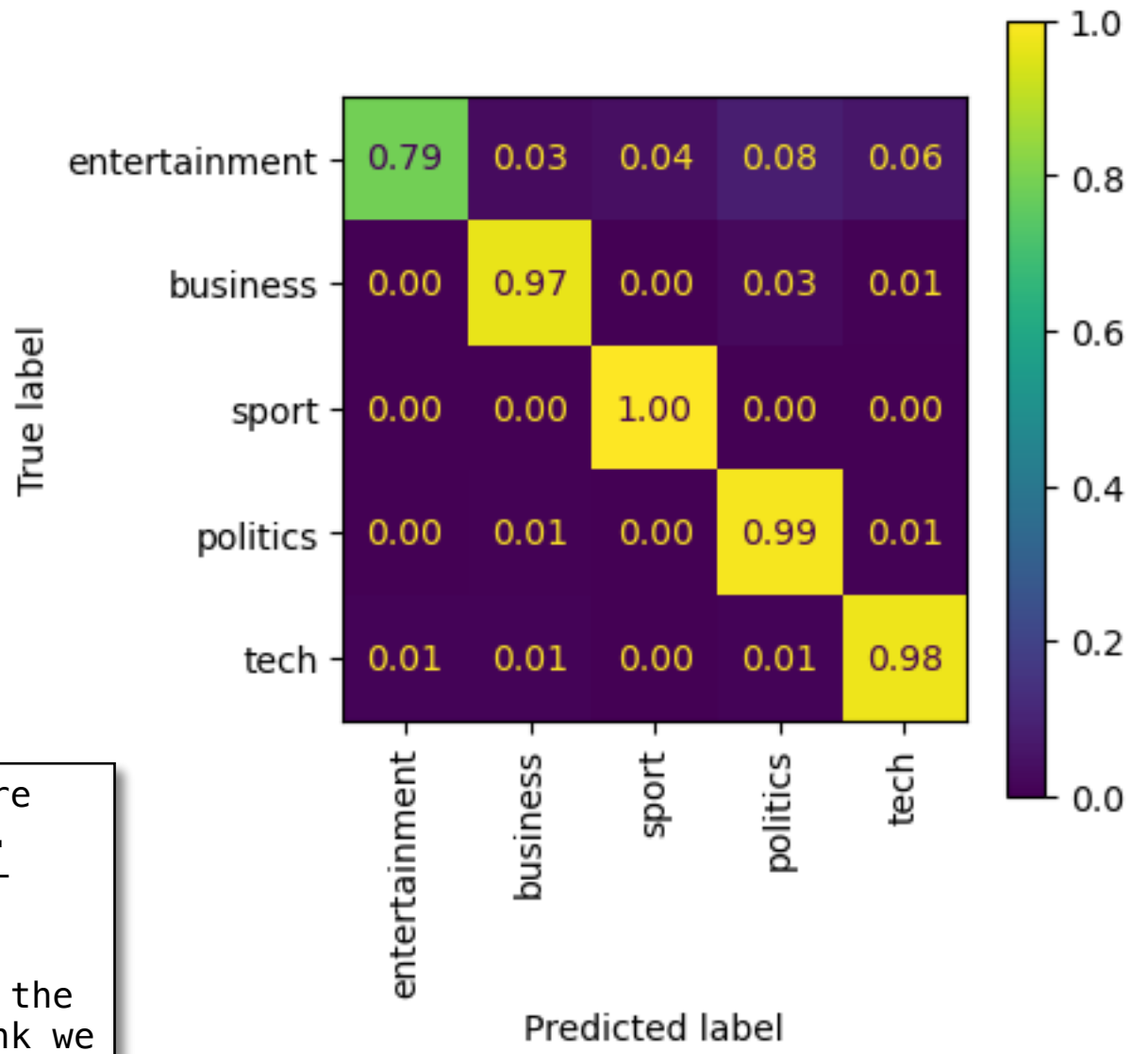
- BBC News text classification
  - Five Classes
    - http://derekgreene.com/bbc/

```
British Airways has blamed high fuel prices for a 40%
drop in profits.  Reporting its results for the three
months to 31 December 2004, the airline made a pre-tax
profit of £75m ($141m) compared with £125m a year
earlier. Rod Eddington, BA's chief executive, said the
results were "respectable" in a third quarter when fuel
costs rose by £106m or 47.3%. BA's profits were still
better than market expectation of £59m, and it expects a
rise in full-year revenues.  To help offset the
increased price of aviation fuel
```

```
Germany calls for EU reform  German Chancellor Gerhard
Schroeder has called for radical reform of the EU\'s
stability pact to grant countries more flexibility over
their budget deficits.  Mr Schroeder said existing
fiscal rules should be loosened to allow countries to
run deficits above the current 3% limit if they met
certa
```

```
Scotland manager Walter Smith says he wants to restore
the national team's respectability in world football.
Smith has joined his first squad for a three-day get-
together near Manchester in preference to playing a
friendly. While qualification for the 2006 World Cup
appears to be beyond Scotland, Smith is anxious that the
remainder of the campaign should be positive. "I think we
have got to try to get a bit of respectability back in
whatever way we can," he said. "We will have to approach
each game differently. Obviously we will have to approach
the Italian game ...
```

confusion matrix



94% accuracy

Probability Theory

$P(X)$	Probability of event *X* happening.
$P(X\|Y)$	Conditional probability of event *X* happening, given that event *Y* has happened.

What is the probability of a given hypothesis *h* being true ("the event"), given the observed data *D* ("the evidence")?

Let h denote the hypothesis, D denote the data.

Prior probability of data

$P(D)$: Probability of the data D.

Prior probability of hypothesis - "initial beliefs"

$P(h)$: Probability of the hypothesis h.

Posterior probability

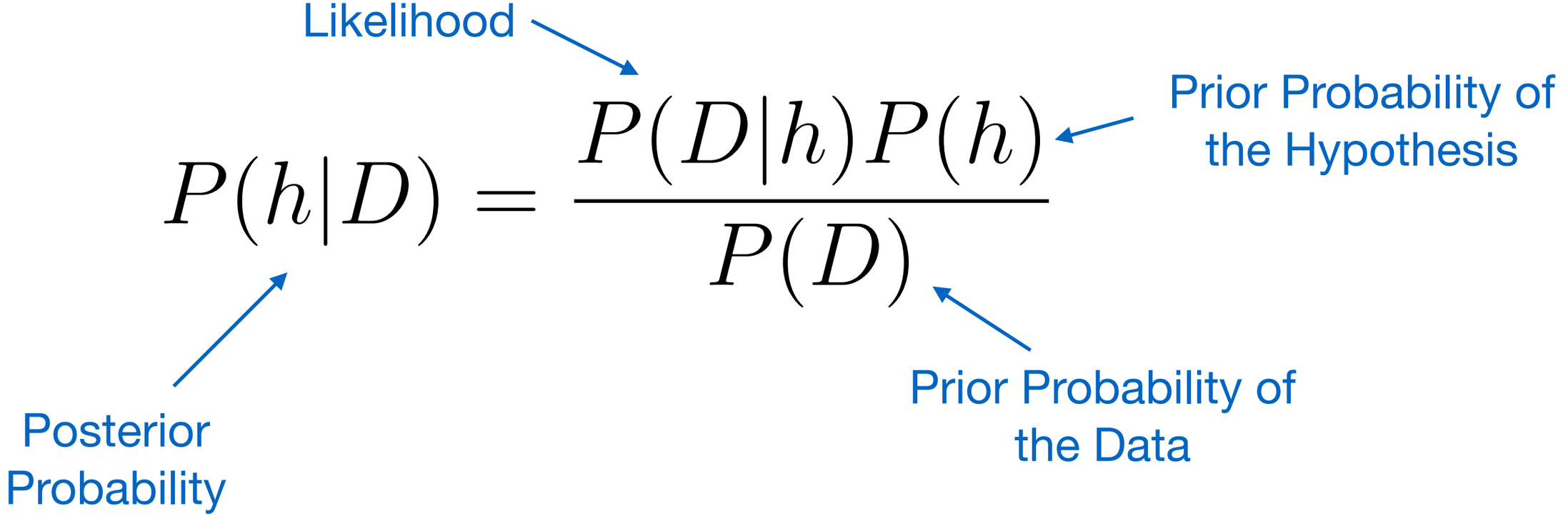
$P(h|D)$: Probability of the hypothesis h given the data D.

Bayes Theorem

"The probability that an event has happened given a set of evidence for it is equal to the probability of the evidence being caused by the event by the probability of the event itself."

- Kelleher et al, 2015

Bayes Theorem: Rule states that for each possible hypothesis *h*



Example: Bayes Theorem

***D*:** Helen is 28 years old, is on a bill-pay plan, and earns €40k.

***h*:** Helen will buy a new iPhone

$P(h\|D)$ Posterior Probability of *h*	Probability that Helen will buy a new iPhone, given that we know her age, plan, and income.
$P(h)$ Prior Probability of *h*	Probability that Helen will buy a new iPhone regardless of age, plan, and income
$P(D\|h)$ Posterior Probability of *D*	Probability that Helen is 28 years old, is on a bill-pay plan, and earns €40k, given that she has bought the iPhone 8.
$P(D)$ Prior Probability of *D*	Probability that a person from our dataset of customers is 28 years old, is on a bill-pay plan, and earns €40k.

We can calculate the Posterior Probability of *h* using Bayes Theorem:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Bayes Classification

- In classification, the posterior probability can be interpreted as: "What is the probability that a particular example x belongs to class *A*, given its observed feature values?"

$$P(h \mid D) : \text{Probability of the hypothesis } h \text{ given the data } D$$

The hypothesis h: Does example x belong to class *A*?

The data D: The set of features values that describe x.

- For classification, each h corresponds to a possible class *A*.
- The prior probabilities $P(h)$, also called class priors, describe the probability of encountering a particular class
- If we knew $P(h|D)$ we could classify the data perfectly.
- Since we generally do not know $P(h|D)$, we try to estimate it from the training data using Bayes Theorem.

Bayes Classification

- We usually want to find the most likely hypothesis for our data.
- Formally, we are looking for the Maximum Aposteriori Hypothesis (MAP):

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$

- Example: Two competing hypotheses h_0 and h_1 for dataset X

$$P(h_0|X) > P(h_1|X) \implies \text{choose } h_0$$

$$P(h_1|X) > P(h_0|X) \implies \text{choose } h_1$$

$$P(h_1|X) = P(h_0|X) \implies \text{choose either}$$

- In classification, we want to find the most likely class label for a given example among all possible class labels.

Definition: Bayes Classifier

Classifier Inputs:

A set of labels $H = \{h_1, h_2, \ldots\}$
A set of examples $X = \{x_1, x_2, \ldots\}$ each represented by features $\{f_1, f_2, \ldots, f_n\}$

Classifier Objective:

Find the most probable class label h for x according to:

$$h_{MAP} = \underset{h_j \in H}{\operatorname{argmax}} P(h_j \mid f_1, f_2, \ldots, f_n)$$

$$h_{MAP} = \underset{h_j \in H}{\operatorname{argmax}} \frac{P(f_1, f_2, \ldots, f_n \mid h_j) P(h_j)}{P(f_1, f_2, \ldots, fn)}$$

$$h_{MAP} = \underset{h_j \in H}{\operatorname{argmax}} P(f_1, f_2, \ldots, f_n \mid h_j) P(h_j)$$

Problem: Difficult to estimate $P(f_1, f_2, \ldots, f_n \mid h_j)$

Naïve Bayes Classifier

- **Key Idea:** Apply Bayes Theorem with the "naïve" assumption that all features in the data are conditionally independent:

$$P(f_1, f_2, \ldots, f_n \mid h_j) = \prod_i P(f_i \mid h_j)$$

i.e. the value of a particular feature is unrelated to the presence or absence of any other feature, given the class label h_j

- Based on this assumption, the objective of the Naïve Bayes classifier becomes:

Find the most probable class label h for x according to:

$$h_{NB} = \underset{h_j \in H}{\operatorname{argmax}}\ P(h_j) \prod_i P(f_i \mid h_j)$$

i.e. (Class Probability) x (Product of Class-Feature Probabilities)

Naïve Bayes Classifier

- The Naïve Bayes classifier is an eager learner, which builds a model in advance. The algorithm operates as follows:
  - **Training Phase:**
    - Estimate the probabilities $P(h_j)$ and $P(f_i \mid h_j)$ based on their frequencies in the training set.
  - **Output after training:**
    - The model, in the form of the probability estimates.
  - **Classification Phase:**
    - Classify new examples using the probability estimates and the Naïve Bayes formula:

$$h_{NB} = \underset{h_j \in H}{\operatorname{argmax}} \; P(h_j) \prod_i P(f_i \mid h_j)$$

i.e. (Class Probability) x (Product of Class-Feature Probabilities)

Example: Naïve Bayes Classifier

Q. "Will we go swimming today?"
Binary classification task (*Swimming = {Yes,No}*), with examples described by 5 categorical weather features:

Example	Rain Recently (RR)	Rain Today (RT)	Temp (T)	Wind (W)	Sunshine (S)	Swimming
1	Moderate	Moderate	Warm	Light	Some	Yes
2	Light	Moderate	Warm	Moderate	None	No
3	Moderate	Moderate	Cold	Gale	None	No
4	Moderate	Moderate	Warm	Light	None	Yes
5	Moderate	Light	Cold	Light	Some	No
6	Heavy	Light	Cold	Moderate	Some	Yes
7	Light	Light	Cold	Moderate	Some	No
8	Moderate	Moderate	Cold	Gale	Some	No
9	Heavy	Heavy	Warm	Moderate	None	Yes
10	Light	Light	Cold	Light	Some	No
X0	Moderate	Moderate	Cold	Light	Some	???

➡ How can we use a Naïve Bayes Classifier to predict for *X0*?

Example: Naïve Bayes Classifier

- How do we build a Naïve Bayes model on this training set?
- We need to estimate the probabilities $P(h_j)$ and $P(f_i \mid h_j)$ based on their frequencies in the training set.

Step 1: Calculate class prior probabilities:

- What are the probabilities of *Swimming = Yes / No* in the training set?

Step 2: Calculate conditional probabilities for each feature:

- What is the probability of each distinct feature value occurring, for the case *Swimming= Yes / No* in the training set?

$$P(f_i = t \mid h_j) = \frac{N_{tij}}{N_j}$$

Probability that feature i has value t when the class is j

N_j *number of samples with class j :*

N_{tij} *number of those where feature i has value t*

See correction in scikit-learn for zero counts <link>

Example: Naïve Bayes Classifier

- In practice, the key step in building a Naïve Bayes model is to calculate a contingency table (probability table), which has the class prior probabilities and conditional probabilities.

  e.g. If we look at the 4 training examples for *Swimming=Yes*:

Example	Rain Recently (RR)	Rain Today (RT)	Temp (T)	Wind (W)	Sunshine (S)	Swimming
1	Moderate	Moderate	Warm	Light	Some	Yes
2	Light	Moderate	Warm	Moderate	None	No
3	Moderate	Moderate	Cold	Gale	None	No
4	Moderate	Moderate	Warm	Light	None	Yes
5	Moderate	Light	Cold	Light	Some	No
6	Heavy	Light	Cold	Moderate	Some	Yes
7	Light	Light	Cold	Moderate	Some	No
8	Moderate	Moderate	Cold	Gale	Some	No
9	Heavy	Heavy	Warm	Moderate	None	Yes
10	Light	Light	Cold	Light	Some	No

Class Probability

P(Yes) = 4/10

Feature: Rain Recently

P(L_RR|Yes) = 0/4

P(M_RR|Yes) = 2/4

P(H_RR|Yes) = 2/4

Feature: Rain Today

P(L_RT|Yes) = 1/4

P(M_RT|Yes) = 2/4

P(H_RT|Yes) = 1/4

...

Example: Naïve Bayes Classifier

Construct full contingency table for all features on both classes:

Swimming	Yes	No
Rain Recently=light	0/4	3/6
Rain Recently=moderate	2/4	3/6
Rain Recently=heavy	2/4	0/6
Rain Today=light	1/4	3/6
Rain Today=moderate	2/4	3/6
Rain Today=heavy	1/4	0/6
Temp=Cold	1/4	5/6
Temp=Warm	3/4	1/6
Wind=Light	2/4	2/6
Wind=Moderate	2/4	2/6
Wind=Gale	0/4	2/6
Sunshine=Some	2/4	4/6
Sunshine=None	2/4	2/6
Class Probabilities (Priors)	4/10	6/10

Example: Naïve Bayes Classifier

Test a new input example for hypothesis 1: *Swimming=Yes*

Example	Rain Recently (RR)	Rain Today (RT)	Temp (T)	Wind (W)	Sunshine (S)	Swimming
X1	Moderate	Moderate	Cold	Moderate	Some	???

(Product of Class-Feature Probabilities) x (Class Probability)

P(Yes) = (2/4 x 2/4 x 1/4 x 2/4 x 2/4) x 4/10

P(Yes) = 0.00625

Test the input example for hypothesis 2: *Swimming=No*

P(No) = (3/6 x 3/6 x 5/6 x 2/6 x 4/6) x 6/10

P(No) = 0.028

We usually normalise probabilities to sum to 1:

$$P(Yes)' = \frac{0.00625}{0.00625+0.028} = 0.18$$

$$P(No)' = \frac{0.028}{0.00625+0.028} = 0.82$$

Output: *Swimming=No*

Example: Naïve Bayes Classifier

Test another input example for hypothesis 1: *Swimming=Yes*

Example	Rain Recently (RR)	Rain Today (RT)	Temp (T)	Wind (W)	Sunshine (S)	Swimming
X2	Moderate	Light	Warm	Light	None	???

(Product of Class-Feature Probabilities) x (Class Probability)

P(Yes) = (2/4 x 1/4 x 3/4 x 2/4 x 2/4) x 4/10

P(Yes) = 0.00938

Test the input example for hypothesis 2: *Swimming=No*

P(No) = (3/6 x 3/6 x 1/6 x 2/6 x 2/6) x 6/10

P(No) = 0.00278

Normalise probabilities to sum to 1:

$$P(Yes)' = \frac{0.00938}{0.00938 + 0.00278} = 0.7714$$

$$P(No)' = \frac{0.00278}{0.00938 + 0.00278} = 0.2286$$

Output: *Swimming=Yes*

COMP47750/COMP47990
Machine Learning with Python

Naïve Bayes Classifier

Part II
Numeric features & scikit-learn

Pádraig Cunningham
original slides by Derek Greene

School of Computer Science

Handling Numeric Features

- How to classify when features take numeric values?

Example	Rain Recently (RR)	Rain Today (RT)	Temp (T)	Wind (W)	Sunshine (S)	Swimming
X0	Moderate	Moderate	9	Light	Some	???

- **Option 1:** Discretise the feature to take fixed number of values.
  e.g. Temp = {cool, mild, hot}

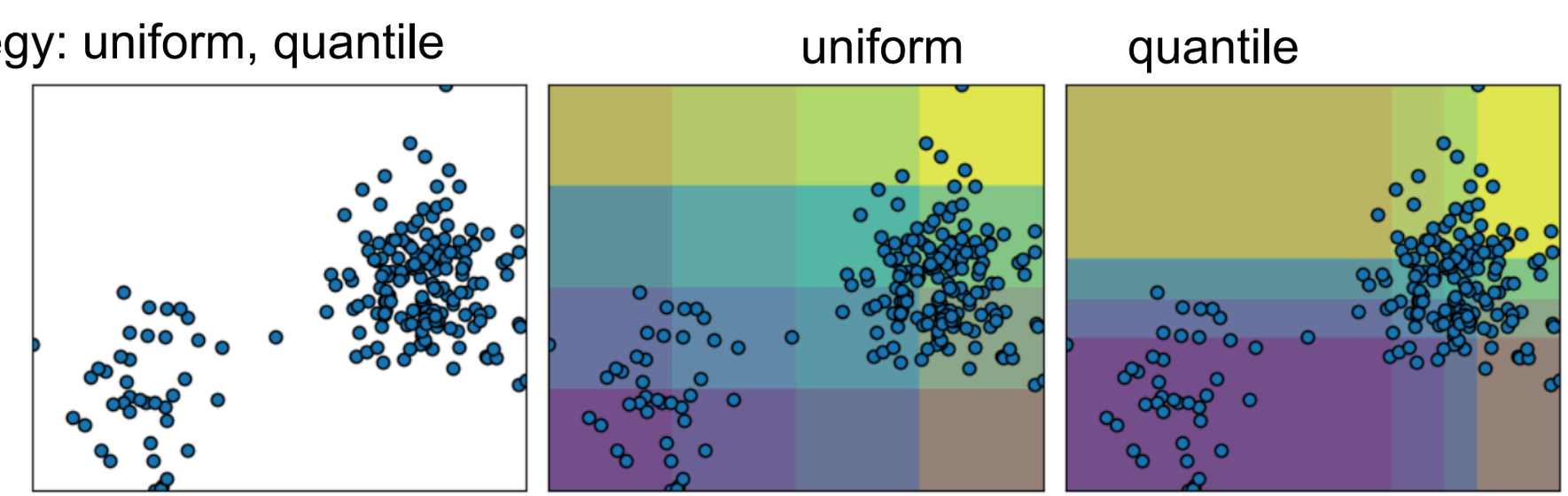
- **Option 2:** Assume that the feature fits to some distribution.
  e.g. for a Normal Distribution:

  1. For numeric feature f_i, store mean μ_i and standard deviation σ_i for each class v_j
  2. When classifying, find the probability that the feature value fits the distribution $N(\mu_i, \sigma_i^2)$

Handling Numeric Features

- Discretisation in scikit-learn
  - KBinsDiscretizer
    - Parameters:
      - n_bins
      - encode: onehot, ordinal
      - strategy: uniform, quantile

Note:
Order information is lost



Encode

```
X = [[-2, 1],
     [-1, 3],
     [ 0, 4],
     [ 2, 5]]
```

ordinal

```
[[0., 0.],
 [0., 1.],
 [1., 2.],
 [2., 2.]])
```

onehot

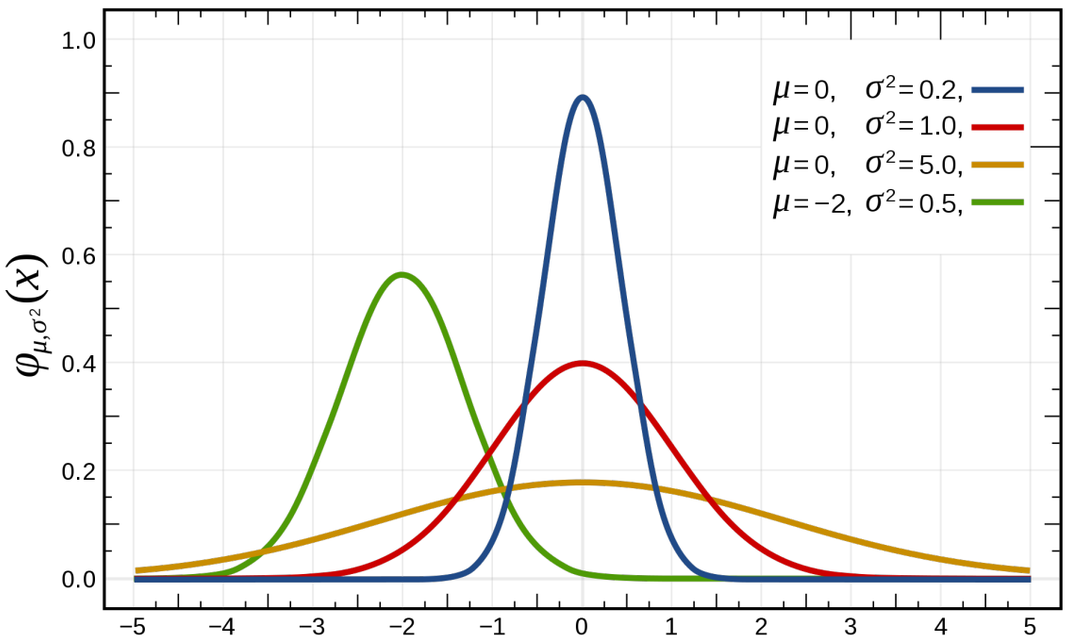
```
[[1., 0., 0., 1., 0., 0.],
 [1., 0., 0., 0., 1., 0.],
 [0., 1., 0., 0., 0., 1.],
 [0., 0., 1., 0., 0., 1.]])
```

Handling Numeric Features

- If data is Normal (Gaussian) the probability density function is

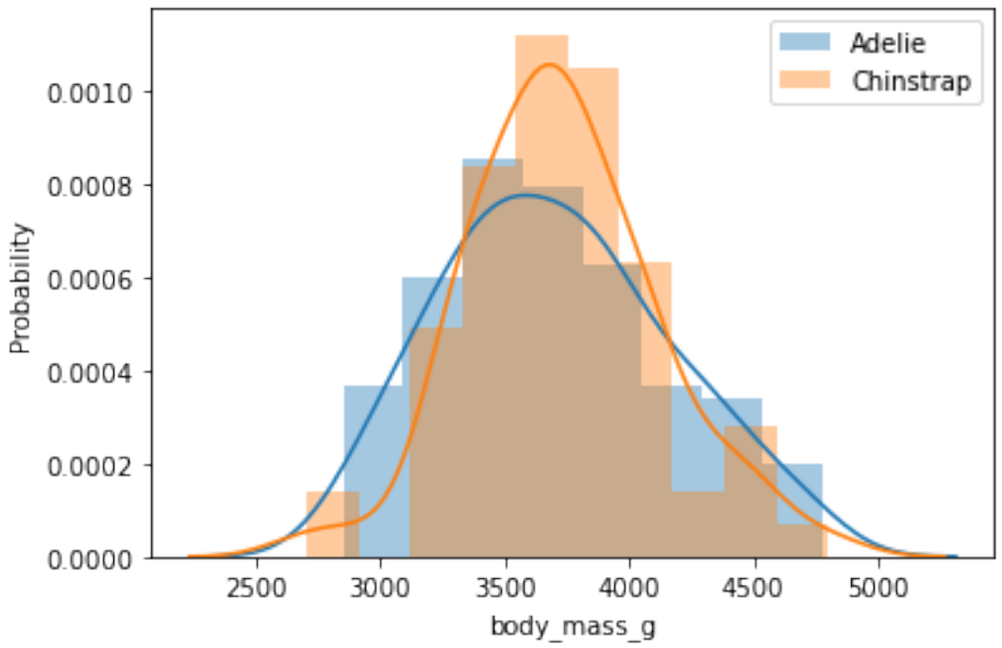
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

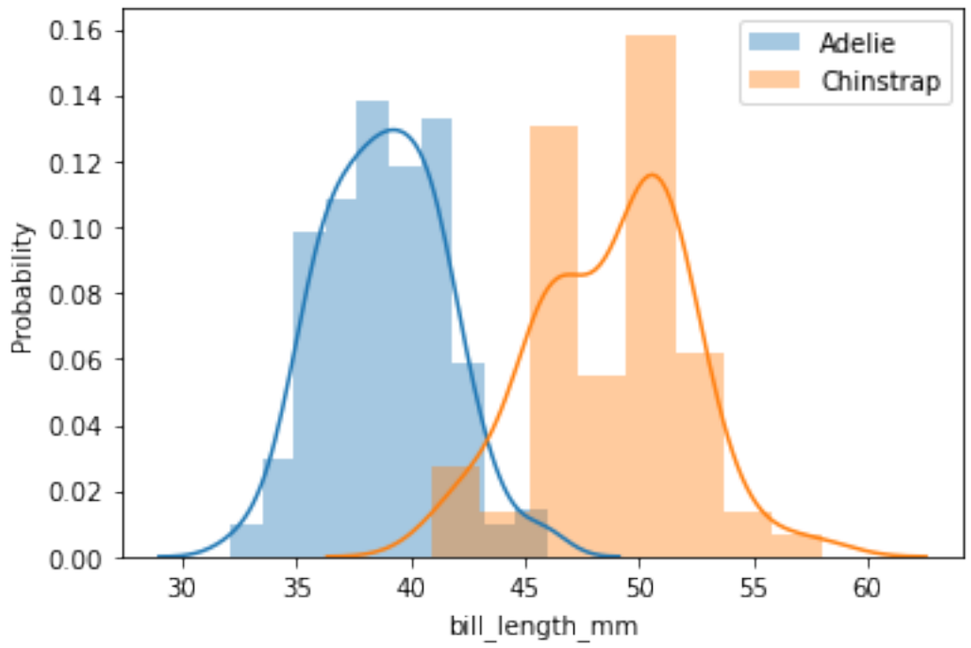
Used in sklearn to estimate conditional probabilities



https://en.wikipedia.org/wiki/Normal_distribution

See spreadsheet





Naive Bayes in scikit-learn

Notebook:
06 Naive Bayes

- Multiple implementations suitable for different types of data.
  - `CategoricalNB` will work with categorical data once it is processed using an `OrdinalEncoder`
  - `GaussianNB` assumes the numeric features have a Gaussian distribution
  - `BernoulliNB` binary data
  - `MultinomialNB` count data, e.g. word counts

	Rain_Recently	Rain_Today	Temp	Wind	Sunshine	Swimming
0	Moderate	Moderate	Warm	Light	Some	Yes
1	Light	Moderate	Warm	Moderate	Overcast	No
2	Moderate	Moderate	Cold	Gale	Overcast	No
3	Moderate	Moderate	Warm	Light	Overcast	Yes
4	Moderate	Light	Cold	Light	Some	No

Categorical Naive Bayes

- Use OrdinalEncoder to convert to numbers
- CategoricalNB will treat these as categories

```
import pandas as pd
from sklearn.preprocessing import OrdinalEncoder

swim = pd.read_csv('Swimming.csv')
y = swim.pop('Swimming').values

ord_encoder = OrdinalEncoder()
swimOE = ord_encoder.fit_transform(swim)
```

swim

	Rain_Recently	Rain_Today	Temp	Wind	Sunshine	Swimming
0	Moderate	Moderate	Warm	Light	Some	Yes
1	Light	Moderate	Warm	Moderate	Overcast	No
2	Moderate	Moderate	Cold	Gale	Overcast	No
3	Moderate	Moderate	Warm	Light	Overcast	Yes
4	Moderate	Light	Cold	Light	Some	No
5	...	...	...	...	...	...

swimOE

```
array([[2., 2., 1., 1., 1.],
       [1., 2., 1., 2., 0.],
       [2., 2., 0., 0., 0.],
       [2., 2., 1., 1., 0.],
       [2., 1., 0., 1., 1.],
       [0., 1., 0., 2., 1.],
       [1., 1., 0., 2., 1.],
       [2., 2., 0., 0., 1.],
       [0., 0., 1., 2., 0.],
       [1., 1., 0., 1., 1.]])
```

Train a Naive Bayes model

- Train and test on training data

```
catNB = CategoricalNB(fit_prior=True,alpha = 0.0001)
swim_catNB = catNB.fit(swimOE,y)
y_dash = swim_catNB.predict(swimOE)

confusion = confusion_matrix(y, y_dash)
print("Confusion matrix:\n{}".format(confusion))
```

```
Confusion matrix:
[[6 0]
 [0 4]]
```

Test with other data

- Set up a dataframe and then do an OrdinalTransform

Query Dataframe

```
squery = pd.DataFrame([["Moderate","Moderate","Warm","Light","Some"],
                       ["Moderate","Moderate","Cold","Moderate","Some"],
                       ["Moderate","Light","Warm","Light","Overcast"]
                      ], columns=swim.columns)
```

OrdinalEncoder Format

```
X_query = ord_encoder.transform(squery)
X_query, X_query.shape
Out[76]:
(array([[2., 2., 1., 1., 1.],
        [2., 2., 0., 2., 1.],
        [2., 1., 1., 1., 0.]]), (3, 5))
```

Predictions

```
In [77]:
y_query = swim_catNB.predict(X_query)
y_query
Out[77]:
array(['Yes', 'No', 'Yes'], dtype='<U3')
```

Probabilities

```
In [78]:
q_probs = swim_catNB.predict_proba(X_query)
q_probs
Out[78]:
array([[0.228592  , 0.771408  ],
       [0.81632203, 0.18367797],
       [0.22858759, 0.77141241]])
```

One-Hot Encoding

- One-Hot encode the other data

```
onehot_encoder = OneHotEncoder(sparse=False)
swimOH = onehot_encoder.fit_transform(swim)
swimOH
```

  - swimOH is a numpy array

```
array([[0., 0., 1., 0., 0., 1., 0., 1., 0., 1., 0., 0., 1.],
       [0., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 1., 0.],
       [0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 1., 0.],
       [0., 0., 1., 0., 0., 1., 0., 1., 0., 1., 0., 1., 0.],
       [0., 0., 1., 0., 1., 0., 1., 0., 0., 1., 0., 0., 1.],
       [1., 0., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 1.],
       [0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 1.],
       [0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 0., 1.],
       [1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 1., 0.],
       [0., 1., 0., 0., 1., 0., 1., 0., 0., 1., 0., 0., 1.]])
```

- Train and test on training data

```
bnb = BernoulliNB()
swim_numNB = bnb.fit(swimOH,y)
y_dash = swim_numNB.predict(swimOH)

confusion = confusion_matrix(y, y_dash)
print("Confusion matrix:\n{}".format(confusion))

Confusion matrix:
[[5 1]
 [1 3]]
```

Test with other data

- Set up a dataframe and then One-Hot

Query Dataframe

```
squery = pd.DataFrame([["Moderate","Moderate","Warm","Light","Some"],
                       ["Moderate","Moderate","Cold","Moderate","Some"],
                       ["Moderate","Light","Warm","Light","Overcast"]
                      ], columns=swim.columns)
```

OneHot Format

```
In [66]:
X_query = onehot_encoder.transform(squery)
X_query, X_query.shape
Out[66]:
(array([[0., 0., 1., 0., 0., 1., 0., 1., 0., 1., 0., 0., 1.],
        [0., 0., 1., 0., 0., 1., 1., 0., 0., 0., 1., 0., 1.],
        [0., 0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 1., 0.]]), (3, 13))
```

Predictions

```
In [67]:
y_query = swim_numNB.predict(X_query)
y_query
Out[67]:
array(['Yes', 'No', 'Yes'], dtype=uint8)
```

Probabilities

```
In [69]:
q_probs = swim_numNB.predict_proba(X_query)
q_probs
Out[69]:
array([[0.19324128, 0.80675872],
       [0.89608238, 0.10391762],
       [0.14709254, 0.85290746]])
```

Naïve Bayes Classifier

- **Recall:** We are applying Bayes Theorem based on the naïve assumption that all features are conditionally independent:

$$P(f_1, f_2, \ldots, f_n \mid h_j) = \prod_i P(f_i \mid h_j)$$

  i.e. the value of a particular feature is unrelated to the presence or absence of any other feature, given the class label h_j

- However, in certain domains, strong violations of the independence assumption can lead to poor performance by Naive Bayes classifiers.

➡ Always need to keep in mind the type of data and the type problem to be solved when choosing a classification model.

References

- J. D. Kelleher, B. Mac Namee, A. D'Arcy. "Fundamentals of Machine Learning for Predictive Data Analytics", 2015.
- T. Mitchell. "Machine Learning". McGraw-Hill, 1997. pp. 55–58.
- Lewis, D. D. "Naive (Bayes) at forty: The independence assumption in information retrieval". Proceedings of ECML 1998.
- A. Muller, S. Guido, Introduction to Machine Learning with Python, 2017