

# **LECTURE 8**

# **MACHINE LEARNING**

**DR. PRAPASSORN TANTIPHANWADI**

INDUSTRIAL ENGINEERING, FACULTY OF ENGINEERING AT KHAMPAENGSAEN

**DECEMBER 2565**

# CONTENT

- What is Machine Learning?
- Type of Machine Learning?
- Overfitting and Underfitting
- ML Algorithms
  - 1) Simple Linear Regression
  - 2) Multiple Linear Regression
  - 3) Logistic Regression
  - 4) Decision Tree
  - 5) Random Forest
  - 6) Support Vector Machine (SVM)
  - 7) Naïive Bayes
  - 8) K-NN
  - 9) PCA
  - 10) K-Mean Clustering

# SUPPORT VECTOR MACHINE (SVM)

Support Vector Machine (SVM) เป็นอัลกอริทึมหนึ่งของ Machine Learning ที่จัดอยู่ในประเภท Supervised Learning

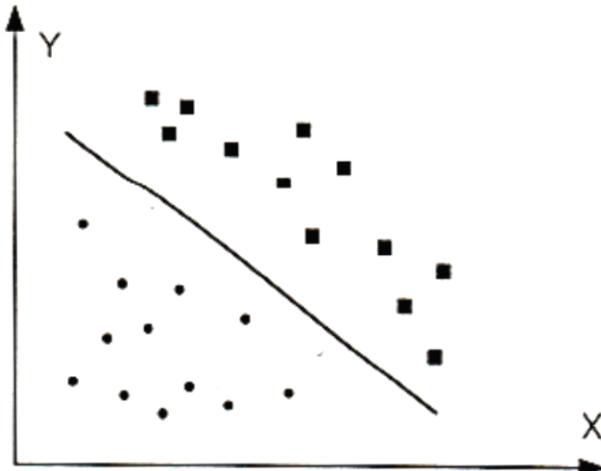
อัลกอริทึมนี้เป็นที่นิยมนำมาใช้งานอย่างกว้างขวาง เพราะมีความแม่นยำในการทำนายสูง เมื่อเทียบกับ Logistic Regression และยังทำนายผลได้เร็วกว่าอัลกอริทึมแบบ Naïve Bayes Classification ที่จะกล่าวถึงในหัวข้อต่อไปด้วย

อีกทั้งอัลกอริทึมนี้ยังสามารถทำนายข้อมูลได้ทั้งแบบ Classification และ Regression ด้วย โดยถ้าเป็นการทำนายข้อมูลแบบ Regression จะเรียกว่า Support Vector Regression (SVR) สำหรับ หนังสือเล่มนี้จะกล่าวถึงเฉพาะ SVM เท่านั้น

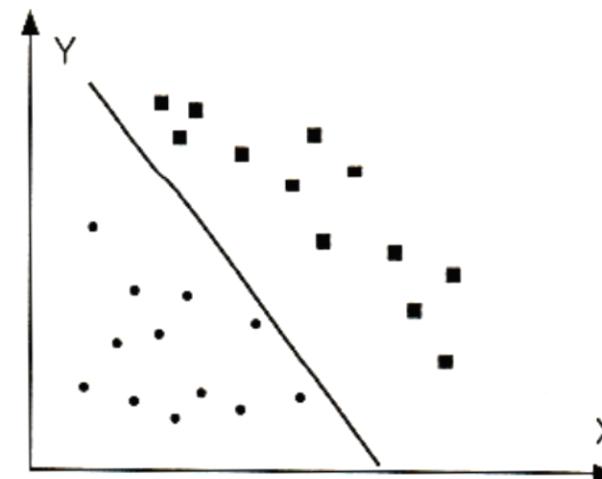
ทั้งนี้ SVM ก็มีข้อเสียตรงที่ไม่เหมาะสมที่จะนำมาใช้กับ Dataset ขนาดใหญ่ เพราะต้องใช้เวลาในการสอนคอมพิวเตอร์

ตัวอย่างของการนำอัลกอริทึม SVM ไปใช้งาน เช่น การตรวจจับและตรวจสอบใบหน้า (Face Recognition), การแบ่งกลุ่มประเภทของยีน (Gene), การรับรู้และจดจำลายมือเขียน (Handwriting recognition) เป็นต้น

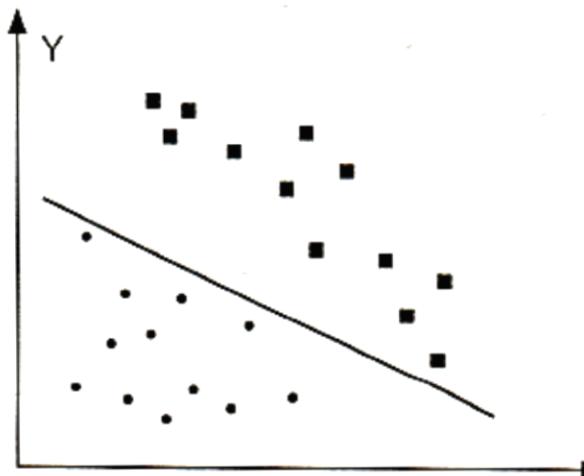
# SUPPORT VECTOR MACHINE (SVM)



โมเดลที่ 1



โมเดลที่ 2



โมเดลที่ 3

# SUPPORT VECTOR MACHINE (SVM)

จากรูป จะเห็นว่าการลากเส้น Hyperplane ทำได้หลากหลายวิธีมาก แล้วเส้นไหนคือเส้นที่ดีที่สุด สามารถแบ่งกลุ่มข้อมูลได้ดีที่สุด ตรงนี้แหละค่ะคือสิ่งที่อัลกอริทึม SVM จะช่วยเรา โดย SVM จะพยายามหาเส้น Hyperplane ที่ดีที่สุดออกมาให้

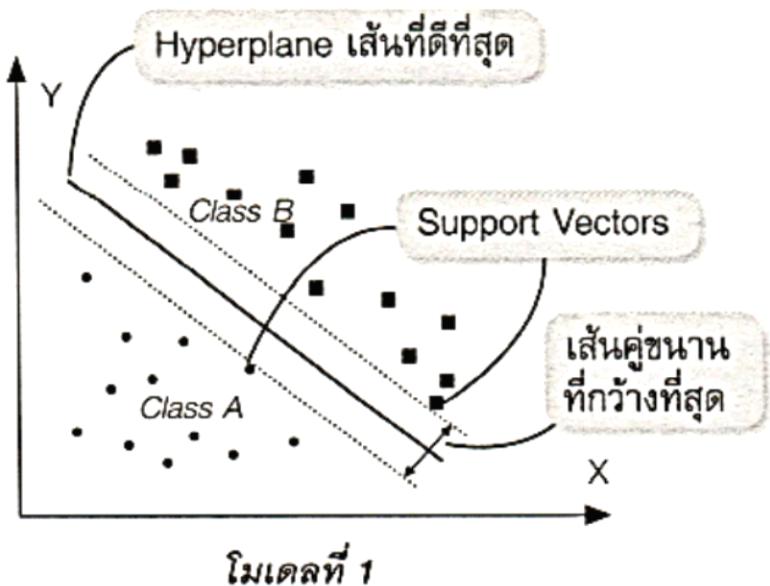
หลักการคือ SVM จะลากเส้นคู่ขนานขึ้นมา 2 เส้นเพื่อประกอบเส้น Hyperplane ซึ่งทำได้ 2 วิธี คือ

- **Hard Margin** คือ จะไม่ยอมให้มีจุดข้อมูลใด ๆ อยู่บนเส้นคู่ขนานเลย ตรงนี้ถ้ามีข้อมูลบางส่วนที่ผิดเพี้ยนกระจายตัวออกจากกลุ่มข้อมูล แต่เราไม่สามารถลากเส้นผ่านจุดข้อมูลใด ๆ ได้ ก็จะทำให้เส้นคู่ขนานแคบลง ซึ่งจะมีผลให้การแบ่งข้อมูลผิดพลาดแน่นอน
- **Soft Margin** คือ ยอมให้มีจุดข้อมูลอยู่บนเส้นคู่ขนานได้บ้าง ทั้งนี้เพื่อไม่ให้เส้นคู่ขนานแคบจนเกินไป ซึ่งก็จะลดความผิดพลาดในการแบ่งข้อมูลได้ดีกว่าแบบ Hard Margin

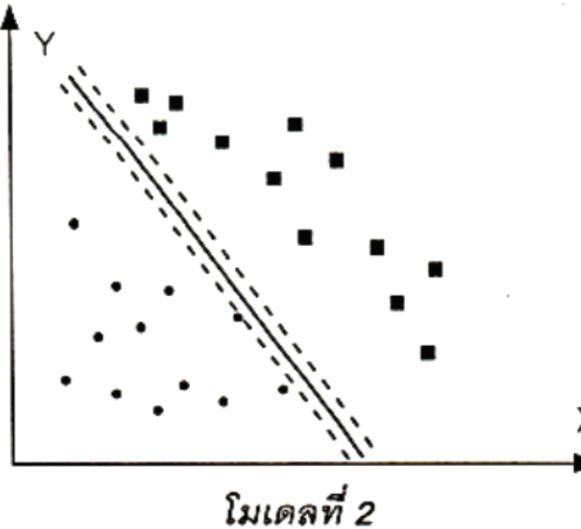
## Note

เราจะเรียกจุดข้อมูลที่อยู่บนเส้นคู่ขนานว่า Support Vectors

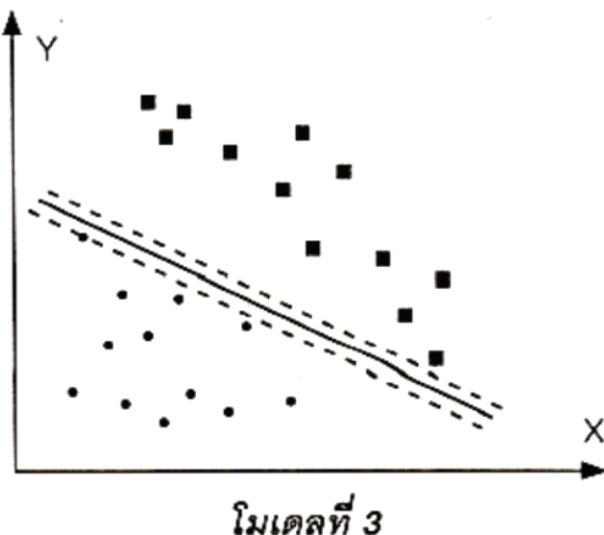
# SUPPORT VECTOR MACHINE (SVM)



โมเดลที่ 1



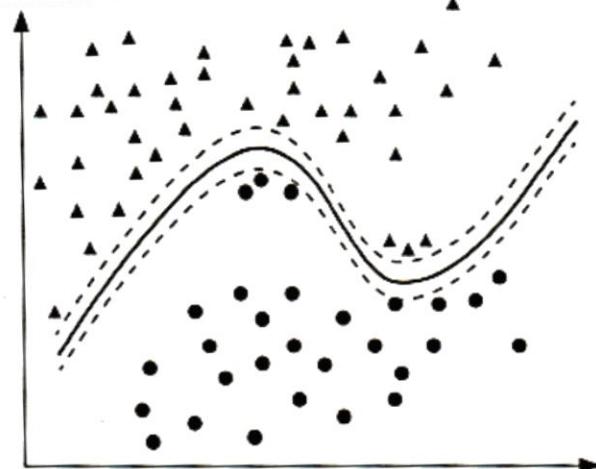
โมเดลที่ 2



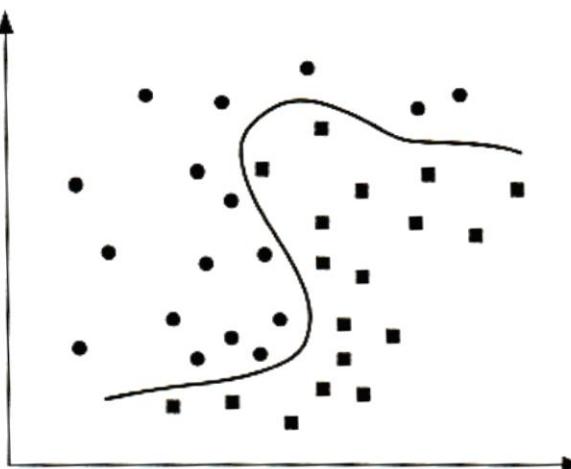
โมเดลที่ 3

การพิจารณาว่าเส้น Hyperplane ที่ดีที่สุดคือเส้นใด มีหลักการ คือ ถ้าเส้นคู่ขนานได้กว้างที่สุด มีระยะห่างระหว่างข้อมูลของแต่ละกลุ่มหรือคลาสสูงที่สุดแล้ว จะถือว่าเส้น Hyperplane ที่อยู่ตรงกลางระหว่างเส้นคู่ขนานนั้นคือเส้น Hyperplane ที่ดีที่สุด หากปัด้านล่าง โมเดลที่ 1 จะได้เส้น Hyperplane ที่ดีที่สุดอ ก มาก

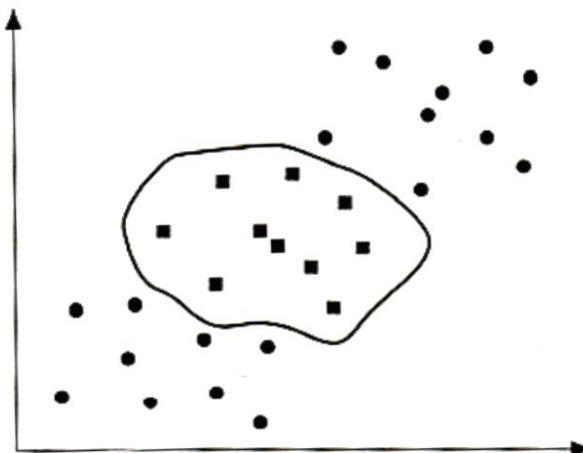
# SUPPORT VECTOR MACHINE (SVM)



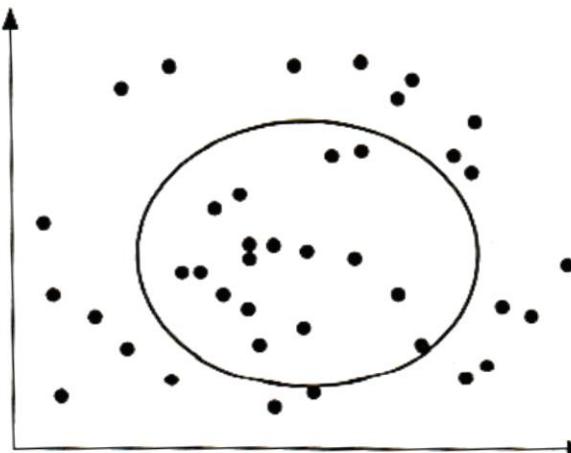
เรียกว่า *Polynomial Kernel*



อย่างไรก็ตี การลากเส้นแบ่งข้อมูลใน  
บางครั้งอาจไม่สามารถลากเป็นเส้น  
ตรงเพื่อแบ่งจุดข้อมูลได้  
โดยเส้นแบ่งข้อมูลอาจอยู่ในรูป<sup>ๆ</sup>  
แบบต่างๆ ดังรูปด้านล่าง



เรียกว่า *Radial Basis Function Kernel* เรียกด้วยๆ ว่า *RBF Kernel*



# SUPPORT VECTOR MACHINE (SVM)

ในการเขียนโปรแกรมเพื่อสร้างโมเดลด้วย SVM นั้นจึงจำเป็นต้องกำหนดว่าจะใช้ kernel ประเภทใดในอัลกอริทึม เพราะ kernel แต่ละประเภทมีอัลกอริทึมการทำงานที่ต่างกัน โดยสามารถกำหนด kernel ได้ดังนี้

Linear SVM

กำหนด kernel = "linear"

Polynomial

กำหนด kernel = "poly"

Radial Basis Function

กำหนด kernel = "rbf"

ถ้าเราไม่กำหนดค่า kernel แล้ว kernel จะมีค่าเป็น rbf โดยอัตโนมัติ ซึ่งหากผู้อ่านไม่ทราบว่า  
เราจะต้องเลือก kernel แบบใด ก็จะต้องลองนำจุดข้อมูลมาพิจารณาด้วย scatter plots ก่อน เพื่อ<sup>เพื่อ</sup>  
ตรวจสอบว่าข้อมูลนั้นมีการกระจายตัวแบบใด

# SUPPORT VECTOR MACHINE (SVM)

## Example 1

ตัวอย่างนี้จะเป็นการวิเคราะห์คำขออนุมัติเงินกู้จากลูกค้าว่าเราจะอนุมัติให้กู้เงินหรือไม่ สมมติเรามีตัวอย่างข้อมูลดังนี้

| รายได้ต่อเดือน | สถานะเครดิตบูโร | อายุ                      | ยอดเงินกู้ | ผลอนุมัติเงินกู้ |
|----------------|-----------------|---------------------------|------------|------------------|
| 0              | 20000           | ปกติ                      | 25         | 1000000          |
| 1              | 55000           | ปิดบัญชีไม่มีหนี้ค้าง     | 45         | 2000000          |
| 2              | 18000           | พกชำระหนี้                | 25         | 850000           |
| 3              | 22000           | มีหนี้ค้างชำระเกิน 90 วัน | 27         | 1220000          |
| 4              | 18500           | อภัยในกระบวนการทางกฎหมาย  | 25         | 870000           |

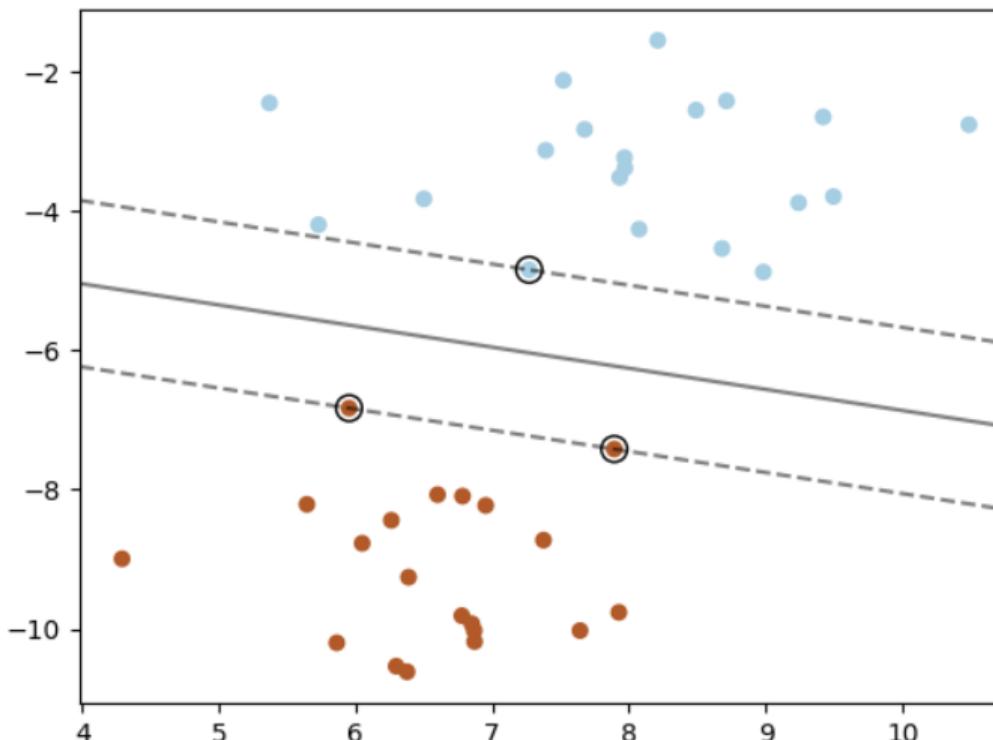
ข้อมูลที่นำมาเป็น Feature สำหรับการสร้างโมเดลประกอบไปด้วย รายได้ต่อเดือน, สถานะเครดิตบูโร, อายุ, และยอดเงินกู้ โดยมีคอลัมน์ ผลอนุมัติเงินกู้ เป็น Target

# SUPPORT VECTOR MACHINE (SVM)

<https://scikit-learn.org/stable/modules/svm.html#>

## SVM: Maximum margin separating hyperplane

Plot the maximum margin separating hyperplane within a two-class separable dataset using a Support Vector Machine classifier with linear kernel.

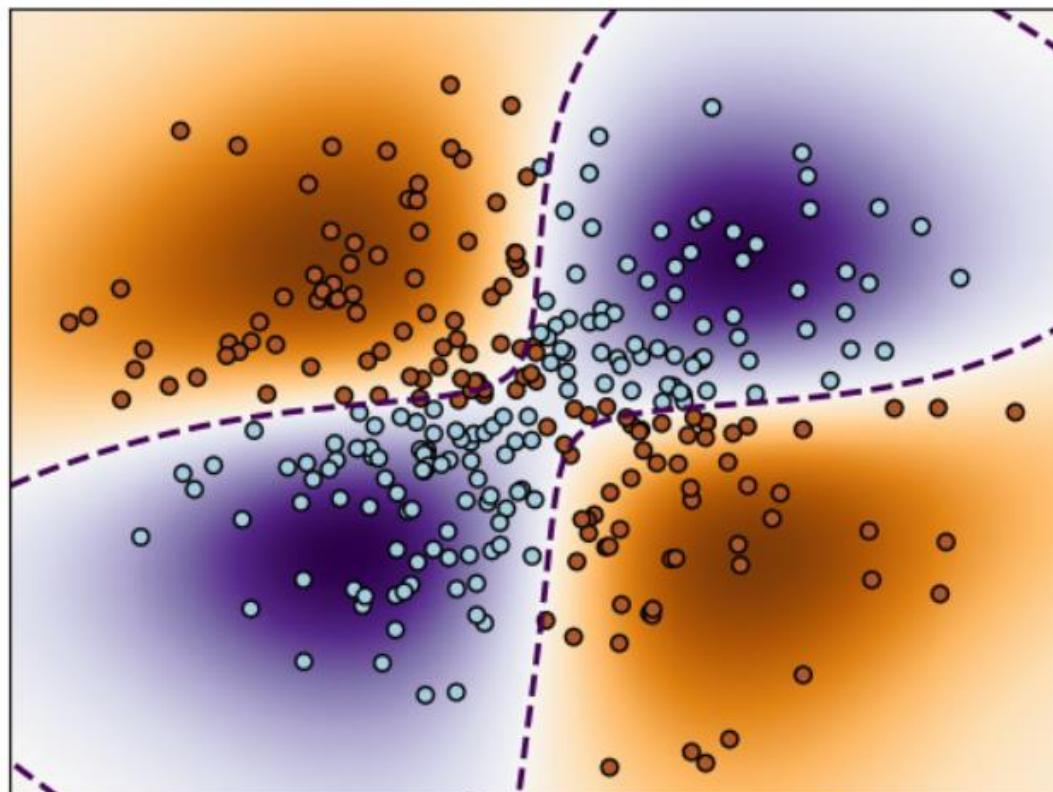


# SUPPORT VECTOR MACHINE (SVM)

## Non-linear SVM

Perform binary classification using non-linear SVC with RBF kernel. The target to predict is a XOR of the inputs.

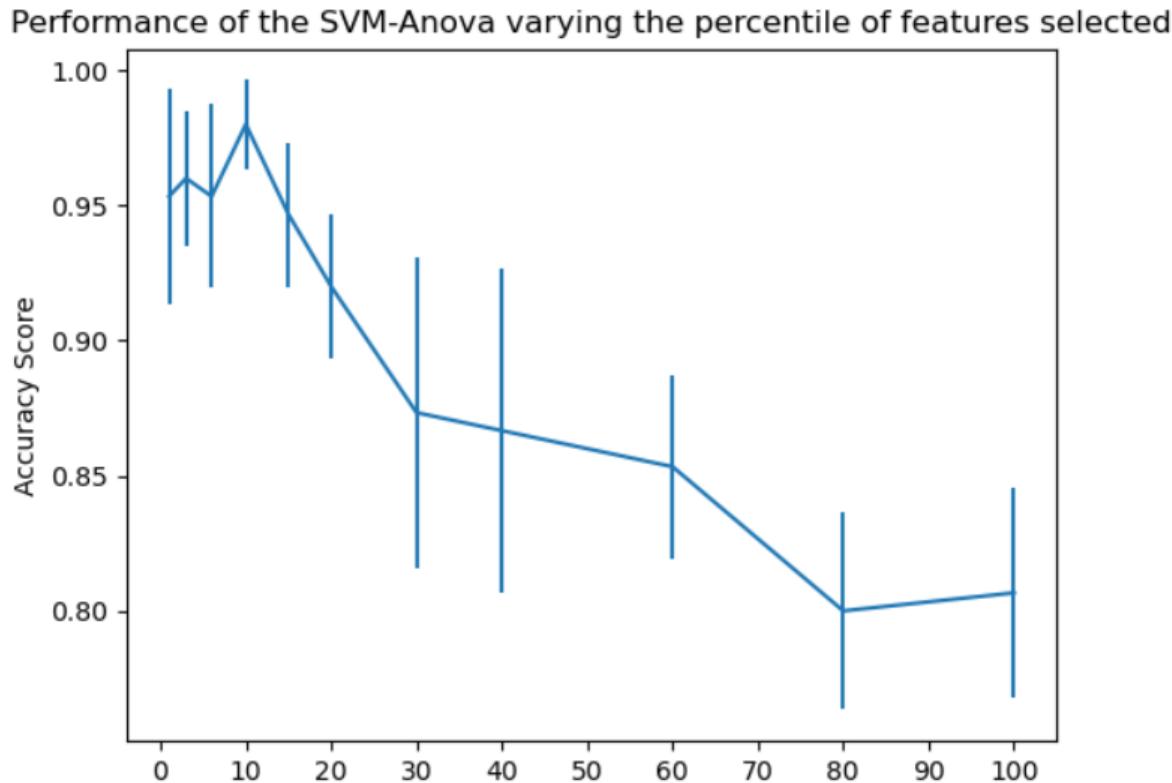
The color map illustrates the decision function learned by the SVC.



# SUPPORT VECTOR MACHINE (SVM)

## SVM-Anova: SVM with univariate feature selection

This example shows how to perform univariate feature selection before running a SVC (support vector classifier) to improve the classification scores. We use the iris dataset (4 features) and add 36 non-informative features. We can find that our model achieves best performance when we select around 10% of features.



# SUPPORT VECTOR MACHINE (SVM)

## # Example: Face Recognition

As an example of support vector machines in action, let's take a look at the facial recognition problem. We will use the Labeled Faces in the Wild dataset, which consists of several thousand collated photos of various public figures. A fetcher for the dataset is built into Scikit-Learn:



# SUPPORT VECTOR MACHINE (SVM)

## Multi-class classification

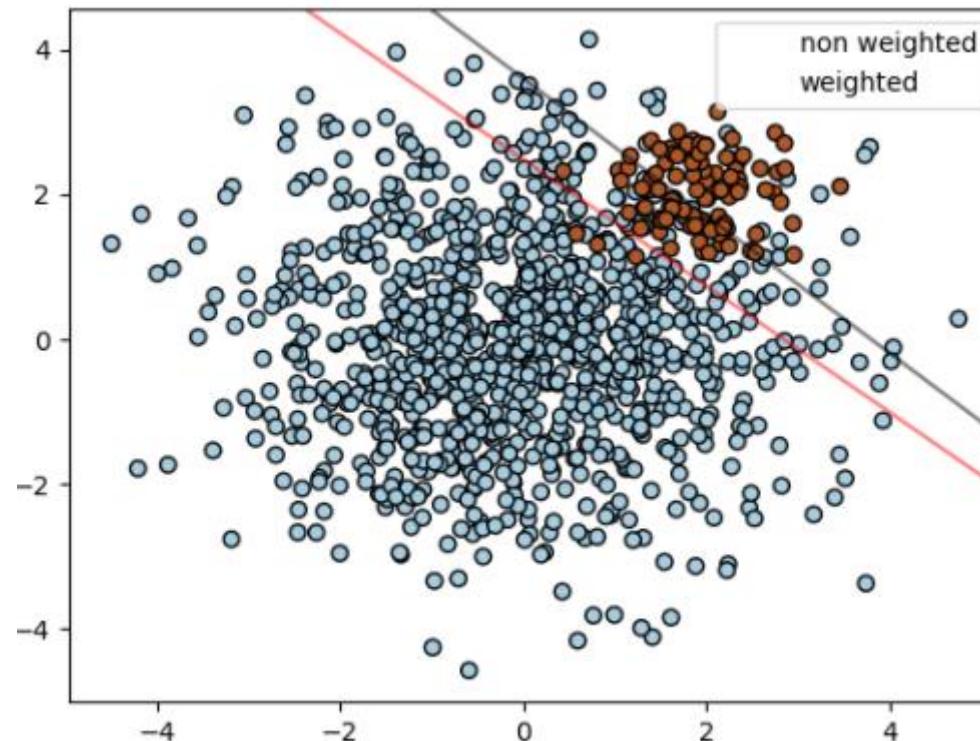
Plot different SVM classifiers in the iris dataset

|                  |                  |                                    |
|------------------|------------------|------------------------------------|
| $\alpha_{0,1}^0$ | $\alpha_{0,2}^0$ | Coefficients<br>for SVs of class 0 |
| $\alpha_{0,1}^1$ | $\alpha_{0,2}^1$ |                                    |
| $\alpha_{0,1}^2$ | $\alpha_{0,2}^2$ |                                    |
| $\alpha_{1,0}^0$ | $\alpha_{1,2}^0$ | Coefficients<br>for SVs of class 1 |
| $\alpha_{1,0}^1$ | $\alpha_{1,2}^1$ |                                    |
| $\alpha_{1,0}^2$ |                  |                                    |
| $\alpha_{2,0}^0$ | $\alpha_{2,1}^0$ | Coefficients<br>for SVs of class 2 |
| $\alpha_{2,0}^1$ | $\alpha_{2,1}^1$ |                                    |

# SUPPORT VECTOR MACHINE (SVM)

## Example 2: Unbalanced data

In problems where it is desired to give more importance to certain classes or certain individual samples, the parameters `class_weight` and `sample_weight` can be used.

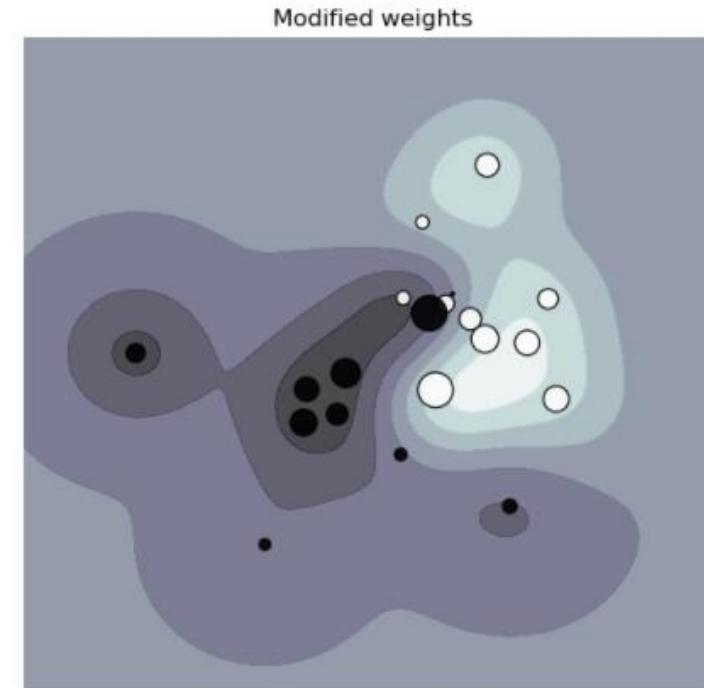
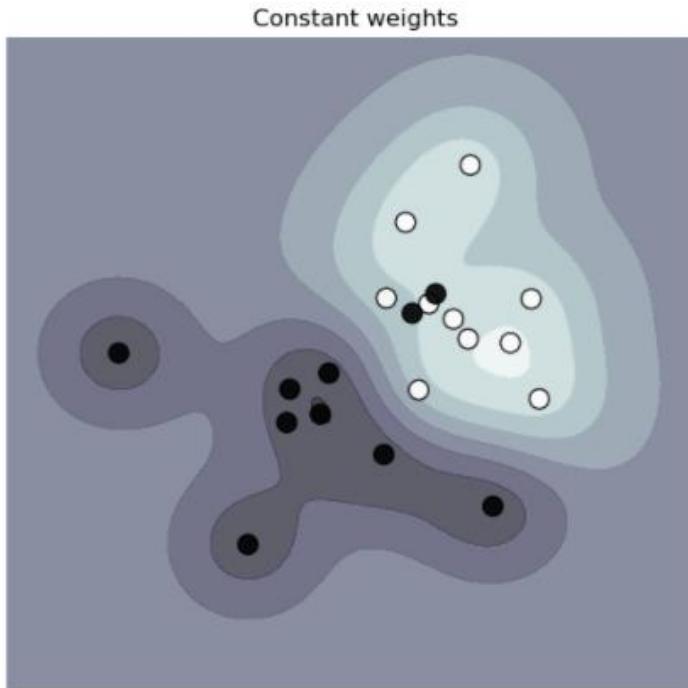


# SUPPORT VECTOR MACHINE (SVM)

## SVM: Weighted samples

Plot decision function of a weighted dataset, where the size of points is proportional to its weight.

The sample weighting rescales the C parameter, which means that the classifier puts more emphasis on getting these points right. The effect might often be subtle. To emphasize the effect here, we particularly weight outliers, making the deformation of the decision boundary very visible.



# NAIVE BAYES CLASSIFICATION

Naïve Bayes Classification เป็นอัลกอริทึมหนึ่งของ Machine Learning ที่จัดอยู่ในประเภท Supervised Learning อัลกอริทึมนี้เป็นการนำหลักการของความน่าจะเป็นมาช่วยจำแนกหรือแยกประเภทข้อมูลตาม class โดยมีสูตรดังนี้

ในที่นี้  $y$  คือ เป้าหมายของการทำนายผล (Target) ส่วน  $X$  คือ คุณลักษณะหนึ่ง ๆ ของข้อมูล (Feature) โดย

$$p(y|X) = \frac{p(X|y)p(y)}{p(X)}$$

- $p(y|X)$  เรียกว่า posterior probability
- $p(X|y)$  เรียกว่า Likelihood
- $p(y)$  เรียกว่า Prior probability
- $p(X)$  เรียกว่า Predictor prior probability

สมมติเรามีตัวอย่างข้อมูลดังตารางด้านไปนี้

| Color | Result |
|-------|--------|
| Red   | Lose   |
| Green | Win    |
| Blue  | Win    |
| Red   | Win    |
| Red   | Win    |
| Green | Win    |
| Blue  | Lose   |
| Blue  | Lose   |
| Red   | Win    |
| Blue  | Win    |
| Red   | Lose   |
| Green | Win    |
| Green | Lose   |
| Blue  | Lose   |

# NAIVE BAYES CLASSIFICATION

เพื่อให้สามารถพิจารณาข้อมูลได้ง่าย เราจะลองนำข้อมูลมาจัดเรียงใหม่โดยแยกข้อมูลออก มาเป็นผลการแข่งขันของแต่ละสี ดังนี้

| Color       | Result |      |
|-------------|--------|------|
|             | Win    | Lose |
| Red         | 3      | 2    |
| Green       | 3      | 1    |
| Blue        | 2      | 3    |
| Total       | 8      | 6    |
| Grand Total | 14     |      |

ลองคำนวณว่าโอกาสที่ผู้เล่นจะ “ชนะ” ถ้าอยู่ทีม “สีแดง” เป็นเท่าไร

$$p(\text{Win}|\text{Red}) = \frac{p(\text{Red}|\text{Win}) * p(\text{Win})}{p(\text{Red})}$$

- $p(\text{Red}|\text{Win})$  คือ นำจำนวนครั้งที่สีแดงชนะ หารด้วย จำนวนครั้งทั้งหมดที่ชนะ จะได้ค่า  $3/8$
- $p(\text{Win})$  คือ นำจำนวนครั้งที่ชนะ หารด้วย จำนวนครั้งทั้งหมด จะได้ค่า  $8/14$
- $p(\text{Red})$  คือ นำจำนวนผู้เล่นที่เป็นสีแดง (ทั้งแพ้และชนะ) หารด้วย จำนวนครั้งทั้งหมด จะได้ค่า  $5/14$

$$\text{ดังนั้น } p(\text{Win}|\text{Red}) = ((3/8) * (8/14)) / (5/14) = 3/5 = 0.6$$

จึงสรุปได้ว่า ความน่าจะเป็นที่ผู้เล่นจะ “ชนะ” ถ้าอยู่ทีม “สีแดง” คือ  $0.6$

# NAIVE BAYES CLASSIFICATION

จะสังเกตว่าตัวอย่างที่ผ่านมาเรามี Feature แค่ 1 ตัว ในกรณีที่เรามี Feature มากกว่า 1 ตัว เราสามารถหาความน่าจะเป็นได้ด้วยสูตรดังต่อไปนี้

$$p(y|X) = p(X_1|y)*p(X_2|y)*...*p(X_n|y)*p(y)$$

สมมติเรามีตัวอย่างข้อมูล  
ดังตารางด้านล่าง

| Weather | Temperature | Running |
|---------|-------------|---------|
| Sunny   | Hot         | No      |
| Rainy   | Cool        | No      |
| Cloudy  | Warm        | Yes     |
| Sunny   | Cool        | Yes     |
| Sunny   | Warm        | Yes     |
| Rainy   | Hot         | Yes     |
| Cloudy  | Cool        | No      |
| Cloudy  | Warm        | Yes     |
| Sunny   | Hot         | No      |
| Cloudy  | Warm        | Yes     |
| Sunny   | Warm        | Yes     |
| Rainy   | Warm        | Yes     |
| Rainy   | Cool        | No      |
| Cloudy  | Hot         | No      |

# NAIVE BAYES CLASSIFICATION

เพื่อให้สามารถพิจารณาข้อมูลได้ง่าย เราจะลองนำข้อมูลมาจัดเรียงใหม่โดยแยกข้อมูลออกตาม  
ตามสภาพอากาศ (Weather) และอุณหภูมิ (Temperature) ดังนี้

| Weather     | Running |    |
|-------------|---------|----|
|             | Yes     | No |
| Sunny       | 3       | 2  |
| Cloudy      | 3       | 2  |
| Rainy       | 2       | 2  |
| Total       | 8       | 6  |
| Grand Total | 14      |    |

| Temperature | Running |    |
|-------------|---------|----|
|             | Yes     | No |
| Hot         | 1       | 3  |
| Warm        | 6       | 0  |
| Cool        | 1       | 3  |
| Total       | 8       | 6  |
| Grand Total | 14      |    |

ลองคำนวณว่าผู้เล่นจะ “ออกไปวิ่ง (Running = “Yes”) หรือไม่ “ออกไปวิ่ง (Running=“No”)  
ถ้าสภาพอากาศ “ฝนตก (Rainy)” และอุณหภูมิ “ร้อน (Hot)”

ในที่นี่เรายากทราบว่าผู้เล่นจะออกไปวิ่งหรือไม่ เราต้องลองหาความน่าจะเป็นของกรณี ฝนตก,  
ร้อน, ออกไปวิ่ง เทียบกับ ความน่าจะเป็นของกรณี ฝนตก, ร้อน, ไม่ออกไปวิ่ง ว่าเป็นเท่าไหร่ แล้วนำ  
ทั้ง 2 กรณีมาเปรียบเทียบกัน ถ้ากรณีใดมีค่าความน่าจะเป็นมากกว่า ผลการคำนวณก็จะเป็นกรณีนั้น

# NAIVE BAYES CLASSIFICATION

กรณีที่ 1 : ฝนตก, ร้อน, ออกไบวิ่ง

$$p(\text{Yes}|\text{Rainy}, \text{Hot}) = p(\text{Rainy}|\text{Yes}) * p(\text{Hot}|\text{Yes}) * p(\text{Yes})$$

- $p(\text{Rainy}|\text{Yes})$  คือ นำจำนวนครั้งที่ Weather="Rainy" และ Running="Yes" หารด้วย จำนวนครั้งทั้งหมดที่ Running= "Yes" จะได้ค่า  $2/8$  (มีค่าเท่ากับ 0.25)
- $p(\text{Hot}|\text{Yes})$  คือ นำจำนวนครั้งที่ Weather="Hot" และ Running="Yes" หารด้วย จำนวนครั้งทั้งหมดที่ Running= "Yes" จะได้ค่า  $1/8$  (มีค่าเท่ากับ 0.125)
- $p(\text{Yes})$  คือ นำจำนวนครั้งที่ Running="Yes" หารด้วย จำนวนครั้งทั้งหมด จะได้ค่า  $8/14$  = (มีค่าเท่ากับ 0.57)

ดังนั้น  $p(\text{Yes}|\text{Rainy}, \text{Hot}) = 0.25 * 0.125 * 0.57 = 0.01$

จึงสรุปได้ว่า ความน่าจะเป็นที่ผู้เล่นจะ "ออกไบวิ่ง (Running = Yes)" ถ้าสภาพอากาศ "ฝนตก (Rainy)" และอุณหภูมิ "ร้อน (Hot)" คือ 0.01

# NAIVE BAYES CLASSIFICATION

จากตัวอย่างที่แสดงจะพบว่าข้อดีของอัลกอริทึมนี้คือเป็นอัลกอริทึมที่ง่ายและทำนายผลได้เร็วมาก สามารถทำนายผลแบบ real-time ได้ และสามารถวิเคราะห์และทำนายผลข้อมูลที่ประกอบด้วยหมวดหมู่จำนวนมากได้ แต่ข้อเสียคือ Feature แต่ละตัวไม่มีความสัมพันธ์ต่อกัน คือเป็นอิสระต่อกันจึงไม่สามารถหาความสัมพันธ์ระหว่าง Feature ได้

ตัวอย่างของการนำอัลกอริทึม Naïve Bayes Classification ไปใช้งาน เช่น

- **Text Classification** คือ การจำแนกแยกหมวดหมู่ของข้อความ หรือ จัดแบ่งประเภทของเอกสาร เช่น การคัดแยกอีเมล การแยกข้อความในแท็บ ให้อยู่ในหมวดหมู่ตามที่กำหนด
- **Sentiment Analysis** คือ การวิเคราะห์ความรู้สึก เช่น social network ทำนายอารมณ์หรือความรู้สึกของลูกค้าจากข้อความ เช่น พอใจไม่พอใจ ชอบไม่ชอบ รู้สึกดี/ปานกลาง/แย่ เพื่อนำไปใช้ประโยชน์ด้านการวิเคราะห์วางแผนการตลาด
- **Spam Filtering** คือ ตรวจจับอีเมลหรือข้อความ sms ว่าเป็นสแปมหรือไม่

# NAIVE BAYES CLASSIFICATION

## ตัวอย่าง : Sentimental Analysis วิเคราะห์ความรู้สึกจากข้อความด้วย Naïve Bayes

ตัวอย่างต่อไปนี้เราจะมาทำ Sentimental Analysis กัน โดยนำประโยคภาษาของมนุษย์มาวิเคราะห์กันว่าประยุคนั้นให้ความรู้สึกบวก (Positive) หรือ ลบ (Negative)

Sentimental Analysis เป็นการทำงานแบบ NLP (Natural Language Processing) ซึ่ง NLP เป็นการทำงานส่วนหนึ่งของ AI ที่นำภาษาธรรมชาติ (Natural Language) มาประมวลผลและวิเคราะห์ด้วย Machine Learning

สมมติเรามีตัวอย่างข้อมูลดังนี้

เราจะมีข้อความต้นแบบ (SentimentText) เก็บไว้เป็นข้อมูลว่าข้อความลักษณะนี้ให้ความรู้สึกบวก (SentimentScore=1) หรือให้ความรู้สึกลบ (SentimentScore=0) เมื่อลูกค้าป้อนข้อความคอมเม้นต์ใหม่เข้ามาเราก็จะนำมาวิเคราะห์จากข้อความต้นแบบว่าลูกค้ามีความรู้สึกบวก (พอใจ) หรือลบ (ไม่พอใจ) กับสินค้าและบริการ เป็นต้น

| SentimentID | SentimentScore | SentimentText                   |
|-------------|----------------|---------------------------------|
| 0           | 101            | I love it                       |
| 1           | 102            | Very nice                       |
| 2           | 103            | Very late                       |
| 3           | 104            | Quick response                  |
| 4           | 105            | Be happy                        |
| 5           | 106            | Bad services                    |
| 6           | 107            | Good luck                       |
| 7           | 108            | I hate it                       |
| 8           | 109            | I like it                       |
| 9           | 110            | I am so glad                    |
| 10          | 111            | Fast shipping                   |
| 11          | 112            | Beautiful and pretty            |
| 12          | 113            | You are the best                |
| 13          | 114            | Good information                |
| 14          | 115            | Well done                       |
| 15          | 116            | I am afraid that it is not good |
| 16          | 117            | Fast shipping                   |
| 17          | 118            | You are professional            |
| 18          | 119            | I am so mad at it               |
| 19          | 120            | Smell terrible                  |

# NAIVE BAYES CLASSIFICATION

ตัวอย่างนี้มีข้อมูลของ SentimentText ประมาณ 20 รายการเพื่อแสดงตัวอย่างให้ดูหลักการทำงานเท่านั้น ในงานจริงควรต้องมี SentimentText เยอะกว่านี้มาก ๆ เพื่อจะได้สามารถวิเคราะห์ข้อความได้หลากหลายมากขึ้นค่ะ

ตัวอย่างที่ 11.7

# NAIVE BAYES CLASSIFICATION

[https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)

## 1.9. Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable  $y$  and dependent feature vector  $x_1$  through  $x_n$ :

$$P(y | x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n | y)}{P(x_1, \dots, x_n)}$$

Using the naive conditional independence assumption that

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y),$$

for all  $i$ , this relationship is simplified to

$$P(y | x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i | y)}{P(x_1, \dots, x_n)}$$

Since  $P(x_1, \dots, x_n)$  is constant given the input, we can use the following classification rule:

$$P(y | x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i | y)$$

↓

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i | y),$$

### 1.9. Naive Bayes

- 1.9.1. Gaussian Naive Bayes
  - 1.9.2. Multinomial Naive Bayes
  - 1.9.3. Complement Naive Bayes
  - 1.9.4. Bernoulli Naive Bayes
  - 1.9.5. Categorical Naive Bayes
  - 1.9.6. Out-of-core naive Bayes
- model fitting

# NAIVE BAYES CLASSIFICATION

## Gaussian Naïve Bayes algorithm

When we have continuous attribute values, we made an assumption that the values associated with each class are distributed according to Gaussian or Normal distribution. For example, suppose the training data contains a continuous attribute  $x$ . We first segment the data by the class, and then compute the mean and variance of  $x$  in each class. Let  $\mu_i$  be the mean of the values and let  $\sigma_i$  be the variance of the values associated with the  $i$ th class. Suppose we have some observation value  $x_i$ . Then, the probability distribution of  $x_i$  given a class can be computed by the following equation –

$$p(x_i|y_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x_i-\mu_j)^2}{2\sigma_j^2}}$$

Naïve Bayes is one of the most straightforward and fast classification algorithm. It is very well suited for large volume of data. It is successfully used in various applications such as :

## Multinomial Naïve Bayes algorithm

With a Multinomial Naïve Bayes model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial ( $p_1, \dots, p_n$ ) where  $p_i$  is the probability that event  $i$  occurs. Multinomial Naïve Bayes algorithm is preferred to use on data that is multinomially distributed. It is one of the standard algorithms which is used in text categorization classification.

## Bernoulli Naïve Bayes algorithm

In the multivariate Bernoulli event model, features are independent boolean variables (binary variables) describing inputs. Just like the multinomial model, this model is also popular for document classification tasks where binary term occurrence features are used rather than term frequencies.

1. Spam filtering
2. Text classification
3. Sentiment analysis
4. Recommender systems

# NAIVE BAYES CLASSIFICATION

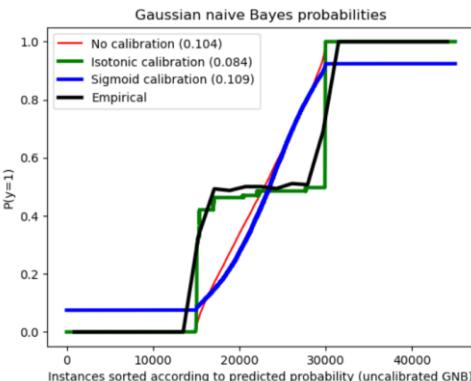
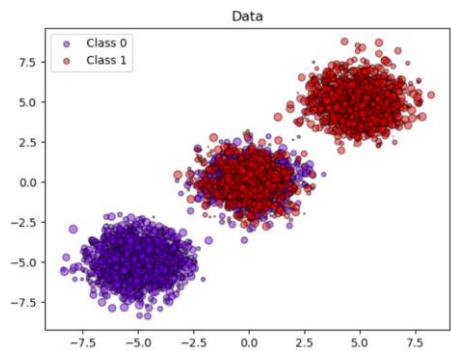
Gaussian Naïve Bayes algorithm

Example: kaggle

## Probability calibration of classifiers

When performing classification you often want to predict not only the class label, but also the associated probability. This probability gives you some kind of confidence on the prediction. However, not all classifiers provide well-calibrated probabilities, some being over-confident while others being under-confident. Thus, a separate calibration of predicted probabilities is often desirable as a postprocessing. This example illustrates two different methods for this calibration and evaluates the quality of the returned probabilities using Brier's score (see [https://en.wikipedia.org/wiki/Brier\\_score](https://en.wikipedia.org/wiki/Brier_score)).

Compared are the estimated probability using a Gaussian naive Bayes classifier without calibration, with a sigmoid calibration, and with a non-parametric isotonic calibration. One can observe that only the non-parametric model is able to provide a probability calibration that returns probabilities close to the expected 0.5 for most of the samples belonging to the middle cluster with heterogeneous labels. This results in a significantly improved Brier score.



# NAIVE BAYES CLASSIFICATION

## `sklearn.naive_bayes.MultinomialNB`

`class sklearn.naive_bayes.MultinomialNB(*, alpha=1.0, fit_prior=True, class_prior=None)`

[\[source\]](#)

Naive Bayes classifier for multinomial models.

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

Example

## Classification of text documents using sparse features

This is an example showing how scikit-learn can be used to classify documents by topics using a bag-of-words approach. This example uses a `scipy.sparse` matrix to store the features and demonstrates various classifiers that can efficiently handle sparse matrices.

The dataset used in this example is the 20 newsgroups dataset. It will be automatically downloaded, then cached.

# NAIVE BAYES CLASSIFICATION

## 1.9.4. Bernoulli Naive Bayes

`BernoulliNB` implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors; if handed any other kind of data, a `BernoulliNB` instance may binarize its input (depending on the `binarize` parameter).

The decision rule for Bernoulli naive Bayes is based on

$$P(x_i | y) = P(i | y)x_i + (1 - P(i | y))(1 - x_i)$$

which differs from multinomial NB's rule in that it explicitly penalizes the non-occurrence of a feature  $i$  that is an indicator for class  $y$ , where the multinomial variant would simply ignore a non-occurring feature.

In the case of text classification, word occurrence vectors (rather than word count vectors) may be used to train and use this classifier. `BernoulliNB` might perform better on some datasets, especially those with shorter documents. It is advisable to evaluate both models, if time permits.

### Example

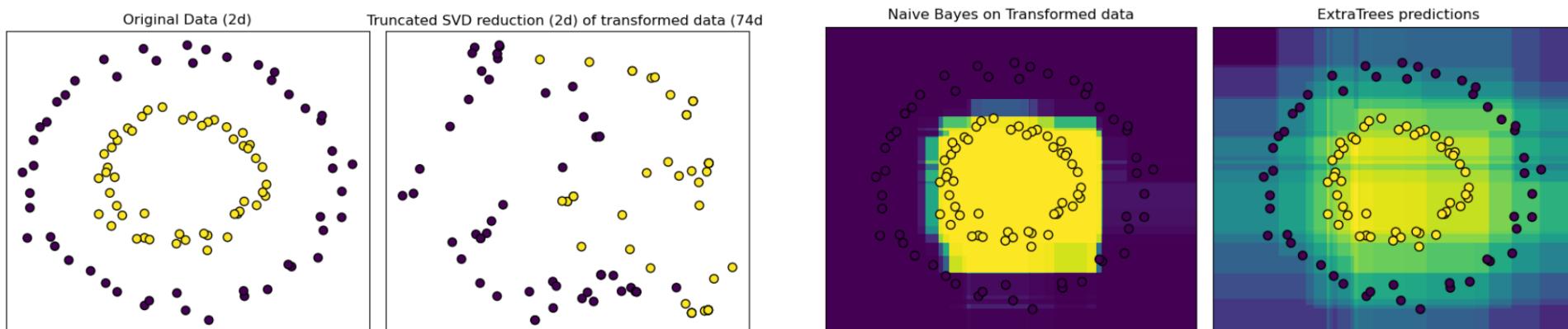
## Hashing feature transformation using Totally Random Trees

RandomTreesEmbedding provides a way to map data to a very high-dimensional, sparse representation, which might be beneficial for classification. The mapping is completely unsupervised and very efficient.

This example visualizes the partitions given by several trees and shows how the transformation can also be used for non-linear dimensionality reduction or non-linear classification.

Points that are neighboring often share the same leaf of a tree and therefore share large parts of their hashed representation. This allows to separate two concentric circles simply based on the principal components of the transformed data with truncated SVD.

In high-dimensional spaces, linear classifiers often achieve excellent accuracy. For sparse binary data, BernoulliNB is particularly well-suited. The bottom row compares the decision boundary obtained by BernoulliNB in the transformed space with an ExtraTreesClassifier forests learned on the original data.



# NAIVE BAYES CLASSIFICATION

## 1.9.5. Categorical Naive Bayes

`CategoricalNB` implements the categorical naive Bayes algorithm for categorically distributed data. It assumes that each feature, which is described by the index  $i$ , has its own categorical distribution.

For each feature  $i$  in the training set  $X$ , `CategoricalNB` estimates a categorical distribution for each feature  $i$  of  $X$  conditioned on the class  $y$ . The index set of the samples is defined as  $J = \{1, \dots, m\}$ , with  $m$  as the number of samples.

The probability of category  $t$  in feature  $i$  given class  $c$  is estimated as:

$$P(x_i = t \mid y = c; \alpha) = \frac{N_{tic} + \alpha}{N_c + \alpha n_i},$$

where  $N_{tic} = |\{j \in J \mid x_{ij} = t, y_j = c\}|$  is the number of times category  $t$  appears in the samples  $x_i$ , which belong to class  $c$ ,  $N_c = |\{j \in J \mid y_j = c\}|$  is the number of samples with class  $c$ ,  $\alpha$  is a smoothing parameter and  $n_i$  is the number of available categories of feature  $i$ .

`CategoricalNB` assumes that the sample matrix  $X$  is encoded (for instance with the help of `ordinalEncoder`) such that all categories for each feature  $i$  are represented with numbers  $0, \dots, n_i - 1$  where  $n_i$  is the number of available categories of feature  $i$ .

### Example

# แบบฝึกหัด

## 1. Support vector machines – Iris datasets

In [1]:

```
from sklearn import datasets  
  
iris = datasets.load_iris()
```

<https://pythoncursus.nl/support-vector-machine/>

In deze dataset zijn de volgende

In [2]:

```
iris.feature_names
```

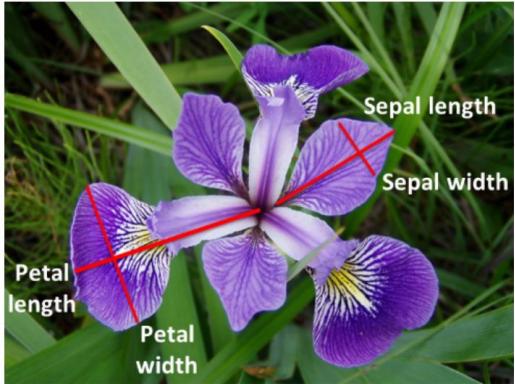
Out[2]:

```
['sepal length (cm)',  
 'sepal width (cm)',  
 'petal length (cm)',  
 'petal width (cm)']
```

# แบบฝึกหัด

## 2. Naïve Bayes – iris flower datasets

<https://www.kaggle.com/vinayshaw/iris-species-100-accuracy-using-naive-bayes>



**iris setosa**



petal

**iris versicolor**



petal

**iris virginica**



petal



sepal

# **THE END**