

Naïve Bayes. Evaluating Machine Learning Methods.

COMP5318 Machine Learning and Data Mining
semester 2, 2024, week 4

Nguyen Hoang Tran

Based on slides prepared by Irena Koprinska (irena.koprinska@sydney.edu.au)

Reference: 1) Witten ch.4.2, Tan ch.5.3

2) Witten ch.5: 128-131, Tan: ch.4.5, Müller & Guido: ch.5



- Bayes theorem
- Naïve Bayes algorithm
- Evaluating ML models
 - Evaluation procedures
 - Holdout method
 - Cross validation
 - Leave-one-out cross validation
 - Cross-validation for parameter tuning
 - Performance measures
 - Accuracy, recall, precision and F1 score; confusion matrix



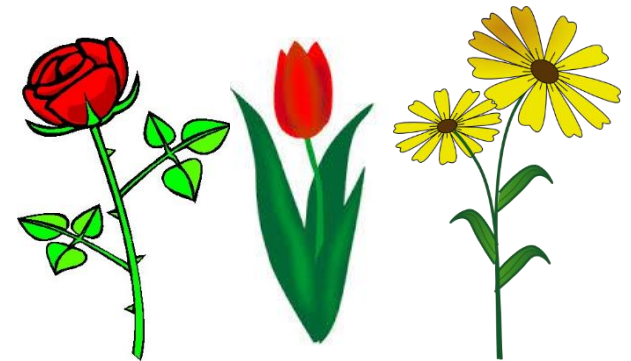
Probabilistic methods: Naïve Bayes

- Probabilistic classification methods compute the **class membership probability**, i.e. the probability that a given example belongs to a particular class
- **Naive Bayes** is a prominent example of this group
- It is based on the **Bayes Theorem**

- Given a hypothesis H and evidence E for this hypothesis, then the probability of H given E is:

$$P(H | E) = \frac{P(E | H)P(H)}{P(E)}$$

- Example:
 - Given are examples of flowers, described by 2 features:
 - color = {red, yellow}
 - stem = {long, short}
 - Suppose that:
 - E is red and long
 - H is the hypothesis that E is a rose



$P(H|E)=?$ $P(H)=?$ $P(E|H)=?$ $P(E)=?$

Probabilities for our example

- E is red and long
 - H is the hypothesis that E is a rose
- $P(H|E)=?$ $P(H)=?$ $P(E|H)=?$ $P(E)=?$

- $P(H|E)$ is the probability of the hypothesis (i.e. that E is a rose), given that we have seen that E is red and long
 - Called **posteriori probability** – probability of an event *after* seeing the evidence
 - Also called **conditional probability** – probability of H given E
- $P(H)$ is the probability that any given example is a rose, regardless of how it looks
 - Called **prior probability** of H – probability of an event *before* seeing evidence
 - It is independent of E

Similarly:


- $P(E|H)$ is the probability that E is red and long, given that we know that E is a rose
 - Called posteriori/conditional probability of E given H
- $P(E)$ is the probability that any given example (flower) is red and long, regardless of the hypothesis
 - Called prior probability of E; it is independent of H

- Uses the Bayes Theorem to solve classification tasks
- Two assumptions:
 - 1) Independence – (the values of the) attributes are conditionally independent of each other, given the class (i.e. for each class value)
 - 2) Equally importance – all attributes are equally important
- Unrealistic assumptions – almost never correct
 - => that's why the algorithm is called **Naive** Bayes
- However, these assumptions lead to a simple and easy to implement algorithm, which works surprisingly well in practice

Applying Naïve Bayes

- Consider the weather data:
- Use Naïve Bayes to predict the class (yes or no) of the new example:

outlook	temp.	humidity	windy	play
sunny	cool	high	true	?

- Bayes theorem:
$$P(H | E) = \frac{P(E | H)P(H)}{P(E)}$$
- What are H and E?
 - the evidence **E is the new example**
 - the hypothesis **H is play=yes** (and there is another **H: play=no**)
- How to use the Bayes theorem?
 - Calculate $P(H|E)$ for each H (class), i.e. $P(\text{yes}|E)$ and $P(\text{no}|E)$ 
 - Compare them and assign E to the class with the highest probability
 - For $P(H|E)$ we need to calculate $P(E)$, $P(H)$ and $P(E|H)$ – how to do this?
 - From the given training data

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

- Step 1: Calculate $P(H|E)$ for each H (class), i.e. $P(\text{yes}|E)$ and $P(\text{no}|E)$

$$P(\text{yes} | E) = \frac{P(E | \text{yes})P(\text{yes})}{P(E)}$$

$$P(\text{no} | E) = \frac{P(E | \text{no})P(\text{no})}{P(E)}$$

where E :

outlook	temp.	humidity	windy	play
sunny	cool	high	true	?

outlook=sunny, temperature=cool,
humidity=high, windy=true

- How to calculate $P(E|\text{yes})$ and $P(E|\text{no})$?
 - Split the evidence E into 4 smaller pieces of evidence (per-attribute evidences):
 $E_1 = \text{outlook=sunny}$, $E_2 = \text{temperature=cool}$
 $E_3 = \text{humidity=high}$, $E_4 = \text{windy=true}$
 - Use the Naïve Bayes's independence assumption:
 - E_1 , E_2 , E_3 and E_4 are independent given the class. Then, their combined probability is obtained by multiplication of per-attribute probabilities:

$$P(E | \text{yes}) = P(E_1 | \text{yes}) P(E_2 | \text{yes}) P(E_3 | \text{yes}) P(E_4 | \text{yes})$$

$$P(E | \text{no}) = P(E_1 | \text{no}) P(E_2 | \text{no}) P(E_3 | \text{no}) P(E_4 | \text{no})$$

- Hence, we obtain:

$$P(\text{yes} | E) = \frac{P(E_1 | \text{yes}) P(E_2 | \text{yes}) P(E_3 | \text{yes}) P(E_4 | \text{yes}) P(\text{yes})}{P(E)}$$

$$P(\text{no} | E) = \frac{P(E_1 | \text{no}) P(E_2 | \text{no}) P(E_3 | \text{no}) P(E_4 | \text{no}) P(\text{no})}{P(E)}$$

- Top parts: the probabilities will be calculated from the training data
- Bottom parts: $P(E)$ in both cases - the same for class yes and no. Since we take the decision by comparing $P(\text{yes}|E)$ and $P(\text{yes}|\text{no})$, there is no need to calculate $P(E)$, we will just compare the top parts.

Calculating probabilities from training data

$E_1 = \text{outlook}=\text{sunny}$, $E_2 = \text{temperature}=\text{cool}$

$E_3 = \text{humidity}=\text{high}$, $E_4 = \text{windy}=\text{true}$

$$P(\text{yes} | E) = \frac{P(E_1 | \text{yes}) P(E_2 | \text{yes}) P(E_3 | \text{yes}) P(E_4 | \text{yes}) P(\text{yes})}{P(E)}$$

$P(E_1|\text{yes})=P(\text{outlook}=\text{sunny}|\text{yes})=?$

$P(E_2|\text{yes})=P(\text{temp}=\text{cool}|\text{yes})=?$

$P(E_3|\text{yes})=P(\text{humidity}=\text{high}|\text{yes})=?$

$P(E_4|\text{yes})=P(\text{windy}=\text{true}|\text{yes})=?$

$P(\text{yes})=?$

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Calculating probabilities from training data

E1 = outlook=sunny, E2 = temperature=cool

E3 = humidity=high, E4 = windy=true

$$P(\text{yes} | E) = \frac{P(E_1 | \text{yes}) P(E_2 | \text{yes}) P(E_3 | \text{yes}) P(E_4 | \text{yes}) P(\text{yes})}{P(E)}$$

$P(E_1 | \text{yes}) = P(\text{outlook}=\text{sunny} | \text{yes}) = ?/9 = 2/9$

$P(E_2 | \text{yes}) = P(\text{temp}=\text{cool} | \text{yes}) = ?$

$P(E_3 | \text{yes}) = P(\text{humidity}=\text{high} | \text{yes}) = ?$

$P(E_4 | \text{yes}) = P(\text{windy}=\text{true} | \text{yes}) = ?$

$P(\text{yes}) = ?$

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

proportions of days when outlook=sunny for days with play=yes
i.e. the probability of outlook=sunny, given that play=yes

Calculating probabilities from training data (2)

Similarly we calculate the other conditional probabilities:

$$P(\text{yes} | E) = \frac{P(E_1 | \text{yes}) P(E_2 | \text{yes}) P(E_3 | \text{yes}) P(E_4 | \text{yes}) P(\text{yes})}{P(E)}$$

$$P(E_1 | \text{yes}) = P(\text{outlook} = \text{sunny} | \text{yes}) = ? / 9 = 2 / 9$$

$$P(E_2 | \text{yes}) = P(\text{temp} = \text{cool} | \text{yes}) = 3 / 9$$

$$P(E_3 | \text{yes}) = P(\text{humidity} = \text{high} | \text{yes}) = 3 / 9$$

$$P(E_4 | \text{yes}) = P(\text{windy} = \text{true} | \text{yes}) = 3 / 9$$

$$P(\text{yes}) = ?$$

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

proportions of days when outlook=sunny for days with play=yes
i.e. the probability of outlook=sunny, given that play=yes

Calculating probabilities from training data (3)

Similarly we calculate the other conditional probabilities:

$$P(\text{yes} | E) = \frac{P(E_1 | \text{yes}) P(E_2 | \text{yes}) P(E_3 | \text{yes}) P(E_4 | \text{yes}) P(\text{yes})}{P(E)}$$

$$P(E_1 | \text{yes}) = P(\text{outlook} = \text{sunny} | \text{yes}) = ?/9 = 2/9$$

$$P(E_2 | \text{yes}) = P(\text{temp} = \text{cool} | \text{yes}) = 3/9$$

$$P(E_3 | \text{yes}) = P(\text{humidity} = \text{high} | \text{yes}) = 3/9$$

$$P(E_4 | \text{yes}) = P(\text{windy} = \text{true} | \text{yes}) = 3/9$$

$$P(\text{yes}) = ?$$

- the prior probability of play=yes - the probability of play=yes without E, i.e. without knowing anything about the particular day
- calculated from the “play” column = 9/14

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

- For play=yes:

$$P(\text{yes} | E) = \frac{\frac{2}{9} \frac{3}{9} \frac{3}{9} \frac{3}{9} \frac{9}{14}}{P(E)} = \frac{0.0053}{P(E)}$$

← 相乘

- Similarly we can calculate the probability for play=no:

$$P(\text{no} | E) = \frac{\frac{3}{5} \frac{1}{5} \frac{4}{5} \frac{3}{5} \frac{5}{14}}{P(E)} = \frac{0.0206}{P(E)}$$

- Since $P(\text{no}|E) > P(\text{yes}|E)$, Naïve Bayes predicts play=no for the new day

- Given is the following dataset where **loan default** is the class. Predict the class of the following new example using Naïve Bayes:

home owner = no, marital status = married, annual income=very high

	home owner	marital status	income	loan default
1	yes	single	very high	yes
2	no	married	high	yes
3	no	single	medium	no
4	yes	married	very high	no
5	yes	divorced	high	yes
6	no	married	low	no
7	yes	divorced	very high	no
8	no	single	high	yes
9	no	married	medium	no
10	no	single	low	yes

$$P(H | E) = \frac{P(E | H)P(H)}{P(E)}$$

- New example:
E = home owner = no, marital status = married, annual income = very high
- Split into 3 pieces of evidence, 1 per each attribute:
E1 is home owner = no
E2 is marital status = married
E3 is annual income = very high
- Calculate $P(\text{yes}|E)$ and $P(\text{no}|E)$ and compare them:

$$P(\text{yes}|E) = \frac{P(E_1|\text{yes})P(E_2|\text{yes})P(E_3|\text{yes})P(\text{yes})}{P(E)}$$

$$P(\text{no}|E) = \frac{P(E_1|\text{no})P(E_2|\text{no})P(E_3|\text{no})P(\text{no})}{P(E)}$$

	home owner	marital status	income	loan default
1	yes	single	very high	yes
2	no	married	high	yes
3	no	single	medium	no
4	yes	married	very high	no
5	yes	divorced	high	yes
6	no	married	low	no
7	yes	divorced	very high	no
8	no	single	high	yes
9	no	married	medium	no
10	no	single	low	yes

E1 is home owner = no, E2 is marital status = married, E3 is annual income=very high

$$P(\text{yes})=5/10$$

$$P(E1|\text{yes})=P(\text{home owner=no}|\text{yes})=3/5$$

$$P(E2|\text{yes})=P(\text{marital status=married}|\text{yes})=1/5$$

$$P(E3|\text{yes})=P(\text{annual income=very high}|\text{yes})=1/5$$

$$P(\text{yes}|E) = \frac{\frac{3}{5} \cdot \frac{1}{5} \cdot \frac{1}{5} \cdot \frac{5}{10}}{P(E)} = \frac{\frac{3}{250}}{P(E)} = \frac{0.012}{P(E)}$$

$$P(\text{no}|E) = \frac{\frac{3}{5} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{5}{10}}{P(E)} = \frac{\frac{9}{125}}{P(E)} = \frac{0.072}{P(E)}$$

$P(\text{no}|E) > P(\text{yes}|E) \Rightarrow$ Naïve Bayes predicts
loan default = no for the new example

$$P(\text{no})=5/10$$

$$P(E1|\text{no})= P(\text{home owner=no}|\text{no})=3/5$$

$$P(E2|\text{no})=P(\text{marital status=married}|\text{no})=3/5$$

$$P(E3|\text{no})=P(\text{annual income=very high}|\text{no})=2/5$$

	home owner	marital status	income	loan default
1	yes	single	very high	yes
2	no	married	high	yes
3	no	single	medium	no
4	yes	married	very high	no
5	yes	divorced	high	yes
6	no	married	low	no
7	yes	divorced	very high	no
8	no	single	high	yes
9	no	married	medium	no
10	no	single	low	yes

1 The “zero-frequency” problem

- What if an attribute value does not occur with every class value?
- E.g. suppose that the training data was different:

outlook=sunny had **always occurred** together with play=no, i.e.
outlook=sunny had **never occurred** together with play=yes

- Then: $P(\text{outlook}=\text{sunny}|\text{yes})=0$

$$P(\text{yes} | E) = \frac{\underbrace{P(E_1 | \text{yes})}_{=0} P(E_2 | \text{yes}) P(E_3 | \text{yes}) P(E_4 | \text{yes}) P(\text{yes})}{P(E)}$$

=> $P(\text{yes}|E)=0$, regardless of the other probability values

- This means that the prediction for new examples with outlook=sunny will always be play=no, completely ignoring the values of the other attributes
- Remedy: add 1 to the numerator and m to the denominator (m - number of attribute values = 3 for outlook)

$$P(E_i|\text{yes}) = (\text{count}(E_i) + 1) / (\text{count}(\text{yes}) + m), \text{ same for } P(E_i|\text{no})$$

- This is called **Laplace correction** or **smoothing**
 - it ensures that the probabilities will never be 0

- Naïve Bayes can easily deal with missing values
- During **classification**: missing value in the new example
 - do not include this attribute, e.g.:
 - **outlook=?**, temperature=cool, humidity=high, windy=true
- Then:

$$P(\text{yes} | E) = \frac{\overline{3} \overline{3} \overline{3} \overline{9}}{\overline{9} \overline{9} \overline{9} \overline{14}} = \frac{0.0238}{P(E)} \quad P(\text{no} | E) = \frac{\overline{1} \overline{4} \overline{3} \overline{5}}{\overline{5} \overline{5} \overline{5} \overline{14}} = \frac{0.0343}{P(E)}$$

outlook is not
included

- During **training**:
 - do not include the missing values in the counts
 - calculate the probabilities based on the actual number of training examples without missing values for each attribute



Naïve Bayes for Numeric Attributes

How to apply Naïve Bayes to numeric attributes?

- Now assume that **temperature** and **humidity** are numeric attributes (**outlook** and **windy** are still nominal)
- We would like to classify the following new example:

outlook	temperature	humidity	windy	play
sunny	66	90	true	?

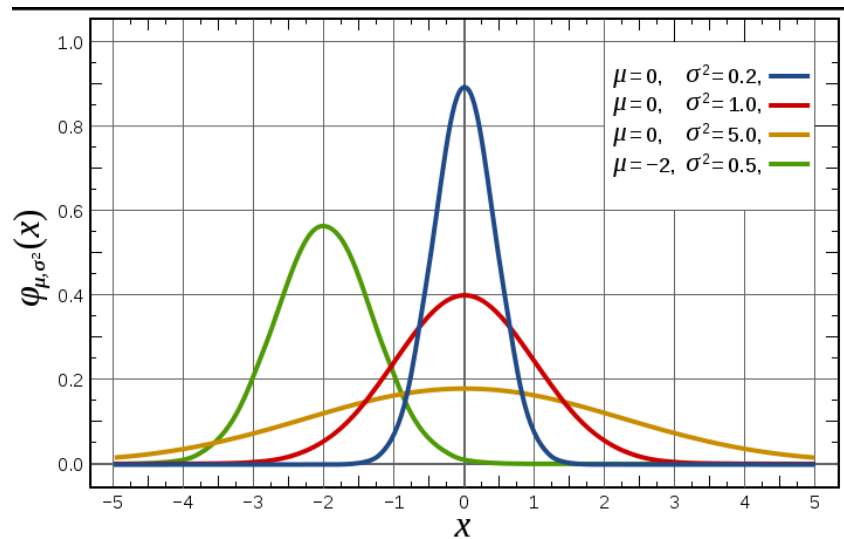
- How to calculate the probabilities for the numeric attributes?
 $P(\text{temperature}=66|\text{yes})=?$, $P(\text{humidity}=90|\text{yes})=?$
 $P(\text{temperature}=66|\text{no})=?$, $P(\text{humidity}=90|\text{no})=?$
- Answer: assume that the numeric attributes follow a *normal* (or *Gaussian*) *distribution* and use *probability density function*

Probability density function for normal distribution

- Probability density function for a *normal* distribution with mean μ and standard deviation σ :

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

- The probability density function is not exactly the probability but it is closely related



Reminder about μ and σ :

$$\mu = \frac{\sum_{i=1}^n x_i}{n} \quad \sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{n-1}}$$

- For each numeric attribute:
 - Split the values per class
 - Calculate μ and σ (for each attribute-class combination)

outlook			temperature		humidity		windy			play	
	yes	no	yes	no	yes	no	yes	no		yes	no
sunny	2	3	83	85	86	85	false	6	2	9	5
overcast	4	0	70	80	96	90	true	3	3		
rainy	3	2	68	65	80	70					
			64	72	65	95					
			69	71	70	91					
			75		80						
			75		70						
			72		90						
			81		75						
sunny	2/9	3/5	mean	73 74.6	mean	79.1 86.2	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	std dev	6.2 7.9	std dev	10.2 9.7	true	3/9	3/5		
rainy	3/9	2/5									

numeric

Probability density function for normal distribution

$P(t = 66 | \text{yes})$

$$f(\text{temperature} = 66 | \text{yes}) = \frac{1}{6.2\sqrt{2\pi}} e^{-\frac{(66-73)^2}{2 \cdot 6.2^2}} = 0.034$$

μ for temp. for play=yes
 σ for temp. for play=yes

- Similarly: $f(\text{humidity} = 90 | \text{yes}) = 0.0221$

$$P(\text{yes}|E) = \frac{\frac{2}{9} 0.034 0.0221 \frac{3}{9} \frac{9}{14}}{P(E)} = \frac{0.000036}{P(E)}$$

$$P(\text{no}|E) = \frac{\frac{3}{5} 0.0279 0.038 \frac{3}{5} \frac{5}{14}}{P(E)} = \frac{0.000137}{P(E)}$$

- $P(\text{no}|E) > P(\text{yes}|E) \Rightarrow$ Naïve Bayes predicts play=no

- The same as before but now **income** is numeric attribute
- Predict the class of the following new example using Naïve Bayes:
home owner = no, marital status = married, annual income=120

	home owner	marital status	income (in K)	loan default
1	yes	single	125	yes
2	no	married	100	yes
3	no	single	70	no
4	yes	married	120	no
5	yes	divorced	95	yes
6	no	married	60	no
7	yes	divorced	220	no
8	no	single	85	yes
9	no	married	75	no
10	no	single	90	yes

1) Calculate the mean and standard deviation values **for income**:

$$\mu = \frac{\sum_{i=1}^n X_i}{n} \quad \sigma^2 = \frac{\sum_{i=1}^n (X_i - \mu)^2}{n-1}$$

We need to do this separately for class=yes and class=no:

class yes income		class no income
125		70
100		120
95		60
85		220
90		75
$\mu_{\text{income_yes}}=99$		$\mu_{\text{income_no}}=109$
$\sigma_{\text{income_yes}}=15.57$		$\sigma_{\text{income_no}}=66.18$

2) Calculate $P(\text{income}=120|\text{yes})$ and $P(\text{income}=120|\text{no})$ using the probability density function for normal distribution:

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

$$f(\text{income} = 120|\text{yes}) = \frac{1}{15.57\sqrt{2\pi}} e^{-\frac{(120-99)^2}{2*15.57^2}} = 0.01032$$

$$f(\text{income} = 120|\text{no}) = \frac{1}{66.18\sqrt{2\pi}} e^{-\frac{(120-109)^2}{2*66.18^2}} = 0.00595$$

class yes income		class no income
125		70
100		120
95		60
85		220
90		75
$\mu_{\text{income_yes}}=99$		$\mu_{\text{income_no}}=109$
$\sigma_{\text{income_yes}}=15.57$		$\sigma_{\text{income_no}}=66.18$

3) Calculating the probabilities $P(\text{yes}|E)$ and $P(\text{no}|E)$ using the Bayes Theorem

- We already have the probabilities for the nominal attributes from the previous exercise:

$$P(\text{yes}|E) = \frac{\frac{3}{5} \frac{1}{5} 0.01032 \frac{5}{10}}{P(E)} = \frac{0.000619}{P(E)}$$

$$P(\text{no}|E) = \frac{\frac{3}{5} \frac{3}{5} 0.00595 \frac{5}{10}}{P(E)} = \frac{0.001071}{P(E)}$$

- $P(\text{no}|E) > P(\text{yes}|E) \Rightarrow$ Naïve Bayes predicts **loan default = no** for the new example

- Probabilities are calculated easily due to the independence assumption
- Fast - requires 1 scan of the training data to calculate all statistics for both nominal and continuous attributes
- In many cases outperforms more sophisticated learning methods
- Robust to isolated noise points – such points have only negligible impact on the conditional probabilities
- Correlated attributes reduce the power of Naïve Bayes - violation of the independence assumption
 - Solution: apply feature selection beforehand to identify and discard correlated (redundant) attributes
- Normal distribution assumption for numeric attributes - many features are not normally distributed – solutions:
- Discretize the data first, i.e. numerical -> nominal attributes
- Use other probability density functions, e.g. Poisson, binomial, gamma



Evaluating Machine Learning Algorithms

- How to evaluate the **generalization performance** of ML models?
 - i.e. the performance on new, unseen data
- Evaluation procedures?
- Performance measures?



Evaluation Procedures

- Holdout method
- Cross validation
- Leave-one-out cross validation
- Cross-validation for parameter tuning

- Split the data randomly into 2 sets: **training set** and **test set**
 - typically 2/3 and 1/3
- **Build the model** using the training data
- **Evaluate the model** on the test data
 - calculate **accuracy** or other performance measures

- Accuracy = proportion of correctly classified examples
 - predicted class = actual class
- We can calculate it on training and test set
 - **accuracy on training set** - overly optimistic, not a good indicator of generalization performance
 - **accuracy on test set** – used to evaluate generalization performance

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

training data

outlook	temp.	humidity	windy	play
sunny	hot	normal	false	no
overcast	mild	normal	false	yes
rainy	hot	normal	false	no
rainy	cool	high	true	no
sunny	mild	high	true	no

test data

Classifier:

if humidity=high then play=yes

elseif humidity=normal then play=no

Accuracy on test set =?

2
5

- Sometimes we need to use a third set: validation set
 - The data is split into: training, validation and test set
- For example, some classification methods (decision trees, neural networks) operate in two stages:
 - Stage 1: Build the classifier
 - Stage 2: Tune its **hyperparameters**
 - The test data can not be used for hyperparameter tuning
- Proper evaluation procedure - 3 datasets:
 - 1) Training set - to build the classifier
 - 2) Validation set - to tune its hyperparameters
 - 3) Test set - to evaluate accuracy

- Hyperparameter – parameter that can be tuned to optimize the performance of a ML algorithm
- Different from basic parameter that is part of a model, such as a coefficient in a linear regression model
- Examples of hyperparameters
 - in k-nearest neighbor algorithm: k
 - in regularized linear/logistic regression: λ
 - in neural networks: number of hidden layers and nodes in them; number of training epochs, etc.

- Since the split into training and test set is random, without stratification some classes might be missing from the training or test sets, or be under-represented
 - e.g. if all examples with a certain class are missing in the training set (they went to the test set), the classifier cannot learn to predict this class
- Solution: stratification
- Can be used together with the holdout method -> an improved holdout method
- Ensures that each class is represented with approximately equal proportions in both data sets (training and testing)
 - e.g. if the class proportion in the whole dataset is 60% class1 and 40% class2, this ratio is maintained in the training and test split

- The holdout method can be made more reliable by repeating the random split into training and test set several times and calculating average accuracy
 - e.g. repeating 10 times: in each of the 10 runs, a certain proportion (e.g. 2/3) is randomly selected for training (possibly with stratification) and the remainder is used for testing
 - the 10 accuracies are averaged to produce an overall average accuracy
- This is called **repeated holdout method**
- We can do better than this, e.g., by preventing the overlapping between the test sets

- Avoids the overlapping test sets
- 10-fold cross-validation – typically used

Step 1: Split data into 10 sets set1,..., set10 of approximately equal size

Step 2: A classifier is built 10 times. Each time the testing is on 1 set (blue) and the training is on the remaining 9 sets together (white)

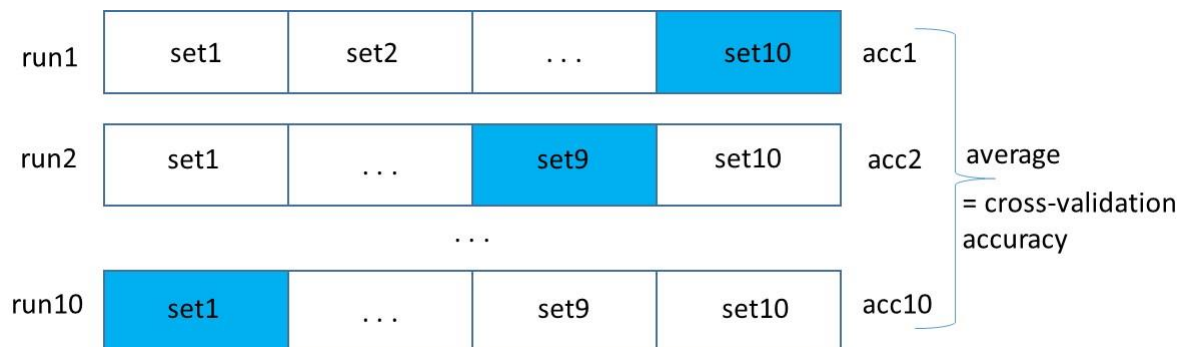
Run1: train on set1+...set9, test on set10 and calculate accuracy (acc1)

Run2: train on set1+...set8+set10, test on set9 and calculate accuracy (acc2)

....

Run10: train on set2+...set10, test on set1 and calculate accuracy (acc10)

Step 3: Calculate the cross validation accuracy = average (acc1, acc2,...acc10)



- **Stratified 10-fold cross-validation** – this is a standard method for evaluation used in ML
 - each subset is stratified
- Why 10?
 - Extensive experiments have shown that this is the best choice to get an accurate estimate
 - There is also some theoretical evidence for this
- Even better: repeated stratified 10-fold cross-validation
 - e.g. 10-fold cross-validation is repeated 10 times and results are averaged – reduces the variance in splitting the data

Leave-one-out cross-validation

- A special form of n-fold cross-validation
 - Set the number of folds to the number of training examples
 - => for n training examples, build classifier n times
- Advantages:
 - Makes the best use of data - the greatest possible amount of data is used for training
 - Deterministic procedure – no random sampling is involved - the same result will be obtained every time
- Disadvantage
 - High computational cost, especially for large datasets

- We can also use cross-validation to search through different parameter combinations and select the best one
- Let's consider k-Nearest Neighbor and 2 of its parameters - k and distance measure type; we can search through the following combinations:
 - number of nearest neighbours $k = 1, 3, 5, 11$ and 13
 - distance measure - Manhattan and Euclidean
- \Rightarrow 5×2 combinations of parameter values
- We would like to find the best combination - the one that will generalise well on new examples
- We will use the following procedure, called **grid-search with cross-validation for parameter tuning**

Grid search with cross-validation for parameter tuning

- Create the parameter grid (i.e. the parameter combinations)
- Split the data into training set and test set
- For each parameter combination:
 - Train a k-NN classifier on the training data using 10-fold cross-validation
 - Compute the cross-validation accuracy `cv_acc`
 - If `cv_acc > best_cv_acc`
 - `best_cv_acc = cv_acc`
 - `best_parameters = current parameters`
- Rebuild the k-NN model using the whole training data and `best_parameters`
- Evaluate it on the test data and report the results

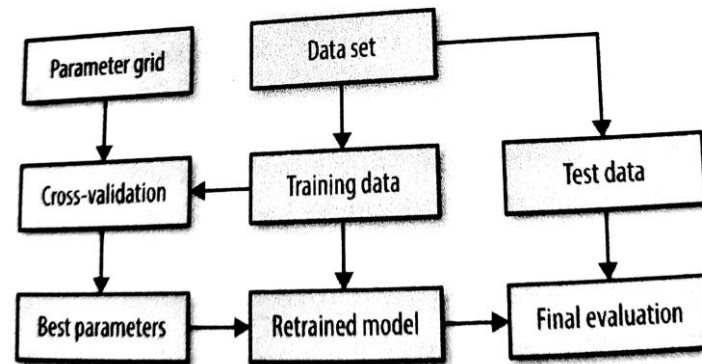


Image from Müller and Guido, Introduction to ML with Python

Grid search with cross-validation for parameter tuning (2)

- The data is split into training set and test set
 - The **cross-validation loop** uses the training data
 - It is performed for every parameter combination
 - Its purpose is to select the best parameter combination - the one with the highest cross-validation accuracy
 - This involves, for every parameter combination, building 10 models on 90% of the training data (9 folds) and evaluating them on the remaining 10% (1 fold)
 - Once this is done, a new model is trained using the selected best parameter combination on the **whole training set** and evaluated on the test set
- In sklearn, we can use GridSearchCV to do this



More Performance Measures

- Confusion matrix
- Recall, precision and F1 score

- 2 class problem: yes and no
- 4 different outcomes - confusion matrix:

examples	# assigned to class yes	# assigned to class no
# from class yes	true positives (tp)	false negatives (fn)
# from class no	false positives (fp)	true negatives (tn)

- accuracy in terms of tp, fn, fp and tn? $\text{accuracy} = (tp + tn) / (tp + fn + fp + tn)$
- The confusion matrix is **not** a performance measure, it allows us to calculate performance measures

Confusion matrix for more than two classes

- E.g. iris data classification - confusion matrix:

a b c <-- classified as

50 0 0 | a = Iris-setosa

0 44 6 | b = Iris-versicolor

0 3 47 | c = Iris-virginica

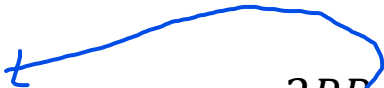
- accuracy = ?

- accuracy =
$$(50+44+47)/(50+0+0+0+44+6+0+3+47)$$
$$=141/150 = 94\%$$

- In addition to accuracy, other performance measures are **precision (P)**, **recall (R)** and their combination - **F1 score** classification

$$P = \frac{tp}{tp + fp}$$

$$R = \frac{tp}{tp + fn}$$

$$F1 = \frac{2PR}{P + R}$$


examples	# assigned to class yes	# assigned to class no
# from class yes	true positives (tp)	false negatives (fn)
# from class no	false positives (fp)	true negatives (tn)