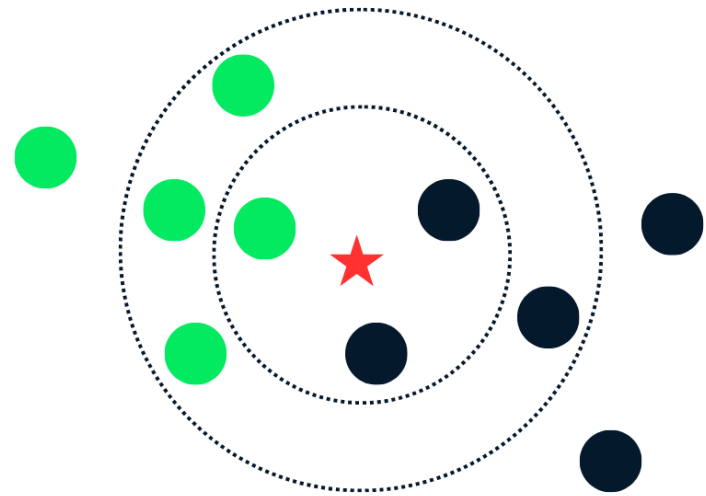


K-Nearest Neighbour (k NN) Classifier



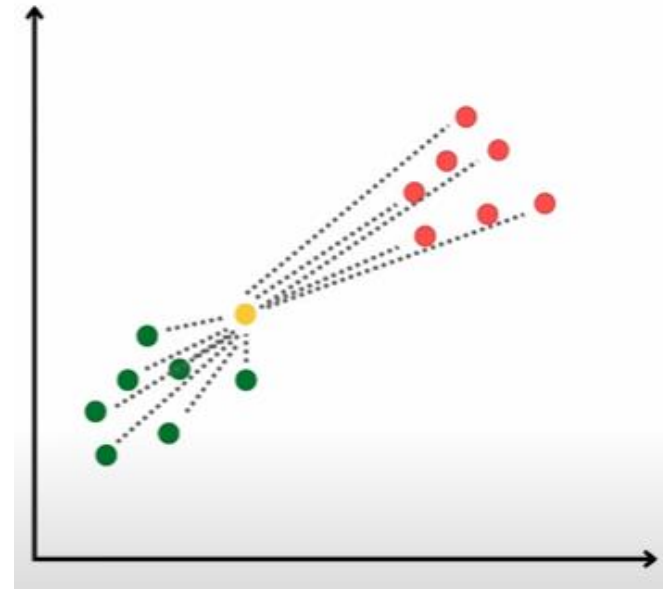
Similarity-Based Learning

- Similarity-based approaches to machine learning come from the idea that:
 - Similar examples have similar label, and
 - Classify new examples like similar training examples
- Requires a measure of similarity
- Algorithm:
 - Given some new example x for which we need to predict its class y
 - Find most similar training examples
 - Classify x “like” these most similar examples
- Questions:
 - How to determine similarity?
 - How many similar training examples to consider?

kNN Algorithm steps

Given a data point:

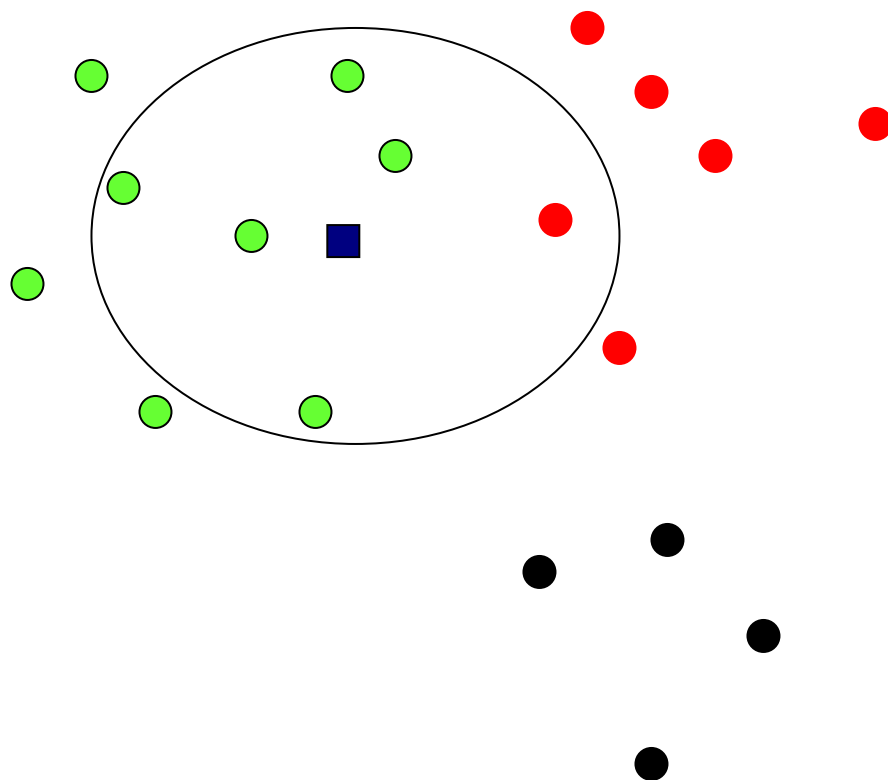
- Calculate its distance from all other data points in the dataset
- Get the closest **K** points
- Regression: Get the average of their values
- Classification: Get the label with majority vote



K-Nearest Neighbour Model

- **Lazy Learner:**
 - It does not build models explicitly
 - Unlike eager learners such as decision tree induction
 - Delaying the decision to the time of classification => ideal where data is constantly changing
- **Instance-based** learning, i.e., **no explicit model**, but rather stores instances of the training data
- **Non-parametric** algorithm, i.e., **no assumptions** about the underlying distribution of the data
- To classify an unseen record: compute its *proximity* to all training instances and locate **k** nearest neighbours
 - Class = majority vote
 - For regression take the average

Example - k=6 (6NN)



● Government

● Science

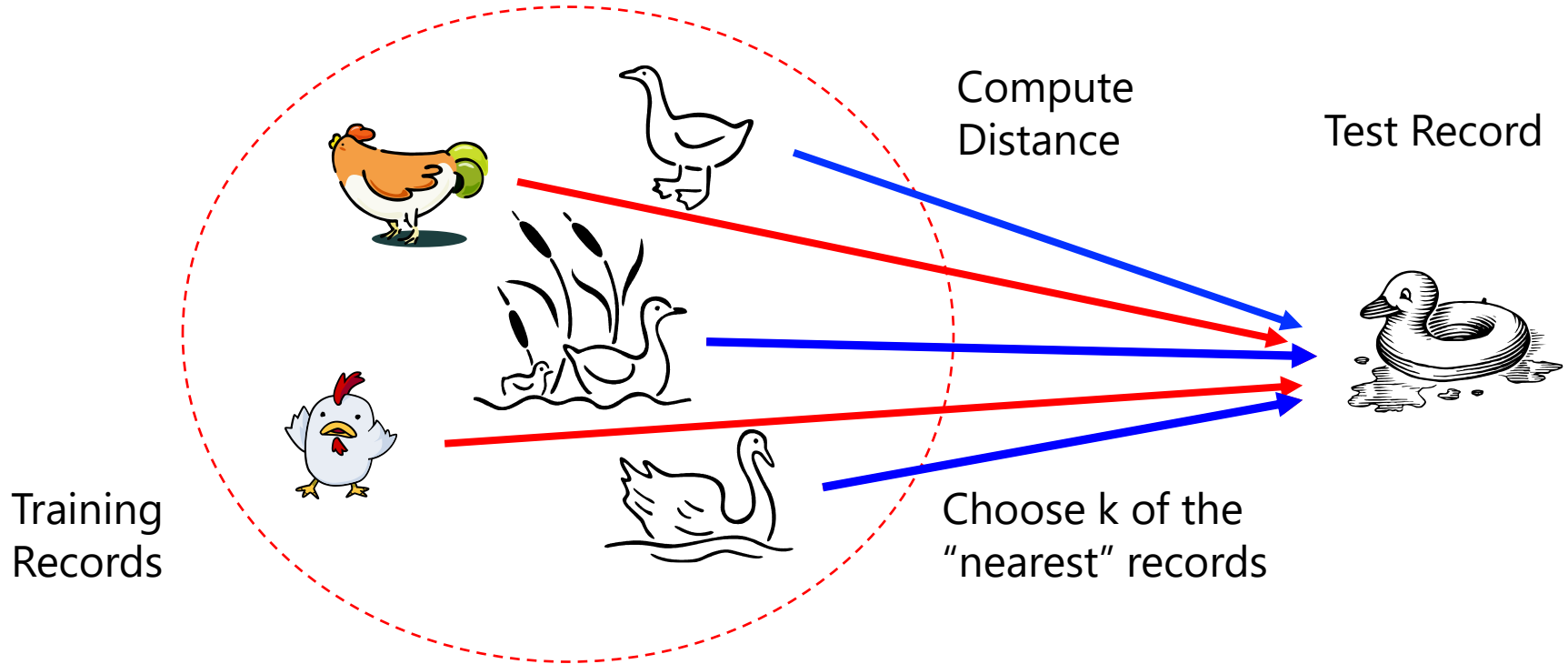
● Arts

A new point ■

$\Pr(\text{Govt} | \blacksquare) = 5/6$

Rationale

- “If it walks like a duck, quacks like a duck, then it’s probably a duck”



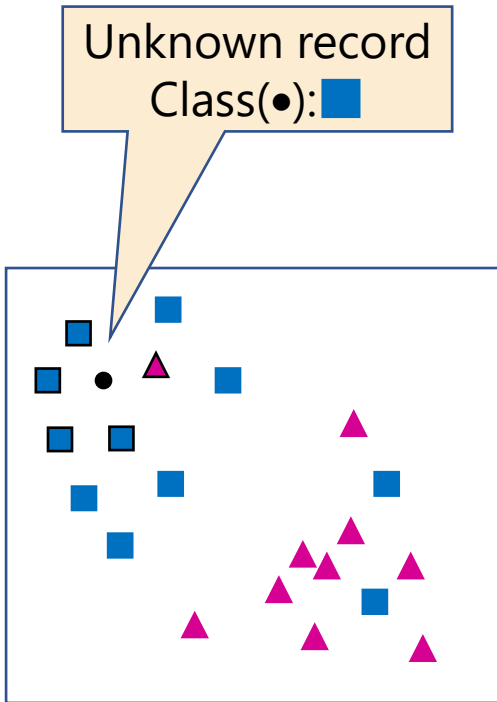
Requirements & Steps

- **Requirements:**

- The set of training records
- **Distance Metric** to compute distance between records
- The value of the **hyperparameter k** , i.e., the number of nearest neighbors to consider

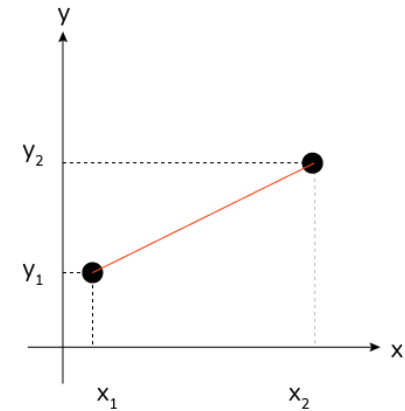
- **To predict an unknown record:**

- Compute distance to other training records
- Identify **k** nearest neighbors
- Take **majority vote** of **class labels** of nearest neighbors to determine the target **class label** of unknown record
 - in the case of a **tie**:
 - Use **odd k** (doesn't solve multi-class)
 - Choose the class label **randomly**
 - Pick the class **dominant** in the entire dataset



Euclidean Distance

- **Euclidean Distance** is a measure of the **straight-line distance** between two points (x_1, y_1) and (x_2, y_2) in a two-dimensional space



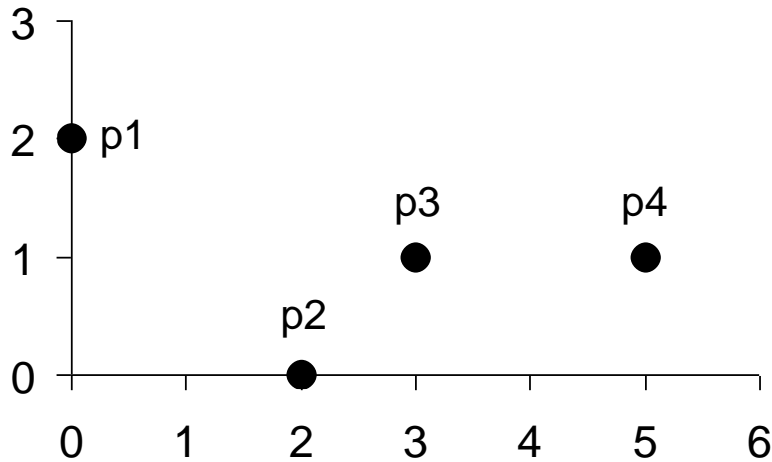
$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- In more general terms, for n-dimensional space, the Euclidean distance between two points **p** and **q** with coordinates (p_1, p_2, \dots, p_n) and (q_1, q_2, \dots, q_n) respectively, is given by:

$$\text{Distance} = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

- Numerical measure of **Dissimilarity/Distance**
 - How **different** are two data objects
 - Lower when objects are more alike

Euclidean Distance - Example



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Distance Matrix

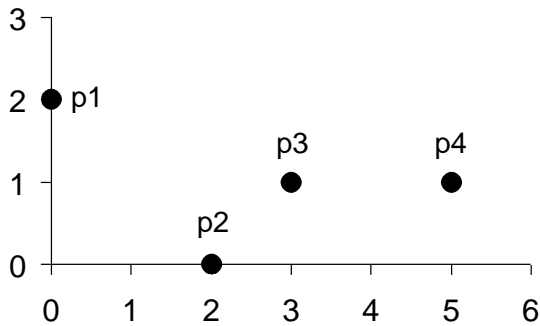
	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance Metrics

- Euclidean distance:
$$d_{Eucl}(a, b) = \sqrt{\sum_{k=1}^m (a_k - b_k)^2}$$
- Manhattan distance:
$$d_{Manh}(a, b) = \sum_{k=1}^m |a_k - b_k|$$
- Minkowski distance:
$$d_{Mink}(a, b) = \sqrt[p]{\sum_{k=1}^m |a_k - b_k|^p}$$
- When $p = 1$, the Minkowski distance is the Manhattan distance
- When $p = 2$, the Minkowski distance is the Euclidean distance
- Although there are infinite number of Minkowski-based distance metrics to choose from, Euclidean distance and Manhattan distance are the most commonly used ones

Minkowski Distance: Example

$$dist = (\sum_{k=1}^n |p_k - q_k|^r)^{\frac{1}{r}}$$



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

L1, r=1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

Distance Matrix

L2, r=2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

L∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Similarity Between Binary Vectors

- Common situation is that objects, p and q , have only binary attributes
- Compute similarities using the following quantities

M_{01} = the number of attributes where p was 0 and q was 1

M_{10} = the number of attributes where p was 1 and q was 0

M_{00} = the number of attributes where p was 0 and q was 0

M_{11} = the number of attributes where p was 1 and q was 1

- Simple Matching and Jaccard Coefficients

$$\begin{aligned}\text{SMC} &= \text{number of matches} / \text{number of attributes} \\ &= (M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00})\end{aligned}$$

$$\begin{aligned}J &= \text{number of 11 matches} / \text{number of not-both-zero attributes values} \\ &= (M_{11}) / (M_{01} + M_{10} + M_{11})\end{aligned}$$

SMC versus Jaccard - Example

$$p = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$$

$$q = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

$M_{01} = 2$ (the number of attributes where p was 0 and q was 1)

$M_{10} = 1$ (the number of attributes where p was 1 and q was 0)

$M_{00} = 7$ (the number of attributes where p was 0 and q was 0)

$M_{11} = 0$ (the number of attributes where p was 1 and q was 1)

$$\begin{aligned} \text{SMC} &= (M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00}) \\ &= (0+7) / (2+1+0+7) = 0.7 \end{aligned}$$

$$J = (M_{11}) / (M_{01} + M_{10} + M_{11}) = 0 / (2 + 1 + 0) = 0$$

Cosine Similarity

- If d_1 and d_2 are two vectors, then

$$\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \|d_2\|$$

where \bullet indicates vector dot product and $\|d\|$ is the length of vector d

- Example:

$$d_1 = \mathbf{3\ 2\ 0\ 5\ 0\ 0\ 0\ 2\ 0\ 0}$$

$$d_2 = \mathbf{1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 2}$$

$$d_1 \bullet d_2 = 3 \times 1 + 2 \times 0 + 0 \times 0 + 5 \times 0 + 0 \times 0 + 0 \times 0 + 0 \times 0 + 2 \times 1 + 0 \times 0 + 0 \times 2 = 5$$

$$\|d_1\| = (3^2 + 2^2 + 0^2 + 5^2 + 0^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2)^{0.5} = (42)^{0.5} = 6.481$$

$$\|d_2\| = (1^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 1^2 + 0^2 + 2^2)^{0.5} = (6)^{0.5} = 2.245$$

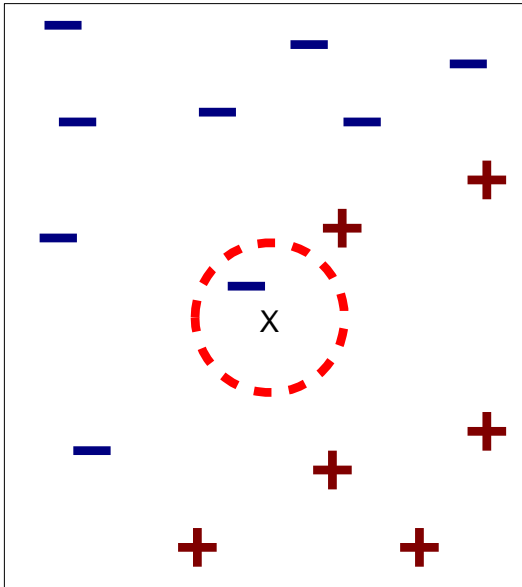
$$\cos(d_1, d_2) = \mathbf{.3150}$$

Extended Jaccard Coefficient (Tanimoto)

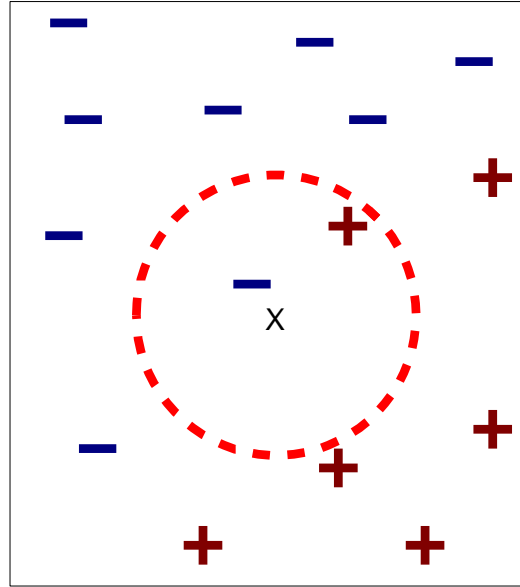
- Variation of Jaccard for continuous or count attributes
 - Reduces to Jaccard for binary attributes

$$T(p, q) = \frac{p \bullet q}{\|p\|^2 + \|q\|^2 - p \bullet q}$$

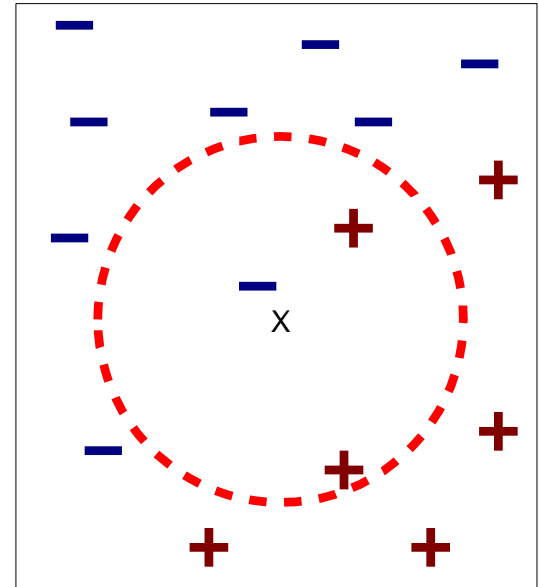
Definition of Nearest Neighbor



(a) 1-nearest neighbor



(b) 2-nearest neighbor

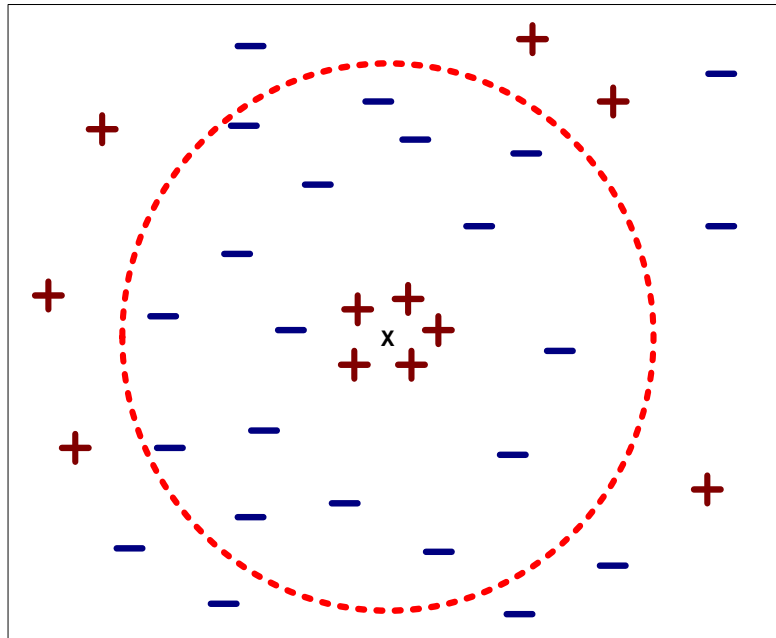


(c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

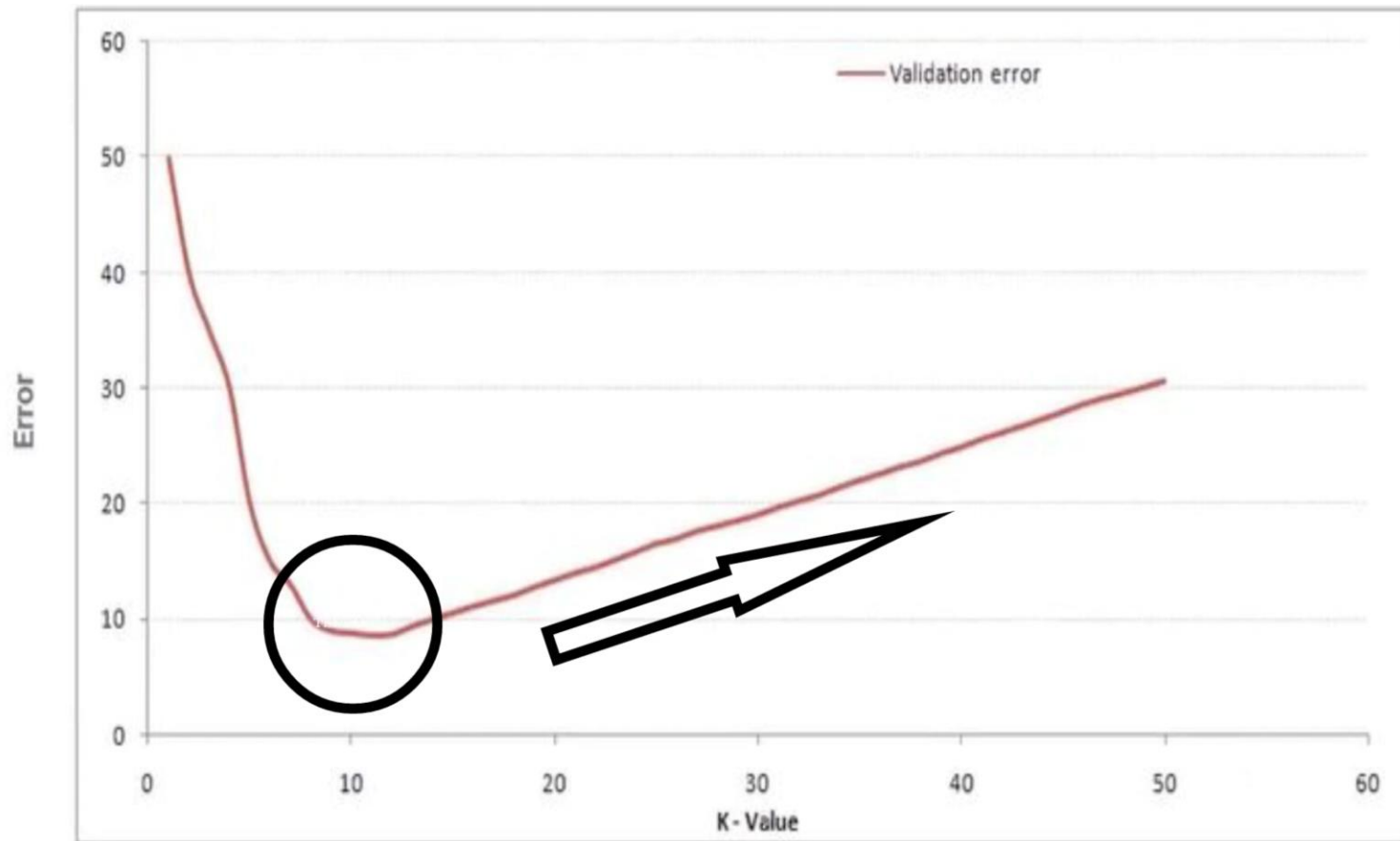
Choosing the value of k

- If k is too small, sensitive to noise points (results in **high variance, overfitting**)
- If k is too large, neighborhood may include points from other classes



The Elbow Method

- Using a validation set



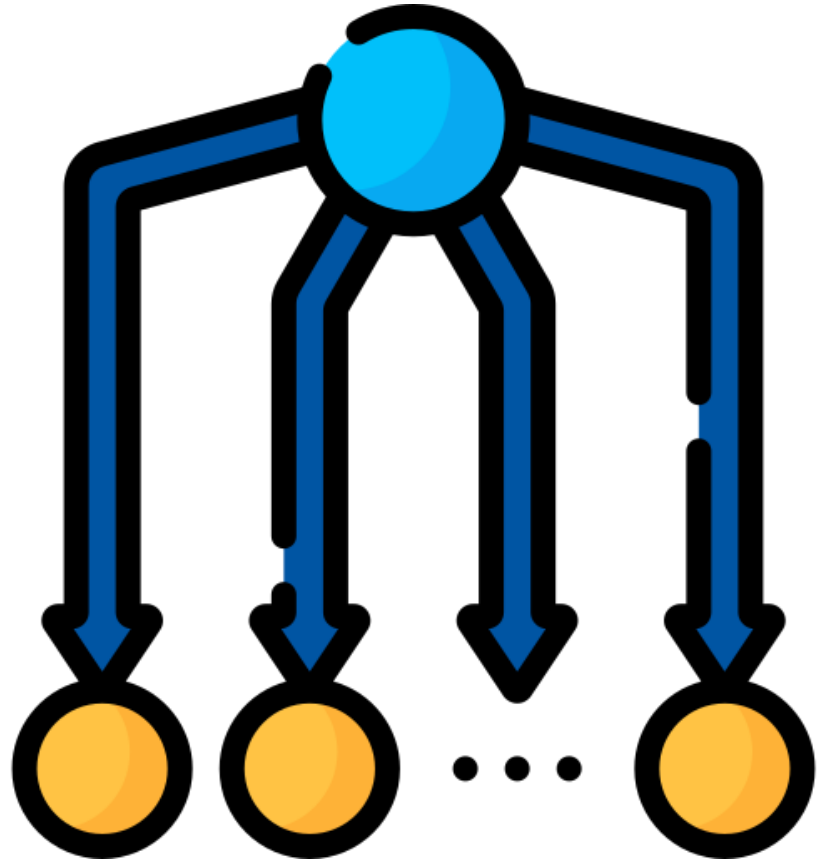
K-NNs Properties

- k-NNs are lazy (or instance-based) learners
 - It does not build models explicitly
 - Unlike eager learners such as decision tree induction
- Simple to implement algorithm
- Feature scaling is necessary, if scales differ, to prevent distance measures from being dominated by attributes having higher magnitude, e.g.,
 - height of a person 1.5m to 1.8m, weight 90lb to 300lb and monthly income QR10K to QR500K
- Requires little tuning
- Often performs quite well!
 - Try it first on a new learning problem
- Predicting unknown records are relatively expensive
 - Must make a pass through the entire dataset for each classification. This can be prohibitive for large data sets

KNN Limitations

- **Slow** when the training data set is large, as it requires calculating distances to every training data point
- It doesn't work well with high-dimensional data, as the **curse of dimensionality** can lead to inaccurate results
 - Prediction accuracy can quickly degrade when number of attributes grows
- **Sensitive** to the choice of distance metric, and different distance metrics may lead to different results
- It doesn't handle **imbalanced data** well, as it may assign the majority class label to new data points
- **Missing values** must be processed with little effect on the distances. Replace with average is reasonable but may have side effects

Naïve Bayes Classifier



Naïve Bayes Classifier

- A probabilistic approach to build a classifier based on Bayes Theorem
- Determine the most probable class label for unseen instance
 - Based on the observed instance attributes
 - Example:
 - Based on the object's attributes {shape, color, weight}
 - A given object that is {spherical, yellow, < 60 grams}, may be classified as a tennis ball
- "naive" as it assumes that the features are independent of each other, which is often not the case in real-world datasets
 - Nevertheless, it is still widely used and can achieve high accuracy in many applications

Naïve Bayesian Classifier - Applications

- **Text classification:** for spam filtering, sentiment analysis, and topic classification
- **Image classification:** for face recognition, object detection, and image segmentation
- **Medical diagnosis:** for disease prediction and risk assessment
- **Recommendation systems:** for personalized product recommendations and content recommendations

Example of Bayes Theorem

- Given:
 - A doctor knows that meningitis causes stiff neck 50% of the time
 - Prior probability of any patient having meningitis is 1/50,000 (*prior knowledge*)
 - Prior probability of any patient having stiff neck is 1/20
- If a patient has stiff neck, what's the probability he/she has meningitis? (**Conditional Probability**)

$$P(M|S) = \frac{P(S|M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

- Heart disease dataset. Class label P means has heart disease, and the class label N means that the disease is absent
- Prior Probabilities for the Class labels: {P, N}**
 - $P(\text{Class} = P) = 6/10 = 0.6$
 - $P(\text{Class} = N) = 4/10 = 0.4$
- Conditional Probabilities given Class = P**
 - $P(\text{BloodPressure}=\text{high} \mid \text{Class}=P) = 4/6$
 - $P(\text{BloodPressure}=\text{normal} \mid \text{Class}=P) = 2/6$
 - $P(\text{BloodSugarLevel}=1 \mid \text{Class}=P) = 1/6$
 - $P(\text{BloodSugarLevel}=2 \mid \text{Class}=P) = 2/6$
 - $P(\text{BloodSugarLevel}=3 \mid \text{Class}=P) = 3/6$
 - $P(\text{Habit}=\text{smoker} \mid \text{Class}=P) = 5/6$
 - $P(\text{Habit}=\text{nonsmoker} \mid \text{Class}=P) = 1/6$
- Conditional Probabilities given Class = N**
 - $P(\text{BloodPressure}=\text{high} \mid \text{Class}=N) = 2/4$
 - $P(\text{BloodPressure}=\text{normal} \mid \text{Class}=N) = 2/4$
 - $P(\text{BloodSugarLevel}=1 \mid \text{Class}=N) = 1/4$
 - $P(\text{BloodSugarLevel}=2 \mid \text{Class}=N) = 2/4$
 - $P(\text{BloodSugarLevel}=3 \mid \text{Class}=N) = 1/4$
 - $P(\text{Habit}=\text{smoker} \mid \text{Class}=N) = 1/4$
 - $P(\text{Habit}=\text{nonsmoker} \mid \text{Class}=N) = 3/4$

Naïve Bayes Classifier - Example



Blood Pressure	Blood Sugar Level	Habit	Class
High	3	Smoker	P
High	3	Nonsmoker	P
High	2	Smoker	P
High	1	Smoker	P
Normal	3	Smoker	P
Normal	2	Smoker	P
High	2	Nonsmoker	N
High	2	Nonsmoker	N
Normal	3	Smoker	N
Normal	1	Nonsmoker	N

Classify patient A (1/2)

- Classify the unseen patient A using the Naïve Bayes classifier

Blood Pressure	Blood Sugar Level	Habit	Class
High	2	Nonsmoker	?

- Calculate **$P(A \mid \text{Class} = P)$** and **$P(A \mid \text{Class} = N)$**

$$\begin{aligned} P(A \mid \text{Class} = P) &= P(\text{BloodPressure} = \text{Normal} \mid \text{Class} = P) \\ &\times P(\text{BloodSugarLevel} = 1 \mid \text{Class} = P) \\ &\times P(\text{Habit} = \text{nonsmoker} \mid \text{Class} = P) \\ &= 4/6 \times 1/6 \times 1/6 = 4 / (6 \times 6 \times 6) \end{aligned}$$

$$\begin{aligned} P(A \mid \text{Class} = N) &= P(\text{BloodPressure} = \text{High} \mid \text{Class} = N) \\ &\times P(\text{BloodSugarLevel} = 1 \mid \text{Class} = N) \\ &\times P(\text{Habit} = \text{nonsmoker} \mid \text{Class} = N) \\ &= 2/4 \times 1/4 \times 3/4 = 6/64 \end{aligned}$$

Classify patient A (2/2)

- Calculate **$P(\text{Class}=\text{P} \mid \text{A})$** and **$P(\text{Class}=\text{N} \mid \text{A})$**

$$\begin{aligned} P(\text{Class}=\text{P} \mid \text{A}) &\approx P(\text{A} \mid \text{Class}=\text{P}) \times P(\text{Class}=\text{P}) \\ &= 4 / (6 \times 6 \times 6) \times (6/10) = 4/360 \end{aligned}$$

$$\begin{aligned} P(\text{Class}=\text{N} \mid \text{A}) &\approx P(\text{A} \mid \text{Class}=\text{N}) \times P(\text{Class}=\text{N}) \\ &= (6/64) \times (4/10) = 6/160 \end{aligned}$$

Since $P(\text{Class}=\text{N} \mid \text{A}) > P(\text{Class}=\text{P} \mid \text{A})$

Therefore, assign the patient A class **N** (not having a heart disease)

Naïve Bayes Classifier

Bayes theorem for a
Single predictor variable A

Probability of C given A is
observed
(i.e., conditional probability)

$$P(C|A) = \frac{P(A|C)P(C)}{P(A)}$$

Now we have object **A** having **m**
attributes: $a_1, a_2, a_3, \dots, a_m$

C is the class label:
 $C \in \{C_1, C_2, \dots, C_n\}$

$$P(C_i|A) = \frac{P(a_1, a_2, \dots, a_m|C_i)P(C_i)}{P(a_1, a_2, \dots, a_m)} \quad i = 1, 2, \dots, n$$

Apply the Naïve Assumption and Remove a Constant

- For observed attributes $A = (a_1, a_2, \dots, a_m)$, we want to compute

$$P(C_i|A) = \frac{P(a_1, a_2, \dots, a_m|C_i)P(C_i)}{P(a_1, a_2, \dots, a_m)} \quad i = 1, 2, \dots, n$$

and assign to the object A the class C_i having the highest probability $P(C_i|A)$

- Two simplifications to the calculations
 - ▶ Apply naïve assumption - each a_j is conditionally independent of each other, then

$$P(a_1, a_2, \dots, a_m|C_i) = P(a_1|C_i)P(a_2|C_i) \cdots P(a_m|C_i) = \prod_{j=1}^m P(a_j|C_i)$$

- ▶ Denominator $P(a_1, a_2, \dots, a_m)$ is a constant and can be ignored

Building a Naïve Bayesian Classifier

- Applying the two simplifications

$$P(C_i|A) = P(C_i|a_1, a_2, \dots, a_m) = \left(\prod_{j=1}^m P(a_j|C_i) \right) P(C_i) \quad i = 1, 2, \dots, n$$

- To build a Naïve Bayesian Classifier, collect the following statistics from the training data:
 - **P(C_i)** for every class labels
 - **P(a_j| C_i)** for every attribute **a_j** given class **C_i**
 - Fraction of training examples of class **C_i** that take the attribute value **a_j**
 - If attribute **a_j** is continuous then discretize first
 - Assign to the object A the class C_i having the highest probability P(C_i| A)

Naïve Bayes Classifier for the Credit Example

- Class labels: {good, bad}
 - $P(\text{Class} = \text{good}) = 19/27 = 0.7$
 - $P(\text{Class} = \text{bad}) = 8/27 = 0.3$
- Conditional Probabilities
 - $P(\text{Housing} = \text{own} \mid \text{bad}) = 4/8 = 0.5$
 - $P(\text{Housing} = \text{own} \mid \text{good}) = 13/19 = 0.68$
 - $P(\text{Housing} = \text{rent} \mid \text{bad}) = 4/8 = 0.5$
 - $P(\text{Housing} = \text{rent} \mid \text{good}) = 6/19 = 0.316$
 - ... and so on

Job	Housing	Savings	Credit_class
self_employed	own	low	good
skilled	own	low	good
skilled	own	high	good
skilled	own	high	good
skilled	own	high	good
skilled	rent	low	good
skilled	own	low	good
skilled	rent	low	good
skilled	own	high	good
skilled	own	high	good
skilled	rent	low	good
skilled	rent	low	good
skilled	own	high	good
skilled	rent	low	good
skilled	own	high	good
skilled	own	high	good
skilled	own	high	good
skilled	own	high	good
skilled	own	low	bad
self_employed	rent	low	bad
self_employed	own	low	bad
self_employed	own	high	bad
self_employed	rent	low	bad
self_employed	rent	low	bad
self_employed	rent	low	bad
self_employed	own	high	bad

Naïve Bayesian Classifier for a Particular Applicant

- Given applicant attributes of

$A = \{\text{Housing} = \text{own}, \text{Job} = \text{self-employed}, \text{savings} = \text{low}\}$

- We need to compare $P(\text{bad}|A)$ against $P(\text{good}|A)$, and we also need to calculate $P(A|\text{bad})$ and $P(A|\text{good})$

$$P(\text{good}|A) \approx P(A|\text{good}) \times P(\text{good})$$

$$P(\text{bad}|A) \approx P(A|\text{bad}) \times P(\text{bad})$$

$$P(\text{good}|A) \approx P(A|\text{good}) \times 0.7$$

$$P(\text{bad}|A) \approx P(A|\text{bad}) \times 0.3$$

a_j	C_i	$P(a_j C_i)$
Housing = own	good	$13/19 = 0.68$
Housing = own	bad	$4/8 = 0.5$
Job = self emp	good	$1/19 = 0.05$
Job = self emp	bad	$6/8 = 0.75$
Savings = low	good	$9/19 = 0.73$
Savings = low	bad	$7/8 = 0.875$

$$\begin{aligned} P(A | \text{good}) &= P(\text{Housing} = \text{own} | \text{good}) \times P(\text{Job} = \text{self-employed} | \text{good}) \times P(\text{Savings} = \text{low} | \text{good}) \\ &= 0.68 \times 0.05 \times 0.73 = 0.2482 \end{aligned}$$

$$\begin{aligned} P(A | \text{bad}) &= P(\text{Housing} = \text{own} | \text{bad}) \times P(\text{Job} = \text{self-employed} | \text{bad}) \times P(\text{Savings} = \text{low} | \text{bad}) \\ &= 0.5 \times 0.75 \times 0.875 = 0.3281 \end{aligned}$$

$$P(\text{good}|A) \approx 0.2482 \times 0.7 = 0.0174$$

$$P(\text{bad}|A) \approx 0.3281 \times 0.3 = 0.0984$$

Since $P(\text{bad}|A) > P(\text{good}|A)$, assign the applicant the label **"bad"** credit

Naïve Bayesian Implementation Considerations

- Zero probabilities due to unobserved attribute/classifier pairs
 - Resulting from rare events
 - Handled by smoothing (adjusting each probability by a small amount)

Laplace smoothing (correction)

Adds **1** to the count and adds the **number of attributes** to the total number of records for each class. Examples:

Test Record = (Attribute1=A, Attribute2=D, Attribute3 =F)

	Before Correction	Laplace Correction
$P(A \mid \text{Yes})$	$2/4$	$(2+1)/(4+3) = 3/7$
$P(D \mid \text{Yes})$	$1/4$	$(1+1)/(4+3) = 2/7$
$P(F \mid \text{Yes})$	$2/4$	$(2+1)/(4+3) = 3/7$
$P(A \mid \text{No})$	$0/4$	$(0+1)/(4+3) = 1/7$
$P(D \mid \text{No})$	$4/4$	$(4+1)/(4+3) = 5/7$
$P(F \mid \text{No})$	$3/4$	$(3+1)/(4+3) = 4/7$

Attribute1	Attribute 2	Attribute 3	Class
A	C	E	Yes
B	D	F	Yes
A	C	F	Yes
B	C	E	Yes
B	D	F	No
B	D	F	No
B	D	F	No
B	D	F	No

Before Correction:

$$P((\text{Attribute1}=\text{A}, \text{Attribute2}=\text{D}, \text{Attribute3}=\text{F}) \mid \text{Yes}) = 2/4 \times 1/4 \times 2/4 = 4/64$$

$$P((\text{Attribute1}=\text{A}, \text{Attribute2}=\text{D}, \text{Attribute3}=\text{F}) \mid \text{No}) = 0/4 \times 4/4 \times 3/4 = 0$$

After Correction:

$$P((\text{Attribute1}=\text{A}, \text{Attribute2}=\text{D}, \text{Attribute3}=\text{F}) \mid \text{Yes}) = 3/7 \times 2/7 \times 3/7 = 18/343 = 0.052$$

$$P((\text{Attribute1}=\text{A}, \text{Attribute2}=\text{D}, \text{Attribute3}=\text{F}) \mid \text{No}) = 1/7 \times 5/7 \times 4/7 = 20/343 = 0.058$$

We have:

$$P(\text{record} \mid \text{Yes}) \times P(\text{Yes}) > P(\text{record} \mid \text{No}) \times P(\text{No})$$

$$(4/64 \times 0.5) > (0 \times 0.5)$$

The record should be classified with class label

Yes

We have:

$$P(\text{record} \mid \text{Yes}) \times P(\text{Yes}) < P(\text{record} \mid \text{No}) \times P(\text{No})$$

$$(0.052 \times 0.5) < (0.058 \times 0.5)$$

The record should be classified with class label

No

Smoothing of Conditional Probabilities

- *Idea*: Instead using very small non-zeros values, such as 0.00001

Original: $P(x|y) = \frac{n_{xy}}{n_y}$

Laplace: $P(x|y) = \frac{n_{xy} + 1}{n_y + |X|}$

Laplace Smoothing is a technique used to **handle the problem of zero probabilities**, when calculating conditional probabilities from a dataset, for cases where certain combinations of variables have not been observed

- n_{xy} count of training examples that have the value y
- n_{xy} count of x given y
- $|X|$ count possible values for variable x

Naïve Bayesian Implementation Considerations

- Numerical underflow
 - Resulting from multiplying several probabilities near zero
 - Preventable by computing the logarithm of the products

$$\prod_{j=1}^m P(a_j | c_i)$$

$$\sum_{j=1}^m \log(P(a_j | c_i))$$

```
print("Smallest Number for a float in Python")
print(sys.float_info.min)

print("\n\nProbabilities 1")
prob1 = np.random.uniform(0.001,0.05,200)
print("Probabilities (first five values)" , prob1[:5]) #The first 5
print("Product of Probabilities" , np.prod(prob1))
print("Sum of the logs of Probabilities" ,np.sum(np.log(prob1)))

print("\n\nProbabilities 2")
prob2 = np.random.uniform(0.02,0.05,200)
print("Probabilities (first five values)" , prob2[:5]) #The first 5
print("Product of Probabilities" , np.prod(prob2))
print("Sum of the logs of Probabilities" ,np.sum(np.log(prob2)))
```

Smallest Number for a float in Python
2.2250738585072014e-308

Probabilities 1
Probabilities (first five values) [0.03739613 0.04330825 0.02155112 0.01259336 0.00890564]
Product of Probabilities 0.0
Sum of the logs of Probabilities -764.5679963402812

Probabilities 2
Probabilities (first five values) [0.04485225 0.04126788 0.03125973 0.02053821 0.02056159]
Product of Probabilities 9.529852558582343e-294
Sum of the logs of Probabilities -674.7055880939956

Naïve Bayesian Implementation Considerations

- Assign the classifier label, C_i , that maximizes the value of

$$P(C_i|A) = \left(\prod_{j=1}^m P(a_j|C_i) \right) * P(C_i)$$

$$\left(\sum_{j=1}^m \log P(a_j|C_i) \right) + \log P(C_i)$$

where $i = 1, 2, \dots, n$ and P denotes the probabilities

```
| P_A_good = np.array([0.68,0.05,0.73])
p_good = 0.7
print("P(A|good) : ", np.prod(P_A_good))
print("P(A|good) * P(good) : ", np.prod(P_A_good)*p_good)
print("sum(log(P(A|good))) : ", np.sum(np.log(P_A_good)))
print("sum(log(P(A|good))) + log(P(good)) : ", np.sum(np.log(P_A_good)) + np.log(p_good))
```

```
P(A|good) : 0.024820000000000002
P(A|good) * P(good) : 0.017374
sum(log(P(A|good))) : -3.6961054992056757
sum(log(P(A|good))) + log(P(good)) : -4.0527804431444086
```

```
| P_A_bad = np.array([0.5,0.75,0.875])
p_bad = 0.3
print("P(A|bad) : ", np.prod(P_A_bad))
print("P(A|bad) * P(bad) : ", np.prod(P_A_bad)*p_bad) #P(A|bad)*P(bad)
print("sum(log(P(A|bad))) : ", np.sum(np.log(P_A_bad)))
print("sum(log(P(A|bad))) + log(P(bad)) : ", np.sum(np.log(P_A_bad)) + np.log(p_bad))
```

```
P(A|bad) : 0.328125
P(A|bad) * P(bad) : 0.0984375
sum(log(P(A|bad))) : -1.114360645636249
sum(log(P(A|bad))) + log(P(bad)) : -2.318333449962185
```



Naïve Bayesian Classifier – Reasons to Choose (+) and Cautions (-)

Reasons to Choose (+)	Cautions (-)
Handles missing values well: Simply ignores them when estimating the probabilities	Naïve assumption may not hold in real-world datasets, which can affect the accuracy
Robust to irrelevant variables	Numeric variables have to be discrete (categorized) intervals
Easy to implement	May not work well with imbalanced datasets, where one class has significantly more samples than the other classes
Resistant to over-fitting	
Computationally efficient Handles very high dimensional problems Handles categorical variables with a lot of levels	