

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY  
UNIVERSITY OF TECHNOLOGY  
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



## DISCRETE STRUCTURES FOR COMPUTING (CO1007)

---

Assignment (Semester 203, Duration: 04 weeks)

# “Bayesian Networks” (Version 0.1)

---

Advisor: Nguyễn Tiến Thịnh  
Nguyễn An Khương  
Students: Trần Công Huy Hoàng - 2052482  
Nguyễn Duy Bảo - 2052399  
Trần Trang Kỳ Phong - 2053325  
Nguyễn Phước Nguyên Phúc - 2053342

HO CHI MINH CITY, AUGUST 2021



## Contents

<b>1</b>	<b>Member list &amp; Workload</b>	<b>2</b>
<b>2</b>	<b>Preliminary knowledge</b>	<b>2</b>
2.1	Probability (Exercise 1)	2
2.1.1	Discrete probability (Exercise 1a)	2
2.1.2	Discrete random variables (Exercise 1b)	7
2.2	Graphs (Exercise 2)	10
2.2.1	Undirected Graphs (Exercise 2a)	10
2.2.2	Directed graphs (Exercise 2b)	15
2.2.3	Paths and circuits (Exercise 2c)	21
2.3	Bayesian networks (Exercise 3)	22
2.3.1	Bayesian models (Exercise 3a)	22
2.3.2	Variable Elimination technique and inference (Exercise 3b)	24
2.3.3	Bayesian model (Exercise 3c)	28
<b>3</b>	<b>Applications</b>	<b>30</b>
3.1	Data preprocessing	30
3.1.1	Prepare a pandas data frame (Exercise 4a)	30
3.1.2	Find and fill in or remove missing values in the data (Exercise 4b)	32
3.1.3	Discretize your data set (Exercise 4c)	36
3.1.4	Save your prepared data as a .csv file for later uses (Exercise 4d)	38
3.2	Problem modeling	38
3.2.1	Overview of the data (Exercise 5a)	38
3.2.2	Plots (Exercise 5b)	39
3.2.3	Prepare Bayesian Model (Exercise 5c)	44
<b>4</b>	<b>Training and predicting</b>	<b>45</b>
4.1	Exercise 6a	45
4.2	Exercise 6b	50
4.3	Exercise 6c	52
4.4	Exercise 6d	53
4.5	Exercise 6e	55

## 1 Member list & Workload

No.	Fullname	Student ID	Problems	Percentage of work
1	Trần Công Huy Hoàng	2052482	- Exercise 2 and 3 - Exercise 4, 5, 6	25%
2	Nguyễn Duy Bảo	2052399	- Exercise 2 and 3 - Exercise 4, 5, 6	25%
2	Trần Trang Kỳ Phong	2053325	- Exercise 1 and 3 - Exercise 4, 5, 6	25%
2	Nguyễn Phước Nguyên Phúc	2053342	- Exercise 1 and 3 - Exercise 4, 5, 6	25%

## 2 Preliminary knowledge

### 2.1 Probability (Exercise 1)

#### 2.1.1 Discrete probability (Exercise 1a)

- **Experiments:**

- **Definition:** Something that is done that produces measurable results.
- **Examples:** toss a coin, roll a dice, pick a random number, deal cards, open a box,...

- **Sample spaces:**

- **Definition:** is a set of all outcomes of an experiment.
- **Examples:** toss a coin{tail,head}, roll a dice{1,2,3,4,5,6}, pick a random number{1,2,3,...,n}, deal cards{2,3,4,...,J,Q,K,Ace}, open a box{red balls,blueballs,...}.

- **Events:**

- **Definition:** A subset of the sample space.
- **Example:** toss a coin{head}, roll a dice{3}, pick a random number{67}, deal cards{K}, open a box{blue ball}.

- **Probability of an event:**

- **Definition:** Probability is a number. It is always greater than or equal to zero, and less than or equal to one. This can be written as  $0 \leq P(A) \leq 1$ . An impossible event, or an event that never occurs, has a probability of 0. An event that always occurs has a probability of 1. An event with a probability of 0.5 will occur half of the time.
- **Formula:** Probability of event to happen:

$$P(E) = \frac{\text{Number of favourable outcomes}}{\text{Total Number of outcomes}}$$

- **Examples:**

- \* The probability of occurring head when tossing a coin is  $\frac{1}{2}$

- \* The probability of occurring 1 when rolling a dice is  $\frac{1}{6}$
- \* The probability of occurring 4 when picking a random number from 1-n ( $n \geq 4$ ) is  $\frac{1}{n}$
- \* The probability of occurring K when deal card is  $\frac{4}{52}$
- \* The probability of picking a blue ball in a box which has 1 red ball and 1 blue ball is  $\frac{1}{2}$

- **Complement event of an event:**

- **Definition:** The possibility that there will be only two outcomes which states that an event will occur or not. Basically, the complement of an event occurring in the exact opposite that the probability of it is not occurring.
- **Formula:**  $P(\neg E) = 1 - P(E)$
- **Examples:**
  - \* The complement of occurring head is tail.
  - \* The complement of occurring 1 when rolling a dice is  $\{2,3,4,5,6\}$
  - \* The complement of 4 when picking a random number is  $\{1,2,3,5,\dots,n\}$
  - \* The complement of having K when dealing cards is  $\{2,3,4,\dots,J,Q,Ace\}$
  - \* The complement of having blue ball when opening a box is  $\{red\ ball\}$

- **Intersection of two events:**

- **Definition:** The intersection of two or more sets is the set of elements that are common to every set. The symbol  $\cap$  is used to denote the intersection.
- **Formula:**  $P(A \cap B) = P(B) * P(A/B)$  or  $= P(A) * P(B)$  if they are independent.
- **Examples:**
  - \* Getting two heads when tossing a coin twice:  $P = \frac{1}{2} * \frac{1}{2} = \frac{1}{4}$
  - \* Getting two 1 when rolling a dice:  $P = \frac{1}{6} * \frac{1}{6} = \frac{1}{36}$
  - \* Picking 4 twice when picking a random number:  $P = \frac{1}{n} * \frac{1}{n} = \frac{1}{n^2}$
  - \* Having a pair of K when dealing cards:  $P = \frac{1}{52} * \frac{1}{51} = \frac{1}{2652}$
  - \* Having blue ball then having red ball:  $P = \frac{1}{2} * \frac{1}{1} = \frac{1}{2}$

- **Union of two events:**

- **Definition:** The union of two or more sets is the set that contains all the elements of the two or more sets. Union is denoted by the symbol  $\cup$ .
- **Formula:**  $P(A \cup B) = P(A) + P(B) - P(A \cap B)$ .  
If the sets are disjoint:  $P(A \cup B) = P(A) + P(B)$
- **Examples:**
  - \* Have 10 or 9 when picking a random number :  $P = P(10) + P(9) = \frac{2}{n}$
  - \* Choose a number that divisible by 5 or 2 from 1 - 100:  $P = P(/5) + P(/2) - P(/10) = \frac{20}{100} + \frac{50}{100} - \frac{10}{100} = 0.6$
  - \* Have K or Diamond when dealing card:  $P = \frac{4}{52} + \frac{13}{52} - \frac{1}{52} = \frac{16}{52}$
  - \* Choose a number that divisible by 10 or 15 from 1 - 100:  $P = P(/10) + P(/15) - P(/30) = \frac{10}{100} + \frac{6}{100} - \frac{3}{100} = \frac{13}{100}$
  - \* Have Ace or Heart when dealing card:  $P = \frac{4}{54} + \frac{13}{54} - \frac{1}{54} = \frac{8}{27}$

- **Conditional probability of an event given that a other event is observed:**

- **Definition:** The probability that an event will take place given the restrictive assumption that another event has taken place, or that a combination of other events has taken place

- **Formula:**  $P(B/A) = \frac{P(A \cap B)}{P(A)}$  . If they are independent,  $P(B/A) = P(B)$

- **Examples:**

- \* Suppose that a coin is flipped 3 times giving the sample space:

$$S = \{HHH, HHT, HTH, THH, TTH, THT, HTT, TTT\}$$

Each individual outcome has probability  $1/8$ . Suppose that B is the event that at least one heads occurs and A is the event that all 3 coins are the same. Then the probability of B given A is  $1/2$ , since  $A \cap B = \{HHH\}$ , which has probability  $1/8$  and  $A = \{HHH, TTT\} = 2/8$ , so  $\frac{1/8}{2/8} = \frac{1}{2}$

- \* Probabilty of having K (B) when having a Diamond (A).  $P(B/A) = \frac{P(B \cap A)}{P(A)} = \frac{1/52}{1/13} = \frac{1}{4}$

- \* Probabilty of having Heart (B) when having a J (A).  $P(B/A) = \frac{P(B \cap A)}{P(A)} = \frac{1/52}{1/13} = \frac{1}{4}$

- \* Probability of having a number that divisible by 10(A) when that number is already divisible by 5(B) when picking a random number from 1 - 100:  $P(A/B) = \frac{P(A \cap B)}{P(B)} = \frac{1/10}{1/20} = \frac{1}{2}$

- **Rule of probability multiplication:**

- **Definition:** is the probability of Intersection events

- **Formula:**  $P(A \cap B) = P(A) * P(B)$  or  $P(A \cap B) = P(B) * P(A/B)$  if A,B are not independent.

- **Example:**

- \* Getting two heads when tossing a coin twice:  $P = \frac{1}{2} * \frac{1}{2} = \frac{1}{4}$

- \* Getting two 1 when rolling a dice:  $P = \frac{1}{6} * \frac{1}{6} = \frac{1}{36}$

- \* Picking 4 twice when picking a random number:  $P = \frac{1}{n} * \frac{1}{n} = \frac{1}{n^2}$

- \* Having a pair of K when dealing cards:  $P = \frac{1}{52} * \frac{1}{51} = \frac{1}{52}$

- \* Having Diamond K :  $P = P(K) * P(D/K) = \frac{4}{13} * \frac{1}{4} = \frac{1}{13}$

- **Law of total probability:**

- **Definition:** The law of total probability is a theorem that, in its discrete case, states if  $\{B_n : B_n : n = 1, 2, 3, \dots\}$  is a finite or countably infinite partition of a sample space (in other words, a set of pairwise disjoint events whose union is the entire sample space) and each event  $B_n$  is measurable, then for any event A of the same probability space:

- **Formula:**

$$P(A) = \sum_n P(A \cap B_n)$$

or, alternatively:

$$P(A) = \sum_n P(A | B_n)P(B_n), P(A) = \sum_n P(A | B_n)P(B_n),$$

- **Examples:**

1. Suppose that two factories supply light bulbs to the market. Factory X's bulbs work for over 5000 hours in 99% of cases, whereas factory Y's bulbs work for over 5000 hours in 95% of cases. It is known that factory X supplies 60% of the total bulbs available and Y supplies 40% of the total bulbs available. What is the chance that a purchased bulb will work for longer than 5000 hours?

Solution:

Applying the law of total probability, we have:

$$\begin{aligned} P(A) &= P(A | B_X) \cdot P(B_X) + P(A | B_Y) \cdot P(B_Y) \\ &= \frac{99}{100} \cdot \frac{6}{10} + \frac{95}{100} \cdot \frac{4}{10} = \frac{594 + 380}{1000} = \frac{974}{1000} \end{aligned}$$

2. A person has undertaken a mining job. The probabilities of completion of job on time with and without rain are 0.42 and 0.90 respectively. If the probability that it will rain is 0.45, then determine the probability that the mining job will be completed on time.

Solution:

Let A be the event that the mining job will be completed on time and B be the event that it rains. We have:  $P(B) = 0.45$

$$P(\text{norain}) = P(B') = 1 - P(B) = 1 - 0.45 = 0.55$$

By multiplication law of probability:

$$P(A|B) = 0.42$$

$$P(A|B') = 0.90$$

Since, events B and B' form partitions of the sample space S, by total probability theorem, we have:

$$P(A) = P(B)P(A|B) + P(B')P(A|B') = 0.45 \times 0.42 + 0.55 \times 0.9 = 0.189 + 0.495 = 0.684$$

3. I have three bags that each contain 100 marbles:

-Bag 1 has 75 red and 25 blue marbles;

-Bag 2 has 60 red and 40 blue marbles;

-Bag 3 has 45 red and 55 blue marbles.

I choose one of the bags at random and then pick a marble from the chosen bag, also at random. What is the probability that the chosen marble is red?

Solution:

Let R be the event that the chosen marble is red. Let  $B_i$  be the event that I choose bag i. We already know that:

$$P(R|B_1) = 0.75$$

$$P(R|B_2) = 0.60$$

$$P(R|B_3) = 0.45$$

We choose our partition as  $B_1, B_2, B_3$ . Note that this is a valid partition because, firstly, the  $B_i$ 's are disjoint (only one of them can happen), and secondly, because their union is the entire sample space as one the bags will be chosen for sure, i.e.,

$$P(B_1 \cup B_2 \cup B_3) = 1.$$

Using the law of total probability, we can write:

$$P(R) = P(R|B_1)P(B_1) + P(R|B_2)P(B_2) + P(R|B_3)P(B_3) = (0.75)13 + (0.60)13 + (0.45)13 = 0.60$$

- **The Bayes formula:**

- **Definition:** describes the probability of occurrence of an event related to any condition. It is also considered for the case of conditional probability. Bayes theorem is also known as the formula for the probability of “causes”.

- **Formula:** Bayes’ theorem is stated mathematically as the following equation:  $P(A | B) = \frac{P(B | A)P(A)}{P(B)}$

where A and B are events and  $P(B) \neq 0$ .  $P(B/A)$  is the probability of event B occurring

when A is true  $P(B)$  is the probability of event B Or:  $P(A | B) = \frac{P(B | A)P(A)}{P(B | A)P(A) + P(B | -A)P(-A)}$

- **Examples:**

- \* A bag I contains 4 white and 6 black balls while another Bag II contains 4 white and 3 black balls. One ball is drawn at random from one of the bags, and it is found to be black. Find the probability that it was drawn from Bag I.

Solution:

Let E1 be the event of choosing bag I, E2 the event of choosing bag II, and A be the event of drawing a black ball.

Then,  $P(E1) = P(E2) = \frac{1}{2}$

Also,  $P(A|E1) = P(\text{drawing a black ball from Bag I}) = \frac{6}{10} = \frac{3}{5}$

$P(A|E2) = P(\text{drawing a black ball from Bag II}) = \frac{3}{7}$

By using Bayes’ theorem, the probability of drawing a black ball from bag I out of two bags,

$$P(E1|A) = \frac{P(E1)P(A|E1)}{P(E1)P(A|E1) + P(E2)P(A|E2)} = \frac{1/2 \cdot 3/5}{1/2 \cdot 3/5 + 1/2 \cdot 3/7} = \frac{7}{12}$$

- \* A man is known to speak the truth 2 out of 3 times. He throws a die and reports that the number obtained is a four. Find the probability that the number obtained is actually a four.

Solution:

Let A be the event that the man reports that number four is obtained.

Let E1 be the event that four is obtained and E2 be its complementary event.

Then,  $P(E1) = \text{Probability that four occurs} = \frac{1}{6}$

$P(E2) = \text{Probability that four does not occur} = 1 - P(E1) = 1 - \frac{1}{6} = \frac{5}{6}$

Also,  $P(A|E1) = \text{Probability that man reports four and it is actually a four} = \frac{2}{3}$

$P(A|E2) = \text{Probability that man reports four and it is not a four} = \frac{1}{3}$

By using Bayes’ theorem, probability that number obtained is actually a four,

$$P(E1|A) = \frac{P(E1)P(A|E1)}{P(E1)P(A|E1) + P(E2)P(A|E2)} = \frac{1/62/3}{1/62/3 + 5/61/3} = \frac{2}{7}$$

- \* Suppose, a particular test for whether someone has been using cannabis is 90% sensitive, meaning the true positive rate (TPR)=0.90. Therefore it leads to 90% true positive results (correct identification of drug use) for cannabis users.

The test is also 80% specific, meaning true negative rate (TNR)=0.80. Therefore the test correctly identifies 80% of non-use for non-users, but also generates 20% false positives, or false positive rate (FPR)=0.20, for non-users.

Assuming 0.05 prevalence, meaning 5% of people use cannabis, what is the probability that a random person who tests positive is really a cannabis user?

The Positive predictive value (PPV) of a test is the proportion of persons who are actually positive out of all those testing positive, and can be calculated from a sample as:

$$\text{PPV} = \text{True positive} / \text{Tested positive}$$

If sensitivity, specificity, and prevalence are known, PPV can be calculated using Bayes theorem. Let  $P(\text{User} | \text{Positive})$  mean "the probability that someone is a cannabis user given that they test positive," which is what is meant by PPV. We can write:

$$\begin{aligned} P(\text{User} | \text{Positive}) &= \frac{P(\text{Positive} | \text{User})P(\text{User})}{P(\text{Positive})} \\ &= \frac{P(\text{Positive} | \text{User})P(\text{User})}{P(\text{Positive} | \text{User})P(\text{User}) + P(\text{Positive} | \text{Non-user})P(\text{Non-user})} \\ &= \frac{0.90 \times 0.05}{0.90 \times 0.05 + 0.20 \times 0.95} = \frac{0.045}{0.045 + 0.19} \approx 19\% \end{aligned}$$

### 2.1.2 Discrete random variables (Exercise 1b)

- **Discrete Random Variable:**

- **Definition:** A random variable that takes on a finite or countably infinite number of values is called a Discrete Random Variable

- **Examples:**

- \* Number of children in a family
- \* The Monday night attendance at a cinema
- \* Number of student in a class
- \* Number of “heads” when flipped coin three time
- \* The number of red marbles in the box

- **Probability mass function:**

- **Definition:** a probability mass function is a function that gives the probability that a discrete random variable is exactly equal to some value. Sometimes it is also known as the discrete density function

- **Examples:**

- \* Toss a fair coin twice, the number of heads observed
- \* Toss an unfair coin repeatedly until I observe a heads for the first time



- \* the number of defective chips we receive, Suppose that 10 percent of the chips produced by a computer hardware manufacturer are defective. If we order 100 such chips
- \* proportion of packages is returned if disks produced by a certain company will be defective with probability 0.01 independently of each other The company sells the disks in packages of 10 and offers a money-back guarantee that at most 1 of the 10 disks is defective
- \* The color of one's eyes is determined by a single pair of genes, probability of eyes color

- **Expected value and variance formulas of a discrete random variable:**

- **Expected value formula:**  $\mu = E(X) = \sum_{i=1}^n x f(x)$
- **Varian:**  $\sigma^2 = V(X) = \sum_{i=1}^n (x - \mu)^2 f(x)$
- **Examples:**
  - \* The number that approve, Suppose 60% of American adults approve of the way the president is handling his job, randomly sample 2 American adults
  - \* Expected Value of a Die Let X be the number that comes up when a fair die is rolled. What is the expected value of X?
  - \* A fair coin is flipped three times, X is number of head in this outcome, the expected value of X?
  - \* A has a test X is a random variable of mark that A could get, What is the expected value of mark
  - \* The profit or loss of restaurants, caculate the expected Value and Standard Deviation

- **Bernoulli trial:**

- **Definition:** is a random experiment with exactly two possible outcomes, "success" and "failure", in which the probability of success is the same every time the experiment is conducted
- **Examples:**
  - \* probability of flipping a coin with "Heads" as success and "Tails" as failure
  - \* rolling a dice with 6 dot as success
  - \* A bag contains 6 red marbles and 4 blue marbles, the number of red marbles is observed. We might let a trial here consist of drawing a marble from the bag and let success be getting a red
  - \* Suppose that a student takes a multiple choice test. The test has 10 questions, each of which has 4 possible answers (only one correct). If the student blindly guesses the answer to each question
  - \* Candidate A is running for office in a certain district. Twenty persons are selected at random from the population of registered voters and asked if they prefer candidate A

- **Binomial distribution:**

- **Definition:** Binomial distribution summarizes the number of trials, or observations when each trial has the same probability of attaining one particular value. The binomial distribution determines the probability of observing a specified number of successful outcomes in a specified number of trials.

– **Properties:**

1. The experiment consists of  $n$  identical trials.
2. Each trial results in one of the two outcomes, called a success  $S$  and failure  $F$ .
3. The probability of success on a single trial is equal to  $p$  and remains the same from trial to trial. The probability of failure is  $1 - p = q$ .
4. The outcomes of the trials are independent.
5. The random variable  $X$  is the number of successes in  $n$  trials.

– **Examples:**

- \* the expected value of the number of heads in 100 trials of head and tails is 50
- \* the chances of success for a free-throw shooter in basketball
- \* probability of 2 score in to goal in 6 attempt
- \* probability to selecting exactly three club from a deck
- \* probability to selecting exactly 2 marble in a box with 5 attempt

● **Geometric distribution:**

– **Definition:** The geometric distribution represents the number of failures before you get a success in a series of Bernoulli trials

1. There are one or more Bernoulli trials with all failures except the last one, which is a success.
2. In theory, the number of trials could go on forever. There must be at least one trial.
3. The probability,  $p$ , of a success and the probability,  $q$ , of a failure are the same for each trial.

– **Examples:**

- \* You throw darts at a board until you hit the center area
- \* You play a game of chance that you can either win or lose until you lose, What is the probability that it takes five games until you lose
- \* The probability of a defective steel rod is 0.01. Steel rods are selected at random. Find the probability that the first defect occurs on the ninth steel rod
- \* The probability that the safety engineer will have to examine at least three reports until she finds a report showing an accident caused by employee failure to follow instructions
- \* a lightbulb manufacturing company and determines that 3 out of every 75 bulbs are defective, the probability that Max will find the first faulty lightbulb on the 6th one that he tested

● **Joint probability mass function:**

– **Definition:** If discrete random variables  $X$  and  $Y$  are defined on the same sample space  $S$ , then their joint probability mass function

– **Examples:**

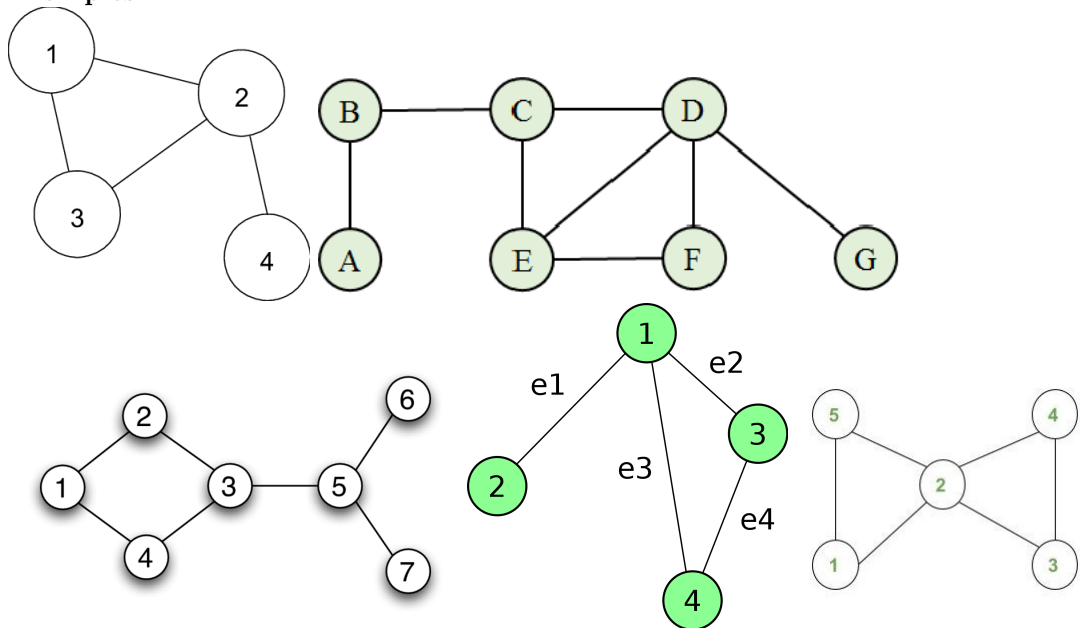
- \* the probability that you get a six in a desk and it is red
- \* probability of rolling the number five twice in a fair six-sided dice
- \* probability of getting a head followed by a tail in a coin toss
- \* probability of drawing a number ten card that is black
- \* the chance of being infected by tuberculosis bacteria for a person if his test result is positive

## 2.2 Graphs (Exercise 2)

### 2.2.1 Undirected Graphs (Exercise 2a)

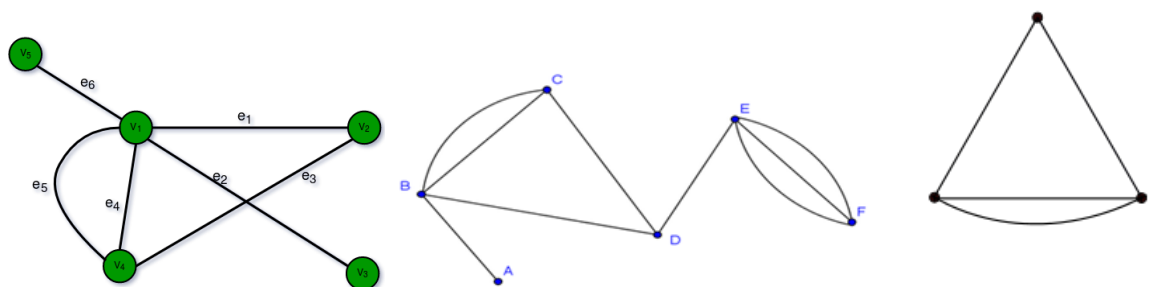
- **Simple graph:** A graph in which each edge connects two different vertices and where no two edges connect the same pair of vertices is called a simple graph.

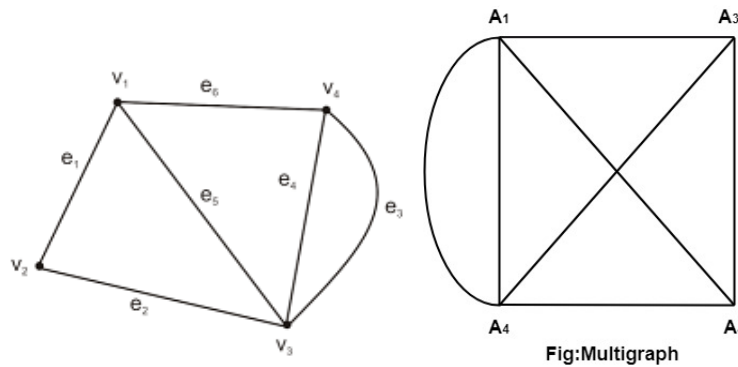
– Examples:



- **Multi-graph:** Graphs that may have multiple edges connecting the same vertices are called multi-graphs.

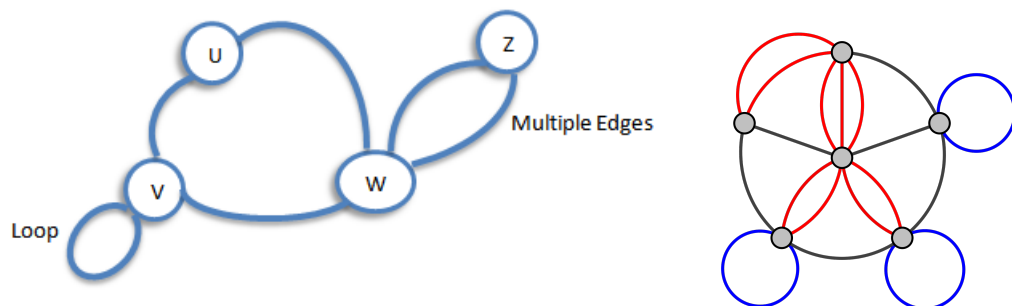
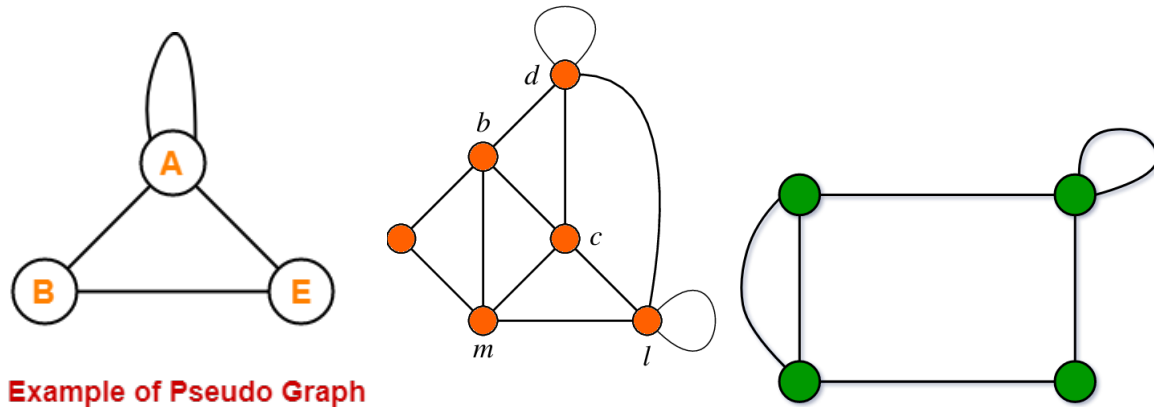
– Examples:





- **Pseudo-graph:** Graphs that may include loops, and possibly multiple edges connecting the same pair of vertices or a vertex to itself, are sometimes called pseudo-graphs.

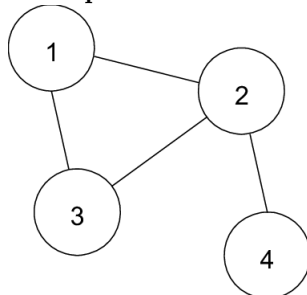
– **Examples:**



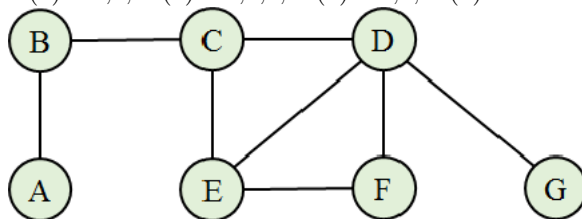
- **Degree of a node (or a vertex):** The degree of a vertex in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex. The degree of the vertex  $v$  is denoted by  $\deg(v)$ .
  - isolated vertex: vertex of degree 0
  - pendant vertex: vertex of degree 1

- **Neighbourhood of a node (or a vertex):** In an undirected graph  $G = (V, E)$ :
  - two vertices  $u$  and  $v \in V$  are called adjacent if they are end-points of edge  $e \in E$ , and we have:
  - $e$  is incident with  $u$  and  $v$
  - $e$  is said to connect  $u$  and  $v$

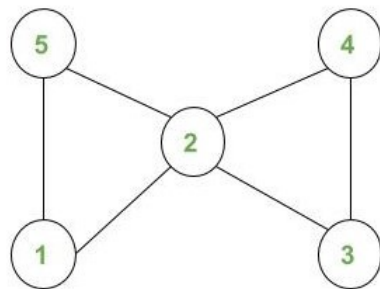
– **Examples:**



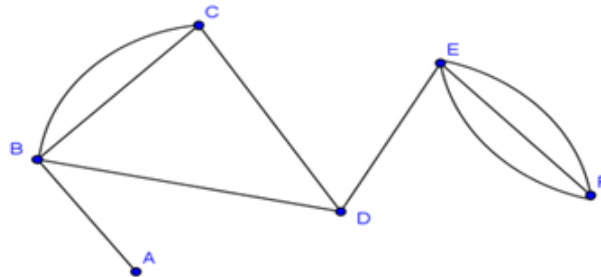
$\deg(1) = 2$ ;  $\deg(2) = 3$ ;  $\deg(3) = 2$ ;  $\deg(4) = 1$   
 $N(1) = 2, 3$ ;  $N(2) = 1, 3, 4$ ;  $N(3) = 1, 2$ ;  $N(4) = 2$ .



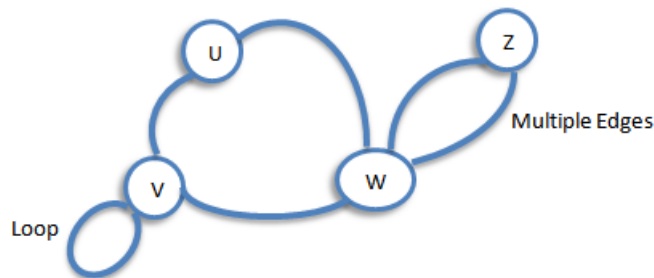
$\deg(A) = 1$ ;  $\deg(B) = 2$ ;  $\deg(C) = 3$ ;  $\deg(D) = 4$ ;  $\deg(E) = 3$ ;  $\deg(F) = 2$ ;  $\deg(G) = 1$   
 $N(A) = B$ ;  $N(B) = A, C$ ;  $N(C) = B, D, E$ ;  $N(D) = C, E, F, G$ ;  $N(E) = C, D, F$ ;  $N(F) = D, E$ ;  $N(G) = D$ .



$\deg(1) = 2$ ;  $\deg(2) = 4$ ;  $\deg(3) = 2$ ;  $\deg(4) = 2$ ;  $\deg(5) = 2$   
 $N(1) = 2, 5$ ;  $N(2) = 1, 3, 4, 5$ ;  $N(3) = 2, 4$ ;  $N(4) = 2, 3$ ;  $N(5) = 1, 2$ .



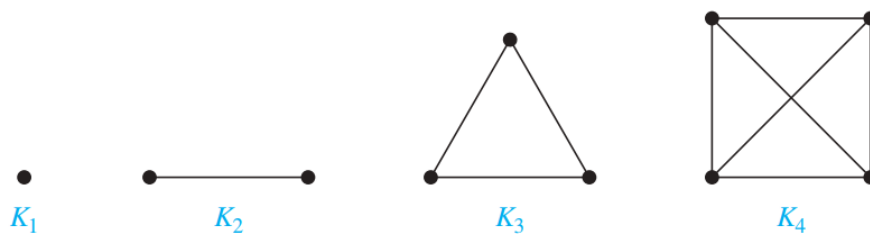
$\deg(A)=1$ ;  $\deg(B)=4$ ;  $\deg(C)=3$ ;  $\deg(D)=3$ ;  $\deg(E)=4$ ;  $\deg(F)=3$   
 $N(A)=B$ ;  $N(B)=A,C,D,E$ ;  $N(C)=B,D$ ;  $N(D)=B,C,E$ ;  $N(E)=B,D,F$ ;  $N(F)=E$ .

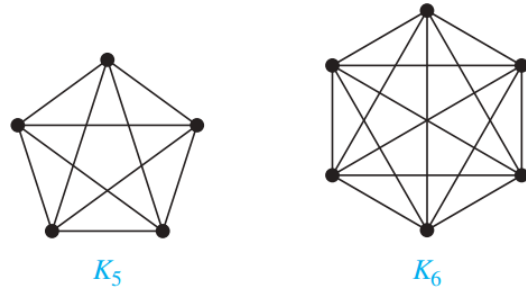


$\deg(u)=2$ ;  $\deg(v)=4$ ;  $\deg(w)=4$ ;  $\deg(z)=2$   
 $N(u)=v,w$ ;  $N(v)=u,w$ ;  $N(w)=u,v,z$ ;  $N(z)=w$ .

- **Completegraph:** A complete graph on  $n$  vertices, denoted by  $K_n$ , is a simple graph that contains exactly one edge between each pair of distinct vertices. The graphs  $K_n$ , for  $n = 1, 2, 3, 4, 5, 6$ , are displayed in Examples. A simple graph for which there is at least one pair of distinct vertex not connected by an edge is called non-complete.

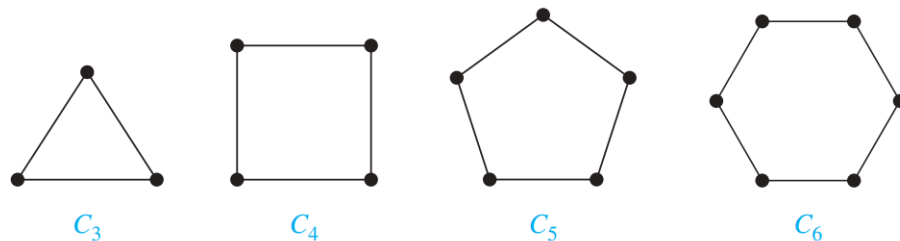
– **Examples:**





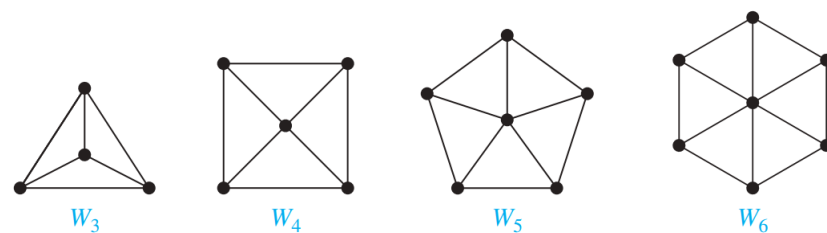
- **Cycles:** A cycle  $C_n$ ,  $n \geq 3$ , consists of  $n$  vertices  $v_1, v_2, \dots, v_n$  and edges  $v_1, v_2, v_2, v_3, \dots, v_{n-1}, v_n$ , and  $v_n, v_1$ . The cycles  $C_3, C_4, C_5$ , and  $C_6$  are displayed in Examples.

– **Examples:**

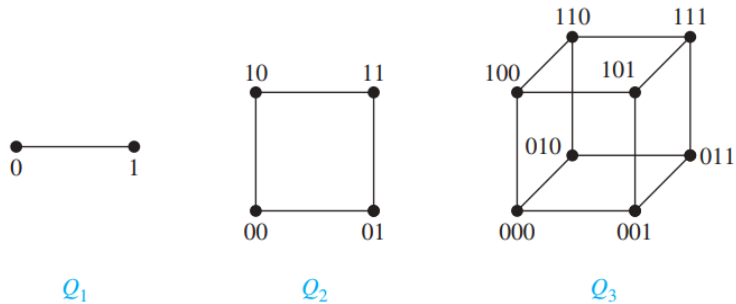


- **Wheels:** We obtain a wheel  $W_n$  when we add an additional vertex to a cycle  $C_n$ , for  $n \geq 3$ , and connect this new vertex to each of the  $n$  vertices in  $C_n$ , by new edges. The wheels  $W_3, W_4, W_5$ , and  $W_6$  are displayed in Examples.

– **Examples:**



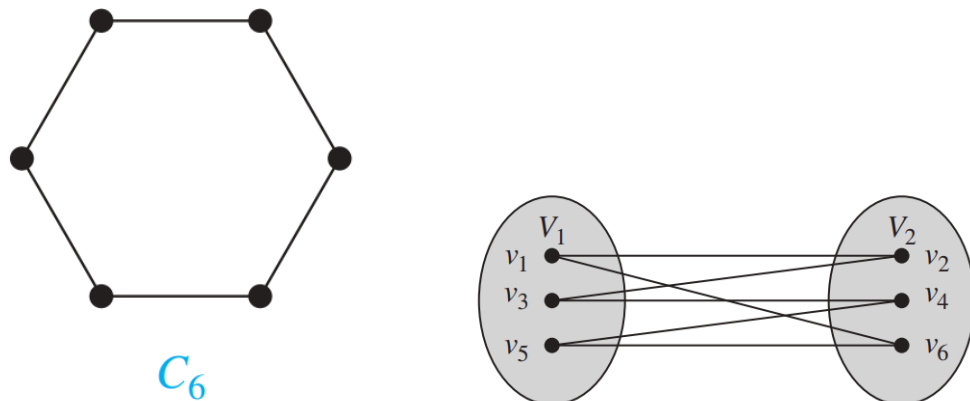
- **n-Cubes:** An  $n$ -dimensional hypercube, or  $n$ -cube, denoted by  $Q_n$ , is a graph that has vertices representing the  $2^n$  bit strings of length  $n$ . Two vertices are adjacent if and only if the bit strings that they represent differ in exactly one bit position. We display  $Q_1, Q_2$ , and  $Q_3$  in Examples.



– Examples:

- **Bipartite graphs:** A simple graph  $G$  is called bipartite if its vertex set  $V$  can be partitioned into two disjoint sets  $V_1$  and  $V_2$  such that every edge in the graph connects a vertex in  $V_1$  and a vertex in  $V_2$  (so that no edge in  $G$  connects either two vertices in  $V_1$  or two vertices in  $V_2$ ). When this condition holds, we call the pair  $(V_1, V_2)$  a bipartition of the vertex set  $V$  of  $G$ .

– Examples:



- **HANDSHAKING Theorem:** Let  $G = (V, E)$  be an undirected graph with  $m$  edges. Then

$$2m = \sum_{v \in V} \deg(v).$$

(Note that this applies even if multiple edges and loops are present.)

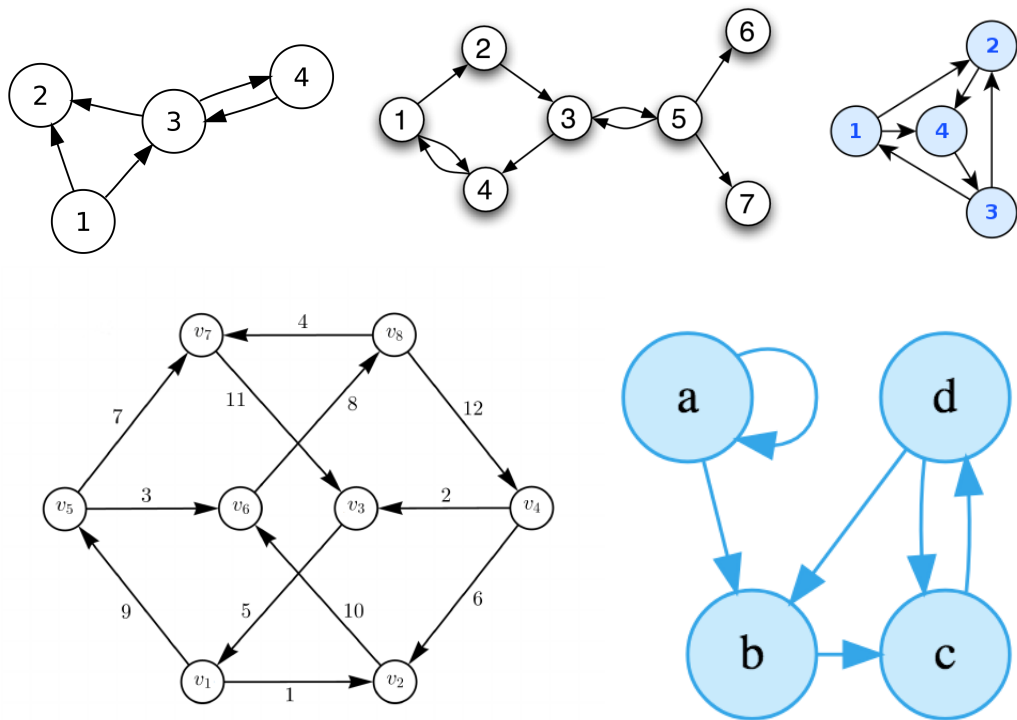
- **Proof for Handshaking Theorem:** Because each edge have to connect two vertices, representing 2 degrees added into each vertex. So, The sum of degrees of all the vertices is two times the number of edges.

### 2.2.2 Directed graphs (Exercise 2b)



### • Directed graph:

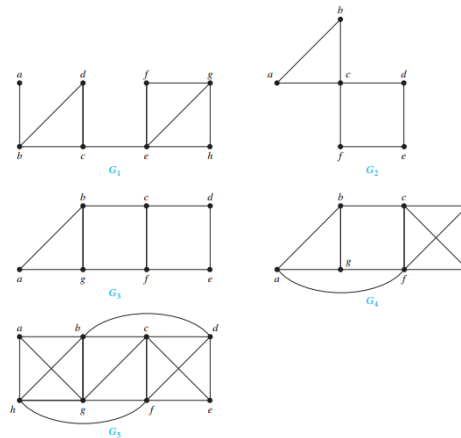
- A directed graph (or digraph)  $(V, E)$  consists of a nonempty set of vertices  $V$  and a set of directed edges (or arcs)  $E$ . Each directed edge is associated with an ordered pair of vertices. The directed edge associated with the ordered pair  $(u, v)$  is said to start at  $u$  and end at  $v$ .
- Examples:



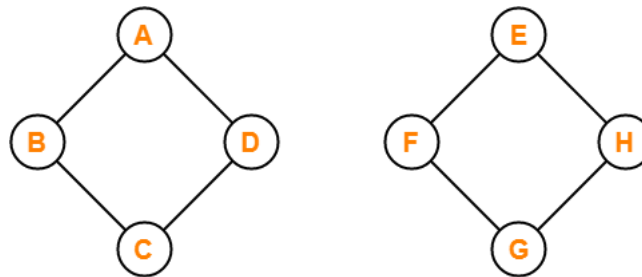
### • Connectivity:

- An undirected graph is called connected if there is a path between every pair of distinct vertices of the graph. An undirected graph that is not connected is called disconnected. We say that we disconnect a graph when we remove vertices or edges, or both, to produce a disconnected subgraph.
- **Example:**
- **Connectedness in Directed Graphs:**
  - \* A directed graph is strongly connected if there is a path from  $a$  to  $b$  and from  $b$  to  $a$  whenever  $a$  and  $b$  are vertices in the graph.
  - \* A directed graph is weakly connected if there is a path between every two vertices in the underlying undirected graph.

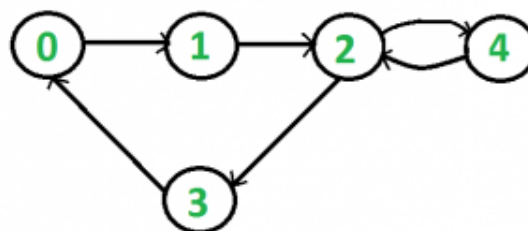
### • In-degree, out-degree and neighborhood in directed graph:



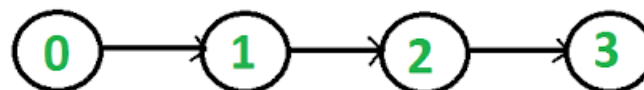
Some connected graph



Example of Disconnected Graph



Strongly Connected



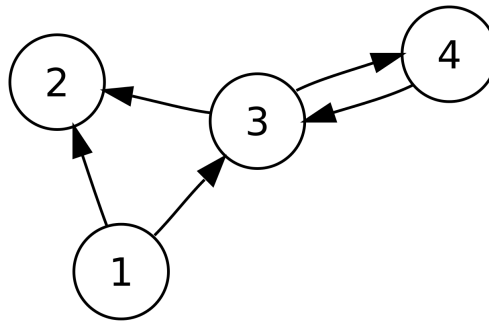
Not Strongly Connected

- **In-degree, out-degree:** In a graph with directed edges the in-degree of a vertex  $v$ , denoted by  $\deg^-(v)$ , is the number of edges with  $v$  as their terminal vertex. The out-

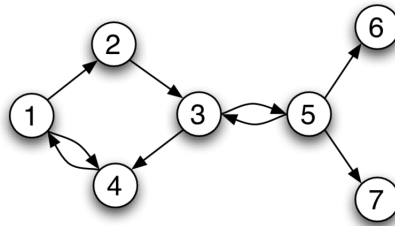
degree of  $v$ , denoted by  $\deg^+(v)$ , is the number of edges with  $v$  as their initial vertex. (Note that a loop at a vertex contributes 1 to both the in-degree and the out-degree of this vertex.)

- **Neighborhood in directed graph:** A vertex  $u$  is an in-neighbor of a vertex  $v$  in a directed graph  $G = (V, E)$  if  $(u, v) \in E$ , and is an out-neighbor of a vertex  $v$  if  $(v, u) \in E$ . we use  $N^+(v)$  to indicate the set of out-neighbors and  $N^-(v)$  to indicate the set of in-neighbors of  $v$ . If we use  $N(v)$  for a directed graph, we mean the set of out-neighbors. The neighborhood of a set of vertices  $U \subseteq V$  is the union of their neighborhoods, e.g.  $N^+(V) = \bigcup_{v \in V} N^+(v)$  or  $N^-(V) = \bigcup_{v \in V} N^-(v)$ .

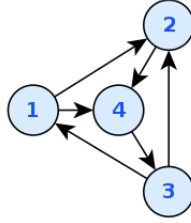
- **Example:**



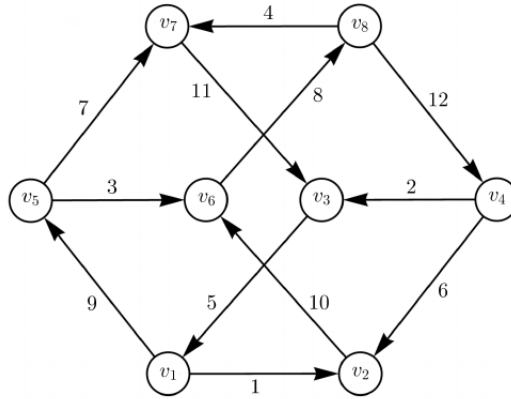
The in-degrees are  $\deg^-(1) = 0$ ,  $\deg^-(2) = 2$ ,  $\deg^-(3) = 2$ ,  $\deg^-(4) = 1$ .  
The out-degrees are  $\deg^+(1) = 2$ ,  $\deg^+(2) = 0$ ,  $\deg^+(3) = 2$ ,  $\deg^+(4) = 1$ .  
In-neighbors:  $N^-(1) = \emptyset$ ,  $N^-(2) = \{1, 3\}$ ,  $N^-(3) = \{1, 4\}$ ,  $N^-(4) = \{3\}$ .  
Out-neighbors:  $N^+(1) = \{2, 3\}$ ,  $N^+(2) = \emptyset$ ,  $N^+(3) = \{2, 4\}$ ,  $N^+(4) = \{3\}$ .



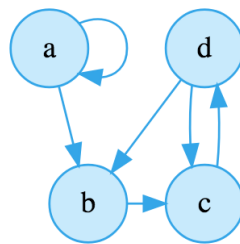
The in-degrees are  $\deg^-(1) = \deg^-(2) = \deg^-(5) = \deg^-(6) = \deg^-(7) = 1$ ,  $\deg^-(3) = \deg^-(4) = 2$ .  
The out-degrees are  $\deg^+(1) = \deg^+(3) = 2$ ,  $\deg^+(2) = \deg^+(4) = 1$ ,  $\deg^+(5) = 3$ ,  $\deg^+(6) = \deg^+(7) = 0$ .  
In-neighbors:  $N^-(1) = \{4\}$ ,  $N^-(2) = \{1\}$ ,  $N^-(3) = \{2\}$ ,  $N^-(4) = \{1, 3\}$ ,  $N^-(5) = \{3\}$ ,  $N^-(6) = \{5\}$ ,  $N^-(7) = \{5\}$ .  
Out-neighbors:  $N^+(1) = \{2, 4\}$ ,  $N^+(2) = \{3\}$ ,  $N^+(3) = \{4\}$ ,  $N^+(4) = \{1\}$ ,  $N^+(5) = \{3, 6, 7\}$ ,  $N^+(6) = \emptyset$ ,  $N^+(7) = \emptyset$ .



The in-degrees are  $\deg^-(1) = 1$ ,  $\deg^-(2) = 2$ ,  $\deg^-(3) = 1$ ,  $\deg^-(4) = 2$ .  
The out-degrees are  $\deg^+(1) = 2$ ,  $\deg^+(2) = 1$ ,  $\deg^+(3) = 2$ ,  $\deg^+(4) = 1$ .  
In-neighbors:  $N^-(1) = \{3\}$ ,  $N^-(2) = \{1, 3\}$ ,  $N^-(3) = \{4\}$ ,  $N^-(4) = \{1, 2\}$ .  
Out-neighbors:  $N^+(1) = \{2, 4\}$ ,  $N^+(2) = \{4\}$ ,  $N^+(3) = \{1, 2\}$ ,  $N^+(4) = \{3\}$ .



The in-degrees are  $\deg^-(v1) = 1$ ,  $\deg^-(v2) = 2$ ,  $\deg^-(v3) = 2$ ,  $\deg^-(v4) = 1$ ,  $\deg^-(v5) = 1$ ,  $\deg^-(v6) = 2$ ,  $\deg^-(v7) = 2$ ,  $\deg^-(v8) = 1$ .  
The out-degrees are  $\deg^+(v1) = 2$ ,  $\deg^+(v2) = 1$ ,  $\deg^+(v3) = 1$ ,  $\deg^+(v4) = 2$ ,  $\deg^+(v5) = 2$ ,  $\deg^+(v6) = 1$ ,  $\deg^+(v7) = 1$ ,  $\deg^+(v8) = 2$ .  
In-neighbors:  $N^-(v1) = \{v3\}$ ,  $N^-(v2) = \{v1, v4\}$ ,  $N^-(v3) = \{v4, v7\}$ ,  $N^-(v4) = \{v8\}$ ,  $N^-(v5) = \{v1\}$ ,  $N^-(v6) = \{v2, v5\}$ ,  $N^-(v7) = \{v5, v8\}$ ,  $N^-(v8) = \{v6\}$ .  
Out-neighbors:  $N^+(v1) = \{v2, v5\}$ ,  $N^+(v2) = \{v6\}$ ,  $N^+(v3) = \{v1\}$ ,  $N^+(v4) = \{v2, v3\}$ ,  $N^+(v5) = \{v6, v7\}$ ,  $N^+(v6) = \{v8\}$ ,  $N^+(v7) = \{v3\}$ ,  $N^+(v8) = \{v4, v7\}$ .



The in-degrees are  $\deg^-(a) = 1$ ,  $\deg^-(b) = 2$ ,  $\deg^-(c) = 2$ ,  $\deg^-(d) = 1$ .  
The out-degrees are  $\deg^+(a) = 2$ ,  $\deg^+(b) = 1$ ,  $\deg^+(c) = 1$ ,  $\deg^+(d) = 2$ .  
In-neighbors:  $N^-(a) = \{a\}$ ,  $N^-(b) = \{a, d\}$ ,  $N^-(c) = \{b, d\}$ ,  $N^-(d) = \{c\}$ .  
Out-neighbors:  $N^+(a) = \{a, b\}$ ,  $N^+(b) = \{c\}$ ,  $N^+(c) = \{d\}$ ,  $N^+(d) = \{b, c\}$ .

### 2.2.3 Paths and circuits (Exercise 2c)

- **Path, Cycle and Circuit in Undirected graphs:**

- Let  $n$  be a nonnegative integer and  $G$  an undirected graph. A path of length  $n$  from  $u$  to  $v$  in  $G$  is a sequence of  $n$  edges  $e_1, \dots, e_n$  of  $G$  for which there exists a sequence  $x_0 = u, x_1, \dots, x_{n-1}, x_n = v$  of vertices such that  $e_i$  has, for  $i = 1, \dots, n$ , the endpoints  $x_{i-1}$  and  $x_i$ . When the graph is simple, we denote this path by its vertex sequence  $x_0, x_1, \dots, x_n$  (because listing these vertices uniquely determines the path). The path is a circuit if it begins and ends at the same vertex, that is, if  $u = v$ , and has length greater than zero. The path or circuit is said to pass through the vertices  $x_1, x_2, \dots, x_{n-1}$  or traverse the edges  $e_1, e_2, \dots, e_n$ . A path or circuit is simple if it does not contain the same edge more than once.
- Let  $n$  be a nonnegative integer and  $G$  a directed graph. A path of length  $n$  from  $u$  to  $v$  in  $G$  is a sequence of edges  $e_1, e_2, \dots, e_n$  of  $G$  such that  $e_1$  is associated with  $(x_0, x_1)$ ,  $e_2$  is associated with  $(x_1, x_2)$ , and so on, with  $e_n$  associated with  $(x_{n-1}, x_n)$ , where  $x_0 = u$  and  $x_n = v$ . When there are no multiple edges in the directed graph, this path is denoted by its vertex sequence  $x_0, x_1, x_2, \dots, x_n$ . A path of length greater than zero that begins and ends at the same vertex is called a circuit or cycle. A path or circuit is called simple if it does not contain the same edge more than once.

- **Path, Cycle, and Circuit length:** Path (or Cycle, or Circuit) of length  $n$  from  $u$  to  $v$ : a sequence of  $n$  edges  $\{x_0, x_1\}, \{x_1, x_2\}, \dots, \{x_{n-1}, x_n\}$ , where  $x_0 = u$  and  $x_n = v$ .

- **Examples:**

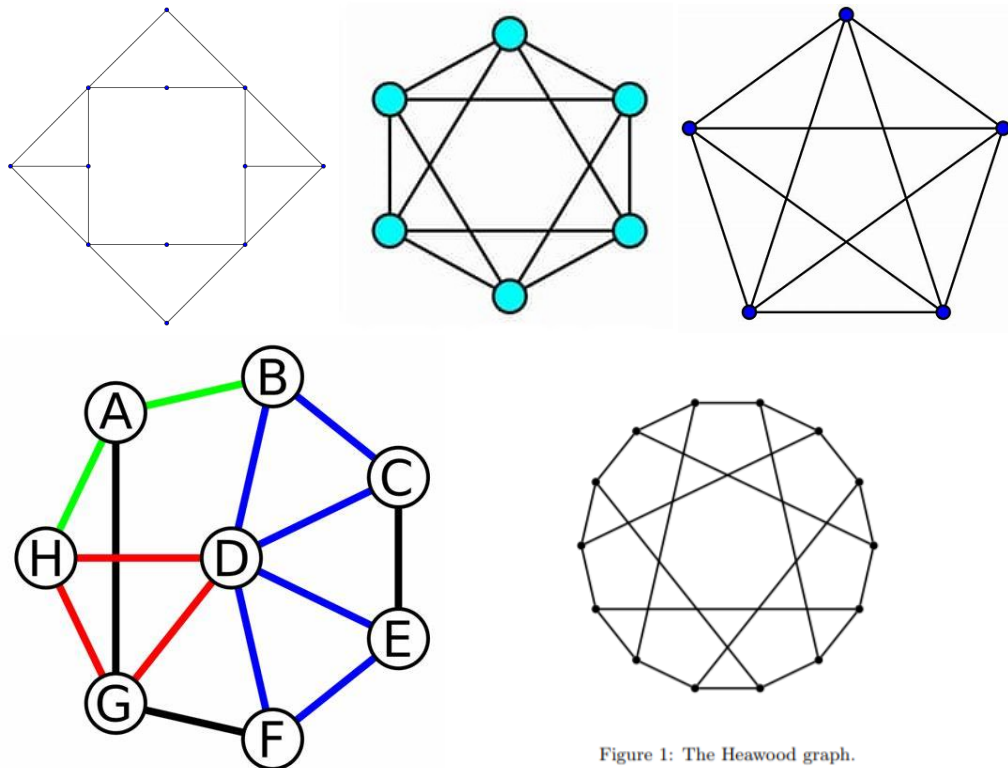


Figure 1: The Heawood graph.

## 2.3 Bayesian networks (Exercise 3)

### 2.3.1 Bayesian models (Exercise 3a)

- Example 1:** Suppose that one person in 100,000 has a particular rare disease for which there is a fairly accurate diagnostic test. This test is correct 99.0% of the time when given to a person selected at random who has the disease; it is correct 99.5% of the time when given to a person selected at random who does not have the disease. Given this information, can we find

F (rare disease)	0.00001
$\neg F$ (not rare disease)	0.99999

Bảng 1: Probability distribution at node F

E F	F (Rare Disease)	$\neg F$ (not Rare Disease)
E (positive)	0.99	0.005
$\neg E$ (non-positive)	0.01	0.995

Bảng 2: Probability distribution at node E

$$P(F|E) = \frac{P(E|F)P(F)}{P(E|F)P(F) + P(E|\neg F)P(\neg F)} = \frac{0.99 \cdot 0.00001}{0.99 \cdot 0.00001 + 0.005 \cdot 0.99999} \approx 0.0019$$

$$P(\neg F|\neg E) = \frac{P(\neg E|\neg F)P(\neg F)}{P(\neg E|\neg F)P(\neg F) + P(\neg E|F)P(F)} = \frac{0.995 \cdot 0.99999}{0.995 \cdot 0.99999 + 0.01 \cdot 0.00001} \approx 0.9999$$

- Example 2:** Suppose that 8% of all bicycle racers use steroids, that a bicyclist who uses steroids tests positive for steroids 96% of the time, and that a bicyclist who does not use steroids tests positive for steroids 9% of the time. What is the probability that a randomly selected bicyclist who tests positive for steroids actually uses steroids?

A (use steroids)	0.08
$\neg A$ (not use steroids)	0.92

Bảng 3: Probability distribution at node A

B A	A (use steroids)	$\neg A$ (not use steroids)
B (positive)	0.96	0.09
$\neg B$ (non-positive)	0.04	0.91

Bảng 4: Probability distribution at node B

$$P(B|A) = \frac{P(A|B)P(B)}{P(A|B)P(B) + P(A|\neg B)P(\neg B)} = \frac{0.96 \cdot 0.08}{0.96 \cdot 0.08 + 0.92 \cdot 0.09} \approx 0.4812$$

- Example 3:** Suppose that 4% of the patients tested in a clinic are infected with avian influenza. Furthermore, suppose that when a blood test for avian influenza is given, 97% of the patients infected with avian influenza test positive and that 2% of the patients not infected with avian influenza test positive. What is the probability that

- a) a patient testing positive for avian influenza with this test is infected with it?  
b) a patient testing positive for avian influenza with this test is not infected with it?  
c) a patient testing negative for avian influenza with this test is infected with it?  
d) a patient testing negative for avian influenza with this test is not infected with it?

I (Infected)	0.04
$\neg I$ (non-infected)	0.96

Bảng 5: Probability distribution at node I

B I	I (Infected)	$\neg I$ (non-infected)
B (positive)	0.97	0.02
$\neg B$ (negative)	0.03	0.98

Bảng 6: Probability distribution at node B

$$a) P(I|B) = \frac{P(B|I)P(I)}{P(B|I)P(I)+P(B|\neg I)P(\neg I)} = \frac{0.97*0.04}{0.97*0.04+0.96*0.02} \approx 0.669$$

$$b) P(\neg I|B) = 1 - P(I|B) = 0.331$$

$$c) P(I/\neg B) = \frac{P(\neg B|I)P(I)}{P(\neg B|I)P(I)+P(\neg B|\neg I)P(\neg I)} = \frac{0.02*0.08}{0.02*0.08+0.97*0.92} \approx 0.00127$$

$$d) P(\neg I/\neg B) = 1 - P(I/\neg B) = 1 - 0.00127 = 0.99873$$

4. **Example 4:** An electronics company is planning to introduce a new camera phone. The company commissions a marketing report for each new product that predicts either the success or the failure of the product. Of new products introduced by the company, 60% have been successes. Furthermore, 70% of their successful products were predicted to be successes, while 40% of failed products were predicted to be successes. Find the probability that this new camera phone will be successful if its success has been predicted.

S (success product)	0.6
$\neg S$ (failed product)	0.4

Bảng 7: Probability distribution at node S

P S	S (success product)	$\neg S$ (failed product)
P (predict to be success)	0.7	0.4
$\neg P$ (predict to be failure)	0.3	0.6

Bảng 8: Probability distribution at node P

$$P(S|P) = \frac{P(S)P(P|S)}{P(S)P(P|S)+P(\neg S)P(P|\neg S)} = \frac{0.6*0.7}{0.6*0.7+0.4*0.4} \approx 0.724$$



5. **Example 5:** A space probe near Neptune communicates with Earth using bit strings. Suppose that in its transmissions it sends a 1 one-third of the time and a 0 two-thirds of the time. When a 0 is sent, the probability that it is received correctly is 0.9, and the probability that it is received incorrectly (as a 1) is 0.1. When a 1 is sent, the probability that it is received correctly is 0.8, and the probability that it is received incorrectly (as a 0) is 0.2.
- a) Find the probability that a 0 is received.
- b) Use Bayes' theorem to find the probability that a 1 was transmitted, given that a 0 was received.

X (0 is transmitted)	0.667
$\neg X$ (1 is transmitted)	0.333

Bảng 9: Probability distribution at node X

Y X	X (0 is transmitted)	$\neg X$ (1 is transmitted)
Y (receive 0)	0.9	0.2
$\neg Y$ (receive 1)	0.1	0.8

Bảng 10: Probability distribution at node Y

- a)  $P(Y) = P(X)P(Y|X) + P(\neg X)P(Y|\neg X) = 0.667*0.9 + 0.333*0.2 = 0.6669 \approx 70\%$
- b)  $P(\neg X|Y) = \frac{P(\neg X)P(Y|\neg X)}{P(\neg X)P(Y|\neg X) + P(X)P(Y|X)} = \frac{0.333*0.2}{0.333*0.2 + 0.667*0.9} \approx 0.1$

### 2.3.2 Variable Elimination technique and inference (Exercise 3b)

1. **Case 1:**  $X_1 \rightarrow X_2 \leftarrow X_3$ : General formula:  $X_1 \rightarrow X_2 \rightarrow \dots \rightarrow X_{n-1} \rightarrow X_n \leftarrow X_{n+1} \leftarrow \dots \leftarrow X_{m-1} \leftarrow X_m$ .
- Let:

$$\tau(X_2) := \sum_{X_1} P(X_2|X_1)P(X_1)$$

$$\tau(X_{m-1}) := \sum_{X_m} P(X_{m-1}|X_m)P(X_m)$$

Then:

$$\begin{aligned} \tau(X_{n-1}) &:= \sum_{X_{n-2}} P(X_{n-1}|X_{n-2})\tau(X_{n-2}) \\ \tau(X_{n+1}) &:= \sum_{X_{n+2}} P(X_{n+1}|X_{n+2})\tau(X_{n+2}) \end{aligned} \quad (1)$$

We have the formula for variable elimination:

$$P(X_n) = \sum_{X_{n-1}} \sum_{X_{n+1}} P(X_n|X_{n-1}, X_{n+1})\tau(X_{n-1})\tau(X_{n+1}) \quad (2)$$

Apply to this case:

$$\tau(X_1) := P(X_1)$$

$$\tau(X_3) := P(X_3)$$

$$P(X_2) = \sum_{X_1} \sum_{X_3} P(X_2|X_1, X_3) \tau(X_1) \tau(X_3)$$

.

Now, we compute  $P(X_1|X_2)$ :

$$P(X_1|X_2) = \frac{P(X_1, X_2)}{P(X_2)}$$

We have:

$$P(X_1, X_2) = \sum_{X_3} P(X_1, X_2, X_3) = P(X_1) \sum_{X_3} P(X_3) P(X_2|X_1, X_3)$$

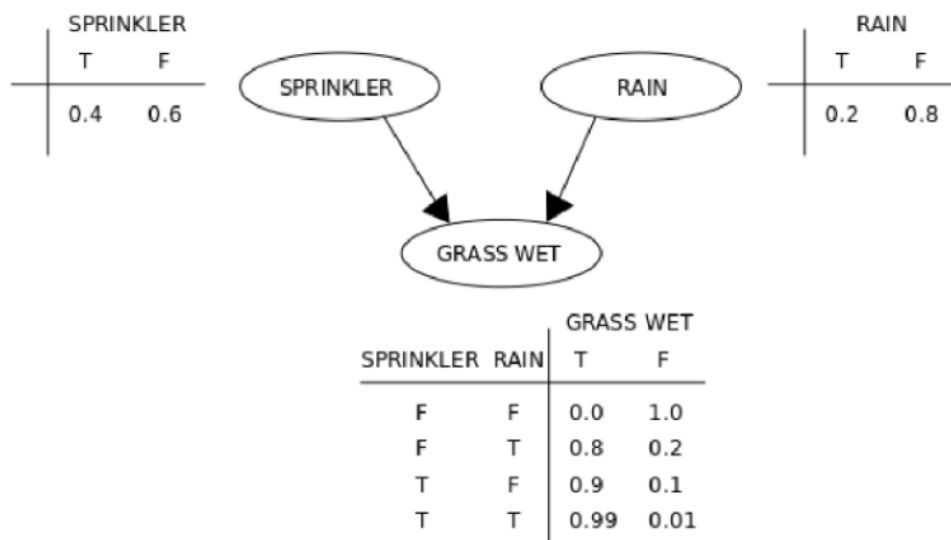
Then:

$$P(X_1|X_2) = \frac{P(X_1) \sum_{X_3} P(X_3) P(X_2|X_1, X_3)}{\sum_{X_1} \sum_{X_3} P(X_2|X_1, X_3) \tau(X_1) \tau(X_2)}$$

Similarly, we can compute  $P(X_3|X_2)$ :

$$P(X_3|X_2) = \frac{P(X_3) \sum_{X_1} P(X_1) P(X_2|X_3, X_1)}{\sum_{X_3} \sum_{X_1} P(X_2|X_3, X_1) \tau(X_3) \tau(X_2)}$$

**Example:**



$$\begin{aligned}
 P(X_2) &= \sum_{X_1} \sum_{X_3} P(X_2|X_1, X_3)P(X_1)P(X_3) \\
 &= P(X_2|X_1, X_3)P(X_1)P(X_3) + P(X_2|\neg X_1, X_3)P(\neg X_1)P(X_3) + P(X_2|X_1, \neg X_3)P(X_1)P(\neg X_3) + \\
 &\quad P(X_2|\neg X_1, \neg X_3)P(\neg X_1)P(\neg X_3) \\
 &= 0.99 * 0.4 * 0.2 + 0.8 * 0.6 * 0.2 + 0.9 * 0.4 * 0.8 + 0 * 0.6 * 0.8 = 0.4632
 \end{aligned}$$

$$\begin{aligned}
 P(X_1, X_2) &= \sum_{X_3} P(X_1, X_2, X_3) \\
 &= P(X_1)P(\neg X_3)P(X_2|X_1, \neg X_3) + P(X_1)P(X_3)P(X_2|X_1, X_3) \\
 &= 0.4 * 0.8 * 0.9 + 0.4 * 0.2 * 0.99 = 0.3672
 \end{aligned}$$

$$P(X_1|X_2) = \frac{P(X_1, X_2)}{P(X_2)} = \frac{0.3672}{0.4632} \approx 0.79$$

## 2. Case 2: $X_1 \leftarrow X_2 \rightarrow X_3$ :

General formula:  $X_1 \leftarrow X_2 \leftarrow \dots \leftarrow X_{n-1} \leftarrow X_n \rightarrow X_{n+1} \rightarrow \dots \rightarrow X_{m-1} \rightarrow X_m$

Let:

$$\begin{aligned}
 P(X_1) &= \sum_{X_2} \dots \sum_{X_{n-1}} \sum_{X_n} \dots \sum_{X_{m-1}} \sum_{X_m} P(X_1|X_2)P(X_2|X_3) \dots P(X_{n-1}|X_n)P(X_n)P(X_{n+1}|X_n) \dots P(X_m|X_{m-1}) \\
 &= \sum_{X_2} P(X_1|X_2) \sum_{X_3} P(X_2|X_3) \dots \sum_{X_n} P(X_{n-1}|X_n)P(X_n) \sum_{X_{n+1}} P(X_{n+1}|X_n) \dots \sum_{X_m} P(X_m|X_{m-1})
 \end{aligned}$$

Since,  $\forall x \in [n+1; m]$ ,  $X_1$  and  $X_x$  are independent. Then, in the formula of  $P(X_1)$ , we always have:

$$\sum_{X_x} P(X_x|X_{x-1}) = 1, \forall x \in [n+1; m]$$

$\Rightarrow$

$$\sum_{X_{n+1}} P(X_{n+1}|X_n) \dots \sum_{X_m} P(X_m|X_{m-1}) = 1 * 1 * 1 \dots * 1 = 1$$

Therefore:

$$P(X_1) = \sum_{X_2} P(X_1|X_2) \sum_{X_3} P(X_2|X_3) \dots \sum_{X_n} P(X_{n-1}|X_n)P(X_n) \quad (3)$$

Just the same as Cascade. We have:

$$\begin{aligned}
 \tau(X_{i+1}) &:= \sum_{X_i} P(X_{i+1}|X_i)\tau_{i-1}(X_i) \\
 &\Rightarrow \tau_{X_2} = \sum_{X_3} P(X_2|X_3)T_{X_3} \\
 \tau_{X_3} &= \sum_{X_4} P(X_3|X_4)T_{X_4} \quad (4)
 \end{aligned}$$

....

$$\begin{aligned}
 \tau_{X_{n-1}} &= \sum_{x_n} P(X_n)P(X_{n-1}|X_n) \\
 &\Rightarrow P(X_1) = \sum_{X_2} P(X_1|X_2)\tau(X_2) \quad (5)
 \end{aligned}$$

Similarly:

$$\begin{aligned}
 P(X_m) &= \sum_{X_m} \sum_{X_{m-1}} \dots \sum_{X_n} P(X_m|X_{m-1})P(X_{m-1}|X_{m-2})\dots P(X_{n+1}|X_n)P(X_n) \\
 &= \sum_{X_{m-1}} P(X_m|X_{m-1})\tau(X_{m-1})
 \end{aligned} \tag{6}$$

Then, apply to this case: We have:

$$P(X_1) = \sum_{X_2} P(X_1|X_2)\tau(X_2)$$

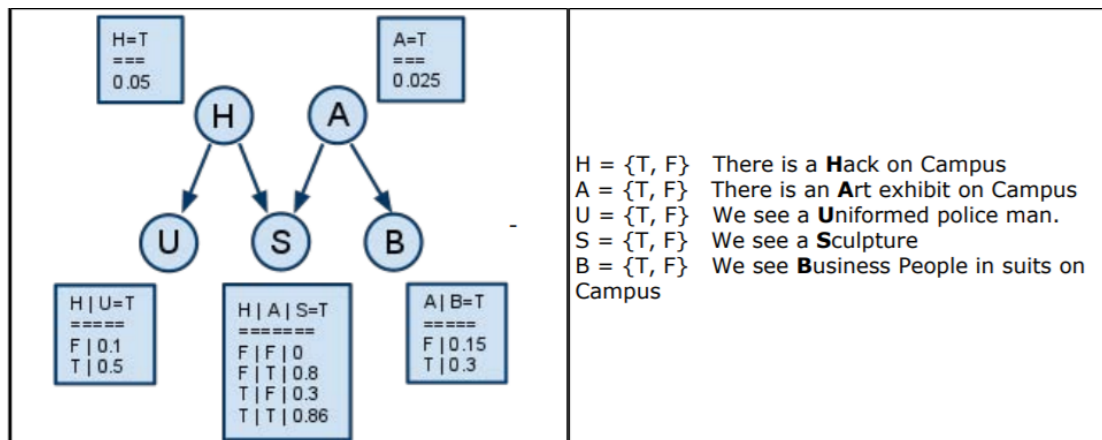
$$P(X_3) = \sum_{X_2} P(X_3|X_2)\tau(X_2)$$

Now, We compute  $P(X_2|X_1)$  and  $P(X_2|X_3)$ :

$$P(X_2|X_1) = \frac{P(X_1, X_2)}{P(X_1)} = \frac{P(X_2)P(X_1|X_2)}{\sum_{X_2} P(X_1|X_2)\tau(X_2)}$$

$$P(X_2|X_3) = \frac{P(X_3, X_2)}{P(X_3)} = \frac{P(X_2)P(X_3|X_2)}{\sum_{X_2} P(X_3|X_2)\tau(X_2)}$$

**Example:**



### Marginalization over the Joint

#### Example: How to compute the probability of P(S)?

We can compute any arbitrary probabilities from joint probabilities by the method of "marginalization" = summing out variables that we don't want.

$$P(S) = \sum_{H,A,U,B} P(H,A,U,S,B)$$

$$P(S) = \sum_{H,A,U,B} P(H)P(A)P(U|H)P(S|H,A)P(B|A)$$

$$P(S) = \sum_H \sum_A \sum_U \sum_B P(H)P(A)P(U|H)P(S|H,A)P(B|A)$$

Next move the sums so that a sum is placed only before all the terms that depend on it.  
e.g.  $P(S|H,A)$  depends on sum A and sum H so those sums occur left of it:

$$P(S) = \sum_A P(A) \sum_H P(H)P(S|H,A) \sum_U P(U|H) \sum_B P(B|A)$$

Notice that:  $\sum_B P(B|A) = 1$ , and same for  $\sum_U P(U|H) = 1$  !

#### Probability

So dropping the B and U terms we get the final summation:

$$P(S) = \sum_A P(A) \sum_H P(H)P(S|H,A)$$

Which works out to:

$$\begin{aligned} P(S) &= P(H)P(A)P(S|H,A) + P(\bar{H})P(A)P(S|\bar{H},A) + P(H)P(\bar{A})P(S|H,\bar{A}) \\ &\quad + P(\bar{H})P(\bar{A})P(S|\bar{H},\bar{A}) \\ &= \mathbf{0.0347} \end{aligned}$$

### 2.3.3 Bayesian model (Exercise 3c)

$X_1 \rightarrow X_2 \rightarrow X_3 \leftarrow X_4 \rightarrow X_5$  Let:

$$\tau(X_2) = \sum_{X_1} P(X_1)P(X_2|X_1)$$

$$\tau(X_4) = P(4)$$

We can see that,  $X_2 \rightarrow X_3 \leftarrow X_4$  has explaining away shape. Therefore, applying (1) and (2):

$$P(X_3) = \sum_{X_2} \sum_{X_4} P(X_3|X_2, X_4) \tau(X_2) \tau(X_4)$$

Similarly,  $X_3 \leftarrow X_4 \rightarrow X_5$  has common parent shape. Therefore, applying (3) and (5):

$$P(X_5) = \sum_{X_4} P(X_5|X_4) \tau(X_4)$$

Now we would like to evaluate  $P(X_1|X_3)$ ,  $P(X_4|X_3)$ ,  $P(X_4|X_5)$ . To do that, first we need to find  $P(X_1, X_3)$ ,  $P(X_3, X_4)$ ,  $P(X_4, X_5)$ :

$$\begin{aligned} P(X_1, X_3) &= \sum_{X_2} \sum_{X_4} \sum_{X_5} P(X_1, X_2, X_3, X_4, X_5) \\ &= \sum_{X_2} \sum_{X_4} P(X_1)P(X_4)P(X_2|X_1)P(X_3|X_2, X_4) \sum_{X_5} P(X_5|X_4) \\ &= \sum_{X_2} \sum_{X_4} P(X_1)P(X_4)P(X_2|X_1)P(X_3|X_2, X_4) \end{aligned}$$

Since  $X_1$  and  $X_3$  are independent with  $X_5$ , then:

$$\sum X_5 P(X_5|X_4) = 1$$

Therefore:

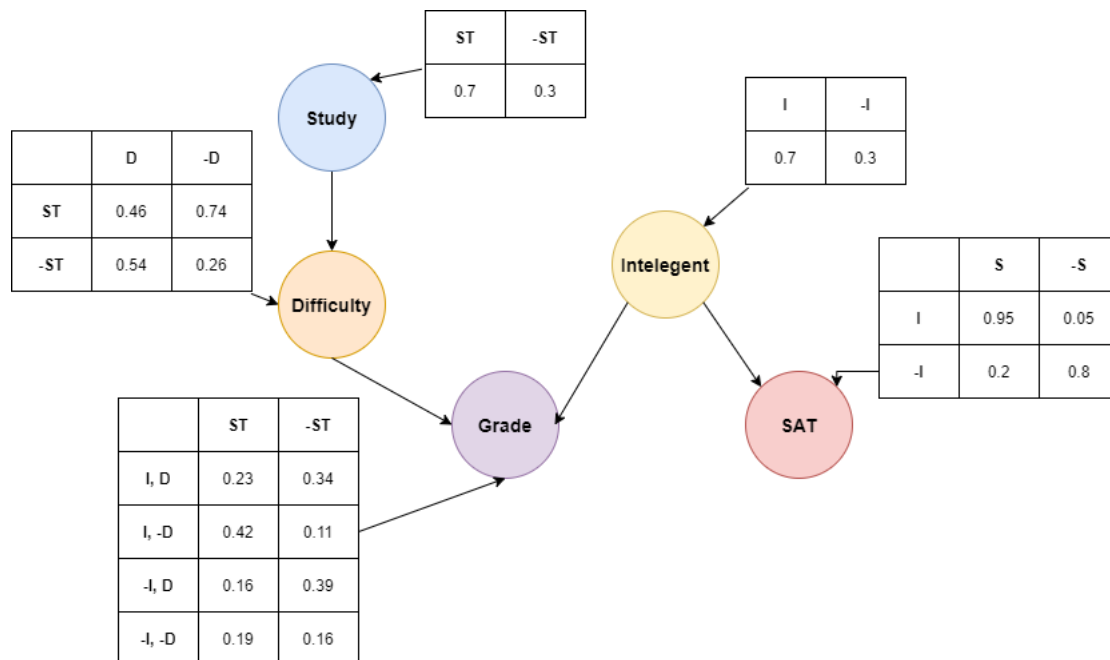
$$P(X_1, X_3) = \sum_{X_2} \sum_{X_4} P(X_1)P(X_4)P(X_2|X_1)P(X_3|X_2, X_4)$$

$$\begin{aligned} P(X_3, X_4) &= \sum_{X_1} \sum_{X_2} \sum_{X_5} P(X_1, X_2, X_3, X_4, X_5) \\ &= \sum_{X_1} \sum_{X_2} \sum_{X_5} P(X_1)P(X_4)P(X_2|X_1)P(X_3|X_2, X_4)P(X_5|X_4) \\ &= \sum_{X_1} P(X_1)P(X_4)P(X_3|X_4) \\ &= P(X_4)P(X_3|X_4) \end{aligned}$$

Case  $P(X_4, X_3)$

$$\begin{aligned} P(X_4, X_5) &= \sum_{X_1} \sum_{X_2} \sum_{X_3} P(X_1, X_2, X_3, X_4, X_5) \\ &= \sum_{X_1} \sum_{X_2} \sum_{X_3} P(X_1)P(X_4)P(X_2|X_1)P(X_3|X_4)P(X_5|X_4) \\ &= \sum_{X_1} \sum_{X_2} P(X_1)P(X_4)P(X_2|X_1)P(X_5|X_4) \\ &= \sum_{X_1} P(X_1)P(X_4)P(X_5|X_4) \\ &= P(X_4)P(X_5|X_4) \end{aligned}$$

**Examples:**



## 3 Applications

### 3.1 Data preprocessing

#### 3.1.1 Prepare a pandas data frame (Exercise 4a)

Import the necessary library:

```
import numpy as np %Help to work with Array
import pandas as pd %Help to work with dataframes
import matplotlib.pyplot as plt %Help to plot the figure in barchart,
import seaborn as sns %Make the dataframes performed in visualized way

matches = pd.read_csv(r"Uefa Euro Cup All Matches.csv")
cups=pd.read_csv(r"Uefa Euro Cup General Statistics.csv") %Read the data from the
excel form

cups %see what's in the dataframe
```

	Year	Host	Champion	Winning coach	Top scorer(s)	Player of the Tournament
0	1960	France	Soviet Union	Gavril Kachalin	François Heutte (2) Valentin Ivanov (2) Vi...	NaN
1	1964	Spain	Spain	José Villalonga	Ferenc Bene (2) Dezső Novák (2) Jesús Marí...	NaN
2	1968	Italy	Italy	Ferruccio Valcareggi	Dragan Džajić (2)	NaN
3	1972	Belgium	West Germany	Helmut Schön	Gerd Müller (4)	NaN
4	1976	Yugoslavia	Czechoslovakia	Václav Ježek	Dieter Müller (4)	NaN
5	1980	Italy	West Germany	Jupp Derwall	Klaus Allofs (3)	NaN
6	1984	France	France	Michel Hidalgo	Michel Platini (9)	NaN
7	1988	West Germany	Netherlands	Rinus Michels	Marco van Basten (5)	NaN
8	1992	Sweden	Denmark	Richard Møller Nielsen	Henrik Larsen (3) Karl-Heinz Riedle (3) De...	NaN
9	1996	England	Germany	Berti Vogts	Alan Shearer (5)	Matthias Sammer
10	2000	Belgium Netherlands	France	Roger Lemerre	Patrick Kluivert (5) Savo Milošević (5)	Zinedine Zidane
11	2004	Portugal	Greece	Otto Rehhagel	Milan Baroš (5)	Theodoros Zagorakis
12	2008	Austria Switzerland	Spain	Luis Aragonés	David Villa (4)	Xavi
13	2012	Poland Ukraine	Spain	Vicente del Bosque	Mario Mandžukić (3) Mario Gómez (3) Mario ...	Andrés Iniesta
14	2016	France	Portugal	Fernando Santos	Antoine Griezmann (6)	Antoine Griezmann

```

matches['HomeTeamName'] = matches['HomeTeamName'].apply(lambda x : x.replace(u'\xa0',
u')).apply(lambda x : x.strip())
matches['AwayTeamName'] = matches['AwayTeamName'].apply(lambda x : x.replace(u'\xa0',
u')).apply(lambda x : x.strip())
% Remove Latin space from AwayTeamName and HomeTeamName

matches.replace('Soviet Union', 'Russia', inplace=True)
matches.replace('West Germany', 'Germany', inplace=True)
cups.replace('Soviet Union', 'Russia', inplace=True)
cups.replace('West Germany', 'Germany', inplace=True)
cups.replace('Belgium Netherlands', 'Netherlands', inplace=True)
cups.replace('Austria Switzerland', 'Switzerland', inplace=True)
cups.replace('Poland Ukraine', 'Ukraine', inplace=True)
% Replace the old names to new names

del matches["Date"]
del matches["Stage"]
del matches["Time"]
del matches["Stadium"]
del matches["City"]
del matches["Attendance"]
del matches["SpecialWinConditions"]
del cups["Winning coach"]
del cups["Top scorer(s)"]
del cups["Player of the Tournament"]
% Delete what information that we don't think it is relevant much to the model. Finally,
we have 2 dataset cups and matches is what we think they are relevant to our model

```



### 3.1.2 Find and fill in or remove missing values in the data (Exercise 4b)

```
matches.isna().sum()
```

`cups.isna().sum()` %Let's detect some blank cell in the dataset. In the shown table, there is no blank cell.

HomeTeamName	0	Year	0
AwayTeamName	0	Host	0
HomeTeamGoals	0	Champion	0
AwayTeamGoals	0	dtype: int64	
Year	0		
dtype: int64			

`matches %matches dataset`

	HomeTeamName	AwayTeamName	HomeTeamGoals	AwayTeamGoals	Year
0	France	Yugoslavia	4	5	1960
1	Czechoslovakia	Russia	0	3	1960
2	Czechoslovakia	France	2	0	1960
3	Russia	Yugoslavia	2	1	1960
4	Spain	Hungary	2	1	1964
...	...	...	...	...	...
281	Germany	Italy	1	1	2016
282	France	Iceland	5	2	2016
283	Portugal	Wales	2	0	2016
284	Germany	France	0	2	2016
285	Portugal	France	1	0	2016

`cups %cups dataset`

	Year	Host	Champion
0	1960	France	Russia
1	1964	Spain	Spain
2	1968	Italy	Italy
3	1972	Belgium	Germany
4	1976	Yugoslavia	Czechoslovakia
5	1980	Italy	Germany
6	1984	France	France
7	1988	Germany	Netherlands
8	1992	Sweden	Denmark
9	1996	England	Germany
10	2000	Netherlands	France
11	2004	Portugal	Greece
12	2008	Switzerland	Spain
13	2012	Ukraine	Spain
14	2016	France	Portugal

```
cups["ChampCount"] = cups.groupby("Champion")["Year"].rank(method="first", ascending=True)
% Group the data to Champion, then count the number that country be the champion
by year.
% method="first": ranks assigned in order they appear in the array
% ascending=True: the elements should be ranked in ascending order

cups %Show the data after counting the number of champion time.
```

	Year	Host	Champion	ChampCount
0	1960	France	Russia	1.0
1	1964	Spain	Spain	1.0
2	1968	Italy	Italy	1.0
3	1972	Belgium	Germany	1.0
4	1976	Yugoslavia	Czechoslovakia	1.0
5	1980	Italy	Germany	2.0
6	1984	France	France	1.0
7	1988	Germany	Netherlands	1.0
8	1992	Sweden	Denmark	1.0
9	1996	England	Germany	3.0
10	2000	Netherlands	France	2.0
11	2004	Portugal	Greece	1.0
12	2008	Switzerland	Spain	2.0
13	2012	Ukraine	Spain	3.0
14	2016	France	Portugal	1.0

```

matches_by_goals1 = matches.groupby(['Year', 'HomeTeamName']).agg('HomeTeamGoals':
['sum'], 'AwayTeamGoals': ['sum', 'size'],)
matches_by_goals1.columns = ['HomeGoalsFor', 'HomeGoalsAgainst', 'HomeMatches']
matches_by_goals1.rename_axis(index='HomeTeamName': 'Country', inplace=True)
matches_by_goals1['HomeAveGoalsFor'] = matches_by_goals1.HomeGoalsFor/matches_by_goals1.HomeMatches
matches_by_goals1['HomeAveGoalsAgainst'] = matches_by_goals1.HomeGoalsAgainst/matches_by_goals1.HomeMatches
matches_by_goals2 = matches.groupby(['Year', 'AwayTeamName']).agg('AwayTeamGoals':
['sum'], 'HomeTeamGoals': ['sum', 'size'],)
matches_by_goals2.columns = ['AwayGoalsFor', 'AwayGoalsAgainst', 'AwayMatches']
matches_by_goals2.rename_axis(index='AwayTeamName': 'Country', inplace=True)
matches_by_goals2['AwayAveGoalsFor'] = matches_by_goals2.AwayGoalsFor/matches_by_goals2.AwayMatches
matches_by_goals2['AwayAveGoalsAgainst'] = matches_by_goals2.AwayGoalsAgainst/matches_by_goals2.AwayMatches
% Calculate the average of GoalsFor and GoalsAgainst

x = pd.concat([matches_by_goals1, matches_by_goals2], axis=1)
x = x.fillna(0)
x['GoalsFor'] = x.HomeGoalsFor + x.AwayGoalsFor
x['GoalsAgainst'] = x.HomeGoalsAgainst + x.AwayGoalsAgainst
x['Matches'] = x.HomeMatches + x.AwayMatches
del x['HomeGoalsFor']
del x['AwayGoalsFor']
del x['HomeGoalsAgainst']
del x['AwayGoalsAgainst']
% We combine two dataset HomeTeam and AwayTeam into one dataset of every Country.
plot = x % This is used for later plotting the figure
x %show the dataset

```

		HomeMatches	HomeAveGoalsFor	HomeAveGoalsAgainst	AwayMatches	AwayAveGoalsFor	AwayAveGoalsAgainst	GoalsFor	GoalsAgainst	Matches
Year	Country									
1960	Czechoslovakia	2.0	1.0	1.500000	0.0	0.000000	0.000000	2.0	3.0	2.0
	France	1.0	4.0	5.000000	1.0	0.000000	2.000000	4.0	7.0	2.0
	Russia	1.0	2.0	1.000000	1.0	3.000000	0.000000	5.0	1.0	2.0
	Yugoslavia	0.0	0.0	0.000000	2.0	3.000000	3.000000	6.0	6.0	2.0
1964	Denmark	1.0	0.0	3.000000	1.0	1.000000	3.000000	1.0	6.0	2.0
...	...	...	...	...	...	...	...	...	...	...
2016	Sweden	1.0	0.0	1.000000	2.0	0.500000	1.000000	1.0	3.0	3.0
	Switzerland	2.0	0.5	0.500000	2.0	1.000000	0.500000	3.0	2.0	4.0
	Turkey	1.0	0.0	1.000000	2.0	1.000000	1.500000	2.0	4.0	3.0
	Ukraine	2.0	0.0	1.500000	1.0	0.000000	2.000000	0.0	5.0	3.0
	Wales	3.0	2.0	0.666667	3.0	1.333333	1.333333	10.0	6.0	6.0

156 rows x 9 columns

```
x['AveGoalsFor'] = x.GoalsFor/x.Matches
x['AveGoalsAgainst'] = x.GoalsAgainst/x.Matches
% Calculate Goals/Matches del x['GoalsFor']
del x['GoalsAgainst']
x = x.reindex(columns=['Matches', 'AveGoalsFor', 'AveGoalsAgainst', 'HomeMatches',
'HomeAveGoalsFor', 'HomeAveGoalsAgainst', 'AwayMatches', 'AwayAveGoalsFor', 'AwayAveGoalsAgainst'])
% Change the order of Columns x.isna().sum()
% Check again if there is any blank cell.

cups1 = cups.groupby(['Year', 'Champion', 'ChampCount']).agg('Host': ['size'],)
cups1.rename_axis(index='Champion': 'Country', inplace=True)
cups1.columns = ['Champion']
cups1 = cups1.reset_index(level=['ChampCount'])
% Group to index Year, Country. Count the number of champion time

cupsfix=pd.DataFrame([[2000, 'Belgium'], [2008, 'Austria'], [2012, 'Poland']], columns=['Year',
'Host'])
cupsfix = pd.merge(cups, cupsfix, how="outer", on=["Year", "Host"])
% Separate and add 3 countries that co-host with three others in the 3 years.
cups2 = cupsfix.groupby(['Year', 'Host']).agg('Champion': ['size'],)
cups2.rename_axis(index='Host': 'Country', inplace=True)
cups2.columns = ['Host']
% Get the Host time of the country by Year.

x = pd.concat([x, cups2, cups1], axis=1)
x.Champion = x.Champion.fillna('false')
x.Host = x.Host.fillna('false')
x.Champion = x.Champion.replace(1, 'true')
x.Host = x.Host.replace(1, 'true')
% Combine cups1 and cups2 into a dataset to use. Replace the value 1 and 0 in Host
and Champion column to true and false
x % show the dataset
```

		Matches	AveGoalsFor	AveGoalsAgainst	HomeMatches	HomeAveGoalsFor	HomeAveGoalsAgainst
Year	Country						
1960	Czechoslovakia	2.0	1.000000	1.500000	2.0	1.0	1.500000
	France	2.0	2.000000	3.500000	1.0	4.0	5.000000
	Russia	2.0	2.500000	0.500000	1.0	2.0	1.000000
	Yugoslavia	2.0	3.000000	3.000000	0.0	0.0	0.000000
1964	Denmark	2.0	0.500000	3.000000	1.0	0.0	3.000000
...	...	...	...	...	...	...	...
2016	Sweden	3.0	0.333333	1.000000	1.0	0.0	1.000000
	Switzerland	4.0	0.750000	0.500000	2.0	0.5	0.500000
	Turkey	3.0	0.666667	1.333333	1.0	0.0	1.000000
	Ukraine	3.0	0.000000	1.666667	2.0	0.0	1.500000
	Wales	6.0	1.666667	1.000000	3.0	2.0	0.666667

AwayMatches	AwayAveGoalsFor	AwayAveGoalsAgainst	Host	ChampCount	Champion
0.0	0.000000	0.000000	false	NaN	false
1.0	0.000000	2.000000	true	NaN	false
1.0	3.000000	0.000000	false	1.0	true
2.0	3.000000	3.000000	false	NaN	false
1.0	1.000000	3.000000	false	NaN	false
...	...	...	...	...	...
2.0	0.500000	1.000000	false	NaN	false
2.0	1.000000	0.500000	false	NaN	false
2.0	1.000000	1.500000	false	NaN	false
1.0	0.000000	2.000000	false	NaN	false
3.0	1.333333	1.333333	false	NaN	false

```
x = x.reset_index(level=['Country'])
M = ["Russia", "Spain", "Italy", "Germany", "Czechoslovakia", "France", "Netherlands",
"Denmark", "Portugal"]
for i in M:
x.loc[x['Country'] == i, 'ChampCount'] = x.loc[x['Country'] == i, 'ChampCount'].fillna(method='ffill')
% Count the ChamCount: The Champion times counted from the year considered before.
x.ChampCount = x.ChampCount.fillna(0)
% Fill all the NaN by 0.

x.isna().sum()
% Check for NaN again
```

### 3.1.3 Discretize your data set (Exercise 4c)

```
x.agg('Matches' : ['min','max'], 'AveGoalsFor' : ['min','max'], 'AveGoalsAgainst'
: ['min','max'], 'HomeMatches' : ['min','max'], 'HomeAveGoalsFor' : ['min','max'],
```

```
'HomeAveGoalsAgainst' : ['min','max'], 'AwayMatches' : ['min','max'], 'AwayAveGoalsFor'
: ['min','max'], 'AwayAveGoalsAgainst' : ['min','max'])
% Find max min to set the interval of every columns.
```

	Matches	AveGoalsFor	AveGoalsAgainst	HomeMatches	HomeAveGoalsFor
min	2.0	0.0	0.166667	0.0	0.0
max	7.0	3.0	3.500000	5.0	4.0

	HomeAveGoalsAgainst	AwayMatches	AwayAveGoalsFor	AwayAveGoalsAgainst
	0.0	0.0	0.0	0.0
	5.0	5.0	3.0	4.0

```
choices = ['low', 'mid', 'high']
% Make choices
conds = [x.Matches.values <= 2, x.Matches.values <=5 , x.Matches.values >5]
x['Matches'] = np.select(conds, choices)

conds = [x.AveGoalsFor.values <= 1 , x.AveGoalsFor.values <=2.5 , x.AveGoalsFor.values
>2.5]
x['AveGoalsFor'] = np.select(conds, choices)

conds = [x.AveGoalsAgainst.values <= 1 , x.AveGoalsAgainst.values <=2.5 , x.AveGoalsAgainst.va
>2.5]
x['AveGoalsAgainst'] = np.select(conds, choices)

conds = [x.HomeMatches.values <= 2 , x.HomeMatches.values <=4 , x.HomeMatches.values
>4]
x['HomeMatches'] = np.select(conds, choices)

conds = [x.HomeAveGoalsFor.values <= 1 , x.HomeAveGoalsFor.values <=2.5 , x.HomeAveGoalsFor.va
>2.5]
x['HomeAveGoalsFor'] = np.select(conds, choices)

conds = [x.HomeAveGoalsAgainst.values <= 1 , x.HomeAveGoalsAgainst.values <=2.5
, x.HomeAveGoalsAgainst.values >2.5]
x['HomeAveGoalsAgainst'] = np.select(conds, choices)

conds = [x.AwayMatches.values <= 2 , x.AwayMatches.values <=4 , x.AwayMatches.values
>4]
x['AwayMatches'] = np.select(conds, choices)

conds = [x.AwayAveGoalsFor.values <= 1 , x.AwayAveGoalsFor.values <=2.5 , x.AwayAveGoalsFor.va
>2.5]
x['AwayAveGoalsFor'] = np.select(conds, choices)

conds = [x.AwayAveGoalsAgainst.values <= 1 , x.AwayAveGoalsAgainst.values <=2.5
, x.AwayAveGoalsAgainst.values >2.5]
```



```
x['AwayAveGoalsAgainst'] = np.select(conds, choices)
% Discrete our dataset
```

X

Year	Country	Matches	AveGoalsFor	AveGoalsAgainst	HomeMatches	HomeAveGoalsFor	HomeAveGoalsAgainst	AwayMatches	AwayAveGoalsFor	AwayAveGoalsAgainst	Host	ChampCount
1960	Czechoslovakia	low	low	mid	low	low	mid	low	low	low	false	0.0
1960	France	low	mid	high	low	high	high	low	low	mid	true	0.0
1960	Russia	low	mid	low	low	mid	low	low	high	low	false	1.0
1960	Yugoslavia	low	high	high	low	low	low	low	high	high	false	0.0
1964	Denmark	low	low	high	low	low	high	low	low	high	false	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
2016	Sweden	mid	low	low	low	low	low	low	low	low	false	0.0
2016	Switzerland	mid	low	low	low	low	low	low	low	low	false	0.0
2016	Turkey	mid	low	mid	low	low	low	low	low	mid	false	0.0
2016	Ukraine	mid	low	mid	low	low	mid	low	low	mid	false	0.0
2016	Wales	high	mid	low	mid	mid	low	mid	mid	mid	false	0.0

156 rows x 13 columns

### 3.1.4 Save your prepared data as a .csv file for later uses (Exercise 4d)

```
y = x.index.values
y = np.unique(y)
y1 = arr = np.array([1964, 1972, 2012])
test = x.loc[y1]
% Take dataset information to the array in 1964, 1972 and 2012, stored in Test.csv
```

```
y2 = [x for x in y if x not in y1]
train = x.loc[y2]
% Take dataset information to the array in other years (84.61%), stored in Train.csv
```

```
train.to_csv(r'Train.csv')
test.to_csv(r'Test.csv')
%send it to excel file
```

## 3.2 Problem modeling

### 3.2.1 Overview of the data (Exercise 5a)

```
from pgmpy.models import BayesianModel
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination

raw = pd.read_csv(r"RawFullData.csv") FullData = pd.read_csv(r"FullData.csv") Train
= pd.read_csv(r"Train.csv") Test = pd.read_csv(r"Test.csv") cups = pd.read_csv(r"cups.csv")
%Read the data from the excel form

FullData = FullData.set_index('Year')
```

```
Train = Train.set_index('Year')
Test = Test.set_index('Year')
% Set index to year

Static = raw
del Static['Year']
Static=Static.groupby('Country').agg('Matches': ['sum']) %Sum all matches by country
Static.columns = ['MatchesSum'] % Change name to MatchesSum
cupsStatic = cups.groupby(['Champion']).agg('Host': ['size'],) % Number become Champion
by country
cupsStatic.columns = ['ChampionTimes'] % Change name to ChampionTimes
Static = pd.concat([Static, cupsStatic], axis=1) %Combine ChampionTimes and MatchesSum
Static.ChampionTimes = Static.ChampionTimes.fillna(0)
Static.sort_values(by='MatchesSum',ascending=False)
%Arrange data
```

	MatchesSum	ChampionTimes		MatchesSum	ChampionTimes
Germany	49.0	3.0	Republic of Ireland	10.0	0.0
Spain	40.0	3.0	Yugoslavia	10.0	0.0
France	39.0	2.0	Hungary	8.0	0.0
Italy	38.0	1.0	Czechoslovakia	8.0	1.0
Netherlands	35.0	1.0	Austria	6.0	0.0
Portugal	35.0	1.0	Scotland	6.0	0.0
England	31.0	0.0	Bulgaria	6.0	0.0
Russia	30.0	1.0	Ukraine	6.0	0.0
Denmark	27.0	1.0	Wales	6.0	0.0
Czech Republic	24.0	0.0	Iceland	5.0	0.0
Sweden	20.0	0.0	Northern Ireland	4.0	0.0
Croatia	18.0	0.0	Slovakia	4.0	0.0
Belgium	17.0	0.0	FR Yugoslavia	4.0	0.0
Greece	16.0	1.0	Norway	3.0	0.0
Romania	16.0	0.0	Latvia	3.0	0.0
Turkey	15.0	0.0	Slovenia	3.0	0.0
Switzerland	13.0	0.0	OS	3.0	0.0
Poland	11.0	0.0	Albania	3.0	0.0

### 3.2.2 Plots (Exercise 5b)

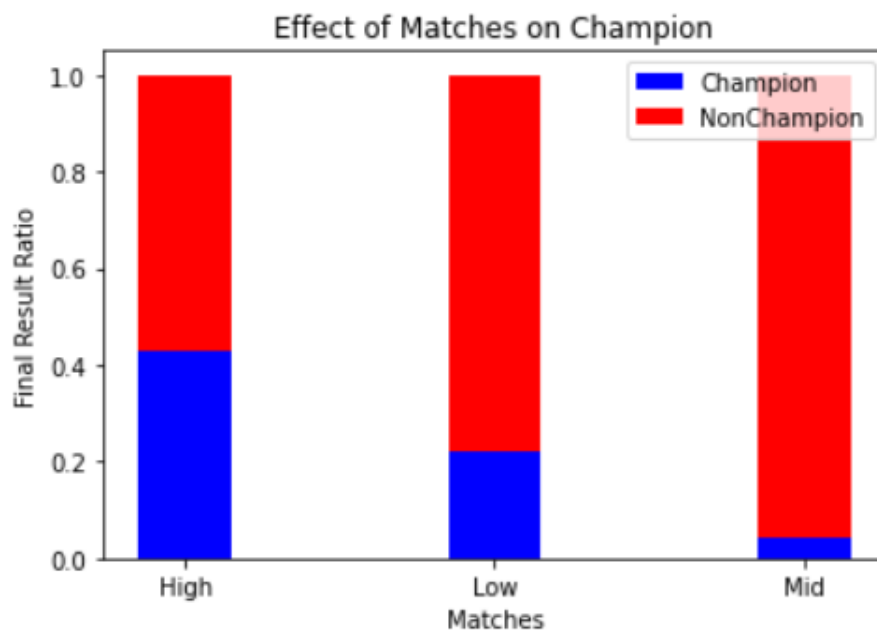
```
plot1 = x.loc[x.Champion == 'true']
plot1 = plot1.groupby('Matches').agg('Champion': ['size'])
plot1.columns = ['Champ']
plot1 = plot1.reset_index(level=['Matches'])
% group champion flowing Discrete of Matches
plot2 = x.loc[x.Champion == 'false']
plot2 = plot2.groupby('Matches').agg('Champion': ['size'])
plot2.columns = ['NonChamp']
plot2 = plot2.reset_index(level=['Matches'])
% group non-champion flowing Discrete of Matches
plot = pd.merge(plot1, plot2, how="outer", on=["Matches"])

combine champoin and non-champion
plot['Champ'] =plot['Champ']/(plot['Champ'] + plot['NonChamp'])
plot['NonChamp'] = 1 - plot['Champ']
% Proportion of champion and non-champion flowing Discrete of Matches
plot
```



	Matches	Champ	NonChamp
0	high	0.428571	0.571429
1	low	0.222222	0.777778
2	mid	0.040323	0.959677

```
Matches = ["High", "Low", "Mid"]
width = 0.3
plt.bar(plot.index, plot.Champ.to_numpy(), width, color = "blue", label="Champion")
plt.bar(plot.index, plot.NonChamp.to_numpy(), width, color = "red", label="NonChampion",
bottom = plot.Champ.to_numpy())
plt.title("Effect of Matches on Champion") plt.xlabel("Matches")
plt.ylabel("Final Result Ratio")
plt.xticks(plot.index, Matches)
plt.legend(loc='best')
plt.show()
```

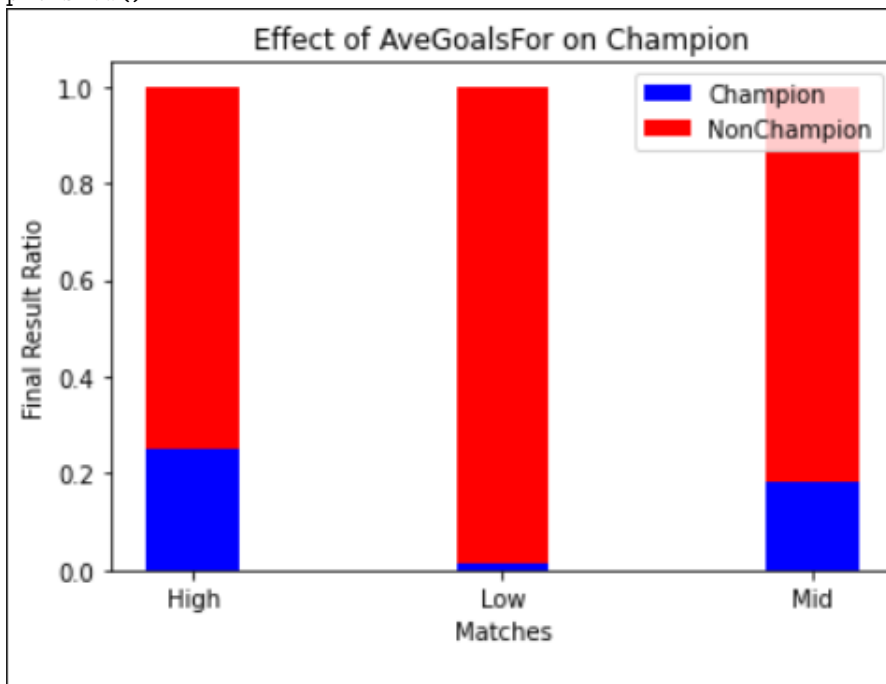


```
plot1 = x.loc[x.Champion == 'true']
plot1 = plot1.groupby('AveGoalsFor').agg('Champion': ['size'])
plot1.columns = ['Champ']
plot1 = plot1.reset_index(level=['AveGoalsFor'])
plot2 = x.loc[x.Champion == 'false']
plot2 = plot2.groupby('AveGoalsFor').agg('Champion': ['size'])
plot2.columns = ['NonChamp']
plot2 = plot2.reset_index(level=['AveGoalsFor'])
plot = pd.merge(plot1, plot2, how="outer", on=["AveGoalsFor"])
```

```
plot['Champ'] = plot['Champ'] / (plot['Champ'] + plot['NonChamp'])
plot['NonChamp'] = 1 - plot['Champ']
plot
```

	AveGoalsFor	Champ	NonChamp
0	high	0.250000	0.750000
1	low	0.012346	0.987654
2	mid	0.183099	0.816901

```
AveGoalsFor = ["High", "Low", "Mid"]
width = 0.3
plt.bar(plot.index, plot.Champ.to_numpy(), width, color = "blue", label="Champion")
plt.bar(plot.index, plot.NonChamp.to_numpy(), width, color = "red", label="NonChampion",
bottom = plot.Champ.to_numpy())
plt.title("Effect of AveGoalsFor on Champion")
plt.xlabel("Matches")
plt.ylabel("Final Result Ratio")
plt.xticks(plot.index, AveGoalsFor)
plt.legend(loc='best')
plt.show()
```

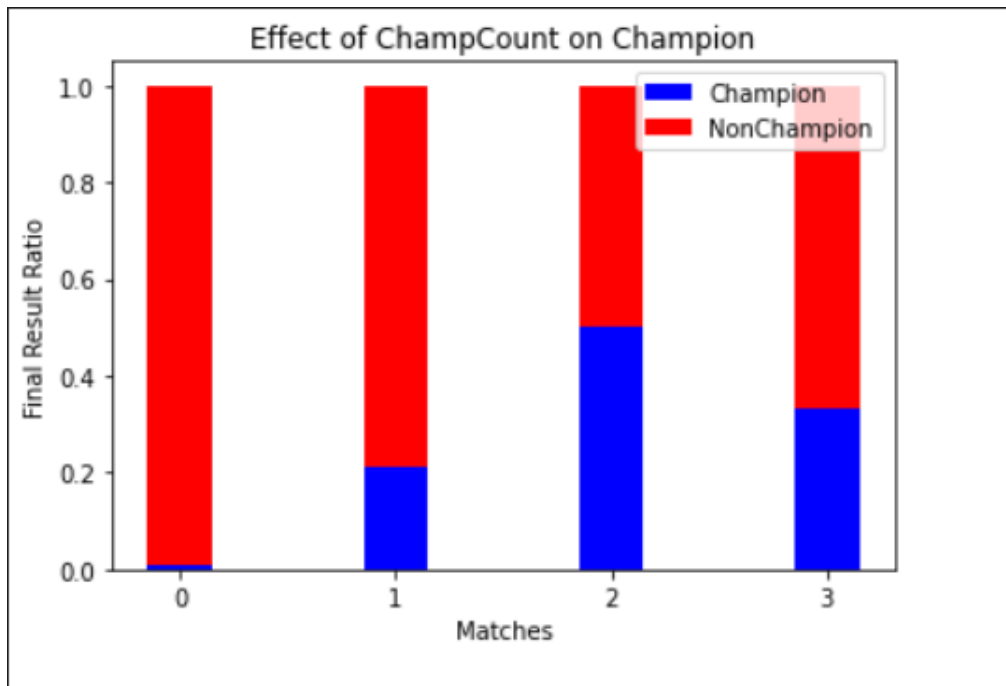


```
plot1 = x.loc[x.Champion == 'true']
plot1 = plot1.groupby('ChampCount').agg('Champion': ['size'])
plot1.columns = ['Champ']
plot1 = plot1.reset_index(level=['ChampCount'])
```

```
plot2 = x.loc[x.Champion == 'false']  
plot2 = plot2.groupby('ChampCount').agg('Champion': ['size'])  
plot2.columns = ['NonChamp']  
plot2 = plot2.reset_index(level=['ChampCount'])  
plot = pd.merge(plot1, plot2, how="outer", on=["ChampCount"])  
plot['Champ'] = plot['Champ'] / (plot['Champ'] + plot['NonChamp'])  
plot['NonChamp'] = 1 - plot['Champ']  
plot
```

	ChampCount	Champ	NonChamp
0	0.0	0.009901	0.990099
1	1.0	0.209302	0.790698
2	2.0	0.500000	0.500000
3	3.0	0.333333	0.666667

```
ChampCount = ["0", "1", "2", "3"]  
width = 0.3  
plt.bar(plot.index, plot.Champ.to_numpy(), width, color = "blue", label="Champion")  
plt.bar(plot.index, plot.NonChamp.to_numpy(), width, color = "red", label="NonChampion",  
bottom = plot.Champ.to_numpy())  
plt.title("Effect of ChampCount on Champion")  
plt.xlabel("Matches")  
plt.ylabel("Final Result Ratio")  
plt.xticks(plot.index, ChampCount)  
plt.legend(loc='best')  
plt.show()
```

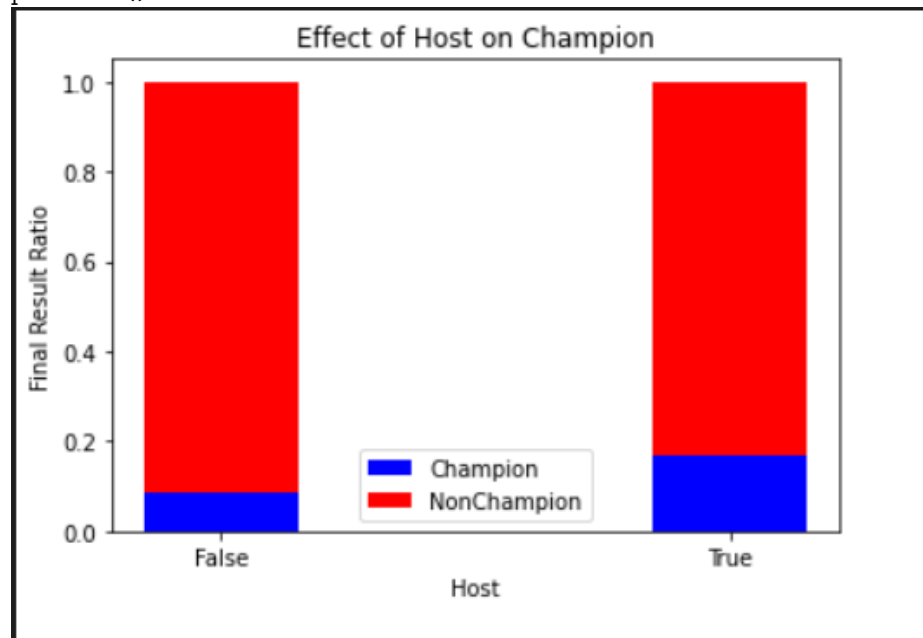


```
plot1 = x.loc[x.Champion == 'true']
plot1 = plot1.groupby('Host').agg('Champion': ['size'])
plot1.columns = ['Champ']
plot1 = plot1.reset_index(level=['Host'])
plot2 = x.loc[x.Champion == 'false']
plot2 = plot2.groupby('Host').agg('Champion': ['size'])
plot2.columns = ['NonChamp']
plot2 = plot2.reset_index(level=['Host'])
plot = pd.merge(plot1, plot2, how="outer", on=["Host"])
plot['Champ'] = plot['Champ'] / (plot['Champ'] + plot['NonChamp'])
plot['NonChamp'] = 1 - plot['Champ']
plot
```

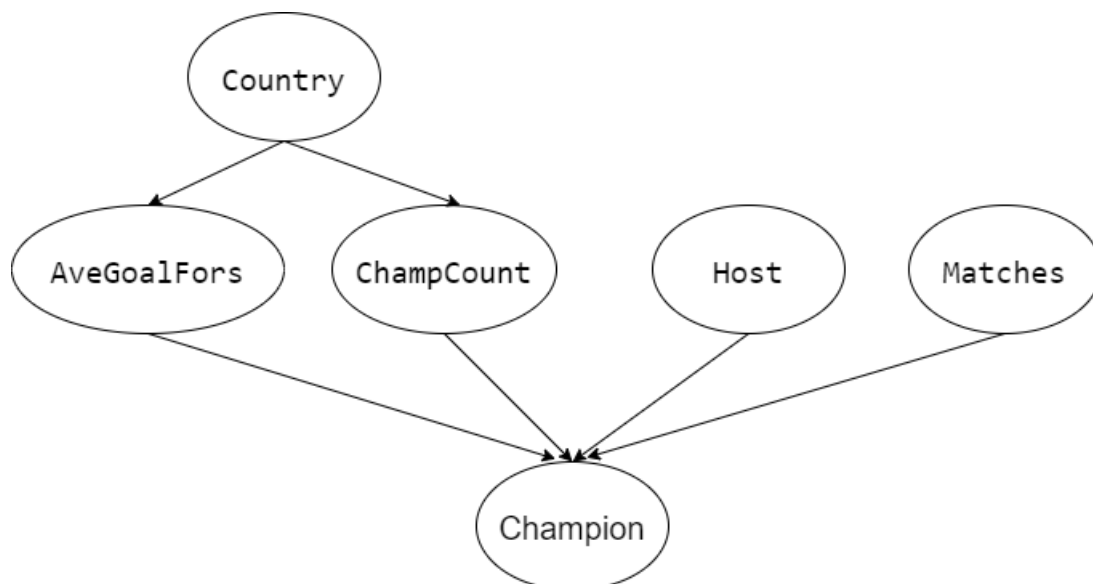
	Host	Champ	NonChamp
0	false	0.086957	0.913043
1	true	0.166667	0.833333

```
Host = ["False", "True"]
width = 0.3
plt.bar(plot.index, plot.Champ.to_numpy(), width, color = "blue", label="Champion")
plt.bar(plot.index, plot.NonChamp.to_numpy(), width, color = "red", label="NonChampion",
bottom = plot.Champ.to_numpy())
plt.title("Effect of Host on Champion")
plt.xlabel("Host")
plt.ylabel("Final Result Ratio")
```

```
plt.xticks(plot.index, Host)  
plt.legend(loc='best')  
plt.show()
```



### 3.2.3 Prepare Bayesian Model (Exercise 5c)



## 4 Training and predicting

### 4.1 Exercise 6a

```
c1 = train
del c1['HomeMatches']
del c1['HomeAveGoalsFor']
del c1['HomeAveGoalsAgainst']
del c1['AwayMatches']
del c1['AwayAveGoalsFor']
del c1['AwayAveGoalsAgainst']
del c1['AveGoalsAgainst']
% Delete unnecessary columns

from pgmpy.models import BayesianModel
% Call the BayesianModel command from library pgmpy.models.
model = BayesianModel([('Country', 'AveGoalsFor'), ('Country', 'ChampCount'), ('AveGoalsFor',
'Champion'), ('Host', 'Champion'), ('ChampCount', 'Champion'), ('Matches', 'Champion')])

from pgmpy.estimators import MaximumLikelihoodEstimator
model.fit(c1, estimator = MaximumLikelihoodEstimator)
% Use command MaximumLikelihoodEstimator to make machine learn the data.

from pgmpy.inference import VariableElimination
infer = VariableElimination(model)
% Use command VariableElimination applied to the model.
print(model.get_cpds('Matches'))
```



```
print(model.get_cpds('Host'))
```

```
+-----+-----+
| Host(false) | 0.893939 |
+-----+-----+
| Host(true)  | 0.106061 |
+-----+-----+
```

```
print(model.get_cpds('Country'))
```

Country	
(Albania)	0.00757576
(Austria)	0.0151515
(Belgium)	0.030303
(Bulgaria)	0.0151515
(CIS)	0.00757576
(Croatia)	0.030303
(Czech Republic)	0.0378788
(Czechoslovakia)	0.0227273
(Denmark)	0.0454545
(England)	0.0606061
(FR Yugoslavia)	0.00757576
(France)	0.0606061
(Germany)	0.0757576
(Greece)	0.0227273
(Hungary)	0.00757576
(Iceland)	0.00757576
(Italy)	0.0606061
(Latvia)	0.00757576
(Netherlands)	0.0606061
(Northern Ireland)	0.00757576
(Norway)	0.00757576
(Poland)	0.0151515
(Portugal)	0.0454545
(Republic of Ireland)	0.0151515
(Romania)	0.0378788
(Russia)	0.0530303
(Scotland)	0.0151515
(Slovakia)	0.00757576
(Slovenia)	0.00757576
(Spain)	0.0606061
(Sweden)	0.0378788
(Switzerland)	0.030303
(Turkey)	0.030303
(Ukraine)	0.00757576
(Wales)	0.00757576
(Yugoslavia)	0.030303



```
print(model.get_cpds('Champion'))
```





AveGoalsFor	ChampCount	Host	Matches	Champion(false)	Champion(true)
high	0.0	false	high	0.5	0.5
high	0.0	false	low	1	0
high	0.0	false	mid	0.5	0.5
high	0.0	true	high	0.5	0.5
high	0.0	true	low	0.5	0.5
high	0.0	true	mid	0.5	0.5
high	1.0	false	high	0.5	0.5
high	1.0	false	low	1	0
high	1.0	false	mid	0.5	0.5
high	1.0	true	high	0.5	0.5
high	1.0	true	low	0.5	0.5
high	1.0	true	mid	1	0
high	2.0	false	high	0.5	0.5
high	2.0	false	low	0.5	0.5
high	2.0	false	mid	0.5	0.5
high	2.0	true	high	0.5	0.5
high	2.0	true	low	0.5	0.5
high	2.0	true	mid	0	1
high	3.0	false	high	0.5	0.5
high	3.0	false	low	0.5	0.5
high	3.0	false	mid	0.5	0.5
high	3.0	true	high	0.5	0.5
high	3.0	true	low	0.5	0.5
high	3.0	true	mid	0.5	0.5
mid	0.0	false	high	1	0
mid	0.0	false	low	0.75	0.25
mid	0.0	false	mid	1	0
mid	0.0	true	high	1	0
mid	0.0	true	low	1	0
mid	0.0	true	mid	1	0
mid	1.0	false	high	0.6	0.4
mid	1.0	false	low	0	1
mid	1.0	false	mid	0.727272727	0.272727273
mid	1.0	true	high	0.5	0.5
mid	1.0	true	low	0	1
mid	1.0	true	mid	1	0
mid	2.0	false	high	0	1
mid	2.0	false	low	0.5	0.5
mid	2.0	false	mid	1	0
mid	2.0	true	high	0.5	0.5
mid	2.0	true	low	0.5	0.5
mid	2.0	true	mid	0.5	0.5
mid	3.0	false	high	0	1
mid	3.0	false	low	0.5	0.5
mid	3.0	false	mid	1	0
mid	3.0	true	high	1	0
mid	3.0	true	low	0.5	0.5
mid	3.0	true	mid	0.5	0.5



AveGoalsFor	ChampCount	Host	Matches	Champion(false)	Champion(true)
low	0.0	false	high	0.5	0.5
low	0.0	false	low	1	0
low	0.0	false	mid	1	0
low	0.0	true	high	0.5	0.5
low	0.0	true	low	0.5	0.5
low	0.0	true	mid	1	0
low	1.0	false	high	0.5	0.5
low	1.0	false	low	0.5	0.5
low	1.0	false	mid	1	0
low	1.0	true	high	0.5	0.5
low	1.0	true	low	0.5	0.5
low	1.0	true	mid	0.5	0.5
low	2.0	false	high	0.5	0.5
low	2.0	false	low	0.5	0.5
low	2.0	false	mid	1	0
low	2.0	true	high	0.5	0.5
low	2.0	true	low	0.5	0.5
low	2.0	true	mid	0.5	0.5
low	3.0	false	high	0.5	0.5
low	3.0	false	low	0.5	0.5
low	3.0	false	mid	1	0
low	3.0	true	high	0.5	0.5
low	3.0	true	low	0.5	0.5
low	3.0	true	mid	0.5	0.5

Country	Champion(true)	Champion(False)
Albania	0.0495	0.9505
Austria	0.0495	0.9505
Belgium	0.0417	0.9583
Bulgaria	0.0495	0.9505
CIS	0.0495	0.9505
Croatia	0.0339	0.9661
Czech Republic	0.0401	0.9599
Czechoslovakia	0.1801	0.8199
Denmark	0.1605	0.8395
England	0.0436	0.9564
FR Yugoslavia	0.0339	0.9661
France	0.1827	0.8173
Germany	0.2678	0.7322
Greece	0.0765	0.9235
Hungary	0.0339	0.9661
Iceland	0.0339	0.9661
Italy	0.1737	0.8263
Latvia	0.0495	0.9505
Netherlands	0.2496	0.7504
Northern Ireland	0.0495	0.9505
Norway	0.0495	0.9505
Poland	0.0495	0.9505
Portugal	0.0566	0.9434
Republic of Ireland	0.0495	0.9505
Romania	0.0495	0.9505
Russia	0.0406	0.9594
Scotland	0.0495	0.9505
Slovakia	0.0495	0.9505
Slovenia	0.0339	0.9661
Spain	0.1900	0.8100
Sweden	0.0432	0.9568
Switzerland	0.0495	0.9505
Turkey	0.0456	0.9544
Ukraine	0.0495	0.9505
Wales	0.0339	0.9661
Yugoslavia	0.1497	0.8503

## 4.2 Exercise 6b

```
Test1 = Test
del Test1['HomeMatches']
del Test1['HomeAveGoalsFor']
del Test1['HomeAveGoalsAgainst']
del Test1['AwayMatches']
del Test1['AwayAveGoalsFor']
del Test1['AwayAveGoalsAgainst']
del Test1['Champion']
del Test1['AveGoalsAgainst']
```

```
del Test1['AveGoalsFor']  
Test1
```

	Country	Matches	Host	ChampCount
Year				
1964	Denmark	low	False	0.0
1964	Hungary	low	False	0.0
1964	Russia	low	False	0.0
1964	Spain	low	True	1.0
1972	Belgium	low	True	0.0
1972	Germany	low	False	1.0
1972	Hungary	low	False	0.0
1972	Russia	low	False	0.0
2012	Croatia	mid	False	0.0
2012	Czech Republic	mid	False	0.0
2012	Denmark	mid	False	1.0
2012	England	mid	False	0.0
2012	France	mid	False	3.0
2012	Germany	mid	False	1.0
2012	Greece	mid	False	0.0
2012	Italy	high	False	1.0
2012	Netherlands	mid	False	1.0
2012	Poland	mid	True	0.0
2012	Portugal	mid	False	0.0
2012	Republic of Ireland	mid	False	0.0
2012	Russia	mid	False	0.0
2012	Spain	high	False	2.0
2012	Sweden	mid	False	0.0
2012	Ukraine	mid	True	0.0

```
y = Test1.index.values  
y = np.unique(y)  
for i in y:  
    print(model.predict(Test1.loc[i]))
```

```
Finding Elimination Order: : 100%|██████████| 4/4 [00:00<00:00, 6.02it/s]

AveGoalsFor  Champion
0           low    False
1           mid    False
2           low    False
3           low    False

100%|██████████| 4/4 [00:00<00:00, 800.13it/s]

AveGoalsFor  Champion
0           mid    False
1           mid    True
2           mid    False
3           low    False

100%|██████████| 16/16 [00:00<00:00, 1454.46it/s]

AveGoalsFor  Champion
0           mid    False
1           mid    False
2           low    False
3           low    False
4           mid    False
5           mid    False
6           low    False
7           low    False
8           mid    False
9           low    False
10          mid    False
11          low    False
12          mid    False
13          mid    True
14          low    False
15          low    False
```

### 4.3 Exercise 6c

Actual \ Predict	Champion (P)	Not Champion (N)
Champion (P)	2 (TP)	1 (FN)
Not Champion (N)	21 (FP)	0 (TN)

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} = \frac{2}{2 + 21} \approx 0.08696$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} = \frac{2}{2 + 1} = \frac{2}{3} \approx 0.6667$$

#### 4.4 Exercise 6d

```
matches = pd.read_csv(r"Matches.csv")
cups = pd.read_csv(r"cups.csv")
raw = pd.read_csv(r"RawFullData.csv")
FullData = pd.read_csv(r"FullData.csv")
Train1 = pd.read_csv(r"Train.csv")
Train2 = pd.read_csv(r"Train.csv")
Test = pd.read_csv(r"Test.csv")
MatchesTrain = pd.read_csv(r"MatchesTrain.csv")
MatchesTest = pd.read_csv(r"MatchesTest.csv")

FullData = FullData.set_index('Year')
Train1 = Train1.set_index('Year')
Train2 = Train2.set_index('Year')
Test = Test.set_index('Year')

Static = raw
del Static['Year']
Static=Static.groupby('Country').agg('Matches': ['sum'])
Static.columns = ['MatchesSum']
cupsStatic = cups.groupby(['Champion']).agg('Host': ['size'],)
cupsStatic.columns = ['ChampionTimes']
Static = pd.concat([Static, cupsStatic], axis=1)
Static.ChampionTimes = Static.ChampionTimes.fillna(0)
Static.sort_values(by='MatchesSum',ascending=False)

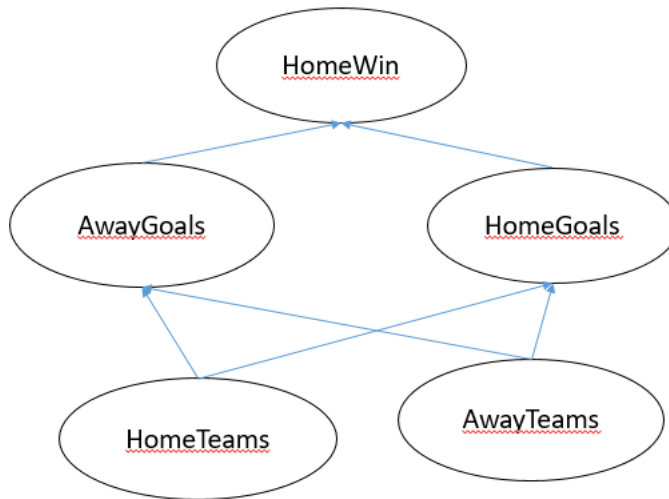
del Train2['HomeMatches']
del Train2['HomeAveGoalsFor']
del Train2['HomeAveGoalsAgainst']
del Train2['AwayMatches']
del Train2['AwayAveGoalsFor']
del Train2['AwayAveGoalsAgainst']
Train2

model1 = BayesianModel([(('Country', 'AveGoalsFor'), ('Country', 'ChampCount'),
('AveGoalsFor', 'Champion'), ('Host', 'Champion'), ('ChampCount', 'Champion'), ('Matches',
'Champion'),('Country', 'AveGoalsAgainst'))])
model1.fit(Train2, estimator = MaximumLikelihoodEstimator)
model1.check_model()
```

	AveGoalsFor	AveGoalsAgainst	Champion
0	low	mid	False
1	mid	mid	False
2	low	low	False
3	low	low	False
100% ██████████  4/4 [00:00<00:00, 1333.54it/s]			
	AveGoalsFor	AveGoalsAgainst	Champion
0	mid	low	False
1	mid	low	True
2	mid	mid	False
3	low	low	False
100% ██████████  16/16 [00:00<00:00, 64.78it/s]			
	AveGoalsFor	AveGoalsAgainst	Champion
0	mid	low	False
1	mid	mid	False
2	low	mid	False
3	low	low	False
4	mid	low	False
5	mid	low	False
6	low	mid	False
7	low	low	False
8	mid	low	False
9	low	low	False
10	mid	low	False
11	low	low	False
12	mid	low	False
13	mid	low	True
14	low	mid	False
15	low	mid	False

% We add the columns AveGoalsAgainst to the Train dataset. But there is no change in Precision and Recall

## 4.5 Exercise 6e



```

model2 = BayesianModel([('HomeTeamName', 'HomeTeamGoals'), ('HomeTeamName', 'AwayTeamGoals'), ('AwayTeamName', 'HomeTeamGoals'),
('AwayTeamName', 'AwayTeamGoals'), ('HomeTeamGoals', 'HomeWin'), ('AwayTeamGoals', 'HomeWin')])
model2.fit(MatchesTrain, estimator = MaximumLikelihoodEstimator)
model2.check_model()

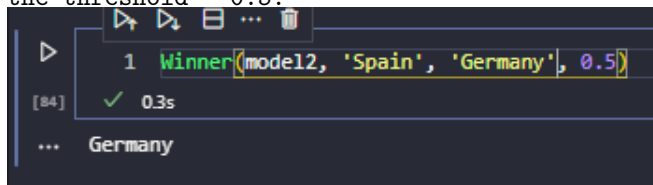
```

```

def Winner(model, Home, Away, threshold):
    d = {'HomeTeamName': [Home], 'AwayTeamName': [Away]}
    df = pd.DataFrame(data=d)
    data = model.predict_probability(df)
    if data.iloc[0]['HomeWin_True'] >= threshold:
        print(Home)
    else:
        print(Away)

```

Winner(model, Home, Away, threshold)  
% For example, we want to predict result of Spain vs Germany match, with model2 and the threshold = 0.5:



```

1 Winner(model2, 'Spain', 'Germany', 0.5)
[84] ✓ 0.3s
... Germany

```

% If we enter the threshold = 0.3, then the result will be different.





```
1 Winner(model2, 'Spain', 'Germany', 0.3)
```

[85] ✓ 0.1s

... Spain

## References

- [1] UEFA Euro Championship data <https://www.kaggle.com/mohammedessam97/uefa-euro-championship>.
- [2] <https://stackoverflow.com/>
- [3] <https://pandas.pydata.org/>
- [4] <https://www.overleaf.com/>