Assignment 1

Question 1: Associate Rules and its Application in Recommendation

Part (a)

According to Bayes rule, $Pr(Y|X) = \frac{Pr(X \cap Y)}{Pr(X)}$.
Therefore we can conclude that $conf(X \rightarrow Y)$ is only relevant to $Pr(X \wedge Y)$ and $Pr(X)$, but not $Pr(Y)$
The main disadvantage of ignoring $Pr(Y)$ is that confidence is lack of comparative power among the whole dataset because the underlying effect of two observed rules are not comparable if they have different support even the confidence are the same.
For example, it is the first time that Alice watchs a scary movie, and then she clicks in another comedy in the recommendation list. The sample size is too small to conclude that "Alice awalys click on a comedy after watching scary movies."
However, by defenition, Lift and Conviction have already incorporate the effect of support by multiplication.

Part(b)

Part(c)

conf Lift

Part(c)

User `align` to create the math environment. and `align` to write the matrix:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \tag{1}$$

If you would like to ignore the number of the matrix, you can use `align*` like that:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Question 2

To write inline mathematics like $x \in \mathbb{Z}$ place your expression inside single $.
To write standalone mathematics like:

$$x \in \mathbb{N}$$

use double $$ or `\[ ... \]`.
Other useful commands:

- For subscript use _. E.g. `S_{a,b}` gives: $S_{a,b}$

- For superscript use `^`. E.g. `S^{a,b}` gives $S^{a,b}$
- To write a set use `\{ ... \}` like this: $\{a, b, c\}$
- `\emptyset \mathcal{U}`: $\emptyset, \mathcal{U}$
- `\cap \cup`: $A \cap B$, $A \cup B$
- `\setminus`: $A \setminus B$
- `\gcd`: gcd
- `\pmod{n}`: $\pmod{n}$
- `\%`: %

Question 3

The `enumerate` package lets you do tailored enumerates:

I First

II Second

III Third

This can be useful if you want to do a simple algorithm:

Step 1. Make a sandwich

Step 2. Eat the sandwich

If you want to break an enumerate with text and then resume, use `\setcounter{enumi}{..}` to start the counter from somewhere specific. For example

Step 0. Buy sandwich ingredients

Question 4

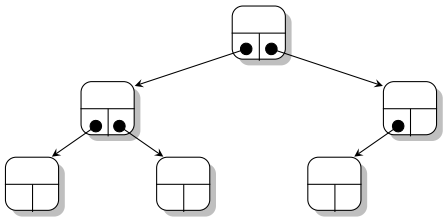
The `listings` package lets you list code verbatim, with some syntax highlighting:

```
\documentclass{article}

\usepackage{listings}

% recursion alert: stack overflow
```

A list of supported languages can be found here.

Question 5: Figures

You can use `\includegraphics` to import figures from many filetypes: jpg, pdf, eps, png, etc.`TikZ` is a very useful and powerful package for drawing pictures directly in LaTeX. The documentation and library of examples are quite extensive, but the results can be pretty:



Here is an example algorithm:

procedure CHECKNUMBERS(A,B)

▷ A and B are two lists of integers

```
    count = 0
    for i = 1, ..., n do
        for j = i ... m do
            if A[i] ≥ B[j] then
                count = count +1
                break
            end if
        end for
    end for
end procedure
```

It will be inserted exactly where it appears in the document. This is not recommended because it is better to group the entire algorithm together and "float" the group to somewhere where it will all fit: this will make things more readable. As an example here is a second, much longer, algorithm (see Algorithm 1).

Observe that the `algorithm` environment also adds nice visual structures around your algorithm.

Part (a)

You should use `\ref` and `\label` to refer to your algorithm in the text so that the reader can find it. Note that if you use `\ref` and `\label` you need to compile your latex document twice: the first time grabs all the labels (and puts them into the .aux file) and the second time it can insert the numbers into the file. Otherwise your references look like this: **??**.

You can use `\ref` and `\label` to refer to other parts of your document, including other floating objects (e.g. `figure` for graphics, `table` for tables).

Question 6

There is almost always a LaTeX package for your typesetting needs. Most packages are well documented and many of the popular ones have copious amounts of examples of their capabilities. Google is your friend.

Algorithm 1 A longer algorithm

```
procedure LongerCheckNumbers(A,B)
                                          ▷ A and B are two lists of integers
    count = 0
    for i = 1,...,n do
        for j = i...m do
            if A[i] ≥ B[j] then
                count = count +1
                break
            end if
        end for
    end for
    count = 0
    for i = 1,...,n do
        for j = i...m do
            if A[i] ≥ B[j] then
                count = count +1
                break
            end if
        end for
    end for
    count = 0
    for i = 1,...,n do
        for j = i...m do
            if A[i] ≥ B[j] then
                count = count +1
                break
            end if
        end for
    end for
    count = 0
    for i = 1,...,n do
        for j = i...m do
            if A[i] ≥ B[j] then
                count = count +1
                break
            end if
        end for
    end for
end procedure
```