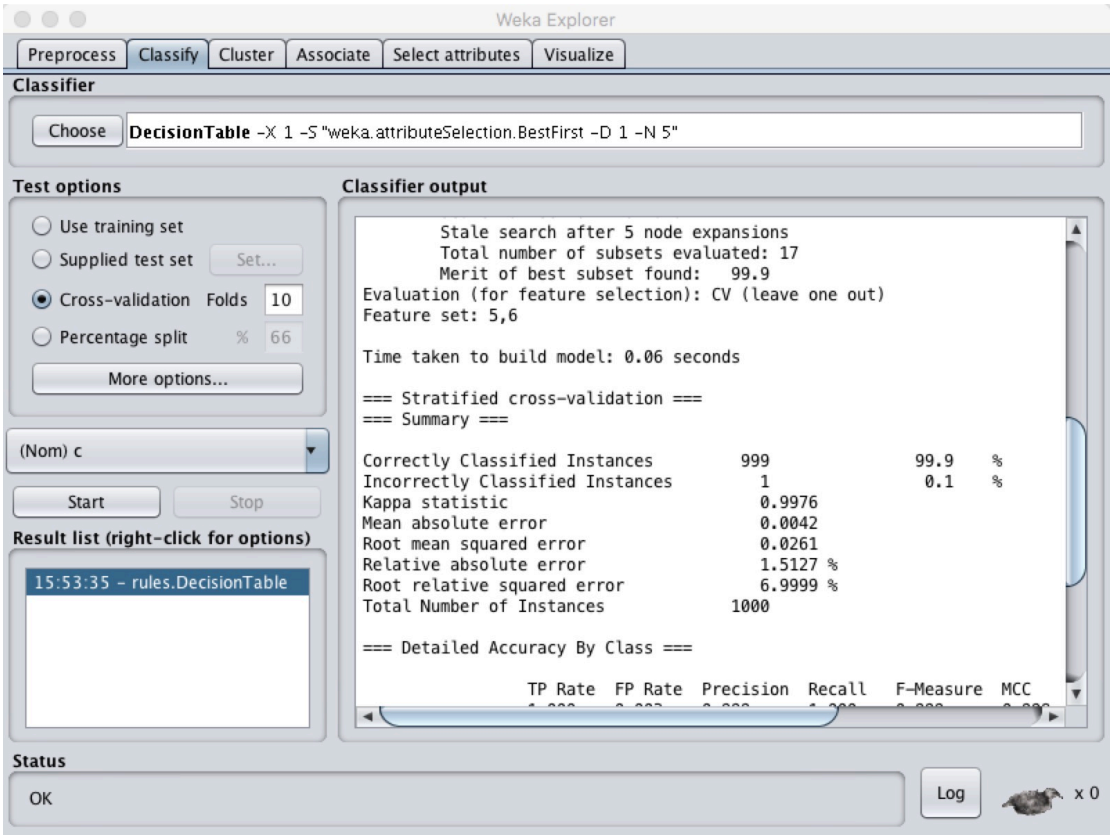
HW4 Boyu Xu

Part 1

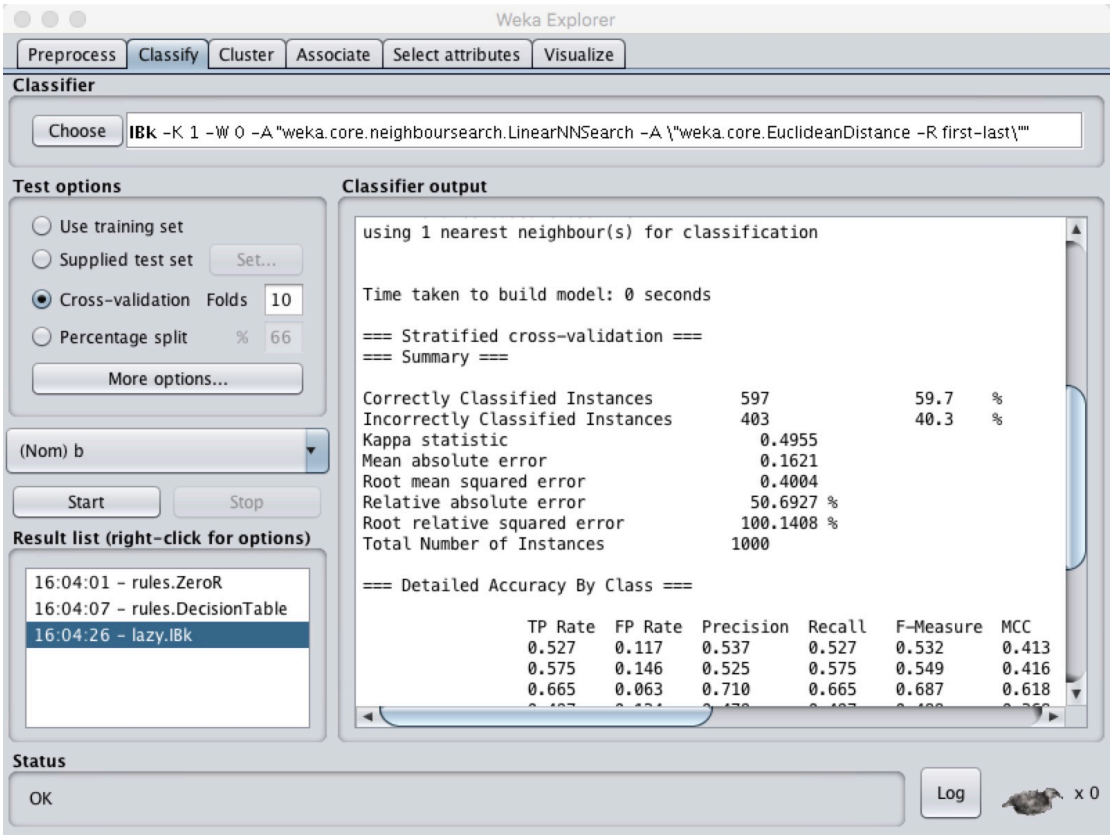
1.
I generate the dataset in python with the code:

```
import csv
import random
import math
csvfile = open('/Users/apple/Desktop/hw4-21.csv','w')
writer = csv.writer(csvfile)
n = 0
head = []
while (n < 10):
    head.append(n)
    n = n + 1
writer.writerow(head)
while(m < 1000):
    column0 = random.uniform(0, 100)
    column1 = random.uniform(0, 100)
    column2 = random.uniform(0, 100)
    column3 = random.uniform(0, 100)
    column4 = random.uniform(0,100)
    column5 = random.uniform(0,100)
    if(column5 < 20):
     column6 = 'a'
    elif(column5 >= 20 and column5 < 40):
     column6 = 'b'
    elif(column5 >= 40 and column5 < 60):
     column6 = 'c'
    elif(column5 > 60 and column5 <= 80):
     column6 = 'd'
    else:
     column6 = 'e'
    writer.writerow([column0,column1,column2,column3,column4,column5,column6]
    m = m + 1
```

The data file is q1.csv
The 10-fold cross-validation performance of decision tree (J48) in WEKA is:



The 10-fold cross-validation performance of nearest neighbor (IBk) in WEKA is:



|J48 accuracy – IBk accuracy| = 40.2% > 30%.

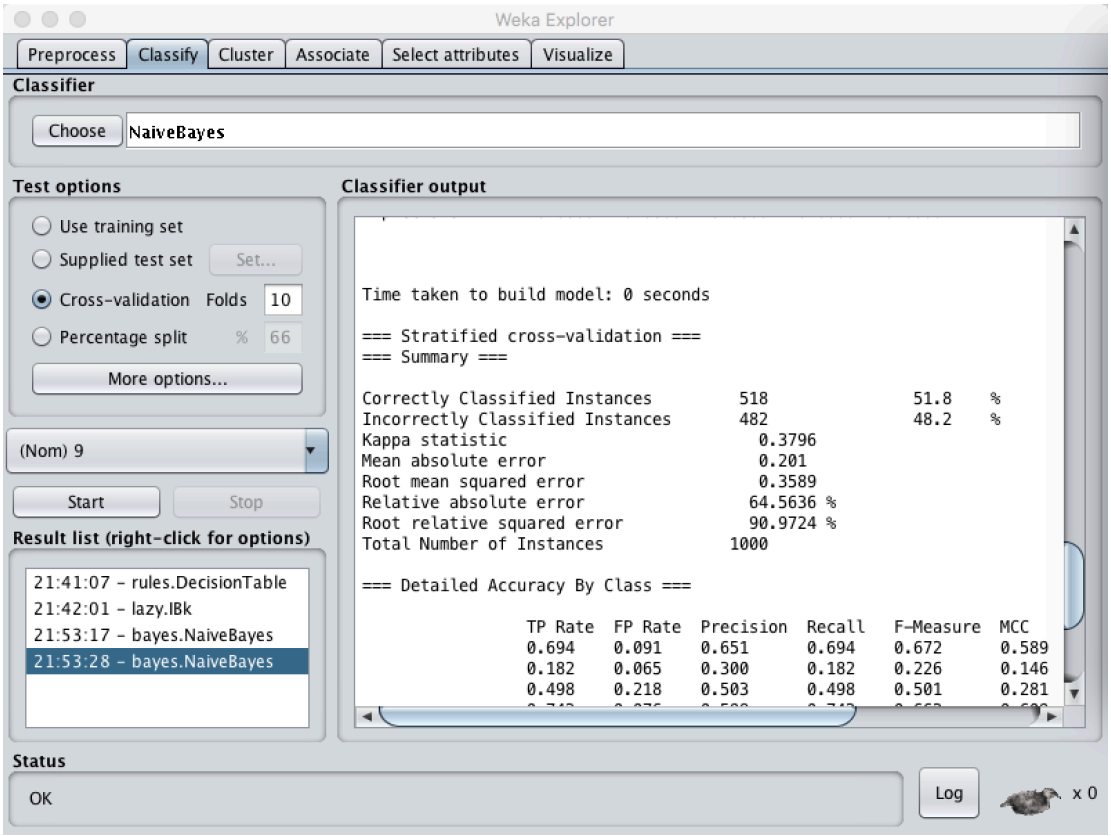
2.

I generate the dataset in python with the code as:

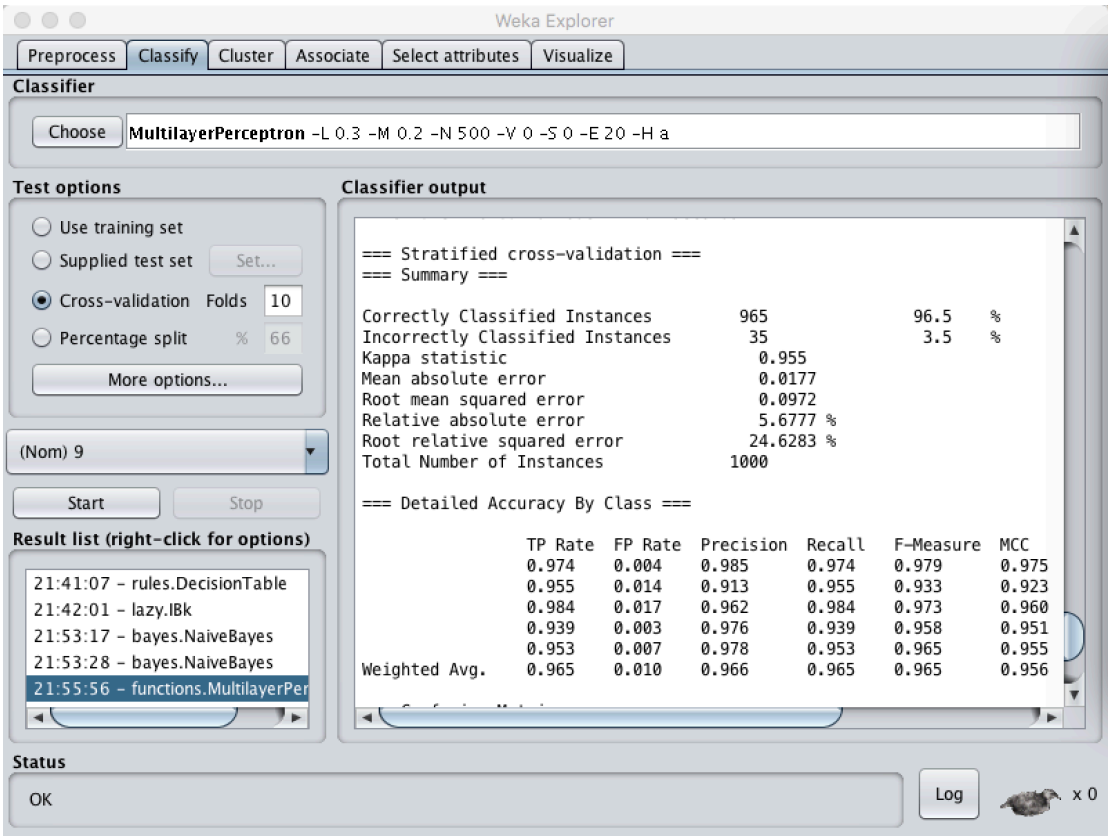
```
import csv
import random
import math
csvfile = open('/Users/apple/Desktop/hw4-21.csv','w')
writer = csv.writer(csvfile)
n = 0
head = []
while (n < 10):
    head.append(n)
    n = n + 1
writer.writerow(head)
while(m < 1000):
    column0 = random.uniform(0, 100)
    column1 = random.uniform(0, 100)
    column2 = column1 + random.uniform(0, 100)
    column3 = column0 + random.uniform(0, 100)
    column4 = 2*column2+random.uniform(0,100)
    column5 = column3 -random.uniform(0,100)
    column6 = column3 + 0.2 * random.uniform(0, 100)
    column7 = column5 + random.uniform(0, 100)
    column8 = column6 + column4 - random.uniform(0, 100)
    if(column8-column7 > 300):
     column9='a'
    elif(column8-column7 > 220 and ):
     column9='b'
    elif(column8-column7 > 150):
     column9='c'
    elif(column8-column7 > 100):
     column9='d'
    else:
     column9='e'
writer.writerow([column0,column1,column2,column3,column4,column5,column6,col
umn7,column8,column9])
    m = m + 1
```

The data file is q2.csv.

The 10-fold cross-validation performance of multi-layer perceptrons in WEKA is:



The 10-fold cross-validation performance of Naive Bayes classifiers in WEKA is:



|MLP accuracy – NB accuracy| = 44.7% > 30%.

3.
In total, there are 9 cases in HoursOfSleep, 3 cases in Studied, 3 cases in LikesMaterial, 3 cases in Exams.
a.
There are 243 cases in total, to achieve the aim to describe probability of the entire space, independent parameters in total:
9*3*3*3 - 1 = ***242***
b.
For condition distribution, prior equals to 9 * 3 * 3 = 81, for each prior event we need 3 – 1 = 2 independent parameters. So in total, independent parameters shall be:
9 * 3 * 3 * (3 - 1) = ***162***
c.
Under the assumption of Naïve Bayes, the total number of independent parameters shall be:
3 * 8 + 3 * 2 + 3 * 2 + 2 = ***38***

Part 2

4.
The function cosine_similarity() is in the ps4.py file.

5.
pixel:
cosine similarity between 'mj1' and 'mj2': 0.370861946454
cosine similarity between 'mj2' and 'cat': 0.619743990598
cosine similarity between 'mj1' and 'cat': 0.473088523579

VGG:
cosine similarity between 'mj1' and 'mj2': 20.960011026178
cosine similarity between 'mj2' and 'cat': 0.141834212136
cosine similarity between 'mj1' and 'cat': 0.152535112667

The pair of "mj1" and "mj2" in VGG is the most similar pair, while the pair of "mj2" and "cat" shares the most similarity in Pixel

6.
The problem is that using pixel method leads us to a wrong answer. We find that picture "mj2" and "cat" share great similarity by using pixel method. We can find that the content in "mj1" and "mj2" are basically the same, except Michael Jordan's position in picture. While the content in "cat" is just a cat, which is totally different

from “mj2”. The reason is as follows:

The pixel representation can only compare the pixels on the same position for pictures, attributes being valued are RGN values of each pixel, which means it can just measure the color information and its distribution. Specifically, it would lose the local image vectors. Simply comparing information in the same position for different picture means that it cannot capture the translation and transformation of objects in images, that’s the reason why using pixel, it fails to detect the translation of Michael Jordan.

Convolution neural network is helpful to solve this problem. By using convolution, the CNN can measure the local features of image content with different resolution and spatial relation. It is able to divide the whole image into different parts and each describes the inner pattern of neighboring pixels, and some groups of pixels can well summarize high level semantic features in the images. CNN captures the ability of shift invariant and pattern recognition. In this example, using VGG from CNN, Michael Jordan can be recognized in both “mj1” and “mj2” pictures, so they share a great similarity.

Therefore, using VGG from CNN layers, the player can actually be recognized from both mj1 and mj2, so the similarity using VGG representation is very large; while the cat image doesn’t share the same content with mj images, so the similarity is very low.

7.

As required, the results are recorded in vgg.txt and pixel.txt. The code for this problem is in ps4.py.

The result comes from using VGG representation is much better than the result comes from using pixel representation. The analysis makes sense for most images. The captions by using VGG can successfully describe the image’s content, capture its essence such as “man with a cell phone”, “bathroom with shower and sink”, etc.

However, pixel representation renders a rather dissatisfied result, only a small amount of captions is relevant to the actually content in the test images. For example, test sample uses “A man and woman are holding a surfboard and two dogs are standing on the surfboard.” to describe an image with two dogs on the surfboard which is kept by a man and a woman. The recognized result is “A bench on a beach near the water”. So using pixel representation can only compare some low level features, such as the main

color, brightness, background etc. between images and it's hard to perform well in this caption task.

8.

The error example using vgg representation is shown below:

The true caption is "A MAN IN A SUIT LICKING A TIE THAT IS HANGING ON A RACK"

The learned caption is "A man stands and talks on his cell phone."

In this learning case, the relevant thing is “A man stands”, and in the test picture, he acts strange and licks his tie. It is totally different as the learning result says “talks in the phone”. In the training image, there is no such man that licking his tie, so the most similar feature by using VGG is “man” and also the way he holds his tie kind of like holding a phone which leads the caption as “A man stands and talks in the phone”

The error using pixel is shown below:

The true caption is “A man and woman are holding a surfboard and two dogs are standing on the surfboard.”
The learned caption is “a bench on a beach near the water.”

The caption learned from training set using pixel representation basically shares no relevant with the test image, maybe the only similar part is “water”. The learning result totally ignores the man, the women and 2 dogs, the surfboard, instead, it recognizes it as a “bench”. The possible reason for the mistake is that learning by pixel focuses on the color or background, it only captures the background “water”, discarding the core elements of the picture. Pixel is sensitive to color distribution but cannot capture the high level semantic features.