EECS 349 Problem Set 4
Weihan Chu (wcm350)

1. My code for this problem:

```
import csv
import random

csvfile= open('/Users/weihanchu/Desktop/hw4-1.csv', 'w')
writer=csv.writer(csvfile)
n=1
head=[]
while(n<9):
        head.append(n)
        n=n+1
writer.writerow(head)

m=1
while(m < 1001):
   column1 = 100*random.random()
   column2 = 100*random.random()
   column3 = 100*random.random()
   column4 = random.uniform(0,100)
   column5 = random.uniform(0,100)
   column6 = random.uniform(0,100)
   column7 = 5*random.random()

   if(column7 < 1):
        column8='a1'
   elif(column7 < 2):
        column8='a2'
   elif(column7 < 3):
        column8='a3'
   elif(column7 < 4):
        column8='a4'
   else:
        column8='a5'

writer.writerow([column1,column2,column3,column4,column5,column6,column7,col
umn8])
```

```
    m = m + 1

csvfile.close()
```

The result is:
a.For the decision tree:

=== Summary ===

Correctly Classified Instances	996	99.6 %
Incorrectly Classified Instances	4	0.4 %
Kappa statistic	0.995	
Mean absolute error	0.0016	
Root mean squared error	0.04	
Relative absolute error	0.5005 %	
Root relative squared error	10.0046 %	
Total Number of Instances	1000	

b. for the KNN
=== Summary ===

Correctly Classified Instances	568	56.8 %
Incorrectly Classified Instances	432	43.2 %
Kappa statistic	0.4597	
Mean absolute error	0.1736	
Root mean squared error	0.4145	
Relative absolute error	54.3042 %	
Root relative squared error	103.6846 %	
Total Number of Instances	1000	

So the difference of accuracy is 99.6%-56.8%=42.8%

The basic idea of this problem is that:
I create a dataset which has 8 attributes, and only the 7th attribute influence the classification. other numeric attributes has nothing to do with the classification. Since KNN will calculate the distance through all the attributes, no matter whether this attribute matters classification or not, So KNN will have a low accuracy for this data set.

2.My code for this problem:

```
csvfile= open('/Users/weihanchu/Desktop/hw4-2.csv', 'w')
writer=csv.writer(csvfile)
n=1
head=[]
while(n<9):
        head.append(n)
        n=n+1
writer.writerow(head)

m=1
while(m < 1001):
   column1 = 100*random.random()
   column2 = 100*random.random()
   column3 = column2+column1
   column4 = 2*column2+random.uniform(0,100)
   column5 = column3-0.5*random.uniform(0,100)
   column6 = 2*column4-column5
   column7 = column3+2*column5

   if(column7-column6 > 100):
        column8='a1'
   elif(column7-column6 > 30):
        column8='a2'
   elif(column7-column6 > -30):
        column8='a3'
   elif(column7-column6 > -100):
        column8='a4'
   else:
        column8='a5'

writer.writerow([column1,column2,column3,column4,column5,column6,column7,col
umn8])
   m = m + 1

csvfile.close()
```

The result is:
a.For the multi-layer perceptrons
=== Summary ===

Correctly Classified Instances	974	97.4 %
Incorrectly Classified Instances	26	2.6 %
Kappa statistic	0.9668	
Mean absolute error	0.0162	
Root mean squared error	0.091	
Relative absolute error	5.1487 %	
Root relative squared error	22.981 %	
Total Number of Instances	1000	

b.For the Naive Bayes
=== Summary ===

Correctly Classified Instances	565	56.5 %
Incorrectly Classified Instances	435	43.5 %
Kappa statistic	0.4417	
Mean absolute error	0.1966	
Root mean squared error	0.3391	
Relative absolute error	62.6567 %	
Root relative squared error	85.6256 %	
Total Number of Instances	1000	

So the difference of accuracy is 97.4%-56.5%=40.9%

The basic idea of this problem is that:
The naive bases pretend that all attributes are independent. So I made the attributes not independent. So the accuracy of the naive bayes will be very low.

3.

a. Genetic Algorithms

This algorithm is good for solving problems like 8 Queen, which can be easily represented as strings. This problem is not easily solved by other two algorithms because it can not be represented as differentiable function. And it may get a local minimum by hill climbing. So the genetic algorithm is good to solve genetic algorithms.

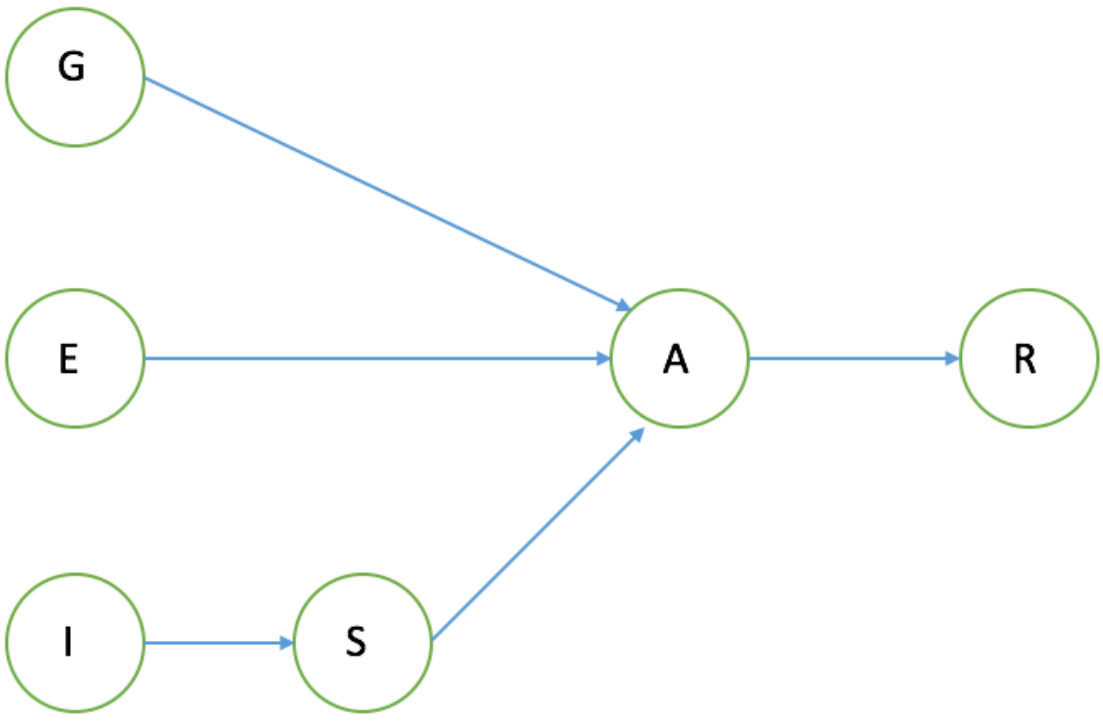
b. Gradient Descent

This algorithm is good for solving problems like find min value of a convex differentiable math function. Since this problem is hard to be represented by strings , is can not be solved by genetic algorithm. Also, hill climbing is easy to get stuck at a local maximum.

C.Hill-Climbing

This algorithm is good for solving problems like 8-puzzle, since we can define the heuristic as Manhattan distance, then it is easy to get the answer. It can not be solve by genetic algorithm because it can bot be defined as strings. It is also to be solved by gradient descent because we can not get the differentiable function.

4.a.



b. 1for G, 1 for E, 1for I, 2 for S, 8 for A, 2 for R, so the total=1+1+1+2+8+2=15

c. Since it is full joint, the total number is 2^6-1=63

d. Yes, E and G are independent. Because it is a “head to head” condition in D-seperation. Since we don't know A, E and G are independent.

e. No, E and G are not independent, because information can be transmitted through A.

5.P(C)=P(C|A^B)*P(A^B)+P(C|A^ ¬B)*P(A^¬B)+P(C|¬A^B)*P(¬A^B)+
P(C|¬A^¬B)*P(¬A^¬B)
=1*0.7*0.5+1*0.7*0.5+0.7*0.3*0.5+0.3*0.3*0.5
=0.35+0.35+0.105+0.045
=0.85