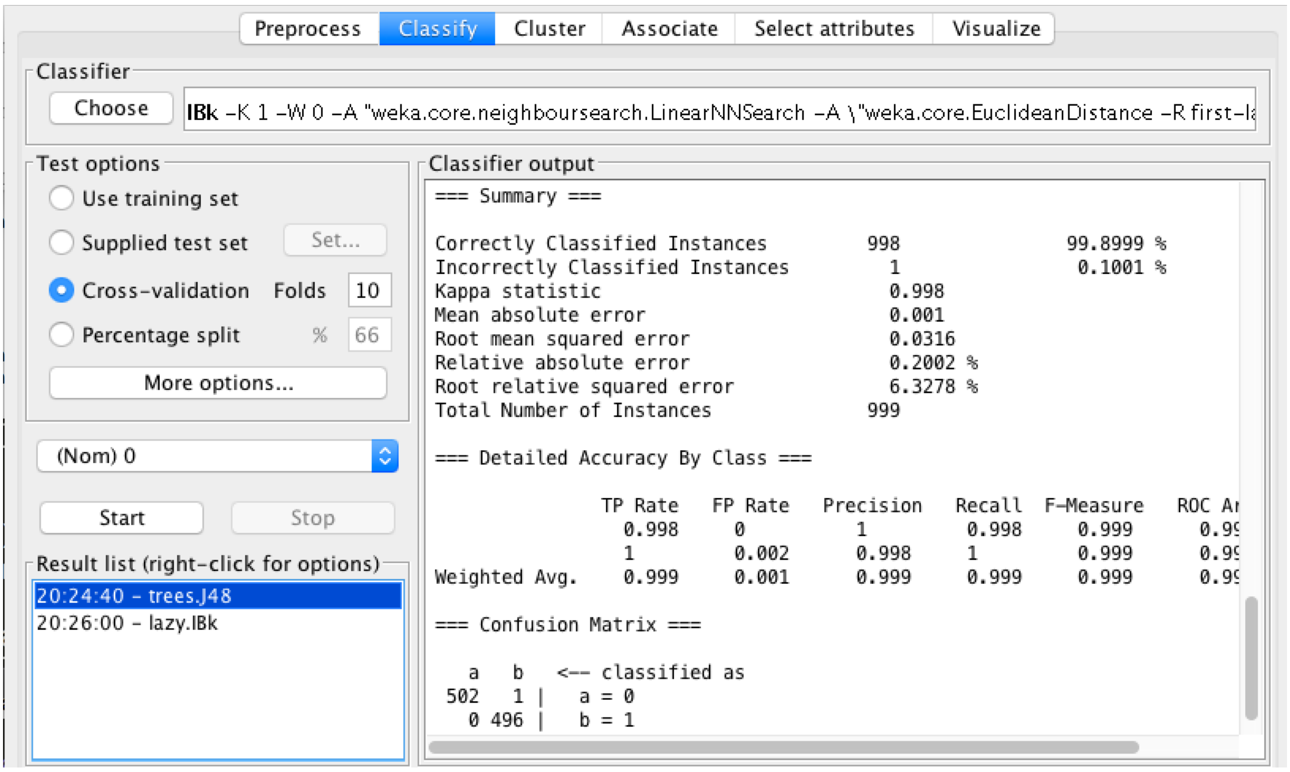
Problem Set 4

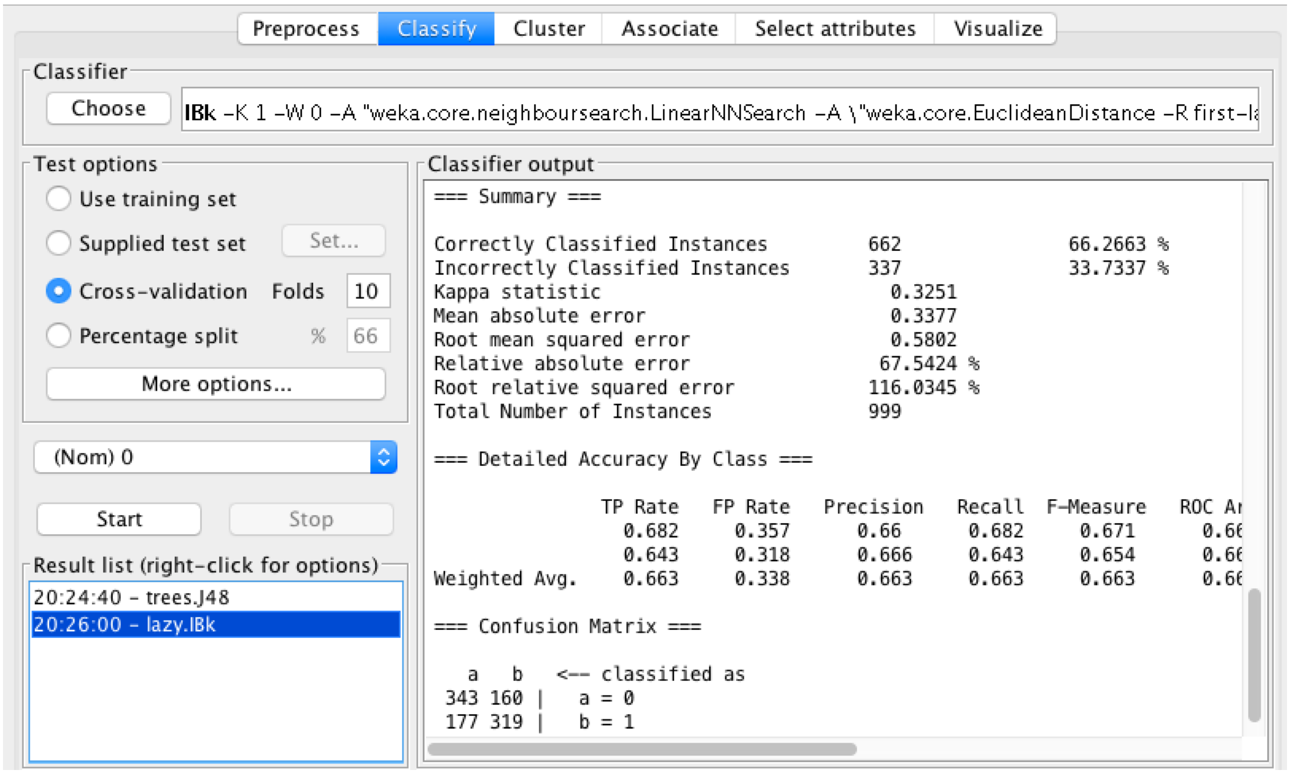
1. **Create dataset using code below. Let the results only depend on the first attribute.**

```
import random
import csv

def create_dataset():
    attr_size = 50
    data_size = 1000
    train_set = [ [ 0 for j in range(attr_size) ] for i in range(data_size) ]
    for i in range(data_size):
        for j in range(attr_size):
            train_set[i][j] = random.uniform(j,j+1)
        if train_set[i][0] < 0.5:
            train_set[i].append(0)
        else:
            train_set[i].append(1)

    with open("./train.csv", "wb") as f:
        writer = csv.writer(f)
        writer.writerows(train_set)
```

Accuracy of J48:



Accuracy of IBK:

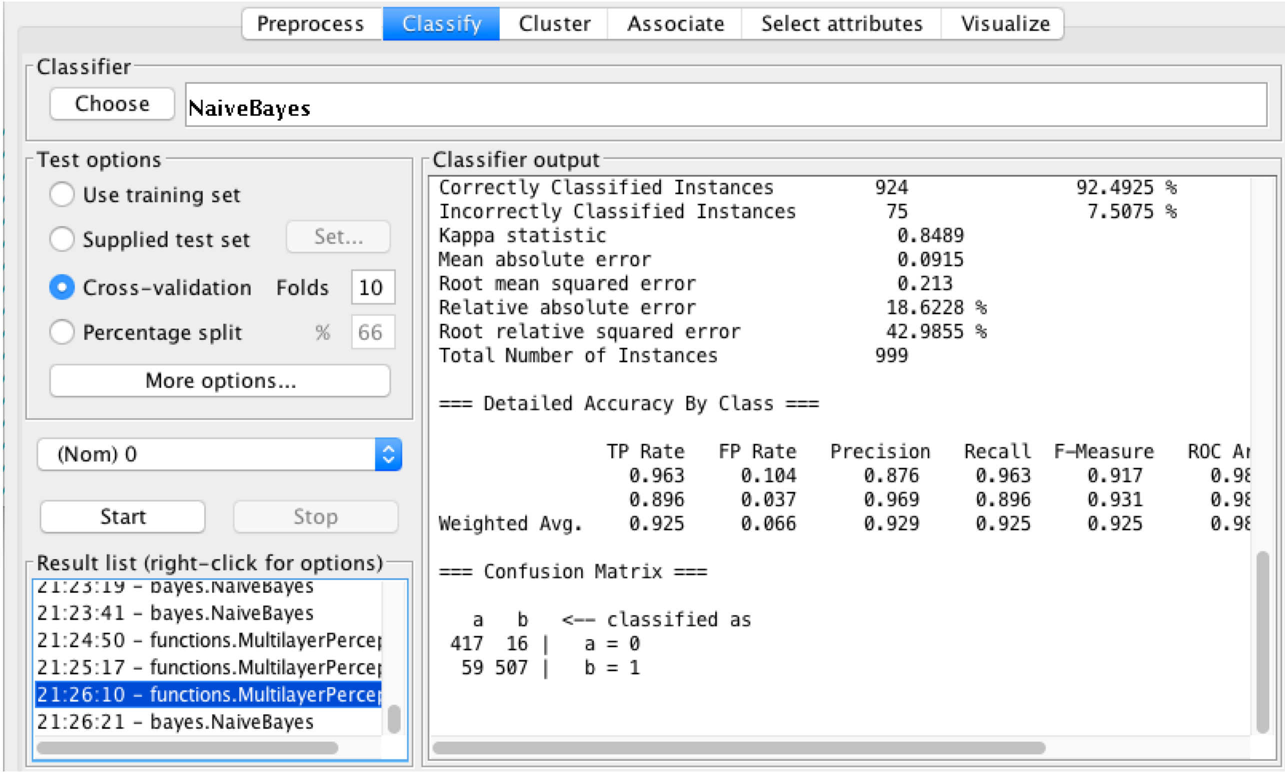


The difference in accuracy is 99.89% - 66.26% = 33.63%

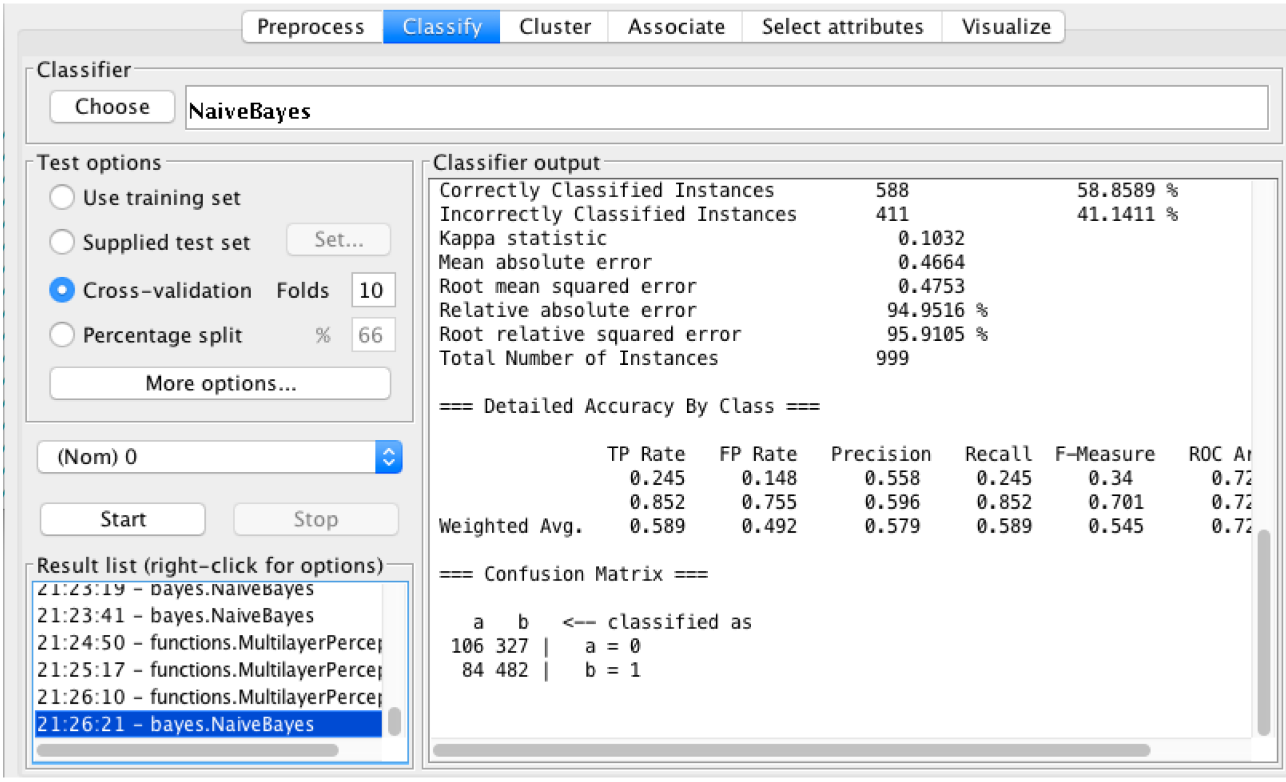
2. **Create dataset using code below. Let the results depend on the Covariance of attributes.**

```
import random
import csv
import numpy as np

def create_dataset():
    attr_size = 2
    data_size = 1000
    train_set = [ [ 0 for j in range(attr_size) ] for i in range(data_size) ]
    for i in range(data_size):
        for j in range(attr_size):
            train_set[i][j] = random.uniform(0,30)
        if np.cov([train_set[i][0], train_set[i][1]]) > 50:
            train_set[i].append(0)
        else:
            train_set[i].append(1)

    with open("./2-train.csv", "wb") as f:
        writer = csv.writer(f)
        writer.writerows(train_set)
```

Multilayer Perceptron:



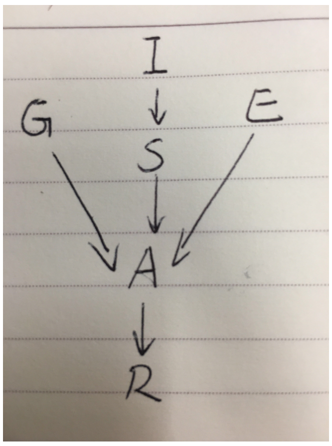
Naïve Bayes:



The difference in accuracy is 92.49% - 58.85% = 33.64%

3. **Genetic Algorithm, Gradient Decent & Hill Climbing**
    1. Problems that have local minimum or maximum, e.g. N-Queue Problem, is more easily solved by Genetic Algorithm, but is not well-suit for Hill Climbing and Gradient Decent. The reason that N-Queue problem is well-suit for Genetic Algorithm is that successor state is generated by combining two best parent states, which will get the best solution. The reason that N-Queue problem is not well-suit for Hill Climbing and Gradient Decent is that N-Queue has local minimum state.
    2. Problems like finding minimum value in a high dimensional function is more easily solved by Gradient Decent. Because we can easily find the direction that the function is descending most quickly using Gradient Decent, which helps us get the solution. However, Genetic Algorithm will be very slow in these problems and Hill Climbing is also time consuming because of the high dimension.
    3. Problem like finding minimum value in a discrete function is more easily solved by Hill Climbing. Because we can easily find the maximum value searching follow the trend of the function. However, Genetic Algorithm will search states more randomly which makes it inefficient. Gradient Decent will perform bad because discrete function has no gradient.

4. **Solutions:**
    1. As shown in figure below.



    2. 1 for P(I), P(E) and P(G); 2 for P(S|I); 8 for P(A|S,E,G) and 2 for P(R|A). The minimum number of probabilities we need is 15.
    3. There are six binary variables without independencies, the number of parameters we need is 2^6 -1 = 63.
    4. Yes, E is independent of G if we are not given the value of any other variables. We know that P(E,G,A) = P(E)*P(G)*(A|E,G). If we are not given the value of A, we will add them up with all possible value of A. Then we will get P(E,G) = P(E)*P(G), which means E and G are independent.
    5. No. Once we know the value of A, E is no longer independent of G. Now we have P(E,G|A) = P(E,G,A)/P(A) = P(E)*P(G)*P(A|E,G)/P(A), which is not equal to P(E)*P(G). Thus E is not independent of G given A.

5. $P(C) = P(C|A,B)*P(A)*P(B) + P(C|\neg A,B)*P(\neg A)*P(B) + P(C|A,\neg B)*P(A)*P(\neg B) + P(C|\neg A,\neg B)*P(\neg A)*P(\neg B) = 0.35 + 0.35 + 0.105 + 0.045 = 0.85$.