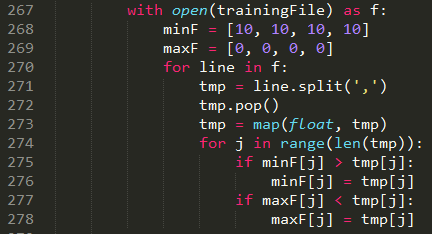
**HW 6 classification**

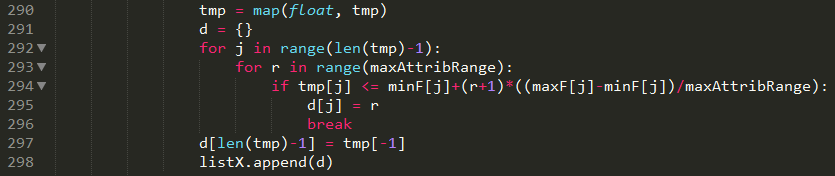
**ส่วน Decision tree**

**1. กระบวนการ pre-process**

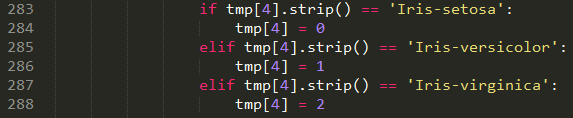
จะวนลูปข้อมูลทั้งหมดเพื่อหาค่า min-max ของ Input ที่เป็น predictive attributes ของแต่ละ attribute ก่อน 1 รอบ ดังโค๊ดข้างล่าง

 จากนั้นจะวนลูปอีกรอบเพื่อปรับค่า predictive attributes ให้เป็นช่วงๆตามที่กำหนดจาก parameter -r จาก command เช่น

python "hw classification Decision tree.py" -t iris.data -r 3

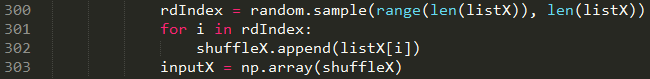
จาก command นี้จะใช้ค่า min max ที่ได้มาปรับค่าทั้ง 4 attribute เป็น 3 ช่วง เริ่มจาก 0 ถึง 2 ที่ต้องแบ่งเป็นช่วงๆเพราะว่าข้อมูลเป็นตัวเลขทศนิยม จึงต้องแบ่งเป็นช่วงเพื่อจะได้มาทำเป็น tree ได้ ดังโค๊ดข้างล่าง

ในส่วนของ class attribute จะเปลี่ยนจากตัวหนังสือให้เป็นตัวเลขเพื่อความเร็วในตอนเปรียบเทียบ ดังโค๊ดข้างล่าง

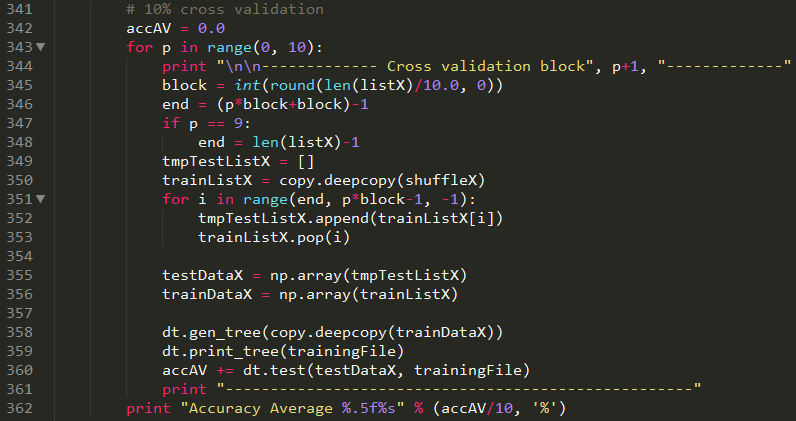


**2.ขั้นตอนการออกแบบการทดลองการ แบ่ง train และ test**

ทำการ shuffle ข้อมูลก่อนนำไปทำ 10 fold cross validation ดังโค๊ดข้างล่าง



จากนั้นแบ่งข้อมูลทีละ 10% เพื่อนำไป test และข้อมูล 90% ที่เหลือนำไป train เพื่อสร้าง tree จนครบ 10 fold โดยจะแสดง accuracy ของแต่ละ fold และจะแสดง accuracy เฉลี่ยทั้ง 10 fold ในตอนท้าย ดังโค๊ดข้างล่าง



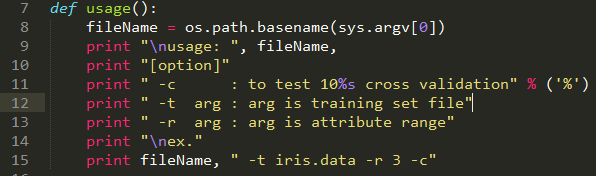
**3. โครงสร้างของโปรแกรมเบื้องต้น**

โปรแกรมเขียนด้วยภาษา python โดยรับ parameter ผ่าน command line รายละเอียดดังนี้

parameter -c เพื่อทำ 10% cross validation

parameter -t เพื่อเลือก training set file

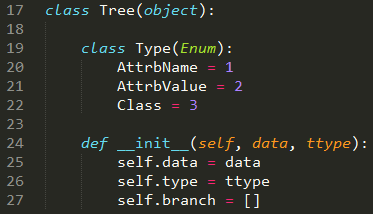
parameter -r เพื่อกำหนดว่าจะแบ่งช่วงค่าของ attribute เป็นกี่ช่วง



ตัวอย่างการเรียกโปรแกรมผ่าน command

python "hw classification Decision tree.py" -t iris.data -r 3

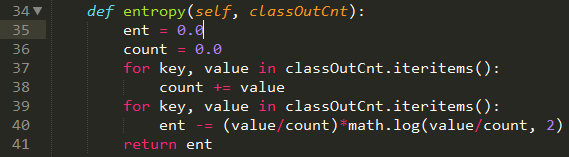
Node ของ tree นั้นจะมีอยู่ 3 แบบ คือ node ที่เป็นชื่อ attribute, node ที่เป็นค่าช่วงของ attribute นั้น ว่าเป็นช่วงที่เท่าไหร่เพื่อใช้เป็นเงื่อนไขในตอนทดสอบ และ node ที่บอกว่าเป็น class อะไร ดังโค๊ดข้างล่าง



ใน class DecisionTree จะแบ่งเป็นฟังก์ชันดังนี้

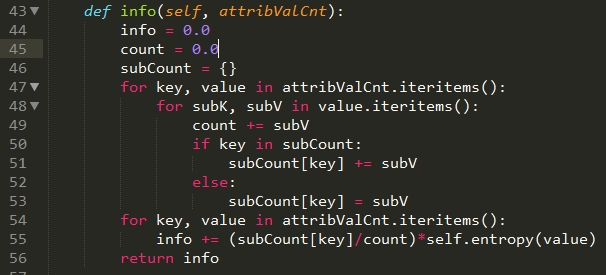
**ฟังก์ชัน entropy(self, classOutCnt)** ไว้หาค่า entropy รวมของแต่ input ตามสูตร





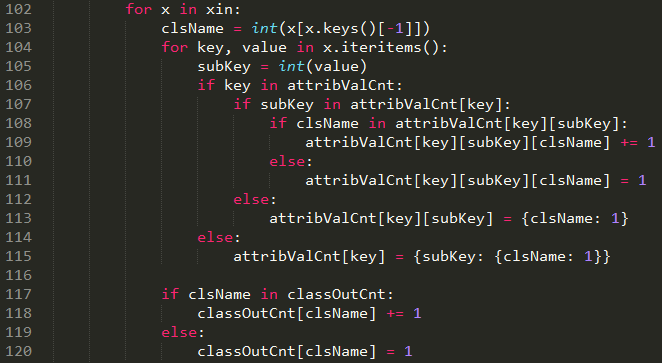
**ฟังก์ชัน info(self, attribValCnt)** ไว้หาค่า information ของแต่ละ attribute ตามสูตร



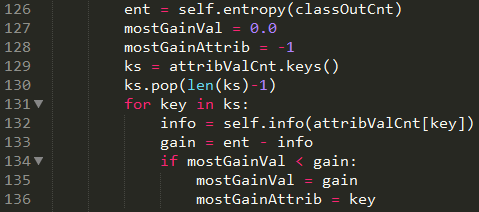


**ฟังก์ชัน gen\_tree(self, xin) และฟังก์ชัน gen\_sub\_tree(self, xin)** ไว้สร้าง tree โดยจะวนลูปอ่านข้อมูลทั้งหมด และนับข้อมูลใน class attribute ว่าแต่ละ class มีอยู่กี่ sample โดยเก็บค่าในตัวแปร classOutCnt

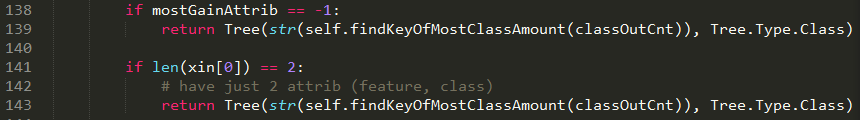
และนับค่าว่า attribute ที่เท่าไหร่, ช่วง attribute ที่เท่าไหร่ ตอบเป็น class อะไรจำนวนเท่าไหร่ โดยเก็บค่าในตัวแปร attribValCnt โดยทั้งค่า classOutCnt และค่า attribValCnt จะเอาไว้ใช้ในตอนหาค่า entropy และ information

หลังจากได้ค่า classOutCnt แล้ว ถ้าทุก sample ตอบแค่ class เดียว จะ return node นั้นเป็นคำตอบ class เลย

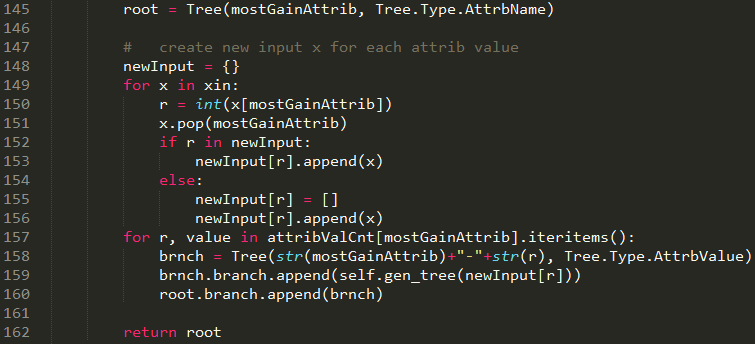
แต่ถ้าไม่ใช่ จะหาค่า entropy รวมและค่า information และ information gain ของแต่ละ attribute และเก็บค่า attribute ที่มี information gain มากสุดไว้



ถ้า information gain ทุกตัวเท่ากันหรือเหลือ predictive attributes ตัวเดียว จะ return node นั้นเป็น class ที่มีจำนวนมากสุดในชุด sample นั้นเลย

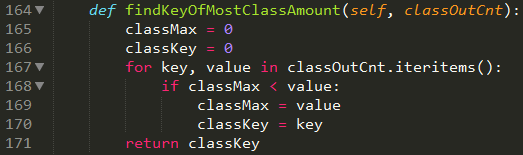


จากนั้นจะสร้าง tree จาก attribute ที่มี information gain มากสุด และต่อ branch ย่อยจาก value ทั้งหมดที่เป็นไปได้ของ attribute นั้น โดย recursive จากฟังก์ชันนี้ โดยจะส่งค่า input sample ที่เป็นแค่ช่วงของแต่ละ value ของ attribute นั้นที่เก็บในตัวแปร newInput เช่น attribute A มีค่าที่เป็นไปได้คือ 1, 2, 3, 4 input ที่จะส่งเข้าฟังก์ชันเพื่อหา branch ย่อยที่ 1 คือ sample ที่ attribute A มีค่าเป็น 1 เท่านั้น input branch ย่อยที่ 2 ก็คือ sample ที่ attribute A มีค่าเป็น 2 เท่านั้น ทำจนครบทุก branch ซึ่งแต่ละ branch ก็จะ recursive แบบนี้ไปเรื่อยๆ สุดท้ายก็จะได้ tree เต็มออกมา ที่จะใช้ในการ test



**ฟังก์ชัน findKeyOfMostClassAmount(self, classOutCnt)**

ไว้หาค่า attribute ที่มีจำนวนมากสุดในชุด sample นั้น



**ฟังก์ช้น test(self, listX, trainingFile)**

ไว้ใช้ในการ test ข้อมูล โดยใช้ tree ที่ได้มาจากฟังก์ชัน gen\_tree(self, xin)

**4.ผลการทดลอง และโครงสร้าง tree ที่ดีที่สุดในการทดลอง**

**การทดลองที่ 1 ทดลองกับข้อมูลจากตัวอย่างใน slide**

โดย preprocess ข้อมูลเป็น

1,3,1,1,no

1,3,1,2,no

2,3,1,1,yes

3,2,1,1,yes

3,1,2,1,yes

3,1,2,2,no

2,1,2,2,yes

1,2,1,1,no

1,1,2,1,yes

3,2,2,1,yes

1,2,2,2,yes

2,2,1,2,yes

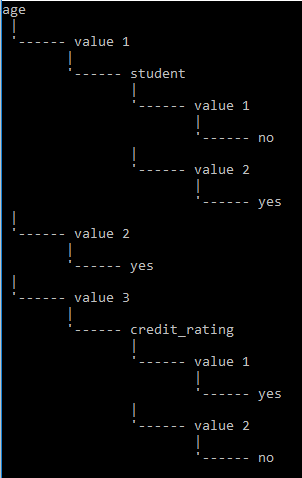
2,3,2,1,yes

3,2,1,2,no

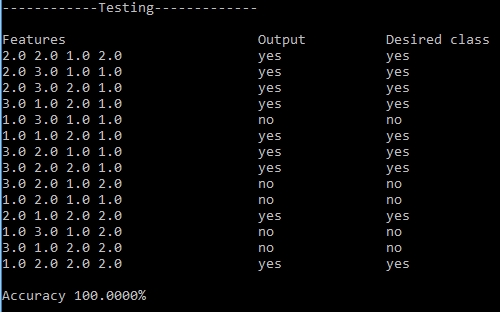
รันจาก command

python "hw classification Decision tree.py" -t test.data

ได้ tree ผลลัพธ์ดังนี้



ซึ่งเป็น tree ที่ตรงกับใน slide และนำข้อมูลทีละตัวไป test กับ tree ที่ได้ ผลออกมาคือ tree สามารถตอบได้ถูกทุกตัว

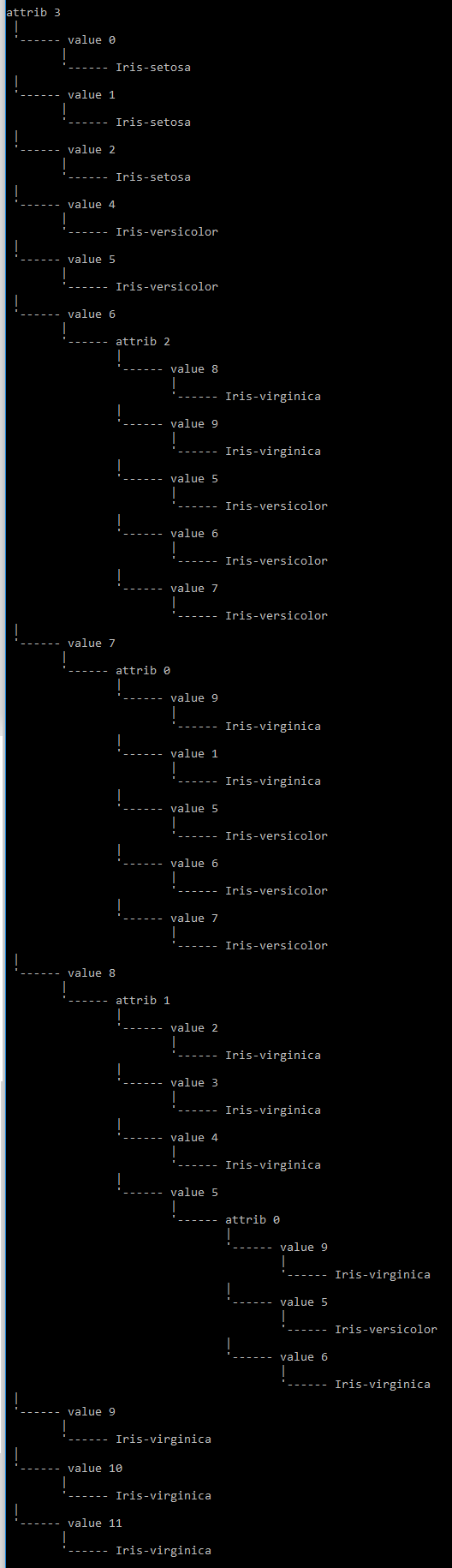


**การทดลองที่ 2 ทดลองกับ iris.data ที่ได้รับมอบหมายโดยไม่ได้ทำ 10 fold cross validation**

รันจาก command

python "hw classification Decision tree.py" -t iris.data -r 12

ได้ tree ผลลัพธ์ดังนี้

****

และนำข้อมูลทีละตัวไป test กับ tree ที่ได้ ผลออกมาคือ tree สามารถตอบได้ถูกทุกตัว

------------Testing-------------

Features Output Desired class

6 3 7 8 Iris-virginica Iris-virginica

2 6 1 0 Iris-setosa Iris-setosa

6 1 6 5 Iris-versicolor Iris-versicolor

9 4 9 7 Iris-virginica Iris-virginica

3 1 5 4 Iris-versicolor Iris-versicolor

2 5 0 0 Iris-setosa Iris-setosa

7 4 8 7 Iris-versicolor Iris-versicolor

4 4 6 5 Iris-versicolor Iris-versicolor

6 3 9 10 Iris-virginica Iris-virginica

6 0 7 6 Iris-versicolor Iris-versicolor

7 2 9 8 Iris-virginica Iris-virginica

6 3 7 8 Iris-virginica Iris-virginica

2 6 1 1 Iris-setosa Iris-setosa

3 8 1 0 Iris-setosa Iris-setosa

0 5 1 0 Iris-setosa Iris-setosa

1 4 0 0 Iris-setosa Iris-setosa

4 11 1 1 Iris-setosa Iris-setosa

2 7 0 0 Iris-setosa Iris-setosa

1 6 1 0 Iris-setosa Iris-setosa

4 3 6 4 Iris-versicolor Iris-versicolor

8 4 9 10 Iris-virginica Iris-virginica

7 4 9 10 Iris-virginica Iris-virginica

6 6 10 11 Iris-virginica Iris-virginica

4 4 6 5 Iris-versicolor Iris-versicolor

2 1 4 4 Iris-versicolor Iris-versicolor

2 10 1 0 Iris-setosa Iris-setosa

6 6 8 10 Iris-virginica Iris-virginica

4 4 7 6 Iris-versicolor Iris-versicolor

8 5 9 10 Iris-virginica Iris-virginica

1 2 7 7 Iris-virginica Iris-virginica

10 3 10 8 Iris-virginica Iris-virginica

3 8 1 0 Iris-setosa Iris-setosa

1 5 1 0 Iris-setosa Iris-setosa

3 2 6 5 Iris-versicolor Iris-versicolor

2 6 1 0 Iris-setosa Iris-setosa

2 7 1 0 Iris-setosa Iris-setosa

8 3 7 6 Iris-versicolor Iris-versicolor

5 4 7 6 Iris-versicolor Iris-versicolor

0 5 0 0 Iris-setosa Iris-setosa

6 5 7 6 Iris-versicolor Iris-versicolor

7 5 9 11 Iris-virginica Iris-virginica

0 6 0 0 Iris-setosa Iris-setosa

9 4 9 10 Iris-virginica Iris-virginica

5 0 6 4 Iris-versicolor Iris-versicolor

2 8 1 1 Iris-setosa Iris-setosa

8 5 7 6 Iris-versicolor Iris-versicolor

6 4 9 8 Iris-virginica Iris-virginica

5 4 7 6 Iris-versicolor Iris-versicolor

4 4 6 5 Iris-versicolor Iris-versicolor

6 2 7 6 Iris-versicolor Iris-versicolor

4 3 6 5 Iris-versicolor Iris-versicolor

5 3 6 5 Iris-versicolor Iris-versicolor

5 4 8 8 Iris-virginica Iris-virginica

4 3 8 8 Iris-virginica Iris-virginica

2 8 1 0 Iris-setosa Iris-setosa

0 4 0 0 Iris-setosa Iris-setosa

7 6 9 10 Iris-virginica Iris-virginica

11 8 11 10 Iris-virginica Iris-virginica

4 2 6 5 Iris-versicolor Iris-versicolor

4 3 5 5 Iris-versicolor Iris-versicolor

8 5 8 10 Iris-virginica Iris-virginica

7 4 8 10 Iris-virginica Iris-virginica

6 5 9 8 Iris-virginica Iris-virginica

4 3 6 5 Iris-versicolor Iris-versicolor

11 3 11 9 Iris-virginica Iris-virginica

2 7 0 0 Iris-setosa Iris-setosa

7 4 7 5 Iris-versicolor Iris-versicolor

1 5 1 0 Iris-setosa Iris-setosa

0 1 0 0 Iris-setosa Iris-setosa

1 5 1 0 Iris-setosa Iris-setosa

4 2 8 9 Iris-virginica Iris-virginica

3 6 1 0 Iris-setosa Iris-setosa

7 6 9 11 Iris-virginica Iris-virginica

8 5 9 10 Iris-virginica Iris-virginica

5 4 7 6 Iris-versicolor Iris-versicolor

4 2 5 4 Iris-versicolor Iris-versicolor

7 5 6 6 Iris-versicolor Iris-versicolor

4 2 5 5 Iris-versicolor Iris-versicolor

6 5 8 10 Iris-virginica Iris-virginica

2 3 5 6 Iris-versicolor Iris-versicolor

5 4 7 8 Iris-virginica Iris-virginica

5 2 9 6 Iris-virginica Iris-virginica

4 3 8 8 Iris-virginica Iris-virginica

6 3 8 8 Iris-virginica Iris-virginica

0 4 0 0 Iris-setosa Iris-setosa

9 5 10 8 Iris-virginica Iris-virginica

5 6 7 7 Iris-versicolor Iris-versicolor

8 5 7 6 Iris-versicolor Iris-versicolor

1 5 0 0 Iris-setosa Iris-setosa

11 8 10 9 Iris-virginica Iris-virginica

4 3 8 11 Iris-virginica Iris-virginica

0 5 0 0 Iris-setosa Iris-setosa

6 3 9 10 Iris-virginica Iris-virginica

7 4 6 6 Iris-versicolor Iris-versicolor

8 5 8 10 Iris-virginica Iris-virginica

2 6 0 0 Iris-setosa Iris-setosa

3 1 5 5 Iris-versicolor Iris-versicolor

7 3 7 6 Iris-versicolor Iris-versicolor

1 5 1 0 Iris-setosa Iris-setosa

1 4 0 0 Iris-setosa Iris-setosa

7 5 8 9 Iris-virginica Iris-virginica

1 1 4 4 Iris-versicolor Iris-versicolor

6 4 6 5 Iris-versicolor Iris-versicolor

4 8 1 0 Iris-setosa Iris-setosa

3 2 6 5 Iris-versicolor Iris-versicolor

2 7 0 0 Iris-setosa Iris-setosa

2 7 1 2 Iris-setosa Iris-setosa

3 4 7 6 Iris-versicolor Iris-versicolor

3 7 0 0 Iris-setosa Iris-setosa

6 2 8 8 Iris-virginica Iris-virginica

4 4 5 5 Iris-versicolor Iris-versicolor

2 6 1 1 Iris-setosa Iris-setosa

2 8 1 1 Iris-setosa Iris-setosa

3 9 0 1 Iris-setosa Iris-setosa

4 3 7 9 Iris-virginica Iris-virginica

11 4 10 10 Iris-virginica Iris-virginica

5 0 8 6 Iris-virginica Iris-virginica

7 5 7 6 Iris-versicolor Iris-versicolor

6 4 6 5 Iris-versicolor Iris-versicolor

10 4 11 10 Iris-virginica Iris-virginica

1 6 1 0 Iris-setosa Iris-setosa

6 3 8 6 Iris-virginica Iris-virginica

2 6 0 0 Iris-setosa Iris-setosa

1 5 1 0 Iris-setosa Iris-setosa

2 4 1 0 Iris-setosa Iris-setosa

9 4 10 8 Iris-virginica Iris-virginica

2 0 5 4 Iris-versicolor Iris-versicolor

6 6 7 7 Iris-versicolor Iris-versicolor

5 3 7 5 Iris-versicolor Iris-versicolor

5 3 8 7 Iris-versicolor Iris-versicolor

6 6 9 11 Iris-virginica Iris-virginica

3 10 0 0 Iris-setosa Iris-setosa

5 4 6 6 Iris-versicolor Iris-versicolor

3 1 6 5 Iris-versicolor Iris-versicolor

3 9 1 1 Iris-setosa Iris-setosa

5 5 7 8 Iris-versicolor Iris-versicolor

9 7 10 11 Iris-virginica Iris-virginica

0 4 0 0 Iris-setosa Iris-setosa

2 2 4 5 Iris-versicolor Iris-versicolor

1 4 0 0 Iris-setosa Iris-setosa

7 4 9 8 Iris-virginica Iris-virginica

4 3 7 5 Iris-versicolor Iris-versicolor

2 8 1 0 Iris-setosa Iris-setosa

11 2 11 10 Iris-virginica Iris-virginica

5 4 7 8 Iris-virginica Iris-virginica

3 6 1 1 Iris-setosa Iris-setosa

0 7 0 0 Iris-setosa Iris-setosa

7 4 8 9 Iris-virginica Iris-virginica

4 9 0 0 Iris-setosa Iris-setosa

2 7 0 0 Iris-setosa Iris-setosa

Accuracy 100.0000%

**การทดลองที่ 3 ทดลองกับ iris.data ที่ได้รับมอบหมายโดยทำ 10 fold cross validation**

รันจาก command

python "hw classification Decision tree.py" -t iris.data -r 12

ได้ความถูกต้องเฉลี่ยทั้ง 10 fold ดังนี้



โดยได้ tree ของ fold ที่ดีที่สุดดังนี้

attrib 3

|

'------ value 0

|

'------ Iris-setosa

|

'------ value 1

|

'------ Iris-setosa

|

'------ value 2

|

'------ Iris-setosa

|

'------ value 4

|

'------ Iris-versicolor

|

'------ value 5

|

'------ Iris-versicolor

|

'------ value 6

|

'------ attrib 2

|

'------ value 8

|

'------ Iris-virginica

|

'------ value 9

|

'------ Iris-virginica

|

'------ value 5

|

'------ Iris-versicolor

|

'------ value 6

|

'------ Iris-versicolor

|

'------ value 7

|

'------ Iris-versicolor

|

'------ value 7

|

'------ attrib 0

|

'------ value 9

|

'------ Iris-virginica

|

'------ value 1

|

'------ Iris-virginica

|

'------ value 5

|

'------ Iris-versicolor

|

'------ value 6

|

'------ Iris-versicolor

|

'------ value 7

|

'------ Iris-versicolor

|

'------ value 8

|

'------ attrib 1

|

'------ value 2

|

'------ Iris-virginica

|

'------ value 3

|

'------ Iris-virginica

|

'------ value 4

|

'------ Iris-virginica

|

'------ value 5

|

'------ attrib 0

|

'------ value 9

|

'------ Iris-virginica

|

'------ value 5

|

'------ Iris-versicolor

|

'------ value 6

|

'------ Iris-virginica

|

'------ value 9

|

'------ Iris-virginica

|

'------ value 10

|

'------ Iris-virginica

|

'------ value 11

|

'------ Iris-virginica

โดยได้ผลลัพธ์ fold ที่ดีที่สุดดังนี้



**ส่วน neural network**

**1. กระบวนการ pre-process**

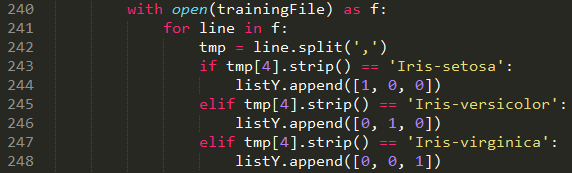
predictive attributes จะใช้ค่าเดิมเลยเพราะเป็นตัวเลข สามารถนำไป input ของ neural ได้เลย ส่วน class attribute จะเปลี่ยนจากตัวหนังสือให้เป็นตัวเลข 3 ตัว โดย

class Iris-setosa จะเปลี่ยนเป็น 1, 0, 0

class Iris- versicolor จะเปลี่ยนเป็น 0, 1, 0

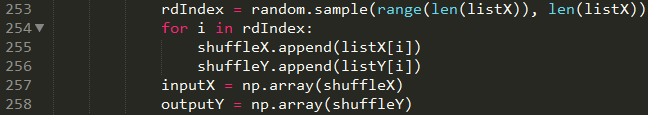
class Iris- virginica จะเปลี่ยนเป็น 0, 0, 1

ที่ต้องเปลี่ยนเป็นเลขเพราะว่า output สุดท้ายที่ได้จาก activation function sigmoid จะมีค่า 0-1 และจะต้องนำไปหา error ต่อไป



**2.ขั้นตอนการออกแบบการทดลองการ แบ่ง train และ test**

ทำการ shuffle ข้อมูลก่อนนำไปทำ 10 fold cross validation ดังโค๊ดข้างล่าง



จากนั้นแบ่งข้อมูลทีละ 10% เพื่อนำไป test และข้อมูล 90% ที่เหลือนำไป train เพื่อสร้าง tree จนครบ 10 fold โดยจะแสดง error, accuracy ของแต่ละ fold และจะแสดง error, accuracy เฉลี่ยทั้ง 10 fold ในตอนท้าย ดังโค๊ดข้างล่าง



**3. โครงสร้างของโปรแกรมเบื้องต้น**

โปรแกรมเขียนด้วยภาษา python โดยรับ parameter ผ่าน command line รายละเอียดดังนี้

parameter -N arg : arg คือ ชื่อโครงสร้าง Neural network เช่น “2-4-1”

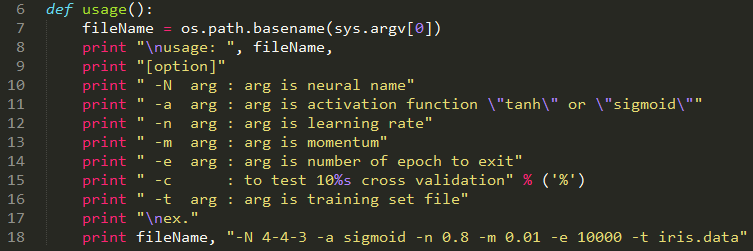
parameter -n arg : arg คือ learning rate

parameter -m arg : arg คือ momentum

parameter -e arg : arg คือ จำนวน epoch

parameter -c : ทำ 10% cross validation

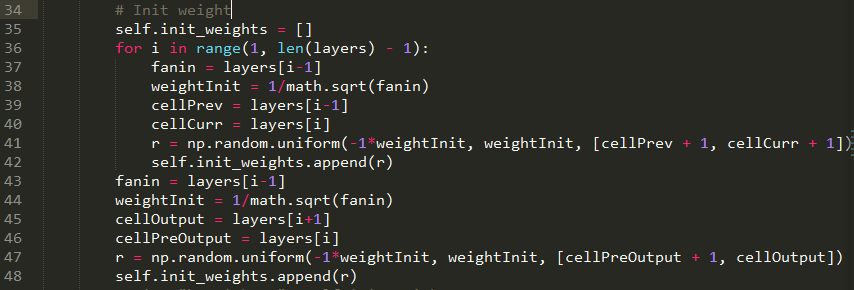
parameter -t arg : arg คือ training set file



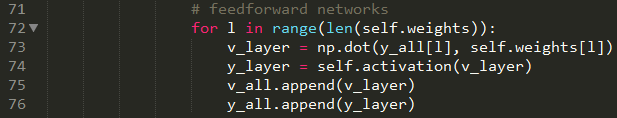
ตัวอย่างการเรียกโปรแกรมผ่าน command

python "hw classification NN.py" -N 4-4-3 -a sigmoid -n 0.8 -m 0.01 -e 100 -t iris.data

เริ่มต้นการทำงาน โปรแกรมจะอ่าน input และ desire-output จากไฟล์มาเก็บใน array จากนั้นนำมา shuffle ก่อนนำเข้า neural network เมื่อเริ่มเข้า neural network จะทำการ random init weight ซึ่งค่าอยู่ระหว่าง  และ  (fanin คือจำนวนเส้น weight ที่เข้า node)



จากนั้นเข้าสู่กระบวนการ train โดยเริ่มจากการทำ feedforward networks โดย output แต่ละ layer ได้จากการ dot product ระหว่าง matrix output ของ layer ก่อนหน้า กับ matrix ของ weight ทุก weight ที่เข้า layer นั้น

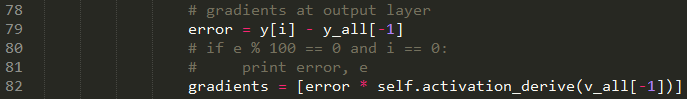


จากนั้น ทำ back propagation โดยเริ่มจาก หา error จากสูตร



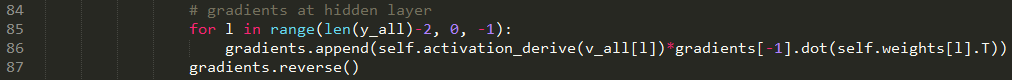
เพื่อนำค่า error ไปหาค่า gradients ที่ output layer จากสูตร





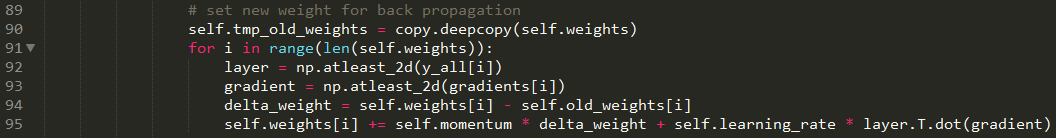
จากนั้น หา gradients ใน hidden layer จากสูตร





เพื่อนำ gradients ที่ได้ของทุก node นำไปปรับ weight จากสูตร





จากนั้นกลับไปเริ่มต้นใหม่ให้ครบรอบจำนวน epoch ที่รับเข้ามา เพื่อปรับ weight ให้ดีขึ้นเรื่อยๆ

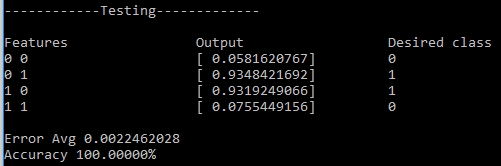
**4.ผลการทดลอง ค่า weight และ bias รวมถึง learning rate ที่ดีที่สุดในการทดลอง**

**การทดลองที่ 1 ทดลองกับปัญหา XOR เพื่อทดสอบความถูกต้องของโปรแกรม**

รันจาก command

python "hw classification NN.py" -N 2-4-1 -a sigmoid -n 0.8 -m 0.01 -e 1000

ได้ผลลัพธ์การทำงานดังนี้

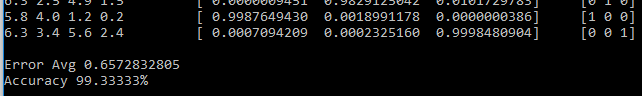


**การทดลองที่ 2 ทดลองกับ iris.data ที่ได้รับมอบหมายโดยไม่ได้ทำ 10 fold cross validation**

รันจาก command

python "hw classification NN.py" -N 4-4-3 -a sigmoid -n 0.8 -m 0.01 -e 15000 -t iris.data

ได้ error และ accuracy ดังนี้

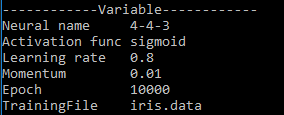


**การทดลองที่ 3 ทดลองกับ iris.data ที่ได้รับมอบหมายโดยทำ 10 fold cross validation**

รันจาก command

python "hw classification NN.py" -N 4-4-3 -a sigmoid -n 0.8 -m 0.01 -e 10000 -t iris.data -c

ตัวแปรต่างๆในการคำนวณดังนี้



ได้ error และ accuracy เฉลี่ยทั้ง 10 fold ดังนี้



โดยได้ weight ของ fold ที่ดีที่สุดดังนี้

weight node 1 layer 1 ไป layer 2

[ 7.37662916e-01 -3.64856362e+01 -1.49287012e+01 -1.08960829e+01 -3.66639018e-02]

weight node 2 layer 1 ไป layer 2

[ 2.99269800e+00 -1.95633280e+01 3.63879270e+00 -8.27856284e+00 2.18265866e+01]

weight node 3 layer 1 ไป layer 2

[ -4.77599212e+00 5.80115246e+01 2.05985264e+01 1.17912165e+01 -3.82919637e-01]

weight node 4 layer 1 ไป layer 2

[ -1.94474364e+00 4.61780392e+01 1.12036767e+01 1.50925606e+01 -5.32952131e+01]

weight node bias ไป layer 2

[ 3.05040288e-01 -7.51541392e+01 -4.80577140e+01 -9.76863637e+00 3.39576399e+01]

weight node 1 layer 2 ไป layer 3

[ -5.94351238 -6.64147599 6.72871249]

weight node 2 layer 2 ไป layer 3

[ -9.95233645 -5.70263968 5.69915129]

weight node 3 layer 2 ไป layer 3

[ -5.40223432 -3.45453736 3.46491247]

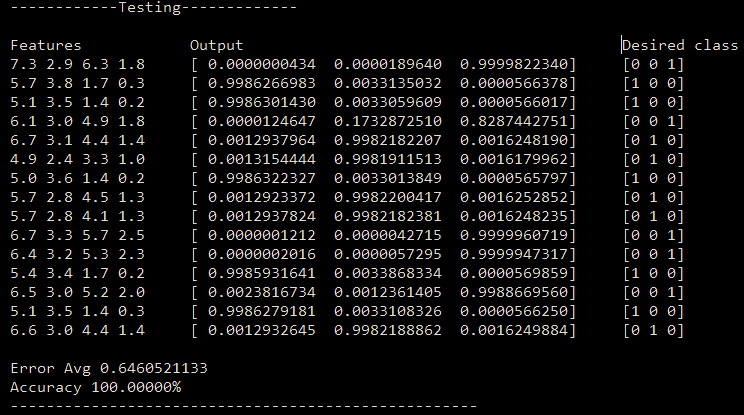
weight node 4 layer 2 ไป layer 3

[ -6.6502021 6.3295615 -6.42039729]

weight node bias ไป layer 3

[ 13.24945572 -12.04516505 -3.36093966]

error และ accuracy ของ fold ที่ดีที่สุดดังนี้



**สรุปและวิเคราะห์ผลการทดลอง**

จากการทดลอง classify โดยใช้ Decision tree และ Neural Network นั้น ผลความถูกต้องมีความใกล้เคียงกันมาก

โดยถ้าไม่ทำ 10 fold cross validation

Decision tree นั้น ถ้าแบ่งข้อมูลเป็น 9 ช่วง (-r 9) จะตอบถูกต้อง 99.3333% ใช้เวลาประมาณ 1 วินาที Neural Network นั้น ถ้า train ถึง 10000 epoch จะตอบถูกต้อง 99.3333% เช่นกันแต่จะใช้เวลาประมาณ 5 นาที แต่ถ้า train แค่ 100 epoch จะตอบถูกต้อง ประมาณ 91-98% (ขึ้นอยู่กับ initial weight ที่ random มา) ซึ่งจะใช้เวลาช้ากว่า Decision tree นิดหน่อย

ถ้าทำ 10 fold cross validation ด้วย

Decision tree ถ้าแบ่งข้อมูลเป็น 12 ช่วง (-r 12) จะตอบถูกต้องประมาณ 90-95% ใช้เวลาประมาณ 3 วินาที Neural Network นั้น ถ้า train ถึง 10000 epoch จะตอบถูกต้องประมาณ 95.33% แต่ต้องใช้เวลาเกือบ 1 ชั่วโมง แต่ถ้า train แค่ 100 epoch จะตอบถูกต้อง ประมาณ 91-95% โดยใช้เวลาประมาณ 20 วินาที

ดังนั้นถ้านำไปใช้งานจริง Decision tree อาจจะเหมาะสมกว่าเรื่องเวลาเพราะใช้เวลาในการสร้าง model เร็วกว่ามาก แต่ข้อมูลที่นำมา train ต้องเพียงพอและครอบคลุมทุกกรณี ไม่งั้น tree จะมีบางค่าของ attribute ที่หายไป เช่น ปกติ attribute A มีค่าที่เป็นไปได้คือ 1, 2, 3, 4 แต่ข้อมูลที่ train มีน้อยไปหรือไม่ครอบคลุม อาจจะทำให้บาง node ของ attribute A มี branch ไม่ครบทั้ง 4 branch ทำให้ไม่สามารถให้คำตอบของบางข้อมูลที่นำมา test ได้ แต่ neural network จะทนทานกับกรณีแบบนี้มากกว่า

**โค๊ดและข้อมูลของ homework ดูได้ที่**

https://github.com/porpeeranut/Data\_Mining\_HW\_Classification