รายงาน

Credit Approval Data

จัดทำโดย

นายอิทธิพร แก้วอำไพ

รหัสนักศึกษา 590610681

รายงานนี้เป็นส่วนหนึ่งของวิชา

Data Mining for CPE

261448

Title: Credit Approval

Data types: Multivariate

Instances: 690

Attribute Types: Categorical, Integer, Real

Attributes: 15 + 1(class attribute)

Attribute Information:

A1 – b, a.

A2 – continuous.

A3 – continuous.

A4 – u, y, l, t.

A5 – g, p, gg.

A6 – c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.

A7 – v, h, bb, j, n, z, dd, ff,o

A8 – continuous.

A9 – t, f.

A10 – t, f.

A11 – continuous.

A12 – t, f.

A13 – g, p, s.

A14 – continuous.

A15 – continuous.

A16 – (+, -) (class attribute)

Missing Attribute Values:

37 cases (5%) have one or more missing values. The missing values from particular attributes are:

A1: 12

A2: 12

A4: 6

A5: 6

A6: 9

A7: 9

A14: 13

Class Distribution:

+: 307 (44.5%)

-: 383 (55.5%)

Source: Submitted by [quinlan@cs.su.oz.au](mailto:quinlan@cs.su.oz.au)

**ขั้นตอนการทำ pre processing**

* จัดการ row ที่มี missing values ตั้งแต่ 20% ขึ้นไปโดยการตัดทั้ง row เนื่องจากมี missing values มากเกินไป โดยจะมีดังนี้ ข้อมูลที่ 207(5), 271(5), 331(5), 457(5), 480(3), 593(5), 602(3), 623(5)
* จัดการเติม missing values ของ A1 โดยการเติม highest frequency value(b) เข้าไปในทุกที่ที่เป็น missing values(A1) เพื่อที่จะได้ไม่ทำให้ข้อมูลเพี้ยนจากเดิม

โดยจะมีดังนี้ ข้อมูลที่ 249, 328, 347, 375, 454, 490, 521, 599, 642, 674

* เติม missing values ของ A2 โดยเติมตาม mean โดย mean = 31.5247 เพื่อไม่ให้ค่า mean เปลี่ยนไปจากเดิม โดยเติมข้อมูลที่ 84, 87, 93, 98, 254, 285, 328, 443, 448, 496, 511, 602
* เติม missing values ของ A6 และ A7 ของข้อมูลที่ 535 โดยค่าที่เติมไปคือ c และ v ตามลำดับโดยนำค่ามาจาก mod ของแต่ละ column
* เติม missing values ของ A14 โดนการนำ mean ไปแทนทุกค่าที่หายไป

**การแบ่งชุดข้อมูลในการ Train และ Test**

โดยในที่นี้ได้แบ่งข้อมูลในการ Train และ Test ดังนี้

Train 90 percent

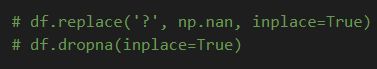
Test 10 percent

โดยการที่แบ่ง Train 90% และ Test 10% ก็เพราะว่า ใน Credit Approval Data นั้นมี Attribute Value ที่มาก จึงจำเป็นต้องนำไป Train มากแต่ไม่มากจนเกินไปไม่งั้นจะเกิดการ overfitting ดังนั้นจึงต้องลองแบ่งข้อมูลหลายๆแบบไปทดลองจริงจนได้ค่า accuracy ที่ดีที่สุดหรือเรียกได้ว่า good fit นั่นเอง

**สมมติฐานการทดลองที่ 1**

การทดลองเพื่อทดสอบผลความถูกต้องจาก การจำแนก โดยการ preprocess ที่ต่างกัน

โดยข้อมูลชุดแรกเป็นข้อมูลที่ไม่ได้ผ่านการ preprocessing by human แต่เป็นการ preprocessing ด้วย code ดังนี้



โดย code ชุดนี้หมายถึง เมื่อ ? หมายถึง missing data ให้ drop ข้อมูลทั้งแถวทิ้งไปและข้อมูลอีกชุดคือข้อมูลที่ผ่านการ preprocessing by human โดยจะทำตามขั้นตอนการ preprocessing ที่กล่าวไว้ก่อนหน้านี้

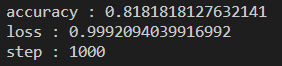
ทำการทดลองโดยนำข้อมูลทั้งสองชุดไปผ่านการ classify และตรวจดู accuracy รวมทั้ง loss ของข้อมูลและเปรียบเทียบกันโดยที่มีเงื่อนไขที่เหมือนกันดังนี้

1. มี 2 hidden layers

2. แต่ละ layer มี node เท่ากันคือ 20

3. ทั้งสองชุดข้อมูลมี step คือ 1000

ข้อมูลชุดที่ผ่านการตัดออกด้วย code ข้อมูลที่ผ่านการ preprocessing ด้วยมือ

จากการทดลองแสดงให้เห็นว่าข้อมูลที่ผ่านการ preprocessing ด้วยคำสั่ง มี loss ที่สูงมากเนื่องจากมีการตัดออกของข้อมูลที่มากเกินไปทำให้มีข้อมูลในการ Train และ Test ไม่พอ รวมทั้งมี accuracy ที่น้อยกว่าข้อมูลที่ผ่านการปรับปรุงด้วยมือ แสดงให้เห็นว่าการ preprocessing ส่งผลต่อการ classify data ที่เรามี

**สมมติฐานการทดลองที่ 2**

การทดลองเพื่อทดสอบโครงสร้าง NN ที่เหมาะสม เช่น มี output layer, มี 1 hidden layer มี 2 hidden layer

สมมติฐาน ถ้ายิ่งมี hidden layer มาก จะยิ่งมี accuracy ที่มากขึ้น

ทดลองโดยกำหนดให้ จำนวน node ของแต่ละ hidden layer คือ 20 และ step คือ 2000

1. Output ของ hidden layer = 1 (20 node per layer , step 2000)

ครั้งที่ 1



ครั้งที่ 2



ครั้งที่ 3



2. Output ของ hidden layer = 2 (20 node per layer , step 2000)

ครั้งที่ 1



ครั้งที่ 2



ครั้งที่ 3



3. Output ของ hidden layer = 3 (20 node per layer , step 2000)

ครั้งที่ 1



ครั้งที่ 2



ครั้งที่ 3



จากการทดลองของ hidden layer สรุปได้ดังตารางนี้

|  |  |  |
| --- | --- | --- |
| จำนวน hidden layer | average accuracy | average loss |
| 1 | 85.02% | 31.59% |
| 2 | 87.44% | 31.58% |
| 3 | 81.64% | 40.58% |

จากกราฟ จะสรุปได้ว่า hidden layer มากไม่ได้ส่งผลดีเสมอไป ดังนั้นแต่ละข้อมูลจะมี hidden layer ที่เหมาะสมต่างกันไป ดังนั้นในข้อมูลชุดนี้ hidden layer ที่เหมาะสมคือ 2 นั่นเอง แต่อาจจะแตกต่างกันไปหากใช้ node ในแต่ละ layer ต่างกันไป

สมมติฐานการทดลองที่ 3

การทดลองเพื่อทดสอบ จำนวน node ที่เหมาะสม

สมมติฐานการทดลอง เมื่อ node ยิ่งมากเท่าไหร่ accuracy ยิ่งมากขึ้นเท่านั้น

ทำการทดลองโดยกำหนดให้

จำนวน hidden layer คือ 1 และ step คือ 2000

ผลการทดลอง

1. จำนวน node 5 ตัว



2. จำนวน node 40 ตัว



3. จำนวน node 250 ตัว



4 จำนวน node 690 ตัว



5 จำนวน node 1000 ตัว



จากการทดลองของ node สรุปได้ดังตารางดังนี้

|  |  |  |
| --- | --- | --- |
| จำนวน node | accuracy | loss |
| 5 | 85.50% | 33.66% |
| 40 | 88.40% | 27.11% |
| 250 | 86.96% | 26.06% |
| 690 | 86.96% | 27.68% |
| 1000 | 79.71% | 65.18% |

จากกราฟแสดงให้เห็นว่าการที่มี node มากไปจะทำให้ output ที่ออกมามีค่า loss ที่สูงมากรวมทั้งค่า accuracy ลดลงดังนั้นจึงสรุปได้ว่า ในแต่ละ data จะมีค่า node ที่เหมาะสมกับข้อมูลอยู่แล้วไม่จำเป็นต้องใช้ node ที่มากเกินไป

สมมติฐานการทดลองที่ 4

ความสัมพันธ์ระหว่างการกำหนดค่า learning rate และ จำนวน epoch

สมมติฐาน ยิ่ง step มากขึ้นเท่าไหร่ accuracy จะมากขึ้นและ loss จะน้อยลง

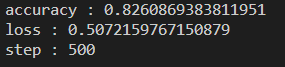
ทดลองโดยมี 2 hidden layer (20 nodes per layer)

ผลการทดลอง

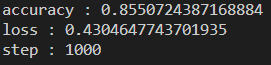
1. 100 steps



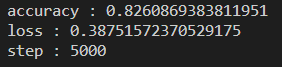
2. 500 steps



3. 1000 steps



4. 5000 steps



จะได้กราฟดังภาพ

จากกราฟสามารถสรุปได้ว่าจำนวน step หรือที่เรียกว่า epoch นั่นจะส่งผลต่อ accuracy แต่ที่เห็นได้ชัดคือ loss ที่ลดลงอย่างเห็นได้ชัดเจอ ยิ่ง step ยิ่งดี แต่ก็จะส่งผลต่อความรวดเร็วในการคำนวณ ดังนั้นการเลือก step ที่เหมาะสมจะช่วยให้การคำนวณเป็นไปอย่างมีประสิทธิภาพโดยการคำนวณไม่ได้ขึ้นอยู่กับค่า step อย่างดียังขึ้นอยู่กับค่าอื่นๆอีกด้วย

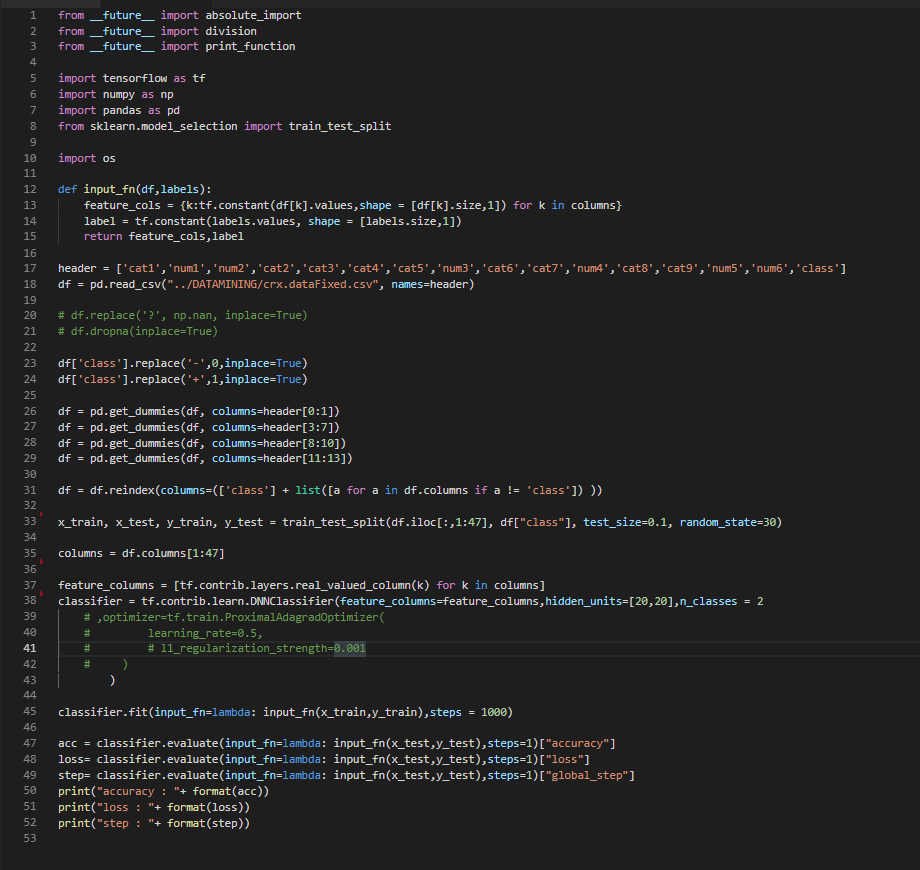
สรุปผลการทดลอง

1. ข้อมูลนี้มี hidden layer ที่เหมาะสมคือ 2

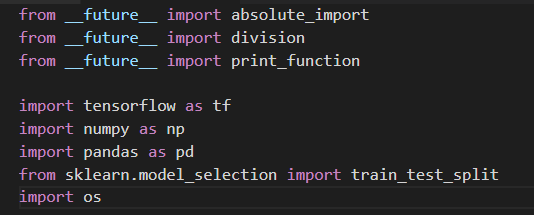
2. มีจำนวน node ต่อ hidden layer คือ 20

3. step ที่เหมาะสมกับข้อมูลชุดนี้คือ 1000

Code โดยรวมทั้งหมด



อธิบาย code ที่ใช้งาน (เขียนในภาษา python)



import สิ่งที่จำเป็นในการทำการสิ่งที่จำเป็นในการทำงานครั้งนี้

tensorflow – เป็นไลบรารีที่ใช้ในการพัฒนา Machine Learning

numpy – เป็นโมดูลส่วนเสริมของ Python ที่มีฟังก์ชันเกี่ยวกับคณิตศาสตร์และการคำนวณต่างๆ มาให้ใช้งาน

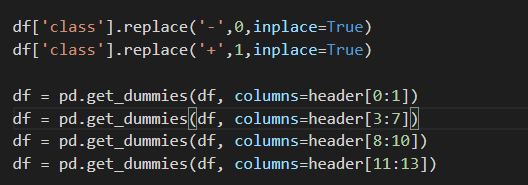
pandas – Library ที่มี Data Structure และ Function สำหรับการจัดการและเตรียมข้อมูลใน Python

ขั้นตอนการนำเข้าข้อมูล



กำหนด header ให้ในแต่ละ column เพื่อให้ง่ายต่อการเรียกใช้ในภายหลัง

ขั้นตอนการจัดการข้อมูล



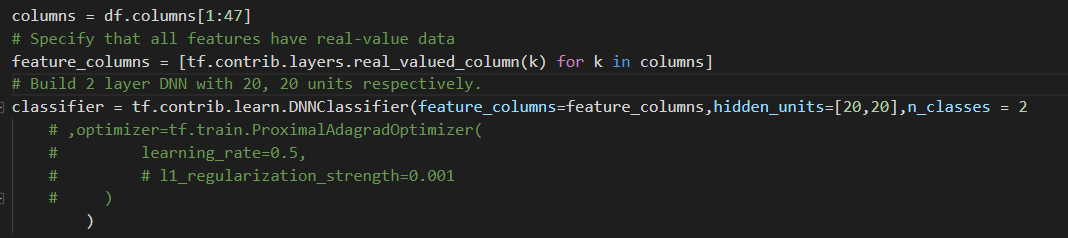
เปลี่ยนค่า class + - ให้เป็นตัวเลขเพื่อนำไปใช้งานต่อรวมทั้งการ get\_dummies ของ pandas จะช่วยให้จัดการข้อมูลที่เป็น categories ได้ง่ายยิ่งขึ้น

แบ่งข้อมูลก่อนนำไป train และ test

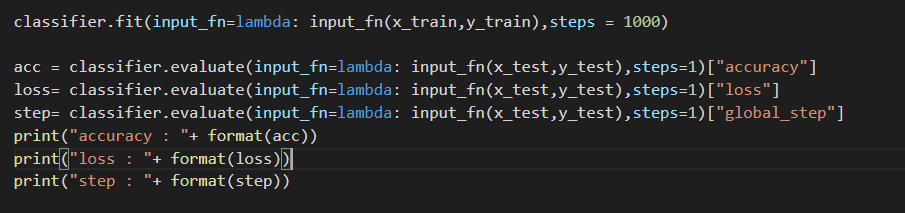
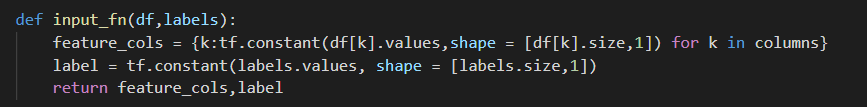


จากภาพจะเห็นว่าพารามิเตอร์ของ test\_size คือ 0.1 หมายถึง train 90% test 10% random\_state คือการสุ่มสลับข้อมูล

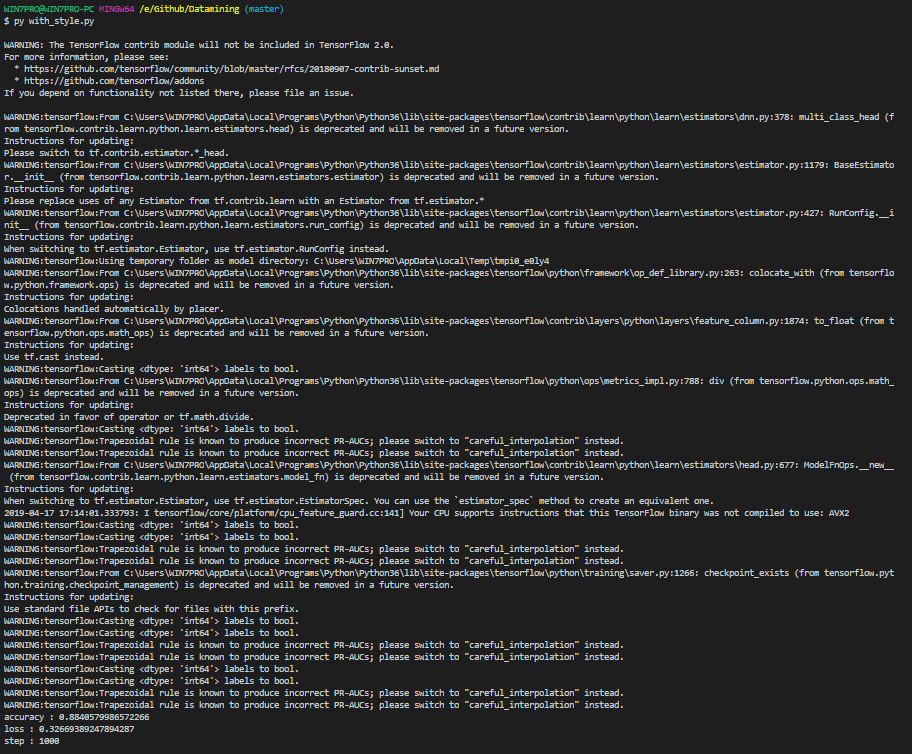
ขั้นตอนการ train test , create model และวิเคราะห์ผล

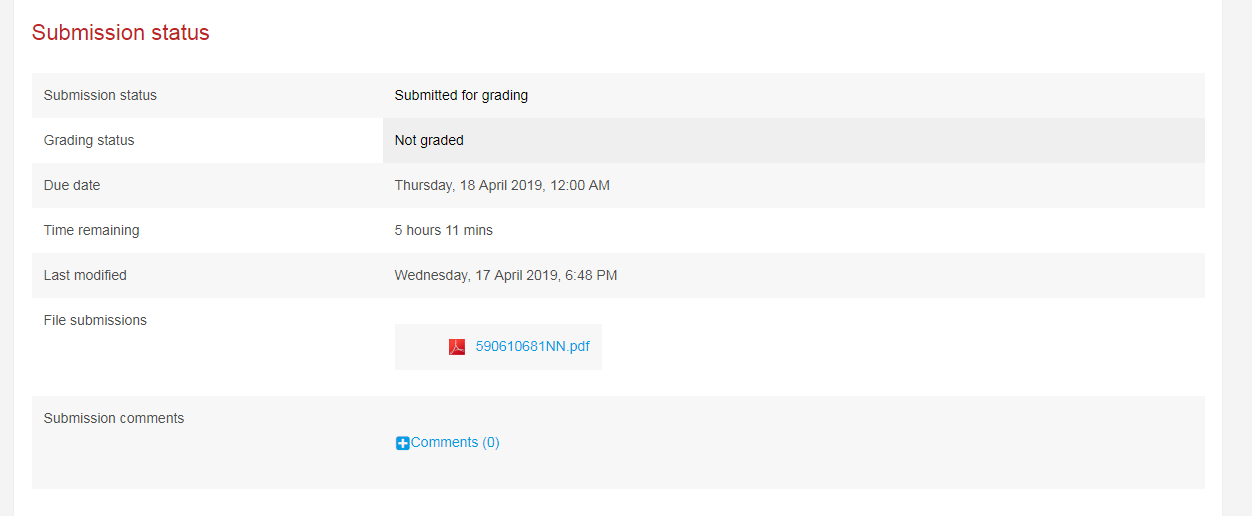


จากภาพเป็นการกำหนด feature columns เพื่อนำไป train โดยมี hidden layers 2 ชั้นดูจาก พารามิเตอร์ hidden\_units และมีการกำหนด class ไว้ว่า 2 ในที่นี้ data มีเพียง 2 class



จากนั้นจะมีการนำไป train ในคำสั่ง classifier.fit และวิเคราะห์ผลออกมาตามที่ต้องการในคำสั่ง classifier.evaluate โดย lambda นั่นหมายถึงการประมวลผลแบบ ไร้เซิร์ฟเวอร์ จะทำให้สามารถประมวลผลได้ในเครื่องของตนเอง

ตัวอย่าง Output



For more information about this project see at : <https://github.com/RAIDSouL/Datamining>