โครงงานที่ 2: การสร้างเครือข่ายประสาทเทียมสำหรับ Lead Generation Identificaiton

รหัสนักศึกษา ____57130500098_____ ชื่อ-นามสกุล _____กรรชัย สดหอม_____

เป้าหมายของโครงงาน:

สร้าง Neural Network ที่สามารถทำนาย conversion rate จากข้อมูลของผู้เยี่ยมชมเว็บไซต์ได้

ชุดข้อมูล:

https://github.com/kulwadeesom/int492_256002/raw/master/exercise01/lead_gen.csv

1. **การทำความเข้าใจข้อมูล (Data Understanding)**

   ข้อมูลที่ได้มาคือข้อมูลของ Lead Generation ซึ่งอธิบายเกี่ยวกับคนที่มีแนวโน้มจะมาเป็นลูกค้าในอนาคตหรือว่าอาจจะเป็นว่าที่ลูกค้าก็ได้ เพื่อจะได้คาดเดาได้ว่าลูกค้าที่เข้ามาใหม่ของแต่ละคนมีแนวโน้มที่จะเป็นลูกค้ามากขนาดไหน เพื่อเป็นประโยชน์ต่อธุรกิจต่อไป ซึ่งข้อมูลที่ได้มามี 6 Column ซึ่งมี Source, Returning, Mobile, Country, Pages_viewed, Lead ซึ่งข้อมูลที่มีอยู่นั้นมีจำนวน 287,742 Records ซึ่งมีอยู่ 2 Column Source กับ Country นั้นเป็น Categorical

```
dataset
```

	source	returning	mobile	country	pages_viewed	lead
0	Taboola	1	1	US	2	0
1	Taboola	0	1	US	6	0
2	Facebook	0	0	US	2	0
3	Taboola	0	0	US	5	0
4	Taboola	0	0	Canada	5	0
5	Adwords	0	1	US	2	0
6	Taboola	0	0	US	6	0
7	Facebook	0	1	UK	1	0
8	Adwords	0	1	Canada	1	0
9	Facebook	0	0	US	6	0
10	Facebook	1	1	US	1	0
11	Facebook	1	1	US	6	0
12	Facebook	0	1	US	4	0
13	Influencer	0	1	Canada	13	0
14	Taboola	0	1	UK	5	0
15	Adwords	0	0	UK	13	1
16	Taboola	0	1	UK	10	0
17	Facebook	0	0	Canada	4	0

2. **การเตรียมข้อมูล (Data Pre-processing)**

   เนื่องจากข้อมูลที่กล่าวไว้ข้างต้นว่ามี 2 Column นั้นเป็น Categorical นั้นก็คือ Source, Country เราก็จับแปลงข้อมูลสอง Column นี้ให้เป็น Numeric เพื่อให้จ่ายต่อการคำนวนทางคณิตศาสตร์

```
## Change categorial data to numerical data
#source -> 1=Taboola 2=Facebook 3=Adwords 4=Influencer
#country -> 1=US 2=Canada 3=UK

dataset["source"] = pd.Categorical(dataset['source'], dataset['source'].unique())
dataset["country"] = pd.Categorical(dataset['country'], dataset['country'].unique())
dataset["source"]  = dataset["source"].cat.rename_categories([1,2,3,4])
dataset["country"]  = dataset["country"].cat.rename_categories([1,2,3])
```

ซึ่งการแปลงก็จะใช้ Pandas ซึ่งเป็น library ตัวนึงแปลงข้อมูล dataset ที่ load มาจาก csv มาเป็น categories ซึ่งจะแทนค่าเป็นตัวเลข 1,2,3,4 ตามลำดับ ส่วน Country ก็จะแปลงเป็น 1,2,3

```
train_data = dataset.sample(frac=0.7,random_state=300)
test_data = dataset.drop(train_data.index)
print("train data size: ", train_data.shape)
print("test data size: ",test_data.shape)
```

```
train data size:  (201419, 6)
test data size:  (86323, 6)
```

ต่อมาก็เป็นการแบ่ง Data สำหรับการ Train และ Test เป็นสัดส่วน 70% และ 30% ตามลำดับก็จะได้จำนวนข้อมูลที่ใช้ในการ Train และ Test

3. การสร้างโมเดลเครือข่ายประสาทเทียม (Model Building)

- ด้าน Architecture ก็จะเลือกใช้ Fully Connected โดยจะ Input 5 Node เข้าไปแล้วมี Hidden Layer อยู่ 2 Layer แล้ว output ก็จะออกเป็น 2 Node ตาม Class
- Hidden Layers จะใช้ Activation แบบ Tanh
- เนื่องจากเป็นการ Classification ดังนั้นเราจะใช้ Loss Function คือ Mean Squared Error และเอาข้อมูล Gradient Descent มาปรับใช้โดยใช้อัลกอลิทึมคือ SGD
- เนื่องจากข้อมูล Output ที่ออกมาเป็น 0-1 เราเลยใช้ Activation ตอน Output เป็น Softmax เพื่อปรับค่าให้อยู่ในช่วง 0-1

4. การเทรนโมเดล (Model Training)

4.1 Macbook Pro 2.7GHz - 128GB 2.7GHz dual-core Intel Core i5 processor (Turbo Boost up to 3.1GHz) with 3MB shared L3 cache and RAM 8 GiB with macOS High Sierra (version 10.13) and Jupyter Lab with Keras version 2.1.6 และใช้ Library numpy และ pandas เพื่อจัดการกับข้อมูล

4.2 ค่า hyperparameter ต่างๆ

- Loss Function ใช้ Mean Squared Error
- Optimizer ใช้ SGD
- Epochs = 100
- batch_size = 128

4.3 รายงานผลการเทรน โดยแสดงค่า Training Accuracy และระยะเวลาทั้งหมดที่ใช้ในการเทรนเน็ตเวิร์ก

```
model = Sequential()
model.add(Dense(10, input_dim = dimof_input_train, activation='relu'))
model.add(Dense(100, activation='tanh'))
model.add(Dense(100, activation='tanh'))
model.add(Dense(dimof_output_train, activation='softmax'))
model.compile(loss='mse', optimizer='sgd', metrics=['accuracy'])
model.fit(X, Y,epochs= 100, batch_size=128, verbose = 2)
loss, accuracy = model.evaluate(X, Y, verbose=0, batch_size=128)
print('loss train: ', loss)
print('accuracy train: ', accuracy)
```

```
Epoch 1/100
 - 3s - loss: 0.0326 - acc: 0.9680
Epoch 2/100
 - 3s - loss: 0.0262 - acc: 0.9689
Epoch 3/100
 - 3s - loss: 0.0165 - acc: 0.9804
Epoch 4/100
 - 4s - loss: 0.0146 - acc: 0.9825
Epoch 5/100
 - 4s - loss: 0.0139 - acc: 0.9830
Epoch 6/100
 - 4s - loss: 0.0135 - acc: 0.9835
Epoch 7/100
 - 3s - loss: 0.0132 - acc: 0.9836
Epoch 8/100
 - 4s - loss: 0.0130 - acc: 0.9838
Epoch 9/100
 - 3s - loss: 0.0128 - acc: 0.9840
Epoch 10/100
 - 3s - loss: 0.0126 - acc: 0.9841
```

ระยะเวลาในการเทรนในแต่ละ Epoch นั้นจะตกที่รอบละประมาณ 3 วินาที เนื่องจากมี 100 Epochs ก็จะใช้เวลาประมาณเกือบ 300 วินาที หรือประมาณ 5 นาทีนั่นเอง

5. การทดสอบประสิทธิภาพของโมเดล (Model Evaluation)

```
Epoch 99/100
 - 6s - loss: 0.0118 - acc: 0.9851
Epoch 100/100
 - 7s - loss: 0.0117 - acc: 0.9853
loss train:  0.011880468708557437
accuracy train:  0.9851851116329641
```

ส่วนเรื่อง Accuracy ที่ได้อยู่ที่ 0.9851 หรือ 98.51%

6. สรุปผลและข้อเสนอแนะ

ผมได้เลือก Activation Function เป็น Tanh ในการทดสอบครั้งนี้ตอนแรกจะใช้ทั้งหมดแต่มันเกิดข้อมูลที่ไม่สมดุลกัน Imbalance Dataset เลยเพิ่ม relu ไว้ในส่วนแรกเพื่อดูผล Accuracy จากการทดสอบซึ่งก็เป็นที่น่าพอใจ แนะนำก็อยากลองแบ่ง Train,Test Data ใหม่ให้มันมีสัดส่วนที่ชัดเจนกว่านี้เช่น 0.6,0.4 หรือ 0.65,0.35 เป็นต้น อาจจะทำให้ผลลัพธ์ที่เกิดขึ้นนั้นดีกว่านี้ก็เป็นได้

****

การส่งงาน: โครงงานนี้เป็นโครงงานเดี่ยว นักศึกษาต้องส่ง

1. **รายงานในรูปแบบไฟล์ PDF (บันทึกชื่อเป็น leadgen-report.pdf)**
2. **ซอร์สโค้ด**

โดยการอัพโหลดข้อมูลทั้งหมดไปไว้บน Dropbox folder ชื่อ **02_LEADGEN**

กำหนดส่ง: ภายใน 30 เมษายน 2561

****