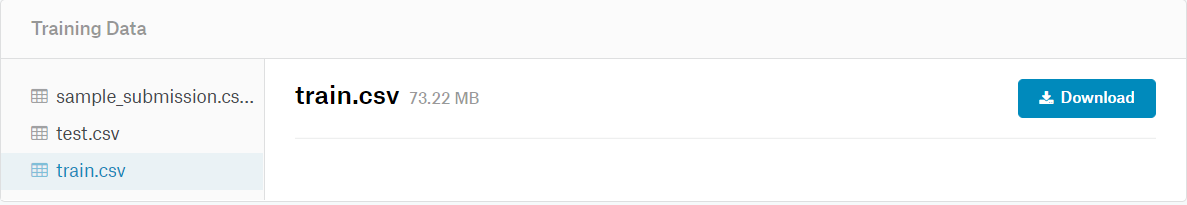
데이터 마이닝

Assignment #3

20121555 강승현

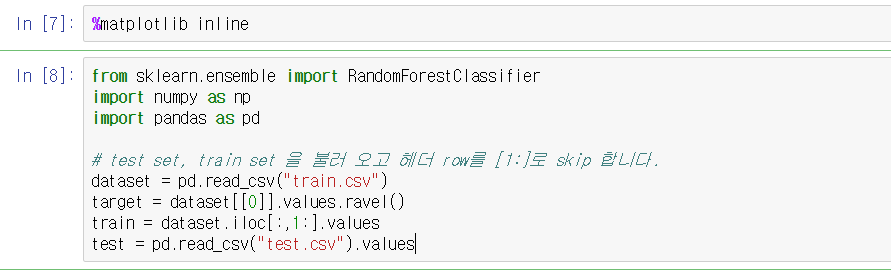
Exercise 1

1. **Test.csv, train.csv 다운**

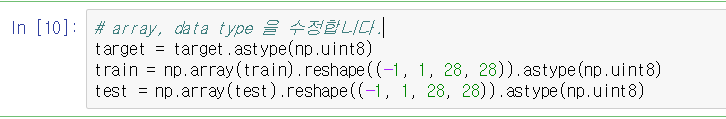


1. **모델 구축 후 예측 값을 .csv 파일로 정리**

**Code 설명**



: train.csv를 읽어 digit labels 를 target 이라는 변수에, 픽셀색의 행을 train이라는 변수에 저장한다. 그리고 test.csv를 읽어 온다.

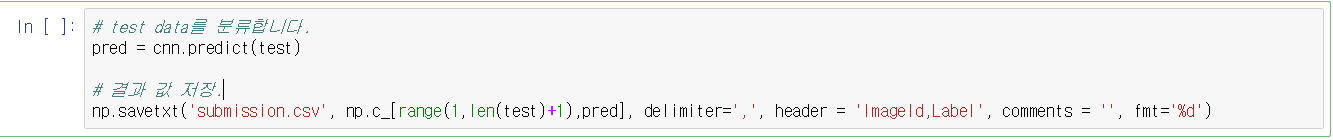


: list 의 값을 numpy array로 바꾸었으며 target 변수는 integer 타입으로 변환하였다.

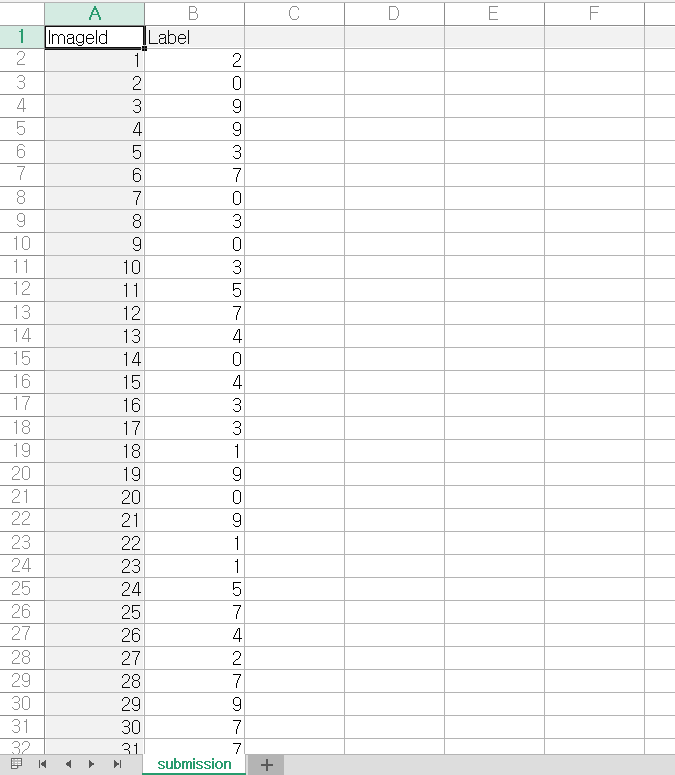
각각의 행은 28 X28 픽셀이므로 train set과 test set을 재배열하였다. (42k, 784) -> (42k,1)



: 신경망 구축 후 트레이닝 과정. 이 과정에서 Theano 의 lasagne 모듈과 nolearn 모듈을 설치하였으며 굉장히 오랜 시간이 걸렸다.

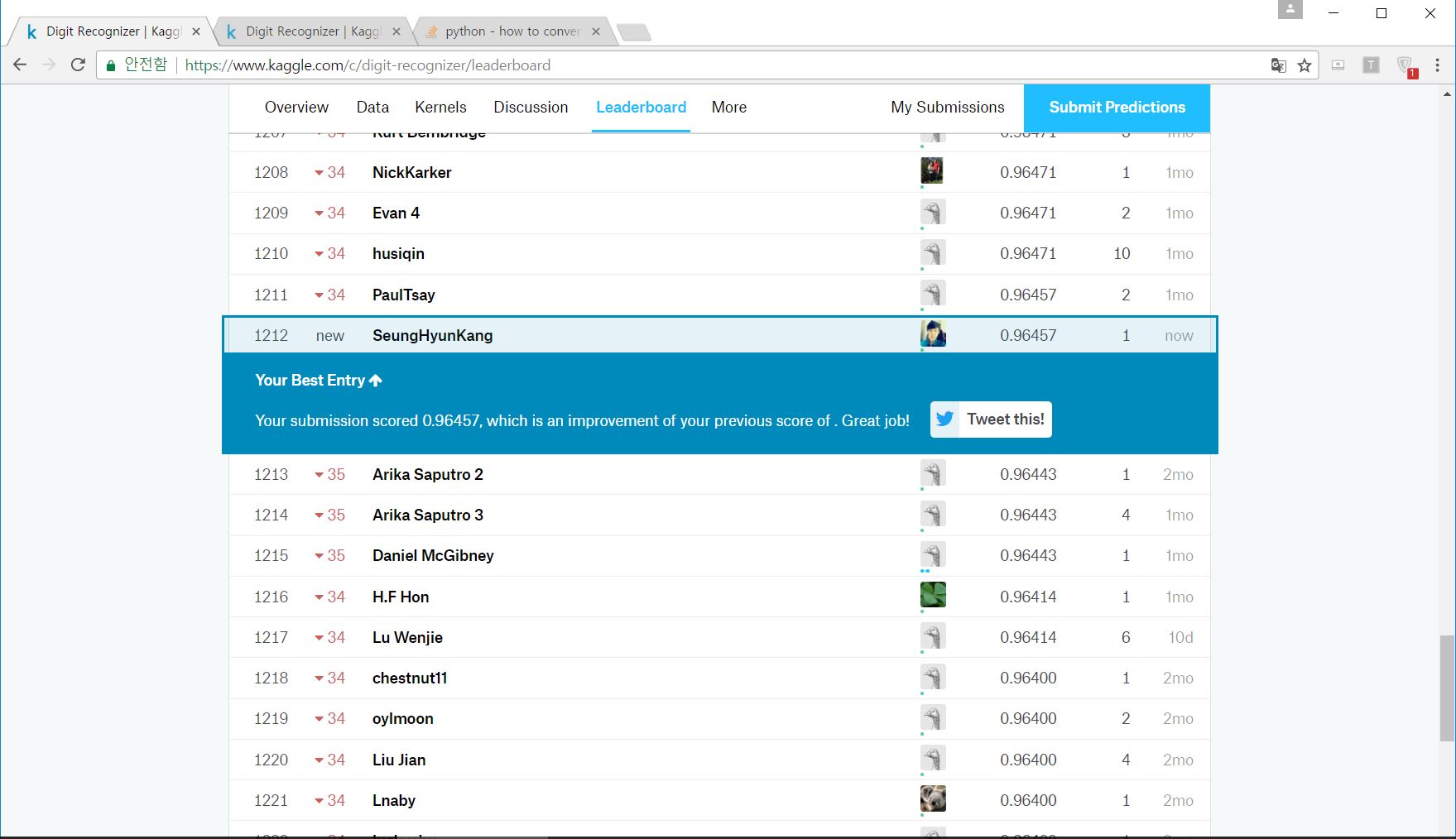


: 트레이닝 후 test data를 구별하고, 예측 값을 submission.csv로 저장하였다.



: Submission.csv 파일 캡처

1. **제출 결과.**



: submission.csv 파일 제출 후 0.96457 의 score 를 받았으며 1212등의 등수를 받았습니다.

**Python code :**

get\_ipython().magic('matplotlib inline')

from sklearn.ensemble import RandomForestClassifier

import numpy as np

import pandas as pd

# test set, train set 을 불러 오고 헤더 row를 [1:]로 skip 합니다.

dataset = pd.read\_csv("train.csv")

target = dataset[[0]].values.ravel()

train = dataset.iloc[:,1:].values

test = pd.read\_csv("test.csv").values

# array, data type 을 수정합니다.

target = target.astype(np.uint8)

train = np.array(train).reshape((-1, 1, 28, 28)).astype(np.uint8)

test = np.array(test).reshape((-1, 1, 28, 28)).astype(np.uint8)

import matplotlib.pyplot as plt

import matplotlib.cm as cm

plt.imshow(train[1729][0], cmap=cm.binary) # draw the picture

import lasagne

from lasagne import layers

from lasagne.updates import nesterov\_momentum

from nolearn.lasagne import NeuralNet

from nolearn.lasagne import visualize

net1 = NeuralNet(

layers=[('input', layers.InputLayer),

('hidden', layers.DenseLayer),

('output', layers.DenseLayer),

],

# layer parameters:

input\_shape=(None,1,28,28),

hidden\_num\_units=1000, # hidden layer 의 unit 개수

output\_nonlinearity=lasagne.nonlinearities.softmax,

output\_num\_units=10, # 0~9 까지의 target vale

# optimization

update=nesterov\_momentum,

update\_learning\_rate=0.0001,

update\_momentum=0.9,

max\_epochs=15,

verbose=1,

)

# Train

cnn = net1.fit(train, target)

#test data를 분류합니다.

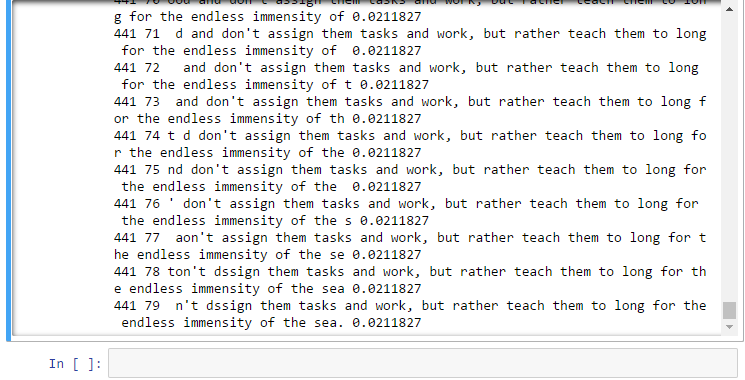
pred = cnn.predict(test)

# 결과 값 저장.

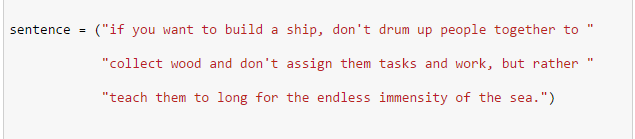
np.savetxt('submission.csv', np.c\_[range(1,len(test)+1),pred], delimiter=',', header = 'ImageId,Label', comments = '', fmt='%d')

Exercise 2

1. 실험 sentence :



(예시로 주신 sentence로 실험하였으나 컴퓨터 성능이 좋지 않아서인지 이 부분에서 더 이상 진행 되지않아 다음과 같이 더 짧은 문장을 사용하였습니다.)



Sequence\_length 변화(초기 20)

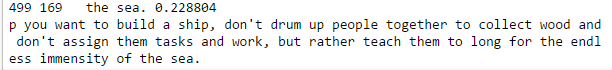


1. Sequence\_lentgh를 40으로 변화

C:\Users\cspclab\Downloads\새 폴더\sequence_20\2.PNG

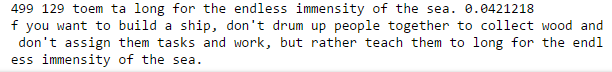
첫 단어 if -> l (94.44%)

1. Sequence\_lentgh를 10으로 변화



첫 단어 if -> p (94.44%)

1. Sequence\_lentgh를 50으로 변화

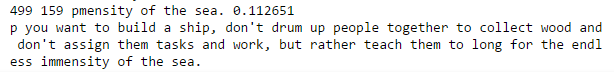


첫 단어 if -> f Loss값도 훨씬 좋아졌고 정확도도 훨씬 좋아졌습니다. (99.4%)

MultiRNNCell의 layer 수 변화(초기 3)

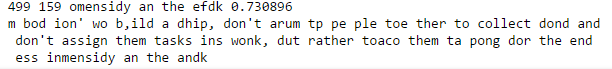


1. Layer = 1



첫 단어 if -> p (94.44%)

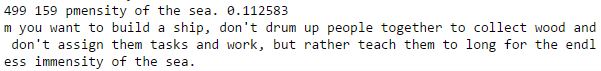
1. Layer = 5



중간중간에 맞는 단어도 있었지만 대체로 틀렸습니다. (41.66%)

Loss 값도 훨씬 높아 높을수록 좋지 않다는 것을 알았습니다.

1. Layer = 2



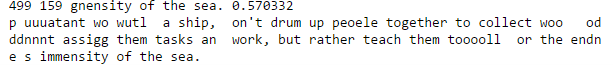
첫 단어 if -> m (94.44%)

Loss가 layer=1 일 때 보다 좋아 layer=2 가 최적인 것을 알게 되었습니다.

Cell의 종류 변화(초기 BasicRNNCell)



1. BasicLSTMCell



중간중간에 맞는 단어도 있었지만 대체로 틀렸습니다. (60.55%)

Loss 값 역시 훨씬 높아 높을수록 좋지 않다는 것을 알았습니다.

1. BasicRNNCell



Loss 값이 굉장히 높을 뿐만 아니라 정확도도 제일 안 좋았습니다. (0%)

기존의 BasicLSTMCELL이 가장 효과가 좋은 것을 확인 할 수 있었습니다.

여러 가지 parameter들을 변화시켜서 실험 해 본 결과, sequence\_length는 클수록 loss를 줄이는데 효과적인 것을 확인 할 수 있었으며, layer 의 개수는 너무 작게도, 크게도 아닌 2~3개가 제일 효과적이었습니다. 그리고 Cell의 종류로는 BasicLSTMCELL이 가장 효과가 좋은 것을 확인 할 수 있었습니다.

최적모델

Layer = 2~3

Seq\_num = 50

Cell = BasicLSTMCELL