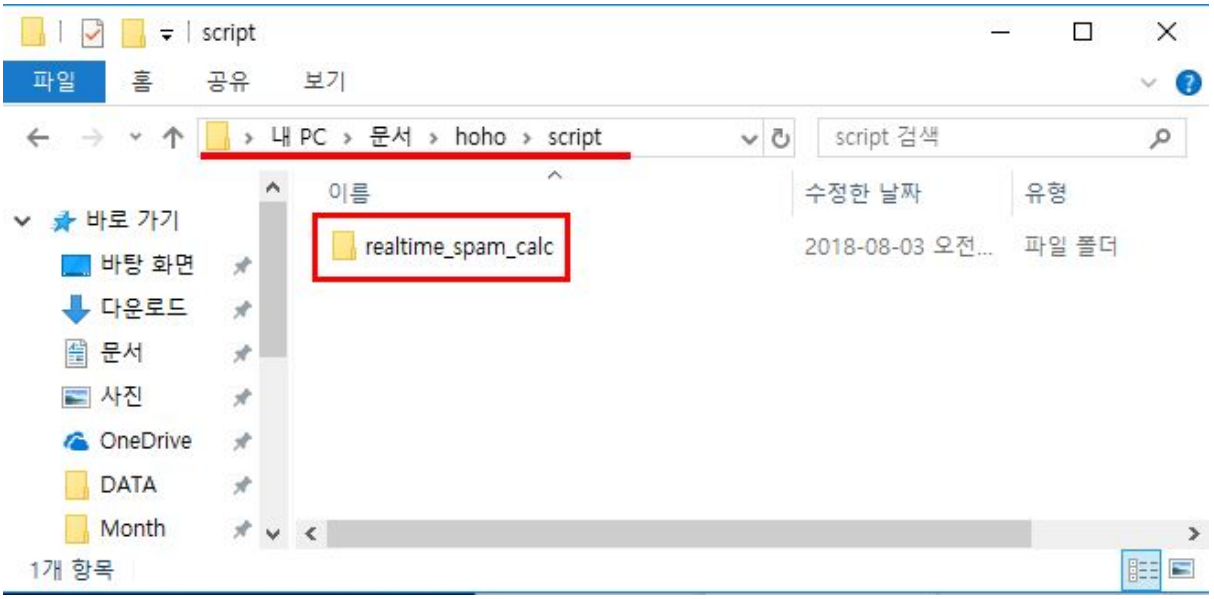
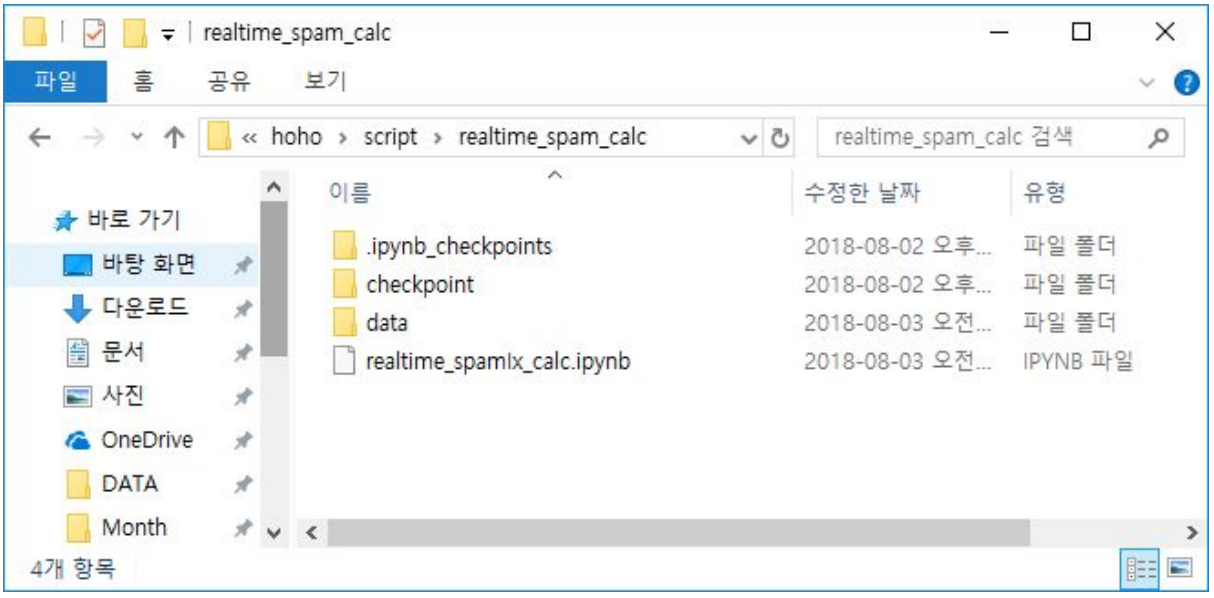
<<실시간 스팸지수 계산기 코드 실행법>>

1. 설치

파일을 받고 압축을 풀어줍니다.

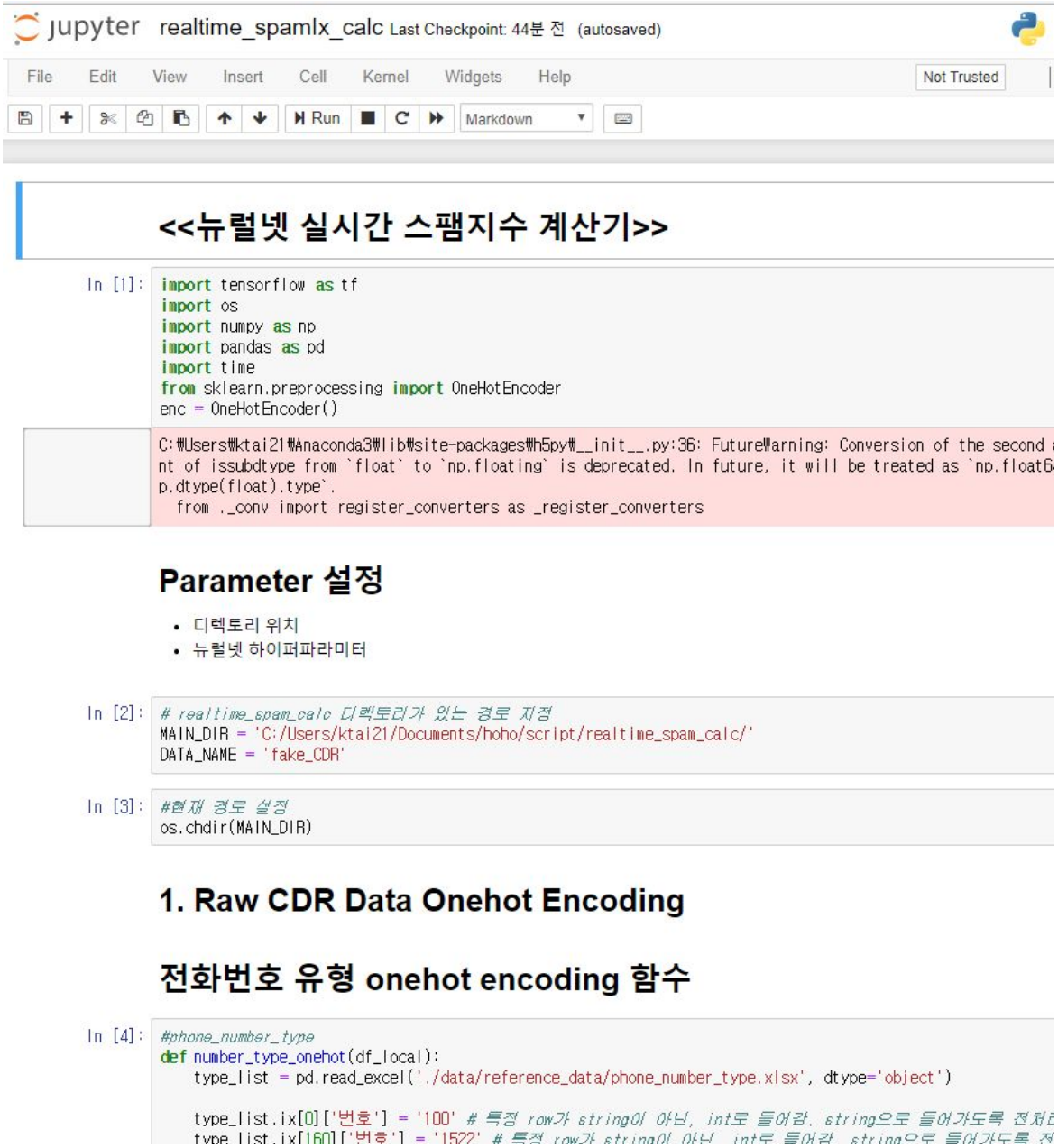


이때 realtime_spam_calc가 HOME 디렉토리가 됩니다.



HOME 디렉토리에 들어가면, ipynb 파일이 있습니다.

jupyter notebook을 이용해서 파일을 열어주면, 뉴럴넷 모델을 이용하여
실시간 스팸지수를 계산할 수 있습니다.



jupyter notebook을 통해 ipynb 파일을 열어보면 사진과 같이,
차례대로 실행할 수 있는 코드들이 나옵니다.

이때 코드를 실행하기 전에 먼저
Parameter 설정을 해주어야 합니다.

2. 경로 설정

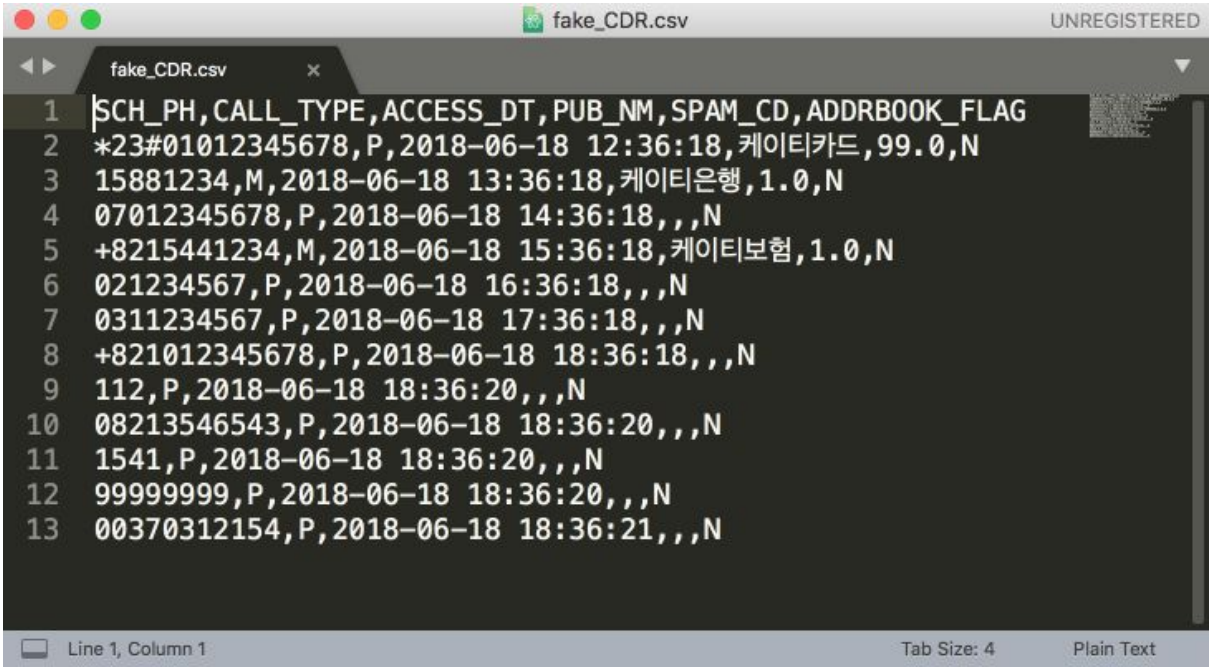
Parameter 설정

- 디렉토리 위치
- 뉴럴넷 하이퍼파라미터

```
# realtime_spam_calc 디렉토리가 있는 경로 지정
MAIN_DIR = 'C:/Users/ktai21/Documents/hoho/script/realtime_spam_calc/'
DATA_NAME = 'fake_CDR'
```

```
#현재 경로 설정
os.chdir(MAIN_DIR)
```

이 위치에 처음에 우리가 realtime_spam_calc 폴더 압축을 풀었던 위치를 적어줍니다.
그리고 그 밑에는 스팸인지 아닌지 식별할 때 사용할 CDR 데이터 이름을 적어줍니다.



데모로 사용할 CDR데이터의 경우
(./realtime_spam_calc/data/original_data/fake_CDR.csv)
이 경로에 저장이 되어있습니다.

이렇게 Parameter 설정이 모두 끝났다면,
ipynb 의 cell들을 순서대로 실행해주면 됩니다.
(cell을 실행하는 단축키는 shift + Enter 입니다.)

3. 코드 구성

코드는 크게 3가지 구성으로 이루어져 있습니다.

1. Raw CDR Data Onehot Encoding

2. Deep Learning Model

3. 비식별 발신번호에 대한 스팸 지수 예측 결과 보여주기(post processing)

1. Raw CDR Data Onehot Encoding

전화번호 유형 onehot encoding 함수

```
#phone_number_type
def number_type_onehot(df_local):
    type_list = pd.read_excel('./data/reference_data/phone_number_type.xlsx', dtype='object')

    type_list.ix[0]['번호'] = '100' # 특정 row가 string이 아닌, int로 들어감, string으로 들어
    type_list.ix[160]['번호'] = '1522' # 특정 row가 string이 아닌, int로 들어감, string으로 들

    type_list['자리수'] = 0
    for i in range(len(type_list)):
        type_list['자리수'][i] = len(type_list['번호'][i])

    df_0 = type_list[type_list['타입 번호']=='0']
    df_0_2 = df_0[df_0['자리수']==2]
    df_0_3 = df_0[df_0['자리수']==3]
    df_0_4 = df_0[df_0['자리수']==4]
    df_0_5 = df_0[df_0['자리수']==5]
```

2. Deep Learning Model

Neural Network Class

```
class NN(object):
    def __init__(self, sess, units=1000, name=None, threshold= 0.5):
        self.sess = sess
        self.units = units
        self.name = name
        self.threshold = threshold
        self._build_net()
        print(self.name)

    def _build_net(self):
        with tf.variable_scope(self.name):
            self.training = tf.placeholder(tf.bool)
            self.X = tf.placeholder(tf.float32, shape=[None, 37])
            self.Y = tf.placeholder(tf.float32, shape=[None, 1])
            self.learning_rate = tf.placeholder(tf.float32)

            L1 = tf.layers.dense(self.X, units=self.units, activation=None, kern
            L1 = tf.layers.batch_normalization(L1, training=self.training, name=
            #L1 = tf.nn.relu(L1, name='relu1')
            L1 = tf.sigmoid(L1, name='sigmoid1')
```

3. 비식별 발신번호에 대한 스팸 지수 예측 결과 보여주기

- (주의, 경계)의 범주로 보여주기

```
def post_process(df_number, logit):
    for idx in range(df_number.shape[0]):
        original_num = df_number['SCH_PH'][idx]
        prediction = logit[idx]
        criteria = max(logit)[0] - (max(logit)[0] - min(logit)[0])/2

        print('============prediction result============')
        if criteria <= logit[idx]:
            print(original_num, ' 는 경계번호일 가능성이 있습니다.')
        else :
            print(original_num, ' 는 주의번호일 가능성이 있습니다.')
```

```
df_number = pd.read_csv('./data/original_data/{}.csv'.format(DATA_NAME), usecols=['SCH_PH'])
post_process(df_number, logit)
```

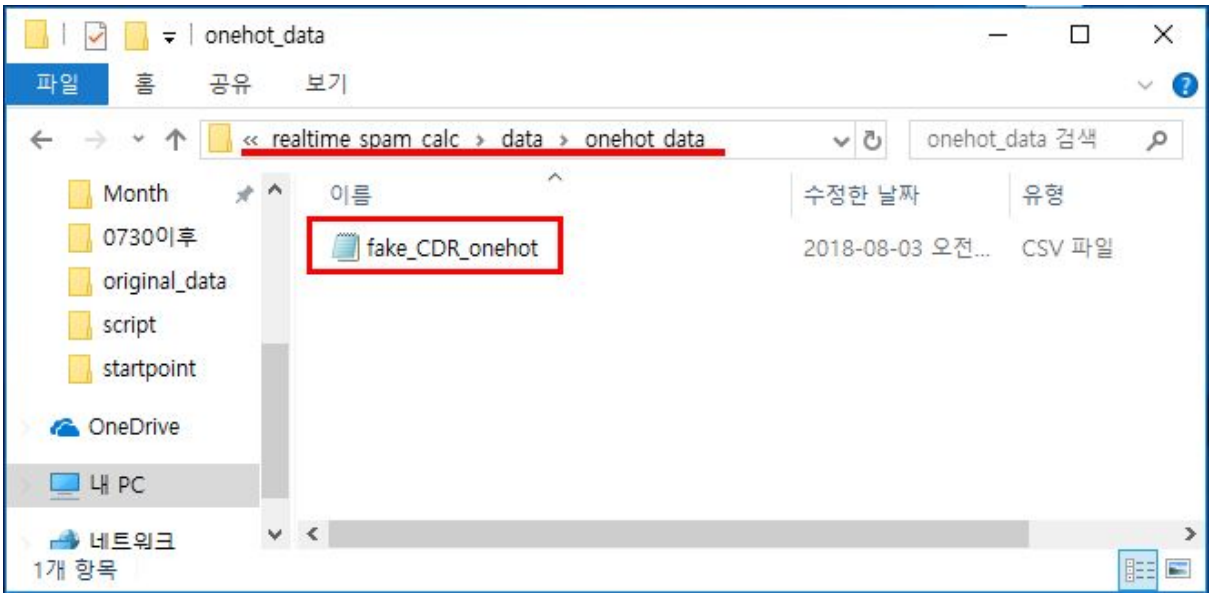
```
============prediction result============
*23#01012345678  는 주의번호일 가능성이 있습니다.
============prediction result============
15881234  는 경계번호일 가능성이 있습니다.
============prediction result============
07012345678  는 주의번호일 가능성이 있습니다.
============prediction result============
+8215441234  는 경계번호일 가능성이 있습니다.
============prediction result============
021234567  는 주의번호일 가능성이 있습니다.
============prediction result============
0311234567  는 주의번호일 가능성이 있습니다.
```

4. Onehot Encoding

Raw CDR Data Onehot Encoding

```
def encoder():

    start_time = time.time()

    #저장할 df 생성
    df_onehot = pd.DataFrame()

    df = pd.read_csv('./data/original_data/{}.csv'.format(DATA_NAME))
    ##################################################

    df_num_onehot = number_type_onehot(df)
    df_onehot = df_num_onehot
```

encoder() 함수가 있는 cell을 실행하게되면,
CDR데이터가 onehot encoding이 되어
아래의 사진과 같은 경로에 csv파일로 저장됩니다.



5. Neural Network Model 사용

Neural Network Class

```
class NN(object):
    def __init__(self, sess, units=1000, name=None, threshold= 0.5):
        self.sess = sess
        self.units = units
        self.name = name
        self.threshold = threshold
        self._build_net()
        print(self.name)

    def _build_net(self):
        with tf.variable_scope(self.name):
            self.training = tf.placeholder(tf.bool)
            self.X = tf.placeholder(tf.float32, shape=[None, 37])
            self.Y = tf.placeholder(tf.float32, shape=[None, 1])
            self.learning_rate = tf.placeholder(tf.float32)

            L1 = tf.layers.dense(self.X, units=self.units, activation=None, ker
            L1 = tf.layers.batch_normalization(L1, training=self.training, name
            #L1 = tf.nn.relu(L1, name='relu1')
            L1 = tf.sigmoid(L1, name='sigmoid1')
            L1 = tf.layers.dropout(L1, rate=0.7, training=self.training, name='
            L2 = tf.layers.dense(L1, units=self.units, activation=None, kernel_
            L2 = tf.layers.batch_normalization(L2, training=self.training, name
            #L2 = tf.nn.relu(L2, name='relu2')
            L2 = tf.sigmoid(L2, name='sigmoid2')
            L2 = tf.layers.dropout(L2, rate=0.7, training=self.training, name='
            self.logits = tf.layers.dense(L2, units=1, activation=tf.sigmoid, k

        #self.cost = tf.reduce_mean(tf.square(self.logits - self.Y))
        #self.cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(
        # softmax는 multinomial classification
        self.cost = -tf.reduce_mean(self.Y * tf.log(self.logits) + (1-self.Y) *
        self.optimizer = tf.train.AdamOptimizer(self.learning_rate).minimize(se
        self.predicted = tf.cast(self.logits > self.threshold, dtype=tf.float32
        self.accuracy = tf.reduce_mean(tf.cast(tf.equal(self.predicted, self.Y)

    def predict(self, test_X, test_Y, training=False):
        return self.sess.run([self.logits, self.predicted], feed_dict={self.X:

    def train(self, train_X, train_Y, learning_rate, training=True):
        return self.sess.run([self.cost, self.optimizer], feed_dict={self.X: tr
```

뉴럴넷 모델은 클래스 형식으로 선언되어 있습니다.
NN 클래스 선언을 해준 뒤

Setting Hyper parameter

```
training_epochs = 100
decay_steps = 1000
decay_rate = 0.9
batch_size = 10000

units = 1024
learning_rate = 0.0001
threshold = 0.756
```

하이퍼 파라미터를 설정해줍니다.
이 ipynb 파일에서는 미리 학습시킨 weight 가 저장된 ckpt파일을 불러올 것이기 때문에
하이퍼 파라미터를 변경하지 않고 실행해주어야 합니다.

Create Model Object

```
model_name = 'layer2_lr{}_node{}'.format(learning_rate, units)


sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
model = NN(sess=sess, name=model_name, units=units, threshold = threshold)
```

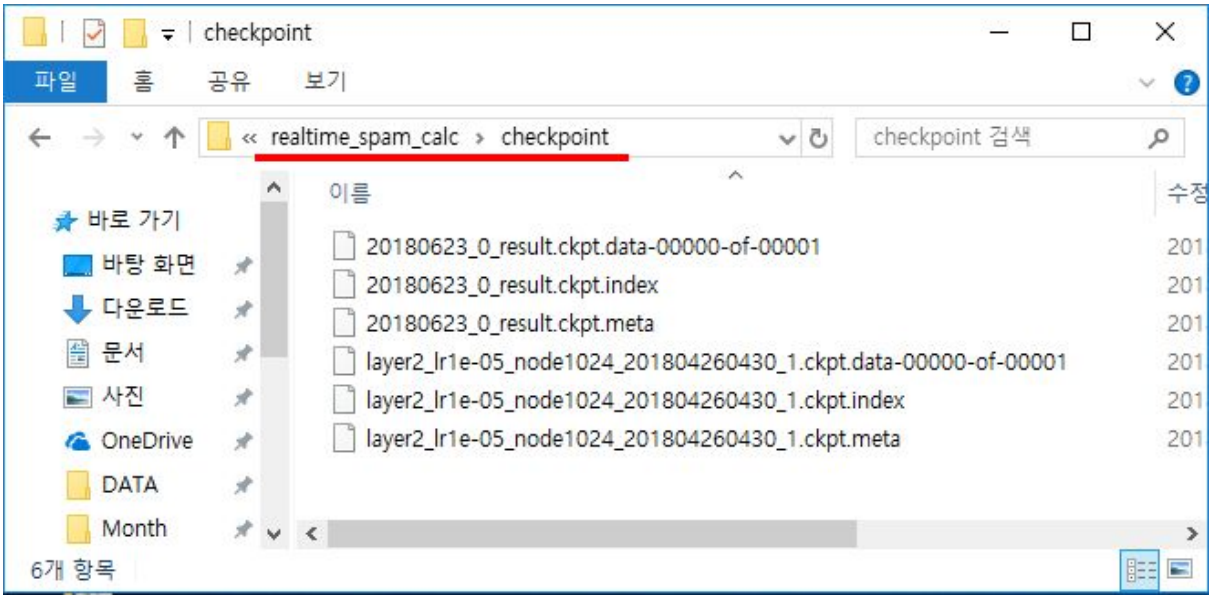
```
layer2_lr0.0001_node1024
```

Restore Weight

```
sess.run(tf.global_variables_initializer())
latest_checkpoint = '20180623_0_result.ckpt'

saver = tf.train.import_meta_graph('./checkpoint/{}.meta'.format(latest_checkpoint))
saver.restore(sess,'./checkpoint/{}'.format(latest_checkpoint))
#tf.trainable_variables()
```

```
INFO:tensorflow:Restoring parameters from ./checkpoint/20180623_0_result.ckpt
```

뉴럴넷 모델 인스턴스를 생성한 뒤
ckpt파일로 미리 학습된 weight를 불러옵니다.



ckpt 파일의 경우 위 사진과 같은 위치에 저장되어 있습니다.

스팸지수 예측(Test)

```
#data 위치
TEST_DATA= './data/onehot_data/{}_onehot.csv'.format(DATA_NAME)

with tf.device('/gpu:0'):
    start = time.time()
    print(TEST_DATA, 'START!')
    data = np.loadtxt(TEST_DATA, dtype=np.int, delimiter=',', skiprows=1)
    print(TEST_DATA, 'Load COMPLETE!')
    x_data = data[:,:37]
    y_data = data[:,-1].reshape(-1,1)
    logit, pred = model.predict(x_data, y_data)
    data = np.concatenate((data[:,:-1], pred), axis=1)
    print(sum(pred))
    np.savetxt(TEST_DATA, data, fmt='%d', delimiter=',')
    print(TEST_DATA, 'COMPLETE!₩t{}s'.format(time.time()-start))
```

```
./data/onehot_data/fake_CDR_onehot.csv START!
./data/onehot_data/fake_CDR_onehot.csv Load COMPLETE!
[0.]
./data/onehot_data/fake_CDR_onehot.csv COMPLETE!        0.21941351890563965s
```

모델을 불러오는 것이 완료되면, 예측 스팸지수를 계산합니다.

6. 결과 확인

3. 비식별 발신번호에 대한 스팸 지수 예측 결과 보여주기

- (주의, 경계)의 범주로 보여주기

```
def post_process(df_number, logit):
    for idx in range(df_number.shape[0]):
        original_num = df_number['SCH_PH'][idx]
        prediction = logit[idx]
        criteria = max(logit)[0] - (max(logit)[0] - min(logit)[0])/2

        print('============prediction result=============')
        if criteria <= logit[idx]:
            print(original_num, ' 는 경계번호일 가능성이 있습니다.')
        else :
            print(original_num, ' 는 주의번호일 가능성이 있습니다.')
```

```
df_number = pd.read_csv('./data/original_data/{}.csv'.format(DATA_NAME), usecols=['SCH_PH'])
post_process(df_number, logit)
```

```
============prediction result=============
+23#01012345678  는 주의번호일 가능성이 있습니다.
============prediction result=============
15881234  는 경계번호일 가능성이 있습니다.
============prediction result=============
07012345678  는 주의번호일 가능성이 있습니다.
============prediction result=============
+8215441234  는 경계번호일 가능성이 있습니다.
============prediction result=============
021234567  는 주의번호일 가능성이 있습니다.
============prediction result=============
0311234567  는 주의번호일 가능성이 있습니다.
============prediction result=============
+821012345678  는 주의번호일 가능성이 있습니다.
============prediction result=============
112  는 주의번호일 가능성이 있습니다.
============prediction result=============
08213546543  는 주의번호일 가능성이 있습니다.
============prediction result=============
1541  는 주의번호일 가능성이 있습니다.
============prediction result=============
99999999  는 주의번호일 가능성이 있습니다.
============prediction result=============
00370312154  는 주의번호일 가능성이 있습니다.
```

마지막으로, 각 번호에 대해 예측한 스팸지수를 이용하여
사용자에게 이 비식별 발신번호가 경계 수준인지, 주의 수준인지 알려줍니다.