In [1]:
```
from sklearn.model_selection import RepeatedStratifiedKFold
import sklearn.metrics
import warnings
import numpy as np
warnings.filterwarnings('ignore')
```

In [2]:
```
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras import regularizers
import matplotlib.pyplot as plt
```

In [3]:
```
from sklearn.datasets import load_breast_cancer
(X, y_) = load_breast_cancer(return_X_y=True)
n, p = X.shape
# y_原來1是良性，0是惡性，這裡把0 1互換
# 或是也可以在算metric那裏設pos_label=0
y = [1 if y_[i]==0 else 0 for i in range(len(y_))]
y = np.array(y)
```

In [4]:
```
n_repeats = 5
n_splits = 10
metrics_list = ['recall', 'precision', 'AUC']
```

In [5]:
```
model_0 = tf.keras.Sequential(name="model_0")
model_0.add(layers.Dense(1,
                         input_shape=(p, )))
```

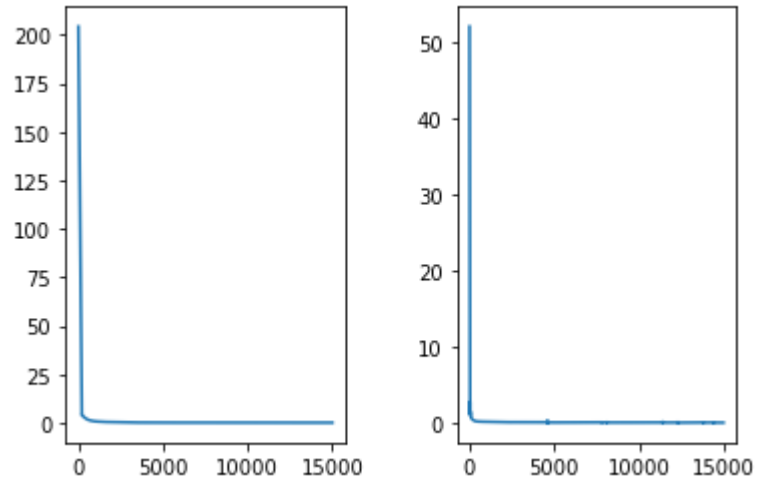
In [6]:
```
model_1 = tf.keras.Sequential(name="model_1")
model_1.add(layers.Dense(10,
                         input_shape=(p, )))
model_1.add(layers.Dense(5,
                         activation='relu',
                         kernel_regularizer=regularizers.l2(0.001)))
model_1.add(layers.Dense(1,))
n_models = 2
```

In [7]:
```
for i in range(n_models):
    locals()['model_{}'.format(i)].compile(
        optimizer='adam',
        loss=tf.keras.losses.BinaryCrossentropy(from_logits=True))
    locals()['loss_model_{}'.format(i)] = []
    for metrics in metrics_list:
        locals()[metrics + '_{}'.format(i)] = []
```

In [8]:
```
def sigmoid(x):
    s = 1 / (1 + np.exp(-x))
    return s
```

In [9]:
```
rskf = RepeatedStratifiedKFold(n_repeats=n_repeats, n_splits=n_splits)

for train_index, test_index in rskf.split(X, y):

    # 切training testing data
    X_tr, X_te = X[train_index], X[test_index]
    y_tr, y_te = y[train_index], y[test_index]

    for i in range(n_models):

        ###fit models###
        locals()['history_{}'.format(i)] = \
        locals()['model_{}'.format(i)].fit(
            X_tr, y_tr, batch_size=n, epochs=300, verbose=0)

        ###把loss存起來###
        locals()['loss_model_{}'.format(i)].extend\
        (locals()['history_{}'.format(i)].history['loss'])

        ###binary的predict###
        pred = (sigmoid(locals()['model_{}'.format(i)].\
                predict(X_te)) > 0.5).astype("int32")

        locals()['recall_{}'.format(i)].append\
        (sklearn.metrics.recall_score(y_te, pred))
        locals()['precision_{}'.format(i)].append\
        (sklearn.metrics.precision_score(y_te, pred))
        locals()['AUC_{}'.format(i)].append\
        (sklearn.metrics.roc_auc_score(y_te, pred))

# 把每次cv完的結果取平均
for i in range(n_models):
    for metrics in metrics_list:
        locals()[metrics + '_{}'.format(i)] = \
        np.mean(locals()[metrics + '_{}'.format(i)])
        # mean of (n_splits * n_repeats) values
```

In [10]:
```
plt.subplots_adjust(wspace=0.4)
# 每300個epochs會換一筆data train
for i in range(n_models):
    plt.subplot(1,2,i+1)
    plt.plot(locals()['loss_model_{}'.format(i)])
```



In [11]:
```
k = locals()
for metrics in metrics_list:
    locals()[metrics+'s'] = \
    [(k[metrics+'_{}'.format(i)]) for i in range(n_models)]
    locals()[metrics+'_max_idx'] = \
    np.argmax(locals()[metrics+'s'])
    locals()[metrics+'_max'] = \
    max(locals()[metrics+'s'])
```

In [12]:
```
# 不想錯過生病的人 --> recall要大 (thredhold低)
# 希望不要嚇到太多沒生病的人 --> precision要大 (threshold高)
# 以上兩點都想要的話 --> AUC要大

task_to_metrics = \
{'不想錯過生病的人':'recall', \
 '希望不要嚇到太多沒生病的人':'precision', \
 '以上兩點都想要的話':'AUC'}

for task, metrics in task_to_metrics.items():
    print("兩個分類器的{}: \n{}".format(metrics, locals()[metrics+'s']))
    best_idx = locals()[metrics+'_max_idx']
    print("如果{}，model_{}是最好的分類器，{}={}".format\
          (task, best_idx, metrics, locals()[metrics+'_max']))
    print("model_{}:".format(best_idx))
    print("confusion matrix: \n", \
          sklearn.metrics.confusion_matrix\
          ((sigmoid(locals()['model_{}'.format(best_idx)].\
                predict(X)) > 0.5).astype("int32"), y))
    print("#"*50)
```

```
兩個分類器的recall:
[0.908008658008658, 0.9323809523809524]
如果不想錯過生病的人，model_1是最好的分類器，recall=0.9323809523809524
model_1:
confusion matrix:
 [[352   4]
 [  5 208]]
##################################################
兩個分類器的precision:
[0.9276100290877572, 0.9595599862634188]
如果希望不要嚇到太多沒生病的人，model_1是最好的分類器，precision=0.9595599862634188
model_1:
confusion matrix:
 [[352   4]
 [  5 208]]
##################################################
兩個分類器的AUC:
[0.9306471861471863, 0.9541349206349207]
如果以上兩點都想要的話，model_1是最好的分類器，AUC=0.9541349206349207
model_1:
confusion matrix:
 [[352   4]
 [  5 208]]
##################################################
```