เอกสารอธิบายการทดลองที่ 3 ตอนที่ 3

การทดลองการค้นหาพารามิเตอร์ที่ดีที่สุดสำหรับโมเดล

ตอนที่ 3 : การทดลองการค้นหาพารามิเตอร์ที่ดีที่สุดสำหรับโมเดล

3.1 สร้างโมเดลรูปแบบโดยใช้ Talos เนื่องจากผู้ทำไม่สามารถใช้ GridSearchCV ได้

- สร้างฟังก์ชันสำหรับการหาค่า parameters ที่ดีที่สุดได้ โดยใช้ library ชื่อ Talos ซึ่งจะเป็นการหาค่าที่ดีที่สุดจากที่กำหนดไว้ โดยกำหนด optimizer ระหว่าง SGD และ Adam batch_size ค่าอยู่ในช่วง 8, 16, 20, 40, 60, 80 และ 100 epochs ค่าอยู่ในช่วง 10, 50 และ 100

```
def search_parameter(model):
    p = {
        # 'activation':['relu', 'sigmoid', 'tanh', 'softmax'],
         'optimizer': ['SGD','Adam'],
         'losses': ['categorical_crossentropy'],
         'batch_size': [8, 16, 20, 40, 60, 80, 100],
         'epochs': [10, 50, 100]
         }
    def cnn_model(X_train, y_train, x_val, y_val, params):
        model = Sequential()
        model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=
(X_train.shape[1:]), padding='same'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))
        model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', padding='sam
e'))
        model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', padding='sa
me'))
        model.add(Flatten())
        model.add(Dense(128 , activation='relu' ))
        model.add(Dense(y_train.shape[1], activation='sigmoid'))
        optimizer = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, amsgrad=
False)
        model.compile( loss='categorical_crossentropy', optimizer=optimizer, met
rics=['accuracy'])
        out = model.fit(X_train, y_train, batch_size=params['batch_size'], epoch
s=params['epochs'], validation_data=[x_val, y_val], verbose=0)
        return out, model

    def lstm_model(X_train, y_train, x_val, y_val, params):
        model = Sequential()
```

```
        model.add(LSTM(32, input_shape=(X_2d_train.shape[1:]) ))
        model.add(Dense(y_train.shape[1], activation='sigmoid'))
        optimizer = Adam(learning_rate=0.003, beta_1=0.9, beta_2=0.999, amsgrad=
False)
        model.compile( loss='categorical_crossentropy', optimizer=optimizer, met
rics=['accuracy'])
        out = model.fit( X_train,y_train, batch_size=params['batch_size'], valid
ation_data=[ x_val, y_val ], epochs=params['epochs'], verbose=0)
        return out, model

    if model == 'cnn' :
      scan_object = ta.Scan(X_train, y_train, model=cnn_model, params=p, experim
ent_name='cnn', fraction_limit=0.1 , x_val=X_2d_valid, y_val=y_valid)
    else :
      scan_object = ta.Scan(X_2d_train, y_2d_train, model=lstm_model, params=p,
experiment_name='lstm', fraction_limit=0.1 , x_val=X_2d_valid, y_val=y_valid)

    return scan_object
```

3.2 นำค่าพารามิเตอร์ที่ดีที่สุดไปสอนโมเดล

- สร้างฟังก์ชันสำหรับการแสดงพารามิเตอร์ที่ดีที่สุด

```
def show_search_res(scan, model) :
  # use Scan object as input
  analyze_object = ta.Analyze(scan)
  print(analyze_object.data)
  print('================================================')
  # get the highest result for any metric
  print('Low validate loss : ', analyze_object.high('val_accuracy'))
  print('================================================')
  # get the round with the best result
  print('\nindex of best result :', analyze_object.rounds2high('val_accuracy'))
  print('================================================')
  # evaluae with k fold
  e = ta.Evaluate(scan)
  model_evaluate = list()
  if model == 'lstm' :
    xx = X_2d_train
    yy = y_train
  else :
    xx = X_train
    yy = y_train
  for i in range(len(analyze_object.data)) :
    evaluate = e.evaluate(xx, yy, folds=10, metric='val_accuracy', task='multi_l
abel', model_id=i)
    model_evaluate.append(np.array(evaluate).mean())
  print('evaluate with kfold', model_evaluate)
```

```
    print('================================================')

    # get the best paramaters
    print('\nbest parameters :')
    print(analyze_object.best_params('val_accuracy', ['acc', 'loss', 'val_loss']))
    analyze_object.plot_line('val_accuracy')
    return analyze_object, model_evaluate
```

- ทำการเรียกใช้ฟังก์ชันด้านบน โดยจะคืนค่าผลลัพธ์ของการค้นหามา แล้วแสดงผลการค้นหา

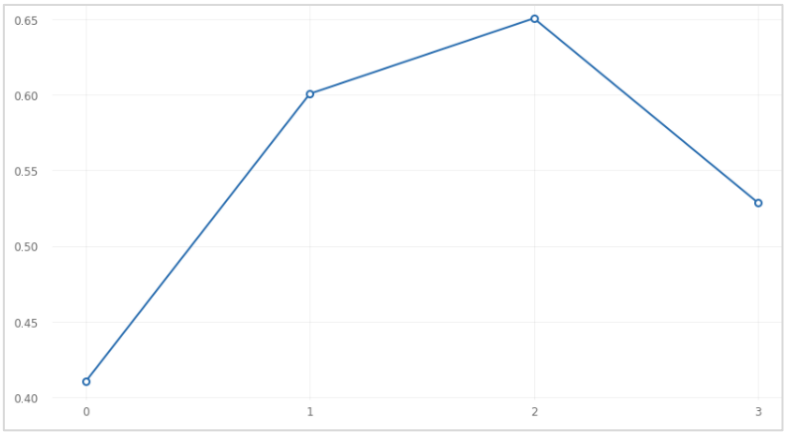
```
scan = search_parameter('cnn')
print(scan.details)
print('================================================')
analyze_object_cnn = show_search_res(scan, 'cnn')

scan = search_parameter('lstm')
print(scan.details)
print('================================================')
analyze_object_lstm = show_search_res(scan, lstm')
```

	round_epochs	val_loss	...	losses	optimizer
0	10	1.316270	...	categorical_crossentropy	Adam
1	50	1.010865	...	categorical_crossentropy	Adam
2	100	0.860332	...	categorical_crossentropy	Adam
3	10	1.169109	...	categorical_crossentropy	Adam

รูปพารามิเตอร์ที่ได้รับจากโมเดล CNN

```
evaluate with kfold [0.11613622881156505, 0.4688153053244834, 0.6201039375192072, 0.3640848097458648]
```

รูปคะแนนค่าแม่นยำเฉลี่ยของแต่ละ fold จากโมเดล CNN ด้านบน



รูปความแม่นยำของผลลัพธ์ในแต่ละรอบจากโมเดล CNN

```
CNN best score :  0.6201039375192072
CNN best parameters :
 round_epochs                          100
val_loss                          0.860332
val_accuracy                      0.650647
loss                              0.837482
accuracy                          0.652797
batch_size                              40
epochs                                 100
losses          categorical_crossentropy
optimizer                            Adam
NaN                                     2
Name: 2, dtype: object
```

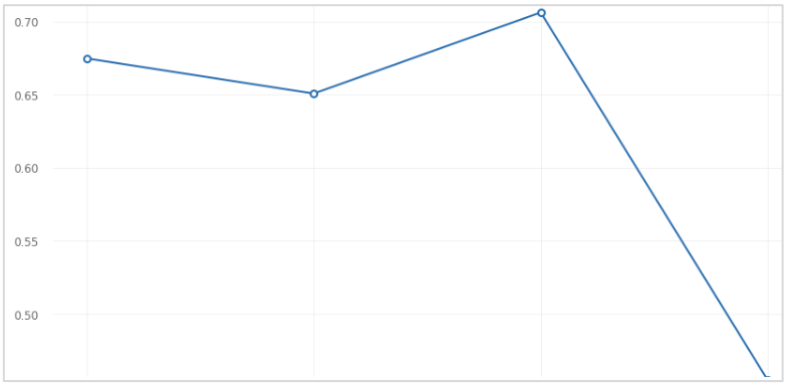
รูปค่าของ parameters ที่ดีที่สุดจากโมเดล CNN

```
0            100  0.877243  ...  categorical_crossentropy       Adam
1            100  0.859741  ...  categorical_crossentropy   adadelta
2            100  0.760143  ...  categorical_crossentropy       Adam
3             10  1.337980  ...  categorical_crossentropy   adadelta
```

รูปพารามิเตอร์ที่ได้รับจากโมเดล LSTM

```
evaluate with kfold [0.6168751723390622, 0.607709605196705, 0.7229203790975777, 0.2372887444272178]
```

รูปคะแนนค่าแม่นยำเฉลี่ยของแต่ละ fold จากโมเดล LSTM ด้านบน



รูปความแม่นยำของผลลัพธ์ในแต่ละรอบจากโมเดล LSTM

```
LSTM best score :  0.7229203790975777
LSTM best parameters :
 round_epochs                          100
val_loss                          0.760143
val_accuracy                        0.7061
loss                               0.59341
accuracy                          0.758206
batch_size                               8
epochs                                 100
losses          categorical_crossentropy
optimizer                            Adam
NaN                                     2
Name: 2, dtype: object
```

รูปค่าของ parameters ที่ดีที่สุดจากโมเดล LSTM

- สร้างฟังก์ชันโมเดล CNN เพื่อให้ง่ายต่อการเรียกใช้งาน

```
def cnn_model(optimizer, losses) :
  model = Sequential()
  model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(X_train.shape[1:
]), padding='same'))
  model.add(MaxPooling2D(pool size=(2, 2)))
  model.add(BatchNormalization())
  model.add(Dropout(0.25))
  model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same'))
  model.add(BatchNormalization())
  model.add(Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same'))
  model.add(BatchNormalization())
  model.add(Flatten())
  model.add(Dense(128 , activation='relu' ))
  model.add(Dense(y_train.shape[1], activation='sigmoid'))
  model.compile( loss=losses, optimizer=optimizer, metrics=['accuracy'])
  return model
```

- สร้างฟังก์ชันโมเดล LSTM เพื่อให้ง่ายต่อการเรียกใช้งาน

```
def lstm_model(optimizer, losses) :

  model = Sequential()
  model.add(LSTM(32, input_shape=(X_2d_train.shape[1:]) ))
  model.add(Dense(y_train.shape[1], activation='sigmoid'))
  model.compile( loss=losses, optimizer=optimizer, metrics=['accuracy'])
  return model
```

- นำค่าของ parameters ที่ดีที่สุดไปสร้างโมเดล โดยใช้ฟังก์ชัน cnn_model ที่สร้างขึ้น

```
model = cnn_model(best_params['optimizer'], best_params['losses'])
model.summary()
history = model.fit(X_train, y_train, batch_size=best_params['batch_size'], vali
dation_data=(X_valid, y_valid), epochs=best_params['epochs'], verbose=1)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 3, 4, 32)          320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 1, 2, 32)          0
_________________________________________________________________
batch_normalization_1 (Batch (None, 1, 2, 32)          128
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 2, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 1, 2, 64)          18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 1, 2, 64)          256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 1, 2, 128)         73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 1, 2, 128)         512
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 645
=================================================================
Total params: 127,109
Trainable params: 126,661
Non-trainable params: 448
```

รูปโครงสร้างโมเดล CNN โดยใช้ parameter จากการค้นหา

- นำค่าของ parameters ที่ดีที่สุดไปสร้างโมเดล โดยใช้ฟังก์ชัน lstm_model ที่สร้างขึ้น

```
model = lstm_model(lstm_best_params['optimizer'], lstm_best_params['losses'])
model.summary()
history = model.fit(X_2d_train, y_2d_train, batch_size=lstm_best_params['batch_s
ize'], validation_data=(X_2d_valid, y_valid), epochs=lstm_best_params['epochs'],
 verbose=1)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 32)                4736
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 165
=================================================================
Total params: 4,901
Trainable params: 4,901
Non-trainable params: 0
```

รูปโครงสร้างโมเดล LSTM โดยใช้ parameter จากการค้นหา

3.3 save ผลลัพธ์จากการทำ Scan ลงบนไฟล์ .csv

```
cnn_best_params.to_csv(root_dir + '/cnn_best_params.csv')
lstm_best_params.to_csv(root_dir + '/lstm_best_params.csv')
```

3.4 ใช้โมเดลที่สอนจากพารามิเตอร์ที่ดีที่สุดมา predict ข้อมูล ชุด x_test สำหรับทั้ง 2 โมเดล

```
y_prediction = model.predict(X_test)
y_pred_single = [np.argmax(p) for p in y_prediction]
y_test_single=[np.argmax(p) for p in y_test]
```

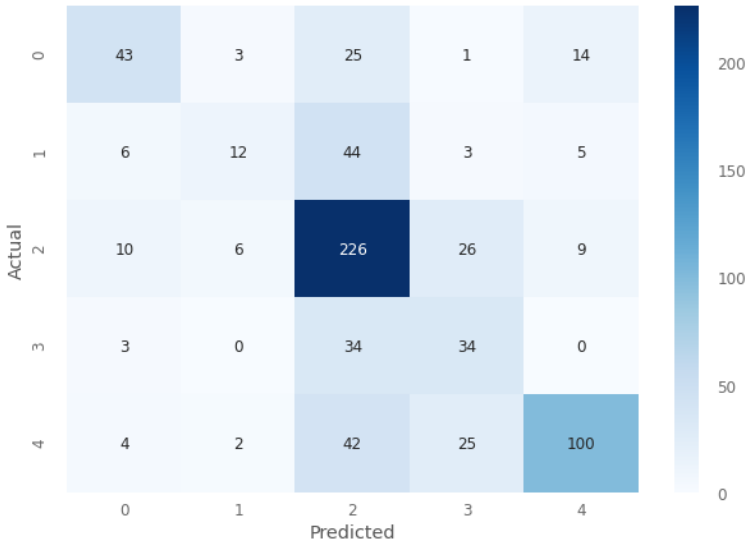
3.5 คำนวณค่าตัววัดประสิทธิภาพของการทำนายจากโมเดลที่ทดสอบด้วย y _test แสดงรูปภาพของ classification_report และ confusion_matrix

- คำนวณค่าตัววัดประสิทธิภาพของการทำนายจากโมเดล CNN ด้วย parameter ที่ดีที่สุด โดยใช้ฟังก์ชัน classification_report จาก sklearn เพื่อดู precision, recall, f1-score, accuracy เป็นต้น และแสดง confusion matrix โดยพบว่าโมเดลมีความแม่นยำร้อยละ 67 จากการทำนายได้ดังนี้

```
print(classification_report(y_test_single, y_pred_single))
conf_mat = confusion_matrix(y_test_single, y_pred_single)
plt.figure(figsize = (10, 7))
ax = sns.heatmap(conf_mat, annot=True, fmt="d", xticklabels='0 1 2 3 4'.split(),
 yticklabels='0 1 2 3 4'.split(), cmap="Blues")
bottom, top = ax.get_ylim()
plt.ylabel("Actual")
plt.xlabel("Predicted")
plt.show()
```

	precision	recall	f1-score	support
0	0.65	0.50	0.57	86
1	0.52	0.17	0.26	70
2	0.61	0.82	0.70	277
3	0.38	0.48	0.42	71
4	0.78	0.58	0.66	173
accuracy			0.61	677
macro avg	0.59	0.51	0.52	677
weighted avg	0.63	0.61	0.60	677

รูปการวัดประสิทธิโมเดล CNN ด้วย classification_report function



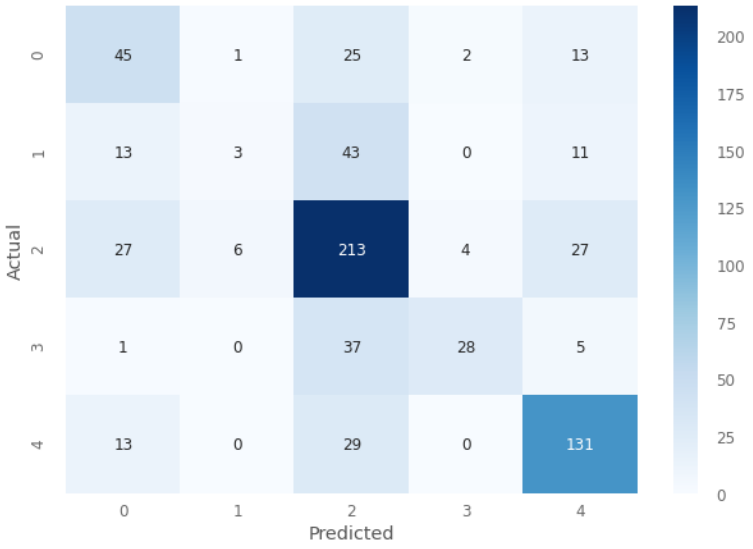
รูปการวัดประสิทธิโมเดล CNN ด้วย confusion matrix

- คำนวณค่าตัววัดประสิทธิภาพของการทำนายจากโมเดล LSTM ด้วย parameter ที่ดีที่สุด โดยใช้ฟังก์ชัน classification_report จาก sklearn เพื่อดู precision, recall, f1-score, accuracy เป็นต้น และแสดง confusion matrix โดยพบว่าโมเดลมีความแม่นยำร้อยละ 58 จากการทำนายได้ดังนี้

```
print(classification_report(y_test_single, y_pred_single))

conf_mat = confusion_matrix(y_test_single, y_pred_single)
plt.figure(figsize = (10, 7))
ax = sns.heatmap(conf_mat, annot=True, fmt="d", xticklabels='0 1 2 3 4'.split(),
 yticklabels='0 1 2 3 4'.split(), cmap="Blues")
bottom, top = ax.get_ylim()
plt.ylabel("Actual")
plt.xlabel("Predicted")
plt.show()
```

	precision	recall	f1-score	support
0	0.45	0.52	0.49	86
1	0.30	0.04	0.07	70
2	0.61	0.77	0.68	277
3	0.82	0.39	0.53	71
4	0.70	0.76	0.73	173
accuracy			0.62	677
macro avg	0.58	0.50	0.50	677
weighted avg	0.61	0.62	0.59	677

รูปการวัดประสิทธิโมเดล LSTM ด้วย classification_report function



รูปการวัดประสิทธิโมเดล CNN ด้วย confusion matrix