**เอกสารอธิบายการทดลองที่ 3 ตอนที่ 3**

**การทดลองการค้นหาพารามิเตอร์ที่ดีที่สุดสำหรับโมเดล**

**ตอนที่ 3 : การทดลองการค้นหาพารามิเตอร์ที่ดีที่สุดสำหรับโมเดล**

* 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='same'))

        model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu', padding='same'))

        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, metrics=['accuracy'])

        out = model.fit(X\_train, y\_train, batch\_size=params['batch\_size'], epochs=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, metrics=['accuracy'])

        out = model.fit( X\_train,y\_train, batch\_size=params['batch\_size'], validation\_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, experiment\_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

* 1. 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\_label', 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('=================================================')

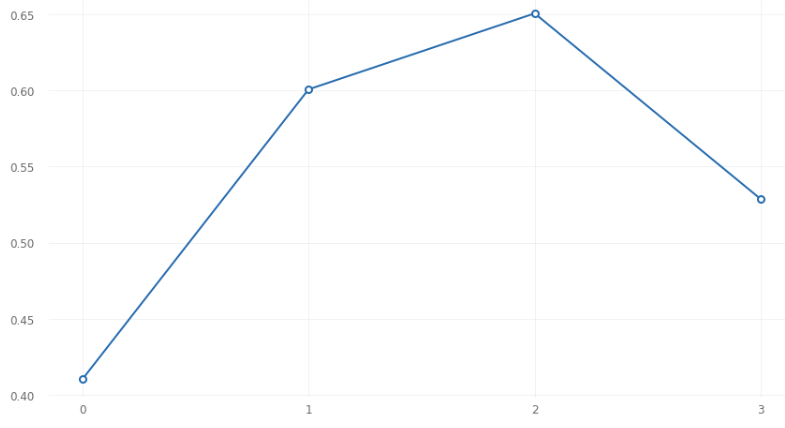
analyze\_object\_lstm = show\_search\_res(scan, lstm')



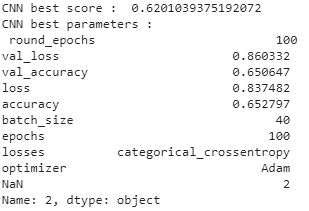
รูปพารามิเตอร์ที่ได้รับจากโมเดล CNN



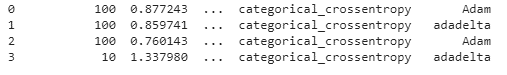
รูปคะแนนค่าแม่นยำเฉลี่ยของแต่ละ fold จากโมเดล CNN ด้านบน



รูปความแม่นยำของผลลัพธ์ในแต่ละรอบจากโมเดล CNN



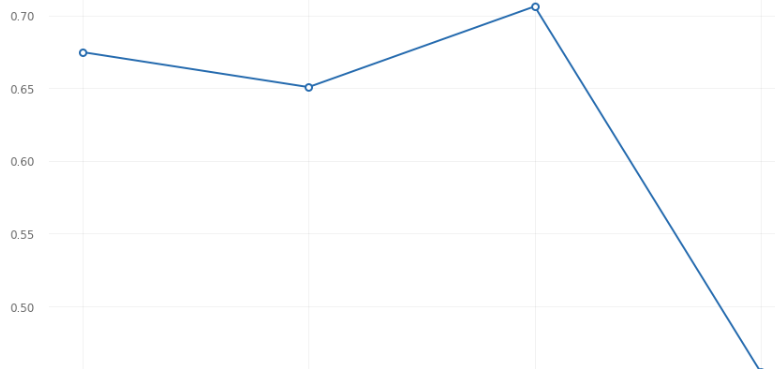
รูปค่าของ parameters ที่ดีที่สุดจากโมเดล CNN



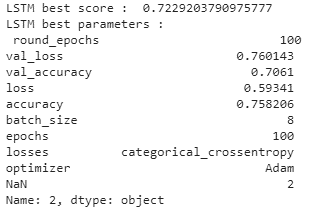
รูปพารามิเตอร์ที่ได้รับจากโมเดล LSTM



รูปคะแนนค่าแม่นยำเฉลี่ยของแต่ละ fold จากโมเดล LSTM ด้านบน



รูปความแม่นยำของผลลัพธ์ในแต่ละรอบจากโมเดล LSTM



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'])

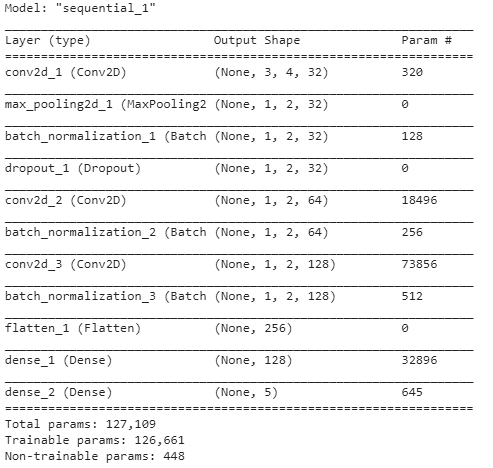
  return model

- 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'], validation\_data=(X\_valid, y\_valid), epochs=best\_params['epochs'], verbose=1)



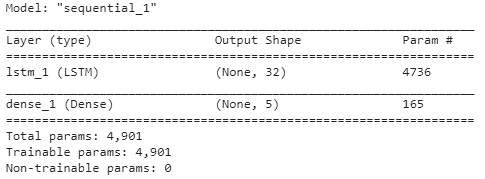
รูปโครงสร้างโมเดล CNN โดยใช้ parameter จากการค้นหา

- 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\_size'], validation\_data=(X\_2d\_valid, y\_valid), epochs=lstm\_best\_params['epochs'], verbose=1)



รูปโครงสร้างโมเดล LSTM โดยใช้ parameter จากการค้นหา

* 1. 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')

* 1. 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]

* 1. 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))

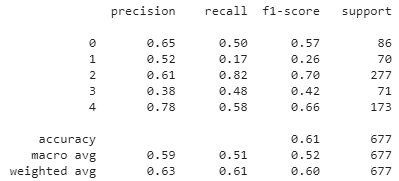
ax = sns.heatmap(conf\_mat, annot=True, fmt="d", xticklabels='0 1 2 3 4'.split(), yticklabels='0 1 2 3 4'.split(), cmap="Blues")

bottom, top = ax.get\_ylim()

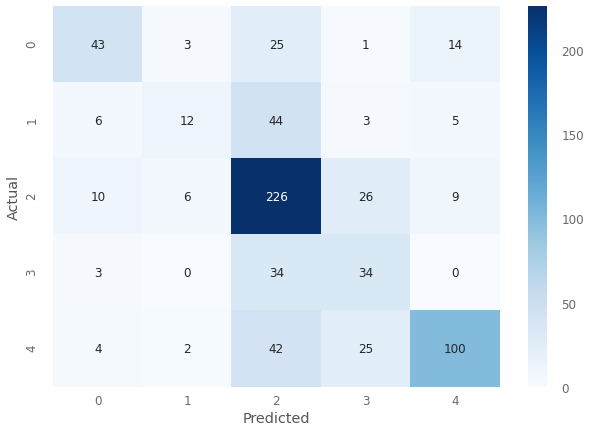
plt.ylabel("Actual")

plt.xlabel("Predicted")

plt.show()



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))

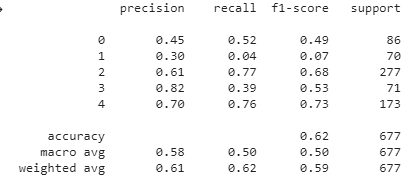
ax = sns.heatmap(conf\_mat, annot=True, fmt="d", xticklabels='0 1 2 3 4'.split(), yticklabels='0 1 2 3 4'.split(), cmap="Blues")

bottom, top = ax.get\_ylim()

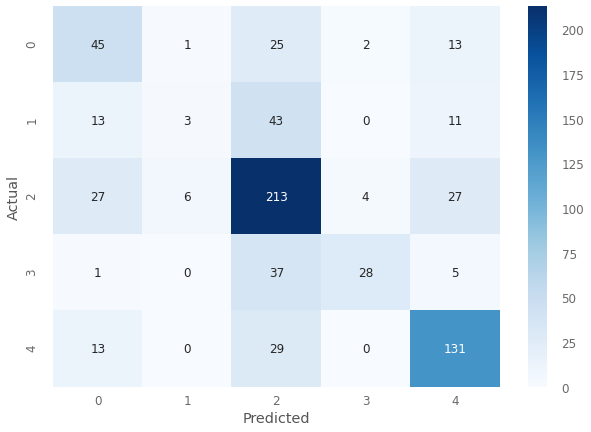
plt.ylabel("Actual")

plt.xlabel("Predicted")

plt.show()



รูปการวัดประสิทธิโมเดล LSTM ด้วย classification\_report function



รูปการวัดประสิทธิโมเดล CNN ด้วย confusion matrix