三軍總醫院北投分院統計及實驗設計課程之八

2021/7/13

ytai1123@gmail.com

使用方法:

1. 使用gmail帳號登入
2. 按"執行階段" -->"全部執行" 以執行全部內容, 若要個別執行可點選每格程式左方箭頭或按 Control + Enter 鍵執行。

```
##0-1
!git  clone  https://github.com/YuehMintTai/RPython.git
```

```
Cloning into 'RPython'...
remote: Enumerating objects: 168, done.
remote: Counting objects: 100% (168/168), done.
remote: Compressing objects: 100% (166/166), done.
remote: Total 168 (delta 92), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (168/168), 3.16 MiB | 14.91 MiB/s, done.
Resolving deltas: 100% (92/92), done.
```

```
##0-2
!pip  install  rpy2
```

```
Requirement already satisfied: rpy2 in /usr/local/lib/python3.7/dist-packages (3.4.5)
Requirement already satisfied: cffi>=1.10.0 in /usr/local/lib/python3.7/dist-packages (from r
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from rpy2) (20
Requirement already satisfied: tzlocal in /usr/local/lib/python3.7/dist-packages (from rpy2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from rpy2) (
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-packages (from cffi
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (fr
```

```
##0-3
%load_ext  rpy2.ipython
```

```
##8-0-0
import  tensorflow  as  tf
import  torch
tf.__version__          ##2.5.0
torch.__version__      ##1.9.0+cu102
```

```
'1.9.0+cu102'
```

```
##8-0-1  區分所有的資料為testing,  validating及training  sets
import  pandas  as  pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
import numpy as np

df=pd.read_csv('RPython/samples.csv')
df['sex']='男'
df.loc[df['性別']==2,'sex']='女'
x=df[['sex','網路成癮分數YDQ','家庭滿意度apgar','年齡']]
x=pd.get_dummies(data=x,drop_first=True)
y=df['自殺意念01'].astype(int).to_frame()
all_df=pd.concat([x,y],axis=1)
all_df['remainder']=all_df.index%6
all_df['group']='training'
all_df.loc[all_df['remainder']==0,'group']='testing'
all_df.loc[all_df['remainder']==1,'group']='validating'
all_df.loc[all_df['remainder']==2,'group']='validating'

all_df['年齡'].groupby(all_df['group']).mean()

all_df['groupNo']=0
all_df.loc[all_df['group']=='validating','groupNo']=1
all_df.loc[all_df['group']=='testing','groupNo']=2
all_df.tail(10)
```

	網路成癮分數YDQ	家庭滿意度apgar	年齡	sex_男	自殺意念01	remainder	group	groupNo
178	1	10.0	25.000000	1	0	4	training	0
179	8	0.0	29.000000	1	1	5	training	0
180	3	8.0	21.333082	0	1	0	testing	2
181	7	3.0	34.000000	1	0	1	validating	1
182	7	5.0	18.000000	1	0	2	validating	1
183	8	9.0	27.000000	1	0	3	training	0
184	5	5.0	27.000000	1	1	4	training	0
185	5	7.0	27.000000	1	0	5	training	0
186	2	0.0	21.000000	1	1	0	testing	2
187	0	0.0	25.000000	1	1	1	validating	1

```
##8-0-2 Chi-square and ANOVA
from sklearn.feature_selection import chi2
from sklearn.feature_selection import f_classif
print(pd.crosstab(all_df['group'],all_df['自殺意念01']))
##chi-square test確定 自殺意念01 沒有過度集中在某組....
Chi2=chi2(all_df['groupNo'].to_numpy().reshape(-1,1),all_df['自殺意念01'].to_numpy())
print('chi2={}, p={}'.format(round(Chi2[0].item(),3),round(Chi2[1].item(),3)))
##ANOVA確定年齡沒有過度集中在某組
print('='*30)
F=f_classif(all_df['年齡'].to_numpy().reshape(-1,1),all_df['group'].to_numpy())
print(all_df.groupby('group')['年齡'].mean())
print('ANOVA F={}, p={}'.format(round(F[0].item(),3),round(F[1].item(),3)))
```

```
print('ANOVA F={}, p={}'.format(round(F[0],3),round(F[1],3)))
```

```
自殺意念01        0   1
group
testing     23   9
training    77  16
validating  53  10
chi2=0.986, p=0.321
==============================
group
testing       21.385385
training      21.752499
validating    20.687376
Name: 年齡, dtype: float64
ANOVA F=1.126, p=0.327
```

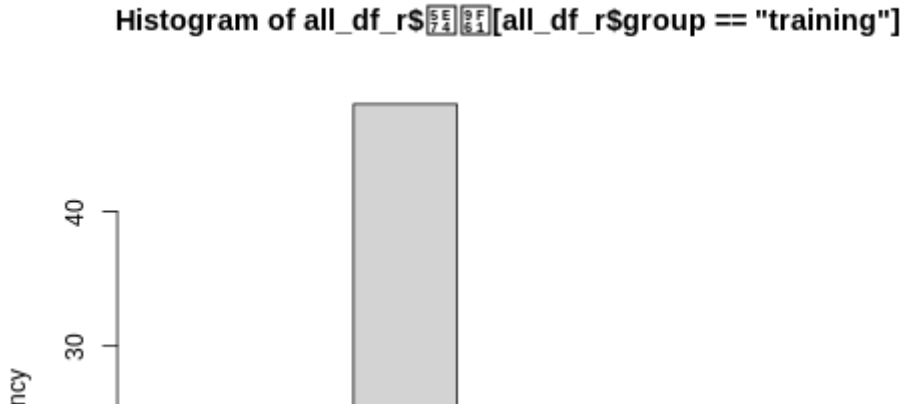
```
##8-0-3 區分 training, validation, testing sets..
y_test=all_df.loc[all_df['group']=='testing']['自殺意念01'].to_frame()
x_test=all_df.loc[all_df['group']=='testing'][['網路成癮分數YDQ','家庭滿意度apgar','年齡','sex_男']
y_vald=all_df.loc[all_df['group']=='validating']['自殺意念01'].to_frame()
x_vald=all_df.loc[all_df['group']=='validating'][['網路成癮分數YDQ','家庭滿意度apgar','年齡','sex_男
y_train=all_df.loc[all_df['group']=='training']['自殺意念01'].to_frame()
x_train=all_df.loc[all_df['group']=='training'][['網路成癮分數YDQ','家庭滿意度apgar','年齡','sex_男
print('cases number of train={}, of validation={}, of test={} '.format(len(x_train),len(x_v
```

```
cases number of train=93, of validation=63, of test=32
```

```
all_df.to_csv('all_df_r.csv')
```

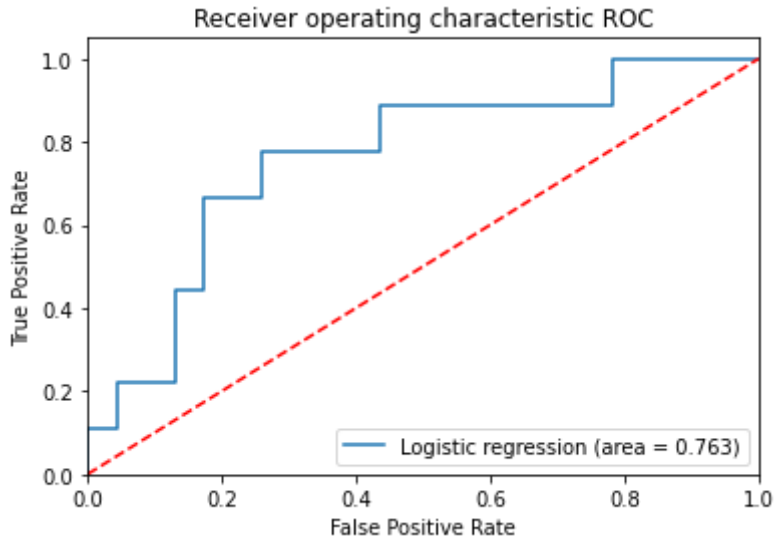
```
##8-0-4 Read Pands.DataFrame in R
%%R
all_df_r<-read.csv('all_df_r.csv')
hist(all_df_r$年齡[all_df_r$group=='training'])
##hist(all_df_r$年齡[all_df_r$group=='validating'])
##hist(all_df_r$年齡[all_df_r$group=='testing'])
formula<-'年齡~group'
ANOVA<-aov(all_df_r$網路成癮分數YDQ~all_df_r$groupNo)
ANOVA<-aov(all_df_r$家庭滿意度apgar~all_df_r$groupNo)
ANOVA<-aov(all_df_r$年齡~all_df_r$group)
summary(ANOVA)
```

```
                 Df Sum Sq Mean Sq F value Pr(>F)
all_df_r$group    2     43   21.36   1.126  0.327
Residuals       185   3510   18.97
```




```
##8-0-5使用sklearn Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
from matplotlib import pyplot as plt
model0=LogisticRegression()
result0=model0.fit(x_train,y_train)
預測機率0=result0.predict_proba(x_test)
AUC面積0=roc_auc_score(y_test,預測機率0[:,1])
fpr, tpr, thresholds = roc_curve(y_test,預測機率0[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic regression (area = %0.3f)' % AUC面積0)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic ROC')
plt.legend(loc="lower right")
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:760: DataConversic
  y = column_or_1d(y, warn=True)
```

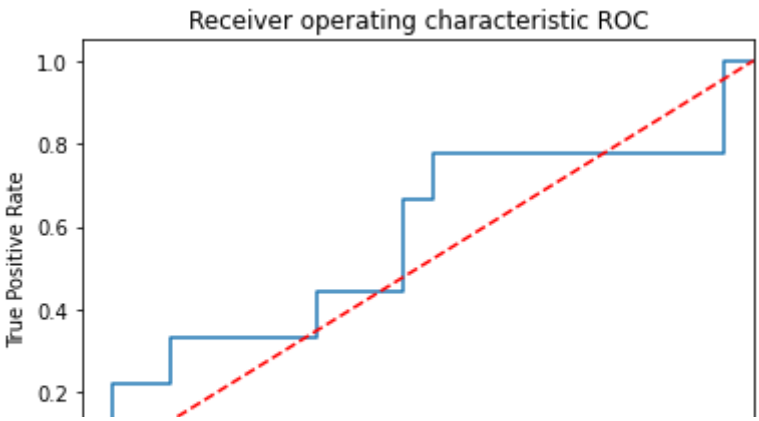



```
##8-1-1 使用Tensorflow Logistic regression ...
```

```
##8-1-1 使用tensorflow Logistic regression ...
import tensorflow as tf
from tensorflow.keras.layers import Dense
from tensorflow.keras import Sequential
from tensorflow.keras.metrics import AUC, Precision, Recall
model1=Sequential()
model1.add(Dense(units=1,input_dim=4,activation='sigmoid')) ###'softmax' is for multiclass out
model1.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy',AUC(),Precision(),Re
model1.fit(x_train, y_train, epochs=20,batch_size=10,validation_data=(x_vald,y_vald))


##8-1-2 計算accuracy...
score1=model1.evaluate(x_test,y_test)
print('{}-->{}'.format(model1.metrics_names[0],score1[0]))
print('{}-->{}'.format(model1.metrics_names[1],score1[1]))
print('{}-->{}'.format(model1.metrics_names[2],score1[2]))
print('{}-->{}'.format(model1.metrics_names[3],score1[3]))
print('{}-->{}'.format(model1.metrics_names[4],score1[4]))

print('='*30)
from sklearn.metrics import roc_auc_score
print('AUC for testing set:')
predicted_prob=model1.predict(x_test)
roc_auc_score(y_test,predicted_prob)
```

```
1/1 [==============================] - 0s 487ms/step - loss: 2.9203 - accuracy: 0.7188 - auc:
loss-->2.920323371887207
accuracy-->0.71875
auc-->0.5
precision-->0.0
recall-->0.0
==============================
AUC for testing set:
0.5652173913043478
```

```
##8-1-3 繪製ROC曲線圖...
from sklearn.metrics import roc_auc_score, roc_curve
from matplotlib import pyplot as plt
AUC面積=roc_auc_score(y_test, predicted_prob) ##0.6973180076628352
fpr,tpr,thresholds=roc_curve(y_test,predicted_prob)
plt.figure()
plt.plot(fpr,tpr,label='Logistic regression (area = %0.3f)'%AUC面積)
plt.plot([0,1],[0,1],'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic ROC')
plt.legend(loc="lower right")
plt.show()
```

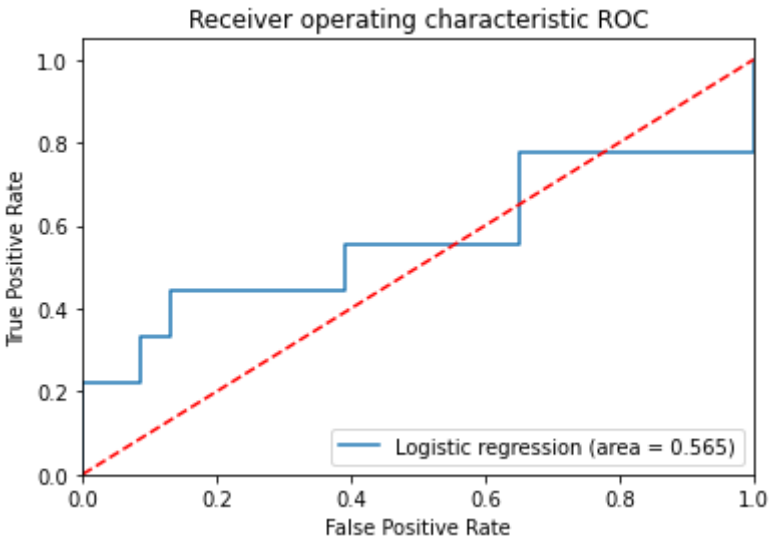



```
##8-2-1使用pytorch
import  torch
x_train_=torch.from_numpy(x_train.to_numpy().astype(np.float32))
x_vald_=torch.from_numpy(x_vald.to_numpy().astype(np.float32))
x_test_=torch.from_numpy(x_test.to_numpy().astype(np.float32))
y_train_=torch.from_numpy(y_train.to_numpy().astype(np.float32))
y_vald_=torch.from_numpy(y_vald.to_numpy().astype(np.float32))
y_test_=torch.from_numpy(y_test.to_numpy().astype(np.float32))


##8-2-2  建立  class
import  torch
class  Logistic_Reg_model(torch.nn.Module):
    def  __init__(self,no_input_features):
        super(Logistic_Reg_model,self).__init__()
        self.layer1=torch.nn.Linear(no_input_features,4)
        self.layer2=torch.nn.Linear(4,1)
    def  forward(self,x):
        y_predicted=self.layer1(x)
        y_predicted=torch.sigmoid(self.layer2(y_predicted))
        return  y_predicted


##8-2-3  訓練model
import  torch
樣本數,變項數=x_train_.shape
model2=Logistic_Reg_model(變項數)
criterion=torch.nn.BCELoss()
optimizer=torch.optim.SGD(model2.parameters(),lr=0.01)
number_of_epochs=20
for  epoch  in  range(number_of_epochs):
    y_prediction=model2(x_train_)
    loss=criterion(y_prediction,y_train_)
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
    if  (epoch+1)%10==0:
        print('epoch:',epoch+1,',  loss=',loss.item())
```

```
epoch: 10 ,  loss= 0.49723049998283386
epoch: 20 ,  loss= 0.4733884334564209
```

```
##8-2-4測試testing  group
```

```
import torch
with torch.no_grad():
    y_pred=model2(x_test_)
    y_pred_class=y_pred.round()
    accuracy=float(y_pred_class.eq(y_test_).sum())/float(y_test_.shape[0])
print(accuracy)
print(y_pred)


##8-2-5 計算AUC
from sklearn.metrics import roc_auc_score, roc_curve
from matplotlib import pyplot as plt
AUC面積=roc_auc_score(y_test, y_pred) ##0.609
fpr,tpr,thresholds=roc_curve(y_test,y_pred)
plt.figure()
plt.plot(fpr,tpr,label='Logistic regression (area = %0.3f)'%AUC面積)
plt.plot([0,1],[0,1],'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic ROC')
plt.legend(loc="lower right")
plt.show()
```




```
##8-3-1 Deep Learning Model...
import tensorflow as tf
import numpy as np
from tensorflow import keras
from tensorflow.keras.layers import Dense
model3=keras.Sequential()
model3.add(Dense(8,input_dim=4,activation='relu'))
model3.add(Dense(5,activation='relu'))
model3.add(Dense(1,activation='sigmoid'))
model3.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
dense_1 (Dense)              (None, 8)                 40
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 45
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 6
=================================================================
Total params: 91
Trainable params: 91
Non-trainable params: 0
_________________________________________________________________
```

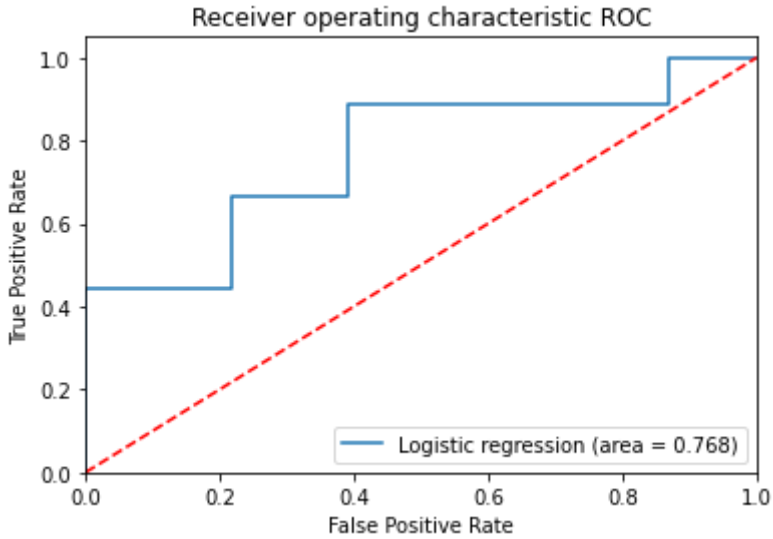
```
##8-3-2 Training model..
from tensorflow.keras.metrics import AUC, Precision, Recall
model3.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy',AUC(),Precision(),Re
model3.fit(x_train, y_train,epochs=200, batch_size=10,validation_data=(x_vald,y_vald))
```

```
##8-3-3 計算accuracy...
score1=model3.evaluate(x_test,y_test)
print('{}-->{}'.format(model3.metrics_names[0],score1[0]))
print('{}-->{}'.format(model3.metrics_names[1],score1[1]))
print('{}-->{}'.format(model3.metrics_names[2],score1[2]))
print('{}-->{}'.format(model3.metrics_names[3],score1[3]))
print('{}-->{}'.format(model3.metrics_names[4],score1[4]))

print('='*30)
from sklearn.metrics import roc_auc_score
print('AUC for testing set:')
predicted_prob=model3.predict(x_test)
roc_auc_score(y_test,predicted_prob)
```

```
1/1 [==============================] - 0s 470ms/step - loss: 0.5864 - accuracy: 0.7812 - auc_
loss-->0.5864367485046387
accuracy-->0.78125
auc_1-->0.7777777910232544
precision_1-->1.0
recall_1-->0.2222222238779068
==============================
AUC for testing set:
0.7681159420289855
```

```
##8-3-4 計算AUC
from sklearn.metrics import roc_auc_score, roc_curve
from matplotlib import pyplot as plt
AUC面積1=roc_auc_score(y_test, predicted_prob) ##0.609
fpr,tpr,thresholds=roc_curve(y_test,predicted_prob)
plt.figure()
plt.plot(fpr,tpr,label='Logistic regression (area = %0.3f)'%AUC面積1)
plt.plot([0,1],[0,1],'r--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic ROC')
```

```
plt.legend(loc="lower  right")
plt.show()
```




```
##8-4-1  save  model
model3.save('./RPython/Model3',save_format='tf')
```

```
INFO:tensorflow:Assets written to: ./RPython/Model3/assets
```

```
##8-4-2
y_test[y_test['自殺意念01']==1]###找出y_test中自殺意念為1的index
x_test.loc[186]                 ##找出index=186的四個X值...
```

```
網路成癮分數YDQ       2.0
家庭滿意度apgar      0.0
年齡              21.0
sex_男            1.0
Name: 186, dtype: float64
```

```
##8-4-3  load  &  test  model
model4=tf.keras.models.load_model('./RPython/Model3')
newCase={'網路成癮分數YDQ':[2],'家庭滿意度apgar':[0],'年齡':[21],'sex_男':[1]}
newCase=pd.DataFrame(newCase)
model4.predict(newCase)[0][0]
```

```
0.5434687
```

```
##8-5-1  Install  falsk-ngrok
!pip  install  flask-ngrok
```

```
##建立Flask網頁...
from  flask_ngrok  import  run_with_ngrok
from  flask  import  Flask,  render_template,request
import  pandas  as  pd
import  tensorflow  as  tf
app=Flask(__name__,template_folder='./RPython')
run_with_ngrok(app)
```

```
@app.route('/')
def  index():
    title='資料表單'
    return  render_template('index.html',title=title)
@app.route('/predict',methods=['POST'])
def  predict():
    title='預測結果'
    YDQ=request.form.get('YDQ')
    APGAR=request.form.get('APGAR')
    AGE=request.form.get('AGE')
    SEX=request.form.get('SEX')
    model4=tf.keras.models.load_model('./RPython/Model3')
    newCase={'網路成癮分數YDQ':[int(YDQ)],'家庭滿意度apgar':[int(APGAR)],'年齡':[int(AGE)],'sex_男'
    newCase=pd.DataFrame(newCase)
    PREDICT=(round(model4.predict(newCase)[0][0]*100,3))
    return  render_template('predict.html',title=title,YDQ=YDQ,APGAR=APGAR,AGE=AGE,SEX=SEX,PREDICT=
app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Running on http://a627953012d4.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040
127.0.0.1 - - [17/Jul/2021 05:00:41] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [17/Jul/2021 05:00:41] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [17/Jul/2021 05:00:43] "POST /predict HTTP/1.1" 200 -
```

```
##以下內容與課程無關...請自動忽略.....

from  flask_ngrok  import  run_with_ngrok
from  flask  import  Flask

app=Flask(__name__)
run_with_ngrok(app)
@app.route("/")
def  index():
    return  "<h1>這是首頁</h1>"
@app.route("/details")
def  details():
    return  "<h1>這是details頁</h1>"
@app.route("/test")
def  test():
    return  "<h1>這是test頁!</h1>"


app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

```
 * Running on http://be945d3d4f66.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040
```

✓ 1 分鐘 26 秒 完成時間：下午1:01