# Python技術應用班

人工智慧/機器學習/個人學習小成果

製作者:江慶宸

信箱: ccrain78990@gmail.com



# 目錄

**CONTENTS** 

## 

The user can demonstrate on a projector or computer or print the it into a film to be used in a wider field

## 2 機器學習

The user can demonstrate on a projector or computer or print the it into a film to be used in a wider field

## 3 MLP多層感知機

The user can demonstrate on a projector or computer or print the it into a film to be used in a wider field

## 4 總結

The user can demonstrate on a projector or computer or print the it into a film to be used in a wider field



### 主要目的

以學習人工智慧領域為目的 嘗試做機器學習、MLP訓練資料 將資料做分類並預測結果(肥胖水平)

主要使用: **?** python™ 3.X

- Tkinter
- Pandas
- NumpySeaborn
- ScikitLearn
- Tensorflow

### 資料來源

### 資料集名稱:

根據飲食習慣和身體狀況估計肥胖程度

#### 資料集摘要:

該數據集包括根據墨西哥、秘魯和哥倫比亞等國的飲食 習慣和身體狀況估計個人肥胖水平的數據。





資料集來源: <a href="https://reurl.cc/gWW8o4">https://reurl.cc/gWW8o4</a>(或掃描上面QRcode)

變數數量

資料筆數

2019 08-27 資料更新日期 執行目標



### 資料變數

特徵值

#### 與飲食習慣相關

- 1.頻繁食用高熱量食物 (FAVC)
- 2.食用蔬菜的頻率 (FCVC)
- 3.主餐次數 (NCP)
- 4.兩餐之間的食物消耗量 (CAEC)
- 5.每日飲水量 (CH20) )
- 6.酒精消耗量 (CALC)

特徵值

### 與身體狀況相關

- 1.卡路里消耗監測 (SCC)
- 2.身體活動頻率 (FAF)
- 3.使用技術設備的時間 (TUE)
- 4.使用的交通工具 (MTRANS)

特徵值

### 其他變量

- 1.性別
- 2.年齡
- 3.身高
- 4.體重

標籤

類變量 (肥胖水平)

體重不足

體重正常

超重級別।

超重級別Ⅱ

肥胖類型 |

肥胖類型 ||

肥胖類型 III

詳細數據說明:



△ 之後內容將藉由1、2、3變數內容去預測4的肥胖水平

△ 註:身高、體重會影響預測,運算時會略過不處理

資料内容 △底下運用pandas顯示簡易資料,僅供參考, △主要重點為機器學習,故無做其他特殊使用

### Unnamed: 0 Gender Age ... CALC\_Code MTRANS\_Code NObeyesdad\_Code 0 Female 21.0 ... 3 1 Female 21.0 ... 2 2 Male 23.0 ... 1 Male 27.0 ... 1 Male 22.0 ... 2

← (左圖)為簡單的資料集內容 以及將資料類別化, 方便後續處理

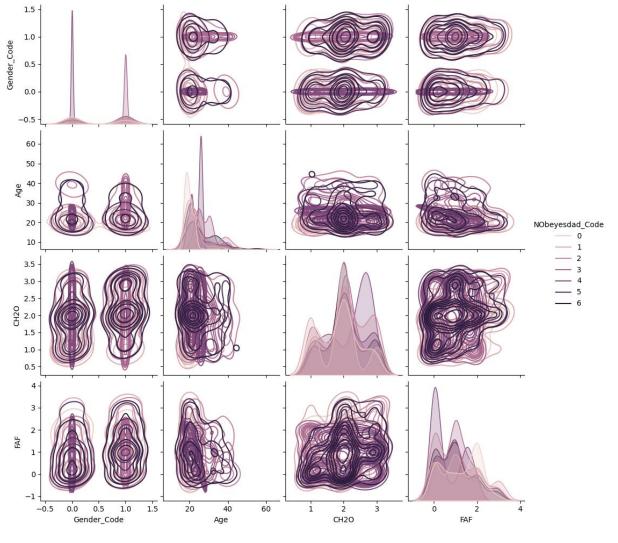
#### print(df.describe()) #統計描述#################

|       | Unnamed: 0 | Age         | MTRANS_Code | NObeyesdad_Code |
|-------|------------|-------------|-------------|-----------------|
| count | 2111.00000 | 2111.000000 | 2111.000000 | 2111.000000     |
| mean  | 1055.00000 | 24.312600   | 2.365230    | 3.015632        |
| std   | 609.53753  | 6.345968    | 1.261423    | 1.952090        |
| min   | 0.00000    | 14.000000   | 0.000000    | 0.000000        |
| 25%   | 527.50000  | 19.947192   | 3.000000    | 1.000000        |
| 50%   | 1055.00000 | 22.777890   | 3.000000    | 3.000000        |
| 75%   | 1582.50000 | 26.000000   | 3.000000    | 5.000000        |
| max   | 2110.00000 | 61.000000   | 4.000000    | 6.000000        |

← (左圖)為簡單的敘述統計 由表大約得知,此資料集 年齡遍布14~61歲之間 平均年齡24歲...等

### Seaborn △簡單看資料整體分布狀況



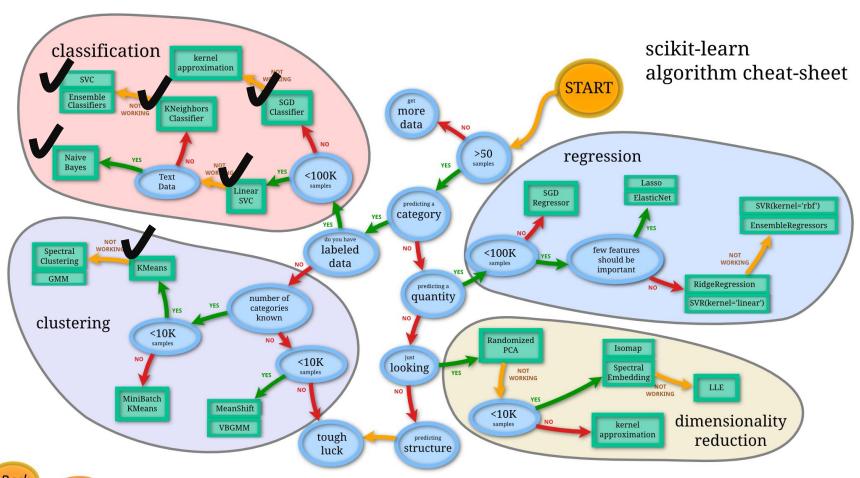


△左圖為整體變數,上圖取局部變數顯示部份狀況 △因整體資料量大,故有上圖供參考



### 關於演算法的挑選

△註1:接下來的頁面以局部程式碼以及結果為主,並無太多理論東西。







△圖片來源: ScikitLearn官方網站

△註2:根據資料處理目標、筆數選擇,外加常見幾個做練習、比較

### KNN(K-近鄰) 程式碼

```
from sklearn.neighbors import KNeighborsClassifier
#
knn = KNeighborsClassifier(7)
knn.fit(X_train, y_train)
print('KNN準確率: %.2f' % knn.score(X_test, y_test))
```

### 結果

```
*********
```

KNN準確率: 0.73

#### △ 以上Code皆只貼出重點部分,並非完整程式碼。

### K-Means(K-平均) 程式碼

```
from sklearn.cluster import KMeans
from sklearn import metrics

#
kmeans = KMeans(n_clusters = 7)
kmeans.fit(X_train)
y_predict=kmeans.predict(X_train)
score = metrics.accuracy_score(y_test_kmeans.predict(X_test))
print('K-means準確率:{0:f}'.format(score))
```

### 結果

## 決策樹 程式碼

```
from sklearn import tree
#
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train_xy_train)
print('決策樹準確率: %.2f' % clf.score(X_test, y_test))
```

## 結果

決策樹準確率: **0.73** 

## 隨機森林 程式碼

```
from sklearn.ensemble import RandomForestClassifier

#

RForest = RandomForestClassifier(n_estimators=100, max_depth=10_random_state=2)

RForest.fit(X_train, y_train)

print('随機森林準確率: %.2f' % RForest.score(X_test, y_test))
```

### 結果

```
********
```

隨機森林準確率: 0.83

△ 以上Code皆只貼出重點部分,並非完整程式碼。

### 具氏分类直器 △ 以下Code皆只貼出重點部分,並非完整程式碼。

### 程式碼

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
predicted = model.predict(X_test)
model.score(X_test_y_test)
print("貝氏分類器準確率", model.score(X_test, y_test))
```

### 結果

```
*********===貝氏分類器===*******
貝氏分類器準確率 0.5566037735849056
```

### SVM支援向量機 △ 以下Code皆只貼出重點部分,並非完整程式碼。

### 程式碼

```
from sklearn import svm
regr = svm.SVC()
regr.fit(X_train, y_train)
print('SVM--SVC準確率: %.2f' % regr.score(X_test, y_test))
```

```
from sklearn import svm
clf = svm.NuSVC(gamma='auto')
clf.fit(X_train, y_train)
print('SVM--Non-linearSVC準確率: %.2f' % clf.score(X_test, y_test))
```

```
from sklearn import svm
lin_clf = svm.LinearSVC()
lin_clf.fit(X_train, y_train)
print('SVM--linearSVC準確率: %.2f' % lin_clf.score(X_test, y_test))
```

### 結果

```
*********
SVM--SVC準確率: 0.46
```

```
*********==SVM--Non-linearSVC==****
SVM--Non-linearSVC準確率: 0.69
```

```
********===SVM--linearSVC==*****
SVM--linearSVC準確率: 0.55
```

### SGD隨機梯度下降 △ 以下Code皆只貼出重點部分,並非完整程式碼。

### 程式碼

```
from sklearn.linear_model import SGDClassifier
clf = SGDClassifier(loss="log", penalty="l1", max_iter=5)
clf.fit(X_train, y_train)
print('SGD準確率: %.2f' % clf.score(X_test, y_test))
```

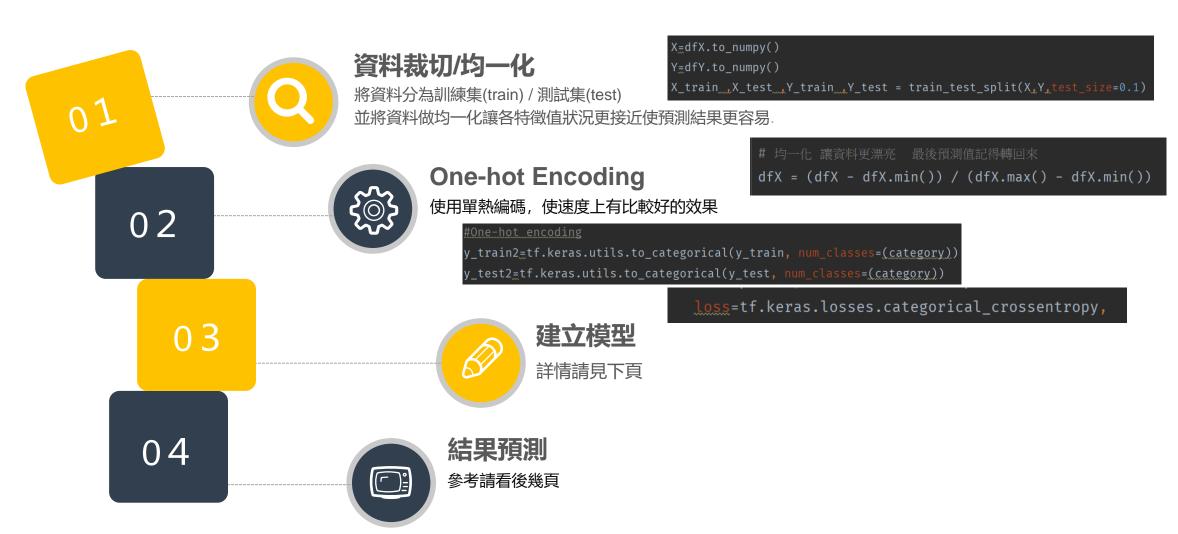
\*\*\*\*\*\*\*\*\*

SGD準確率: 0.48



### 處理MLP小過程

△ 簡單列出執行過程中重要的幾個小細處理&步驟&局部程式碼



### MLP程式片段/訓練結果

- △ 以下 C o d e 皆只貼出重點部分,並非完整程式碼。
- 〉 epochs、神經元參數沒有調很高,主要為簡單小訓練,其結果也是不錯的

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(units=200,
    activation=tf.nn.relu,
    input dim=dim))
model.add(tf.keras.layers.Dense(units=80*t,
    activation=tf.nn.relu_))
model.add(tf.keras.layers.Dense(units=100*t,
    activation=tf.nn.relu_))
model.add(tf.keras.layers.Dense(units=category,
    activation=tf.nn.softmax ))
model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),
    loss=tf.keras.losses.categorical_crossentropy,
    metrics=['accuracy'])
model.fit(x_train, y_train2,
          epochs=300*t,
          batch size=64)
```

dim=14
category=7
t=2

△ 變數數量

△ 分類數

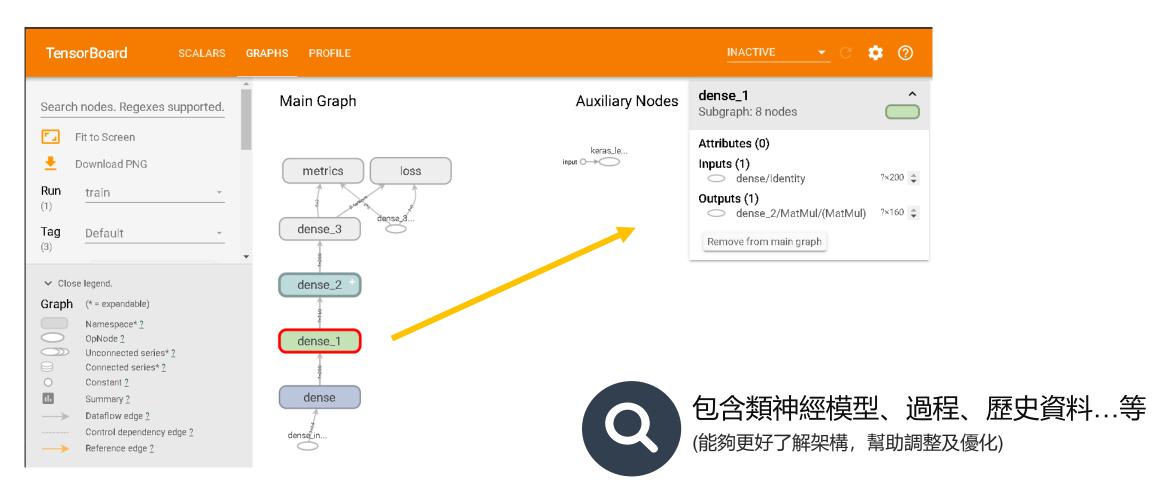
score: [0.09500226759975601, 0.990566]

↑↑↑ 經過訓練後,最後結果 ↑↑↑ (有多訓練幾次,挑選不錯的時候截圖供參考)



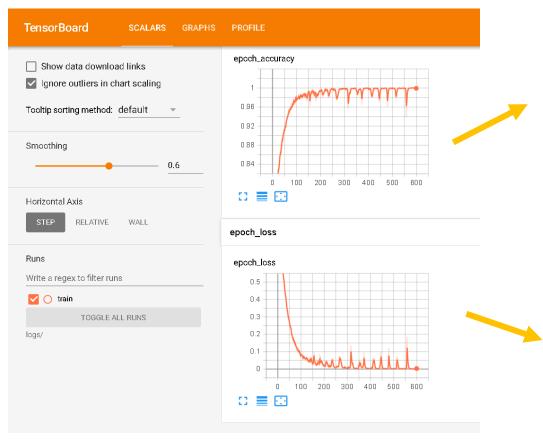
🗅 訓練過程,可參考左邊連結。

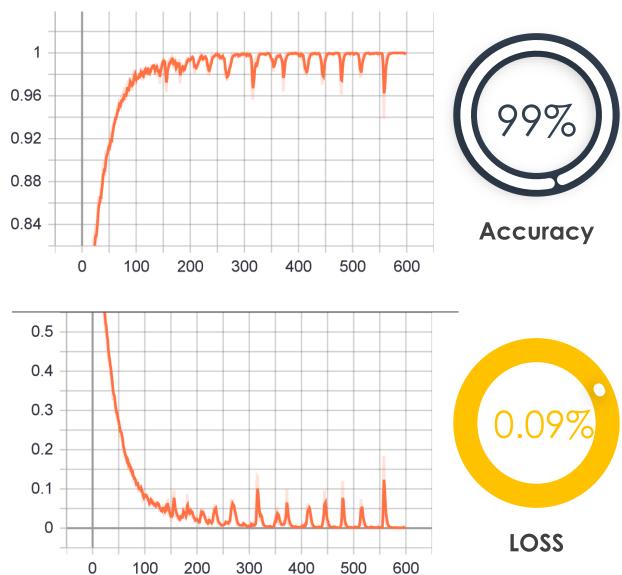
### TensorBoard-1



### TensorBoard-2

#### △ 模型請參考前頁設定





### 結果運用

△ 只要將訓練好的模型與權重運用,結合tknter呈現,讓使用者輸入資料集就可以直接預測

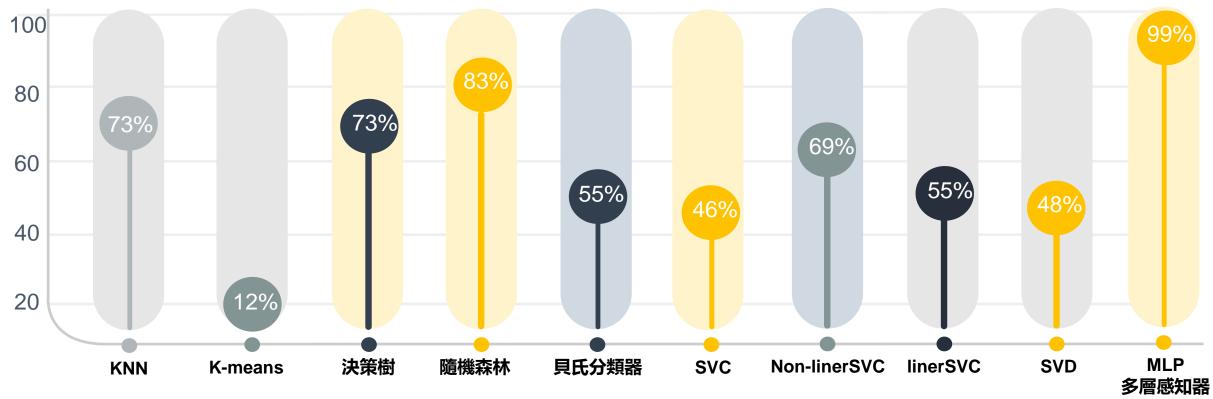




### 綜合整理

△ 以下結果,皆出自於同一筆資料,請參考前幾張運行結果





- △ MLP的效果較好,經過更好的調整更可以到達100%準確率
- △ 其他演算法也有其價值,不過準確率較不理想

### 結語



嘗試幾個學習方法去訓練資料 如何挑選合適演算法也是一個問題 也要懂得資料內容、將資料預處理 2

預測結果方面 透過好的演算法不斷訓練 準確率可以達到100% 對未來使用上有更大的幫助

其他還有更多演算法 也可以預測數字類型資料(ex房價 或是CNN做圖片、影像辨識

運用方面,僅為簡單的一個辦法 將結果結合圖形化介面,使其可視化 可以用在很多領域去預測不同東西



### 结束语/小心得

以上皆為本人關於機器學習上實作之成果小展示。

未來人工智慧、機器學習依然是很重要的學問, 透過學習、訓練資料去判斷、分析人類的行為或未來 趨勢等,以上所有內容主要運用python撰寫,除了練 習自己的程式設計能力,也嘗試了解現在夯的領域,

另外,python的使用上也很廣,除了人工智慧以外,在資料科學上(資料處理、清洗)上也很方便用,像是爬蟲相關技術等,這些都在這段時間有學習到,期待未來自己有更多運用,程式能力也更好!

# 感謝觀看

2021-03-04~2021-06-09