108 智慧化企業整合

只甜你口，不甜你尿
-糖尿病預測篩檢服務

第7組
施美全 李岱諭 周郁淇 許家銘

OUTLINE

01. SCENARIO

02. DATA PREPROCESSING

03. MODEL

04. ANALYSIS

05. CONCLUSION

01
SCENARIO

PROBLEM DEFINITION

Scenario
Background

1 重視飲食與健康

2 過分攝取精緻甜食

3 民眾難以自行判斷

和知名甜點店與在地醫院合作

提供糖尿病篩檢服務

Scenario

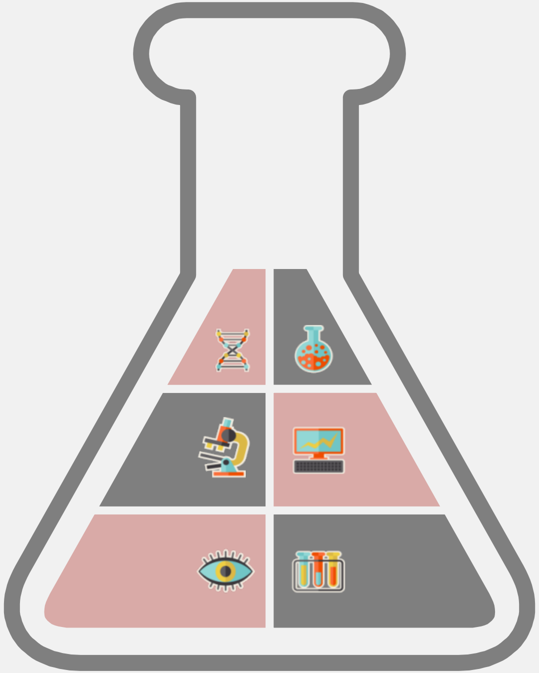
Problem Definition – 5W1H

藉由過往糖尿病患者的資料建立模型，以預測其潛在風險與罹病機率

W **WHAT**
糖尿病的篩檢服務

W **WHEN**
疾病發生前與發生時

W **WHO**
重視自身健康狀況的民眾&高風險族群



WHERE **W**
服務中心與網路

WHY **W**
糖尿病已危害民眾健康，且難以自行判斷

HOW **H**
資料分析、深度學習

02

DATA-PREPROCESSING

Data-Preprocessing

Data Resource

- 來自於Kaggle，數據建模和數據分析競賽平台
- 企業和研究者上發布數據
- 統計學者和數據挖掘專家產生最好的模型

Data-Preprocessing

Data Resource

- 由“National Institute of Diabetes and Digestive and Kidney Diseases”所蒐集並作為糖尿病數據庫

NIDDK

NATIONAL INSTITUTE OF
DIABETES AND DIGESTIVE
AND KIDNEY DISEASES

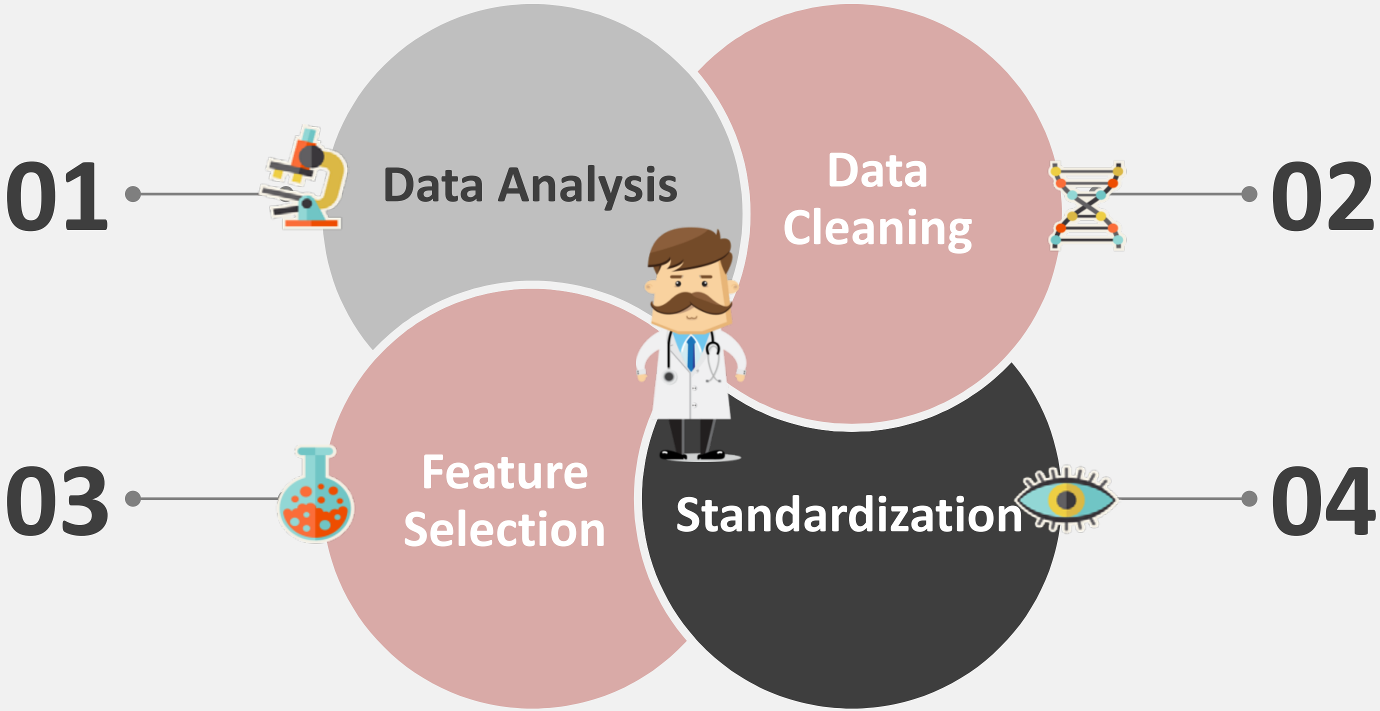
Data-Preprocessing

Data Meaning

特徵	說明
Pregnancies	懷孕週數
Glucose	葡萄糖含量
Blood Pressure	血壓
Skin Thickness	皮褶厚度(全身脂肪含量)
Insulin	血清胰島素
BMI	Body Mass Index
Diabetes Pedigree Function	糖尿病家族函數
Age	病患年齡資料
Outcome	病患是否患有糖尿病

Data-Preprocessing

Summary



Data-Preprocessing

Data Analysis

➢導入資料庫，並利用pandas的head()方法來檢查數據

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
diabetes = pd.read_csv('diabetes2.csv')
diabetes.columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

```
diabetes.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Data-Preprocessing

Data Analysis

- 獲得資料的維度、Outcome資訊
  - 得知此數據共有768筆資料與九個欄位

```
print("Diabetes data set dimensions : {}".format(diabetes.shape))
diabetes.groupby('Outcome').size()
```

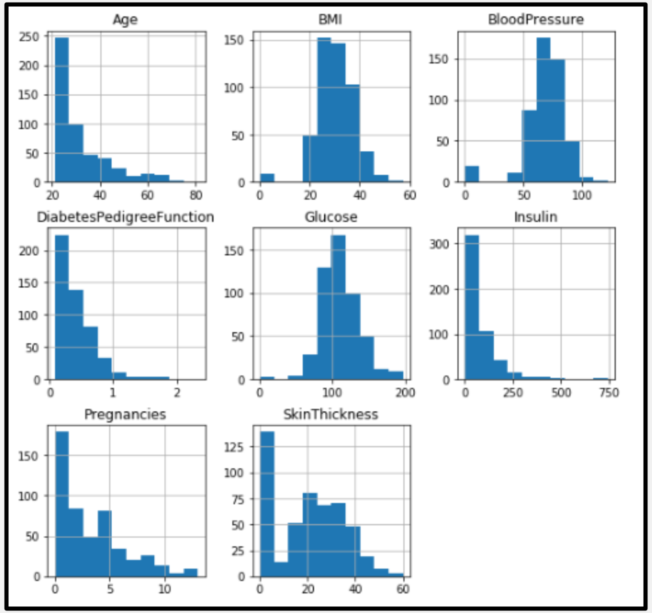
```
Diabetes data set dimensions : (768, 9)
Outcome
0     500
1     268
dtype: int64
```

Data-Preprocessing

Data Analysis

- ➢使用pandas的groupby可視化工具
  - 幫助進行資料分組，並得到各欄位之資料分布

```
diabetes.groupby('Outcome').hist(figsize=(9, 9))
```



Data-Preprocessing

Data Cleaning

- 數據清洗需考量一下幾點：
  - (1)是否有重複之資料
  - (2)錯誤標示的資料
  - (3)缺失或空的資料值
  - (4)異常值

- ✓ 數據來自於“National Institute of Diabetes and Digestive and Kidney Diseases”標準資料庫
- ✓ 故檢查後(1)(2)為已處理過，我們此次將針對(3)(4)來做清洗

Data-Preprocessing
Data Cleaning

- ➢ 針對第三點**是否有缺失或空**的資料值
- 使用函數isnull()和isna()檢查資料是否有空值，回傳boolean值

```
diabetes.isnull().sum()
diabetes.isna().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Data-Preprocessing

Data Cleaning

➢ 透過前面所顯示之直方圖，發現數據擁有異常值，故我們需要透過程式進一步分析

✓ 血壓（Blood pressure）

```
print("Total : ", diabetes[diabetes.BloodPressure == 0].shape[0])
Total :  35
print(diabetes[diabetes.BloodPressure == 0].groupby('Outcome')['Age'].count())

Total :  35
Outcome
0     19
1     16
Name: Age, dtype: int64
```

Data-Preprocessing

Data Cleaning

✓ 血糖（Glucose）

```
print("Total : ", diabetes[diabetes.Glucose == 0].shape[0])
Total :  5
print(diabetes[diabetes.Glucose == 0].groupby('Outcome')['Age'].count())
Total :  5
```

```
Total :  5
Outcome
0    3
1    2
Name: Age, dtype: int64
```

✓ BMI

```
print("Total : ", diabetes[diabetes.BMI == 0].shape[0])
Total :  11
print(diabetes[diabetes.BMI == 0].groupby('Outcome')['Age'].count())
```

```
Total :  11
Outcome
0    9
1    2
Name: Age, dtype: int64
```

Data-Preprocessing

Data Cleaning

- 由於有**異常值**之數據，故採取”移除異常值”
- 將其進行移除，並將其更新新的數據**(diabetes_mod)**
  - 得到新的資料筆數為**724**筆資料，將以此資料量進行**train**與**test**模型

```
diabetes_mod = diabetes[(diabetes.BloodPressure != 0) & (diabetes.BMI != 0)
&(diabetes.Glucose != 0)]
print(diabetes_mod.shape)
```

```
(724, 9)
```

Data-Preprocessing

Feature Selection

- 將資料轉換成特徵，可幫助建模與更高的準確率。
- 採用資料裡所有的特徵，並將其設為X，而預測結果設為y。

```
#選取特徵值
feature_names = [ 'Pregnancies','Glucose', 'BloodPressure',
                 'SkinThickness', 'Insulin', 'DiabetesPedigreeFunction','Age']
X = diabetes_mod[feature_names]
y = diabetes_mod.Outcome
```

Data-Preprocessing

Standardization

- 我們將其每個特徵數據轉為均值為0，標準差為1
- 避免某一特徵在目標函數占主導地位，而使其他特徵被淹沒

```python
#分訓練組與測試組
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = diabetes_mod.Outcome, random_state=66)
```

```python
#標準化re-scale：整理使每一個特徵維度均值為0,變異數為1
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
```

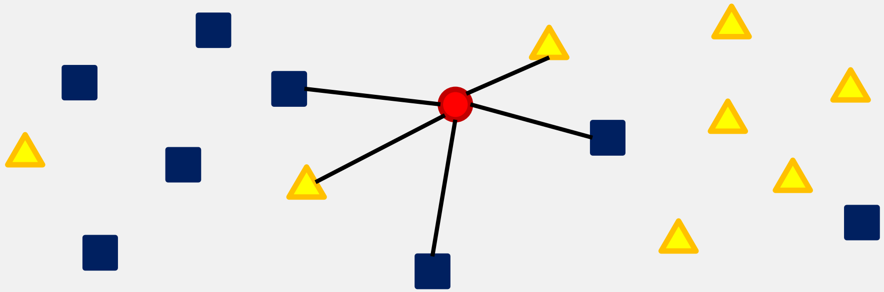
03
MODEL STRUCTURE

INTRODUCTION + SUMMARY

Model Structure

K-NN Architecture

- K-Nearest Neighbors最近鄰居法分類



```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=19)
knn.fit(X_train, y_train)
print('Train accuracy of kNN: {:.3f}'.format(knn.score(X_train, y_train)))
print('Test accuracy of kNN: {:.3f}'.format(knn.score(X_test, y_test)))
```

調整kNN的鄰居數

```
Train accuracy of kNN: 0.750
Test accuracy of kNN: 0.812
```

Model Structure

NN Architecture

- 使用Sequential()建置的Neural Network

```python
from keras.layers import Dropout

model = Sequential()
model.add(Dense(500, input_dim=7, activation='sigmoid'))
model.add(Dropout(0.1))
model.add(Dense(100, activation='sigmoid'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
model.fit(x_train,y_train, epochs=1000, batch_size=70, validation_data=(x_test, y_test))
```

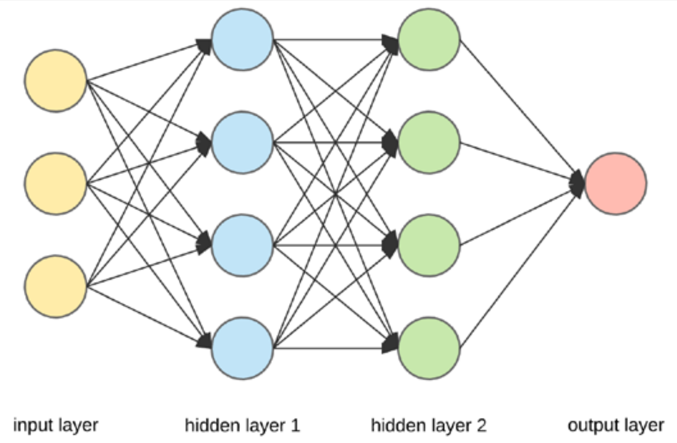
Model Structure

MLP Architecture

- Multilayer Perceptron多層感知器

```python
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(64,64,64),
                    activation='logistic',
                    solver='adam',
                    batch_size='auto',
                    learning_rate='constant',
                    learning_rate_init=0.001,
                    max_iter=1000,
                    random_state=0)
mlp.fit(X_train, y_train)
print("Train accuracy of MLP: {:.3f}".format(mlp.score(X_train, y_train)))
print("Test accuracy of MLP: {:.3f}".format(mlp.score(X_test, y_test)))
```

```
Train accuracy of MLP: 0.823
Test accuracy of MLP: 0.796
```



Model Structure

Model Summary

	kNN	NN	MLP
中文名稱	最近鄰居法	類神經網路	多層感知機
學習程度	machine learning	deep learning	deep learning
套件	sklearn	keras	sklearn
調整空間	少	多	中
調整速度	快	慢	中

04 ANALYSIS

IMPROVEMENT + BRAINSTORMING

Analysis

Improvement

- Data improvement
  - 篩選擾亂因子：Pregnancies

	original data	processed data	improvement
kNN	0.779	0.785	**+0.6%**
NN	0.646	0.669	**+2.3%**
MLP	0.702	0.74	**+3.8%**

- Model Training
  - 模型皆以刪除Pregnancies欄位之資料訓練
  - 調整kNN、NN、MLP參數

Analysis

K-NN Training

- 參數：n_neighbors

```
training_accuracy = []
test_accuracy = []

neighbors_settings = range(1, 50)
for n_neighbors in neighbors_settings:
    knn = KNeighborsClassifier(n_neighbors=n_neighbors)
    knn.fit(X_train, y_train)
    training_accuracy.append(knn.score(X_train, y_train))
    test_accuracy.append(knn.score(X_test, y_test))
```

用迴圈跑鄰居數1~50，看neighbor在哪個數值時準確度較高。

```
plt.plot(neighbors_settings, training_accuracy, label="training accuracy")
plt.plot(neighbors_settings, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend()
plt.savefig('knn_compare_model')
```

作圖

Analysis
K-NN Training

- n_neighbors為19時test data準確度最高

model	n_neighbors	accuracy
original	8	0.785
improve	19	**0.818**

Analysis

NN Training

- Basic version(test accuracy = 0.657)

```
classifier = Sequential()
classifier.add(Dense(input_dim = d, output_dim = 8, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 16, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'relu'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

- 3rd layer activation function = sigmoid(test accuracy = 0.669)

```
#設定NN模型
classifier = Sequential()
classifier.add(Dense(input_dim = d, output_dim = 8, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 16, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Analysis

NN Training

- NN+ model(test accuracy = 0.708)

```
df_label = dataframe['Outcome']
df_features = dataframe.drop('Outcome', 1)
print(df_label.head())
df_features.head()

label = []
for lab in df_label:
    if lab == 1:
        label.append([1, 0])  # class 1
    elif lab == 0:
        label.append([0, 1])  # class 0
```

```
activation='sigmoid')
sigmoid'))
ftmax'))
error', optimizer='adam', metrics=['accuracy'])
=1000, batch_size=70, validation_data=(x_test, y_test))
```

- NN+ add dropout (alpha = 0.1)(test accuracy = 0.75)

model	accuracy
original	0.657
improve	**0.75**

```
model = Sequential()
model.add(Dense(500, input_dim=7, activation='sigmoid'))
model.add(Dropout(0.1))
model.add(Dense(100, activation='sigmoid'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
model.fit(x_train,y_train, epochs=1000, batch_size=70, validation_data=(x_test, y_test))
```

Analysis

MLP Training

hidden_layer_sizes	(100,)
activation	relu
solver	adam
batch_size	auto
learning_rate	constant
learning_rate_init	0.001
max_iter	200
accuracy	0.74

hidden_layer_sizes	(100,)
activation	relu
solver	sgd
batch_size	auto
learning_rate	constant
learning_rate_init	0.001
max_iter	200
accuracy	0.707

hidden_layer_sizes	(100,)
activation	logistic
solver	adam
batch_size	auto
learning_rate	constant
learning_rate_init	0.001
max_iter	200
accuracy	0.773

hidden_layer_sizes	(100,)
activation	logistic
solver	adam
batch_size	auto
learning_rate	constant
learning_rate_init	0.001
max_iter	1000
accuracy	0.786

hidden_layer_sizes	(64,64,64)
activation	logistic
solver	adam
batch_size	auto
learning_rate	constant
learning_rate_init	0.001
max_iter	1000
accuracy	0.796

model	**accuracy**
original	0.74
improve	**0.796**

Analysis

Training Summary

	Training	original data	processed data	optimal model	improvement
kNN	n_neighbor	0.779	0.785	0.818	**+3.9%**
NN	activation encoding dropout	0.646	0.669	0.75	**+ 10.4%**
MLP	hidden_layer_siz es activation max_iter	0.702	0.74	0.796	**+9.4%**

05
CONCLUSION

Conclusion

STEP
01

發現問題

糖尿病潛在風險高
現代人生活習慣改變

數據清洗
特徵選取
資料標準化

資料處理

02
STEP

STEP
03

模型建立

由機器學習到深度學
習的不同模型建立

藉腦力激盪以多種方
法訓練模型，以使準
確度提升

過程分析

04
STEP

Conclusion

- 架設網站提供民眾可自行檢測
- 持續優化與訓練模型，提升準確度
- 應用至其他疾病篩檢，擴大通用性

Thank you