紀老師程式教學網
https://www.facebook.com/teacherchi

機器學習

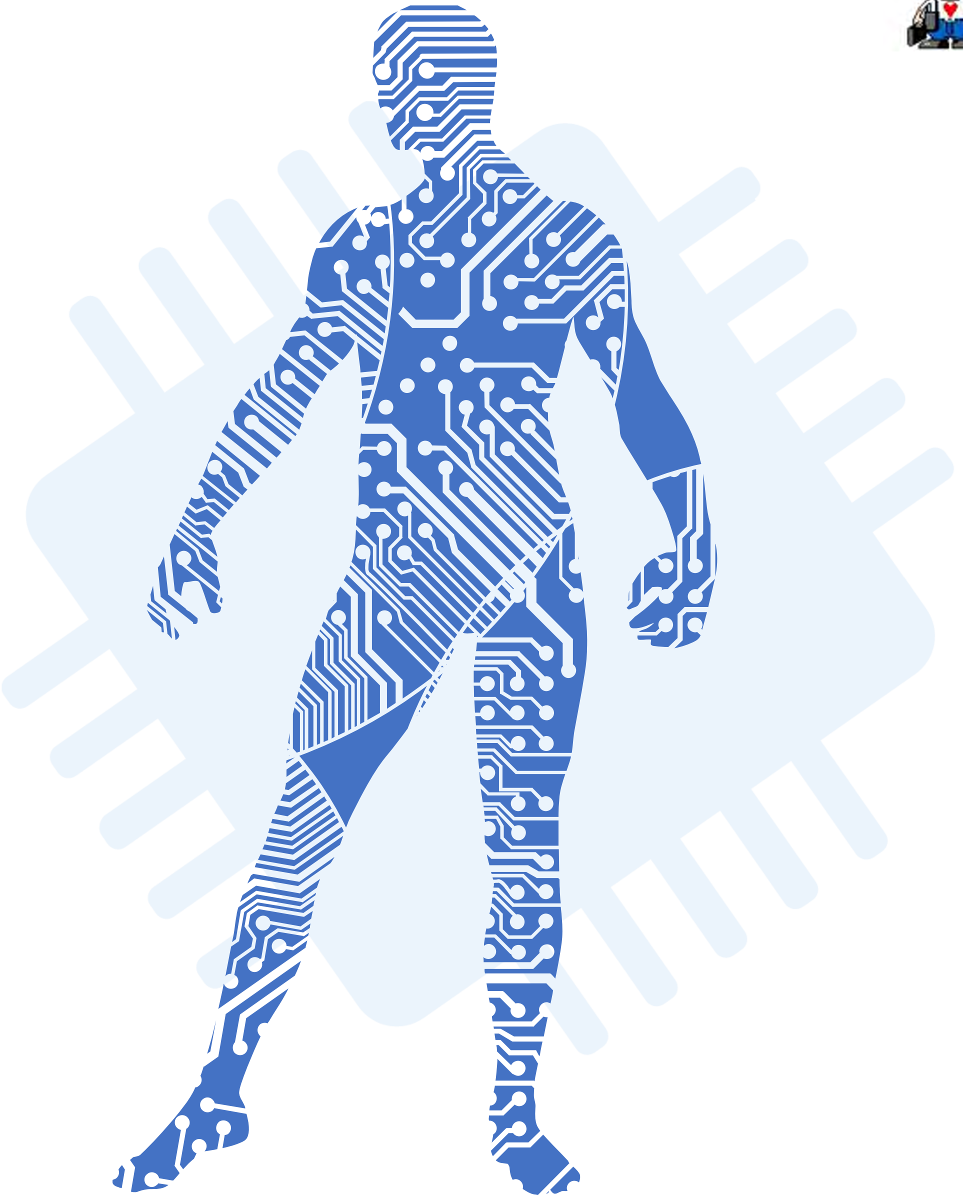
第 10 章 單純貝氏分類器

(Naïve Bayes Classifier)

講師：紀俊男

本章大綱

- 理論說明
- 資料前處理
- 實作單純貝氏分類器
- 交叉驗證
- 將模型視覺化
- 檢查「單純」前提是否成立
- 重點整理



理論說明

為何叫做「單純（Naïve）」

- 單純貝氏分類器**假設前提**

各個「自變數 X_i」獨立！

獨立 = 單純 vs. **不獨立 = 不單純**

使用到的數學定理

- 貝氏定理（Bayes' Theorem）

$$P(Y|X)P(X) = P(X|Y)P(Y)$$

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

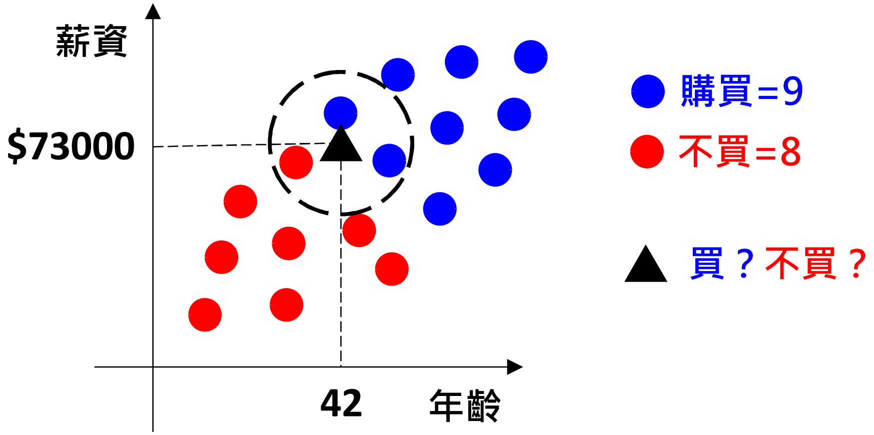
「貝氏定理」解說

$$P(Y|X)=\frac{P(X|Y)P(Y)}{P(X)}$$

$$P(Y=？|X=(42,\$73000))$$

$$=\begin{cases}P(Y=\text{會買}|X=(42,\$73000))\\P(Y=\text{不買}|X=(42,\$73000))\end{cases}$$

$$=\begin{cases}\dfrac{P(X=(42,\$73000)|Y=\text{會買})P(Y=\text{會買})}{P(X=(42,\$73000))}\\[2ex]\dfrac{P(X=(42,\$73000)|Y=\text{不買})P(Y=\text{不買})}{P(X=(42,\$73000))}\end{cases}$$



$$=\begin{cases}\dfrac{2/9\times 9/17}{3/17}=\dfrac{2}{3}\ (\text{會買})\\[2ex]\dfrac{1/8\times 8/17}{3/17}=\dfrac{1}{3}\ (\text{不買})\end{cases}$$

$P(X=(42,\$73000))=$ ◯

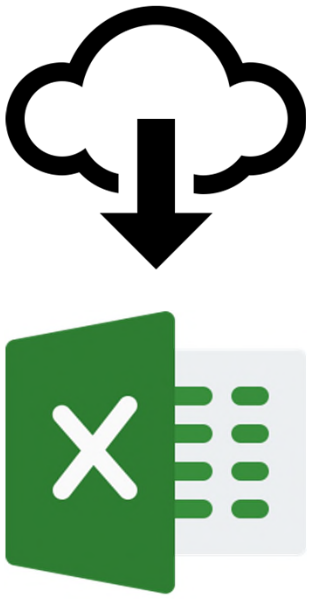
$=$ 與▲相似的機率 $=\dfrac{3}{8+9}=\dfrac{3}{17}$

您覺得是
買？不買？

資料前處理

下載與瀏覽資料集

- 依照講師指示，**下載**並**瀏覽**資料集



Social_Network_Ads.csv

	A	B	C	D	E
1	User ID	Gender	Age	Salary	Purchased
2	15624510	Male	19	19000	0
3	15810944	Male	35	20000	0
4	15668575	Female	26	43000	0
5	15603246	Female	27	57000	0
6	15804002	Male	19	76000	0
7	15728773	Male	27	58000	0
8	15598044	Female	27	84000	0
9	15694829	Female	32	150000	1
10	15600575	Male	25	33000	0

目的：利用「性別、年齡、薪資」
--> 推算「是否購買」

資料前處理

撰寫程式碼

```python
import HappyML.preprocessor as pp

# Load Data
dataset = pp.dataset(file="Social_Network_Ads.csv")

# X, Y decomposition
X, Y = pp.decomposition(dataset, x_columns=[1, 2, 3], y_columns=[4])

# One-Hot Encoder
X = pp.onehot_encoder(ary=X, columns=[0], remove_trap=True)

# Feature Selection
from HappyML.preprocessor import KBestSelector
selector = KBestSelector()
X = selector.fit(x_ary=X, y_ary=Y, verbose=True, sort=True).transform(x_ary=X)

# Split Training / TEsting Set
X_train, X_test, Y_train, Y_test = pp.split_train_test(x_ary=X, y_ary=Y)

# Feature Scaling
X_train, X_test = pp.feature_scaling(fit_ary=X_train, transform_arys=(X_train, X_test))
```

資料前處理流程：

1. 載入資料
2. 切分自變數、應變數
3. 處理缺失資料
   （無缺失資料）
4. 類別資料數位化
   （＋移除虛擬變數陷阱）
5. 特徵選擇
6. 切分訓練集、測試集
7. 特徵縮放
   （僅縮放 X 即可）

資料前處理

- 執行結果

```
The Significant Level: 0.05

--- The p-values of Feature Importance ---
TRUE  <0.05 0.00000000e+00 (EstimatedSalary)
TRUE  <0.05 4.04303193e-100 (Age)
FALSE >0.05 5.44126248e-01 (Gender_Male)

Number of Features Selected: 2
```

dataset

Gender	Age	EstimatedSalary
Male	19	19000
Male	35	20000
Female	26	43000
Female	27	57000
Male	19	76000

→

One Hot Encoding & Remove Dummy

Gender_Male	Age	EstimatedSalary
1	19	19000
1	35	20000
0	26	43000
0	27	57000
1	19	76000

→

Feature Selection

Age	EstimatedSalary
19	19000
35	20000
26	43000
27	57000
19	76000

→

Feature Scaling

Age	EstimatedSalary
0.304495	-0.243348
-1.30142	-0.478655
0.304495	0.0213738
-1.20696	0.462576
1.06022	0.0802008

隨堂練習：資料前處理

- 請撰寫下列程式碼，並予以執行，完成「**資料前處理**」的步驟：

```python
import HappyML.preprocessor as pp

# Load Data
dataset = pp.dataset(file="Social_Network_Ads.csv")

# X, Y decomposition
X, Y = pp.decomposition(dataset, x_columns=[1, 2, 3], y_columns=[4])

# One-Hot Encoder
X = pp.onehot_encoder(ary=X, columns=[0], remove_trap=True)

# Feature Selection
from HappyML.preprocessor import KBestSelector
selector = KBestSelector()
X = selector.fit(x_ary=X, y_ary=Y, verbose=True, sort=True).transform(x_ary=X)

# Split Training / TEsting Set
X_train, X_test, Y_train, Y_test = pp.split_train_test(x_ary=X, y_ary=Y)

# Feature Scaling
X_train, X_test = pp.feature_scaling(fit_ary=X_train, transform_arys=(X_train, X_test))
```

實作單純貝氏分類器

以標準函式庫實作

- 程式碼

1

2

3

```python
from sklearn.naive_bayes import GaussianNB

model = GaussianNB()
model.fit(X_train, Y_train.values.ravel())

Y_pred = model.predict(X_test)
```

1. 引入「高斯分佈」的簡單貝氏分類器
    - **GaussianNB**: 自變數 X_i 呈「高斯分佈（常態）」時使用
    - **MultinomialNB**: 自變數 X_i 呈「多項分佈（多選一）」時使用
    - **BernoulliNB**: 自變數 X_i 呈「白努利分佈（二選一）」時使用
2. 訓練模型
3. 預測結果

- 執行結果

Y_test

	0
0	0
1	1
2	1
3	0
4	0
5	1
6	0
7	0
8	1
9	1
10	0

Y_pred

	0
0	0
1	1
2	1
3	0
4	0
5	1
6	0
7	0
8	1
9	0
10	0

隨堂練習：以標準函式庫實作

- 請撰寫下列程式碼，並執行之：

```python
from sklearn.naive_bayes import GaussianNB

model = GaussianNB()
model.fit(X_train, Y_train.values.ravel())

Y_pred = model.predict(X_test)
```

- 執行完畢後，請比較 Y_test（真實值）與 Y_pred（預測值）的差異。

以快樂版函式庫實作

• 原始程式碼解說（1）：

/HappyML/classification.py

引入必要的套件 →

```
from sklearn.naive_bayes import BernoulliNB, MultinomialNB, GaussianNB
import pandas as pd

class NaiveBayesClassifier:
    __classifier = None
    __y_columns = None

    def __init__(self, type="gaussian"):
        algorithm_dict = {
                "bernoulli" : BernoulliNB(),
                "multinomial" : MultinomialNB(),
                "gaussian" : GaussianNB()
                }
        self.__classifier = algorithm_dict[type]
```

類別成員變數 →

建構函數，根據參數，
初始化不同分佈的單純貝氏核心

以快樂版函式庫實作

- 原始程式碼解說（2）：

/HappyML/classification.py

classifier 的 getter/setter

訓練模型

預測結果

```python
    @property
    def classifier(self):
        return self.__classifier

    @classifier.setter
    def classifier(self, classifier):
        self.__classifier = classifier

    def fit(self, x_train, y_train):
        self.classifier.fit(x_train, y_train.values.ravel())
        self.__y_columns = y_train.columns

        return self

    def predict(self, x_test):
        return pd.DataFrame(self.classifier.predict(x_test), index=x_test.index, columns=self.__y_columns)
```

以快樂版函式庫實作

- 呼叫範例

```python
from HappyML.classification import NaiveBayesClassifier

model = NaiveBayesClassifier()
Y_pred = model.fit(X_train, Y_train).predict(X_test)
```

- 執行結果

Y_test

Index	Purchased
20	1
163	0
348	0
213	0
373	1

Y_pred

Index	Purchased
20	1
163	0
348	0
213	0
373	1

隨堂練習：以快樂版函式庫實作

- 請先將前一小節「**實作單純貝氏分類器**」的程式碼**註解**掉。
- 請用「**變數觀察面板**」，**清除**掉所有變數。
- 輸入下列程式碼，並重跑整個程式：

```python
from HappyML.classification import NaiveBayesClassifier

model = NaiveBayesClassifier()
Y_pred = model.fit(X_train, Y_train).predict(X_test)
```

- 比較 Y_test（真實值）與 Y_pred（預測值）的結果。

交叉驗證

「交叉驗證」簡介

- 何謂「K 次交叉驗證（K-Fold Cross Validation）」

5-fold CV

DATASET

Estimation 1	Test	Train	Train	Train	Train
Estimation 2	Train	Test	Train	Train	Train
Estimation 3	Train	Train	Test	Train	Train
Estimation 4	Train	Train	Train	Test	Train
Estimation 5	Train	Train	Train	Train	Test

accuracy 1
accuracy 2
accuracy 3
accuracy 4
accuracy 5

Total Accuracy

也可以求算
Recall, Precision...等

不受單次抽樣偏頗影響！

以「標準函式庫」實作交叉驗證

- 程式碼解說

引入必要套件

```
from sklearn.model_selection import cross_val_score

accuracies = cross_val_score(estimator=model.classifier, X=X, y=Y.values.ravel(), scoring="accuracy", cv=10, n_jobs=-1)
print("{} Folds Mean Accuracy: {}".format(k_fold, accuracies.mean()))
```

1. esitmiator=：輸入機器學習模型
2. X=：輸入自變數
3. y=：輸入應變數
4. scoring=：輸入此次求算的效能指標。
（可以輸入 “accuracy”，“recall”，“precision”，“f1” 等數值）
5. cv=：Cross Validation 的次數。如： cv=10 → 10-Fold。
6. n_jobs=：使用多少 CPU 核心（如：1, 2, ...）。-1 代表全部使用。
7. accuracies：保存 10 份 Accuracy 的 NDArray 陣列。
8. .mean()：取平均（Average）。

隨堂練習：以「標準函式庫」實作交叉驗證

- 請依照前一頁投影片的說明，以「交叉驗證」評估模型效能：

```
from sklearn.model_selection import cross_val_score

accuracies = cross_val_score(estimator=model.classifier, X=X, y=Y.values.ravel(), scoring="accuracy", cv=10, n_jobs=-1)
print("{} Folds Mean Accuracy: {}".format(k_fold, accuracies.mean()))
```

- 如果要計算「廣度」、「精度」、或「F1-score」，可以參考下列程式碼：

```
from sklearn.model_selection import cross_val_score

k_fold = 10
accuracies = cross_val_score(estimator=model.classifier, X=X, y=Y.values.ravel(), scoring="accuracy", cv=k_fold, n_jobs=-1)
print("{} Folds Mean Accuracy: {}".format(k_fold, accuracies.mean()))

recalls = cross_val_score(estimator=model.classifier, X=X, y=Y.values.ravel(), scoring="recall", cv=k_fold, n_jobs=-1)
print("{} Folds Mean Recall: {}".format(k_fold, recalls.mean()))

precisions = cross_val_score(estimator=model.classifier, X=X, y=Y.values.ravel(), scoring="precision", cv=k_fold, n_jobs=-1)
print("{} Folds Mean Precision: {}".format(k_fold, precisions.mean()))

f_scores = cross_val_score(estimator=model.classifier, X=X, y=Y.values.ravel(), scoring="f1", cv=k_fold, n_jobs=-1)
print("{} Folds Mean F1-Score: {}".format(k_fold, f_scores.mean()))
```

以「快樂版函式庫」實作交叉驗證

- 程式碼解說（1）

/HappyML/performance.py

引入所需的套件

```python
from sklearn.model_selection import cross_val_score

class KFoldClassificationPerformance:
    __k_fold = None
    __x_ary = None
    __y_ary = None
    __classifier = None
    __verbose = None

    def __init__(self, x_ary, y_ary, classifier, k_fold=10, verbose=False):
        self.__x_ary = x_ary
        self.__y_ary = y_ary
        self.k_fold = k_fold
        self.__classifier = classifier
        self.verbose = verbose

    @property
    def k_fold(self):
        return self.__k_fold
```

類別的成員變數

自變數
應變數

機器學習
模型

K 值

是否逐步
顯示過程

建構函數

k_fold 成員變數
的 getter

以「快樂版函式庫」實作交叉驗證

- 程式碼解說（2）

/HappyML/performance.py

k_fold 成員變數的 setter

verbose 成員變數的 getter & setter

classifier 成員變數的 getter

```python
    @k_fold.setter
    def k_fold(self, k_fold):
        if k_fold >=2:
            self.__k_fold = k_fold
        else:
            self.__k_fold = 2

    @property
    def verbose(self):
        return self.__verbose

    @verbose.setter
    def verbose(self, verbose):
        if verbose:
            self.__verbose = 10
        else:
            self.__verbose = 0

    @property
    def classifier(self):
        return self.__classifier
```

以「快樂版函式庫」實作交叉驗證

- 程式碼解說（3）

/HappyML/performance.py

確度

廣度

精度

F 值

```python
    def accuracy(self):
        results = cross_val_score(estimator=self.classifier, X=self.__x_ary, y=self.__y_ary.values.ravel(), scoring="accuracy", cv=self.k_fold, verbose=self.verbose)
        return results.mean()

    def recall(self):
        def recall_scorer(estimator, X, y):
            return recall_score(y, estimator.predict(X), average="macro")

        results = cross_val_score(estimator=self.classifier, X=self.__x_ary, y=self.__y_ary.values.ravel(), scoring=recall_scorer, cv=self.k_fold, verbose=self.verbose)
        return results.mean()

    def precision(self):
        def precision_scorer(estimator, X, y):
            return precision_score(y, estimator.predict(X), average="macro")

        results = cross_val_score(estimator=self.classifier, X=self.__x_ary, y=self.__y_ary.values.ravel(), scoring=precision_scorer, cv=self.k_fold, verbose=self.verbose)
        return results.mean()

    def f_score(self):
        def f1_scorer(estimator, X, y):
            return fbeta_score(y, estimator.predict(X), beta=1, average="macro")

        results = cross_val_score(estimator=self.classifier, X=self.__x_ary, y=self.__y_ary.values.ravel(), scoring=f1_scorer, cv=self.k_fold, verbose=self.verbose)
        return results.mean()
```

以「快樂版函式庫」實作交叉驗證

- **呼叫範例**

引入所需的套件

自變數、應變數

機器學習模型

K 值

是否逐步顯示過程

印出各種指標

```python
from HappyML.performance import KFoldClassificationPerformance

K = 10
kfp = KFoldClassificationPerformance(x_ary=X, y_ary=Y, classifier=model.classifier, k_fold=K, verbose=False)

print("{} Folds Mean Accuracy: {}".format(K, kfp.accuracy()))
print("{} Folds Mean Recall: {}".format(K, kfp.recall()))
print("{} Folds Mean Precision: {}".format(K, kfp.precision()))
print("{} Folds Mean F1-Score: {}".format(K, kfp.f_score()))
```

- **執行結果**

```
10 Folds Mean Accuracy: 0.8775719199499689
10 Folds Mean Recall: 0.8578717948717948
10 Folds Mean Precision: 0.8874708964782494
10 Folds Mean F1-Score: 0.8619361410762924
```

隨堂練習：以「快樂版函式庫」實作交叉驗證

- 請輸入下列程式碼，並觀看您模型「K 次交叉驗證」效能：

```python
from HappyML.performance import KFoldClassificationPerformance

K = 10
kfp = KFoldClassificationPerformance(x_ary=X, y_ary=Y, classifier=model.classifier, k_fold=K, verbose=False)

print("{} Folds Mean Accuracy: {}".format(K, kfp.accuracy()))
print("{} Folds Mean Recall: {}".format(K, kfp.recall()))
print("{} Folds Mean Precision: {}".format(K, kfp.precision()))
print("{} Folds Mean F1-Score: {}".format(K, kfp.f_score()))
```

將模型視覺化

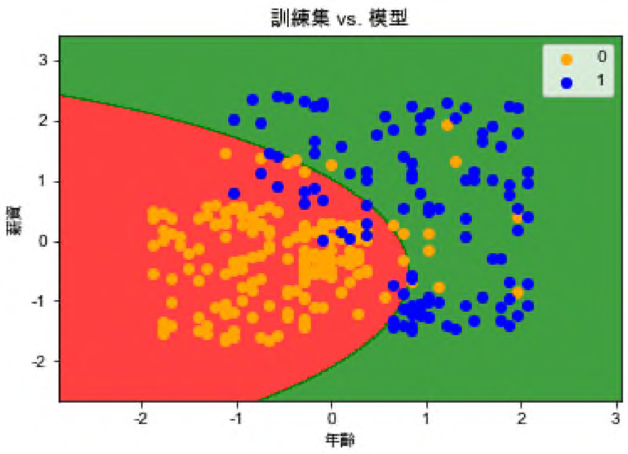
將模型視覺化

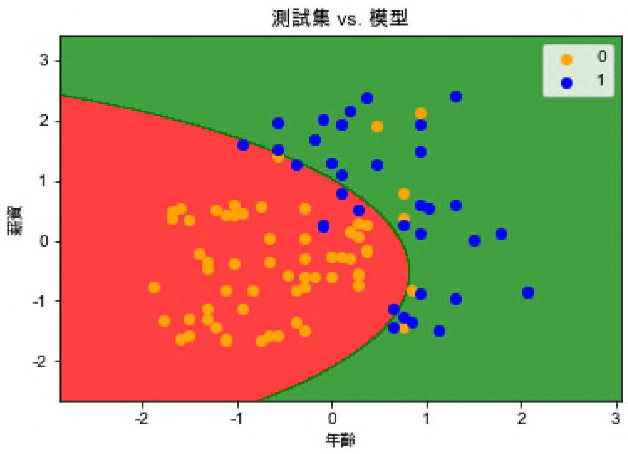
注意！自變數 X
必須是兩個特徵！
且經過縮放！

- 程式碼

```python
import HappyML.model_drawer as md

md.classify_result(x=X_train, y=Y_train, classifier=model.classifier, title="訓練集 vs. 模型", font='DFKai-sb')
md.classify_result(x=X_test, y=Y_test, classifier=model.classifier, title="測試集 vs. 模型", font='DFKai-sb')
```

- 執行結果





隨堂練習：將模型視覺化

- 請輸入下列程式碼，並執行觀看您**模型**的好壞：

```python
import HappyML.model_drawer as md

md.classify_result(x=X_train, y=Y_train, classifier=model.classifier, title="訓練集 vs. 模型", font='DFKai-sb')
md.classify_result(x=X_test, y=Y_test, classifier=model.classifier, title="測試集 vs. 模型", font='DFKai-sb')
```

檢查「單純」前提是否成立

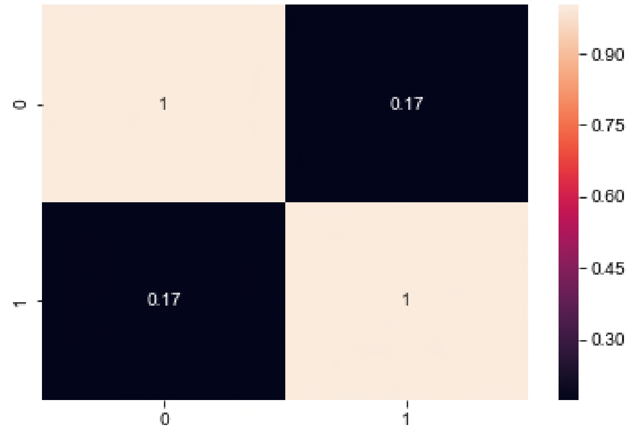
檢查前提是否成立

- 簡單貝氏分類器**前提**：**自變數**各自獨立

```python
from HappyML.criteria import AssumptionChecker

checker = AssumptionChecker(x_train=X_train, x_test=X_test, y_train=Y_train, y_test=Y_test, y_pred=Y_pred)
checker.features_correlation(heatmap=True)
```

- 執行結果

```
*** Check for Correlation of Features ***
--- Features Correlation Matrix ---
        0       1
0  1.0000  0.1735
1  0.1735  1.0000
No Correlation (>=0.8) Found!
```



隨堂練習：檢查前提是否成立

- 請輸入、並執行下列程式碼，檢查**自變數**是否各自**獨立**？

```python
from HappyML.criteria import AssumptionChecker

checker = AssumptionChecker(x_train=X_train, x_test=X_test, y_train=Y_train, y_test=Y_test, y_pred=Y_pred)
checker.features_correlation(heatmap=True)
```

課後作業：糖尿病資料庫

- 說明：
  - 下載「美國國家糖尿病、消化、與腎臟病研究中心」的資料庫
    - https://www.kaggle.com/kandij/diabetes-dataset

  - 各欄位說明如下：
    - Pregnancies：懷孕次數（次）
    - Glucose：口服葡萄糖耐量試驗中血漿葡萄糖濃度
    - BloodPressure：舒張壓（mmHg）
    - SkinThickness：三頭肌組織皮褶厚度（mm）
    - Insulin：兩小時血清胰島素（μU/ ml）
    - BMI：身體質量指數
    - DiabetesPedigreeFunction：糖尿病系統功能
    - Age：年齡（歲）
    - Outcome：是否有糖尿病

課後作業：糖尿病資料庫

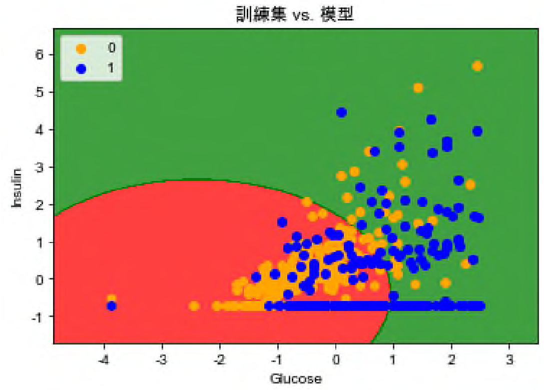
- 要求
  - 使用 **KBestSelector** 挑選「**夠顯著**」的特徵，並**印出**你選中了哪些特徵？
    （提示：使用預設參數 best_k= “auto” 即可）
  - 印出「**K 次交叉驗證**」的「**確度、廣度、精度、F-Score**」，說明你的模型有多好？
  - 檢查您的模型是否符合「單純貝氏」分類器的**前提**（自變數獨立）？
  - 用 **KBestSelector** 將特徵壓至 **2 個**，另外訓練出一個單純貝氏分類器。
  - 用這個「僅有兩個特徵」的**資料集**與**模型**，繪製出「訓練集」與「測試集」的分類結果。

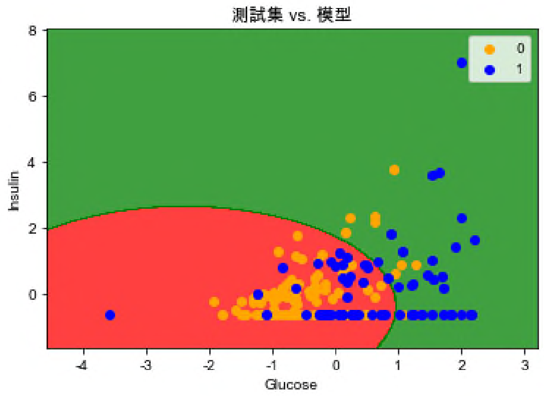
加分題：

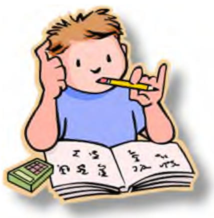
- 這個題目用「單純貝氏」好？還是「邏輯迴歸」模型好？

- 輸出

```
The Significant Level: 0.05

--- The p-values of Feature Importance ---
TRUE  <0.05 0.00000000e+00 (Insulin)
TRUE  <0.05 5.48728628e-309 (Glucose)
TRUE  <0.05 2.51638830e-41 (Age)
TRUE  <0.05 1.32590849e-29 (BMI)
TRUE  <0.05 4.55261043e-26 (Pregnancies)
TRUE  <0.05 3.15697650e-13 (SkinThickness)
TRUE  <0.05 2.71819252e-05 (BloodPressure)
TRUE  <0.05 2.02213728e-02 (DiabetesPedigreeFunction)

Number of Features Selected: 8
10 Folds Mean Accuracy: 0.7564935064935066
10 Folds Mean Recall: 0.7185099715099715
10 Folds Mean Precision: 0.7348495448677373
10 Folds Mean F1-Score: 0.7235726493134441
*** Check for Correlation of Features ***
--- Features Correlation Matrix ---
                          Pregnancies  ...      Age
Pregnancies                    1.0000  ...   0.5530
Glucose                        0.1505  ...   0.2718
BloodPressure                  0.1524  ...   0.2719
SkinThickness                 -0.0670  ...  -0.0680
Insulin                       -0.0567  ...  -0.0379
BMI                            0.0688  ...   0.0719
DiabetesPedigreeFunction       0.0031  ...   0.0688
Age                            0.5530  ...   1.0000

[8 rows x 8 columns]
No Correlation (>=0.8) Found!
```







重點整理

- **單純貝氏分類器成立前提**
  - 自變數各自獨立
- **應該了解的數學定理**
  - 貝氏定理： $P(Y|X)P(X) = P(X|Y)P(Y)$
- **三種單純貝氏分類器的分佈模型**
  - 高斯分佈：GaussianNB - 自變數呈常態分佈時使用
  - 多項分佈：MultinomialNB - 自變數呈多項分佈（多擇一）時使用
  - 白努利分佈：BernoulliNB - 自變數呈白努利分佈（二擇一）時使用
- **K 次交叉驗證**
  - 可以不受**抽樣**影響、公正客觀評判模型的確度、廣度、精度、F 值。
  - 使用 sklearn.model_selection.cross_val_score() 實作
- **何時使用「單純貝氏分類器」**
  - 小樣本（千筆以下）
    - 靠樣本推算母體機率分佈，再拿最可能的母體機率分佈預測未來。
    - 屬「模型生成式（Generative）」推估法，受樣本影響較小。
  - 觀察樣本點，有一個「**決定邊界**」，但該邊界**彎彎曲曲**，並非任何數學方程式可以擬合