# Exercise 1

- 利用 sklearn.datasets.make_classification 造 classification dataset
  - n_samples=300, n_features=2,n_informative=2,flip_y=0.2, n_redundant=0,n_clusters_per_class=1,n_classes=2, class_sep=2,random_state=2030
  - 其餘用default
- 使用 knn model, 畫出(k=1, 3, 8, 20) 在訓練集上的decision boundary
  - 如下頁所示

```
list1 = [1, 3, 8, 20]
for i in list1:
    """

    中間的程式碼省略...

    """
    a=list(('k=1','k=3','k=8','k=20'))
    plt.title(a[b])
```

```
1  print(list(('k=1','k=3','k=8','k=20')))
2  print(['k=1','k=3','k=8','k=20'])
```

[1] ✓ 1.1s

```
['k=1', 'k=3', 'k=8', 'k=20']
['k=1', 'k=3', 'k=8', 'k=20']
```

list((‘k=1’, ‘k=3’)) == [‘k=1’, ‘k=3’]

```
1  list1 = [1, 3, 8, 20]
2  a = ['k=1','k=3','k=8','k=20']
3  for i, title in zip(list1, a):
4      """
5
6      中間的程式碼省略...
7
8      """
9      plt.title(title)
```

i=1, title=‘k=1’;
i=3, title=‘k=3’;
i=8, title=‘k=8’;
i=20, title=‘k=20’;

Zip可以將兩個list中相同位置的item一起迭代

```
print(list(('k=1','k=3','k=8','k=20')))
print(['k=1','k=3','k=8','k=20'])
```

[1] ✓ 1.1s

```
['k=1', 'k=3', 'k=8', 'k=20']
['k=1', 'k=3', 'k=8', 'k=20']
```

list((‘k=1’, ‘k=3’)) == [‘k=1’, ‘k=3’]

```
list1 = [1, 3, 8, 20]
a = ['k=1','k=3','k=8','k=20']
for i, title in zip(list1, a):
    """

    中間的程式碼省略...

    """
    plt.title(title)
```

i=1, title=‘k=1’;

i=3, title=‘k=3’;

i=8, title=‘k=8’;

i=20, title=‘k=20’;

Zip可以將兩個list中相同位置的item一起迭代

```
list1 = [1, 3, 8, 20]
for i in list1:
    """

    中間的程式碼省略...

    """
    plt.title(f"k={i}")
```

# Exercise 2

- 利用 sklearn.datasets.make_regression 造 regression dataset
  - n_samples=1000, n_features=1000, n_informative=10, noise=20, random_state=99
  - 其餘用default
- 使用 linear regression
- 使用 ridge regression (alpha 候選為 $2^0,2^1,2^2,2^3,2^4,2^5,2^6$)
- 評估方式為資料分成訓練集70%與測試集30%
- 評估指標為 mean square error

```python
import matplotlib.pyplot as plt
plt.plot(range(8), lr_mse) # , label='Linear_Regression')
plt.plot(range(8), ridge_mse) # , label='Ridge_Regression')
plt.legend()
plt.show()
```

✓ 0.1s Python

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```

```python
plt.plot(range(8), lr_mse, label='Linear_Regression')
plt.plot(range(8), ridge_mse, label='Ridge_Regression')
plt.legend()
```

要有折線的label才能收集折線的label

```python
def mean_squared_error(yte,yprl):
    yte, yprl = np.array(yte), np.array(yprl)
    return np.square(np.subtract(yte,yprl)).mean()
def mean_squared_error(yte,yprr):
    yte, yprr = np.array(yte), np.array(yprr)
    return np.square(np.subtract(yte,yprr)).mean()
```

✓ 0.2s

```python
from sklearn.metrics import mean_squared_error
```

```python
prmse=mean_squared_error(yte,yprr)
```

```python
plmse=mean_squared_error(yte,yprl)
```

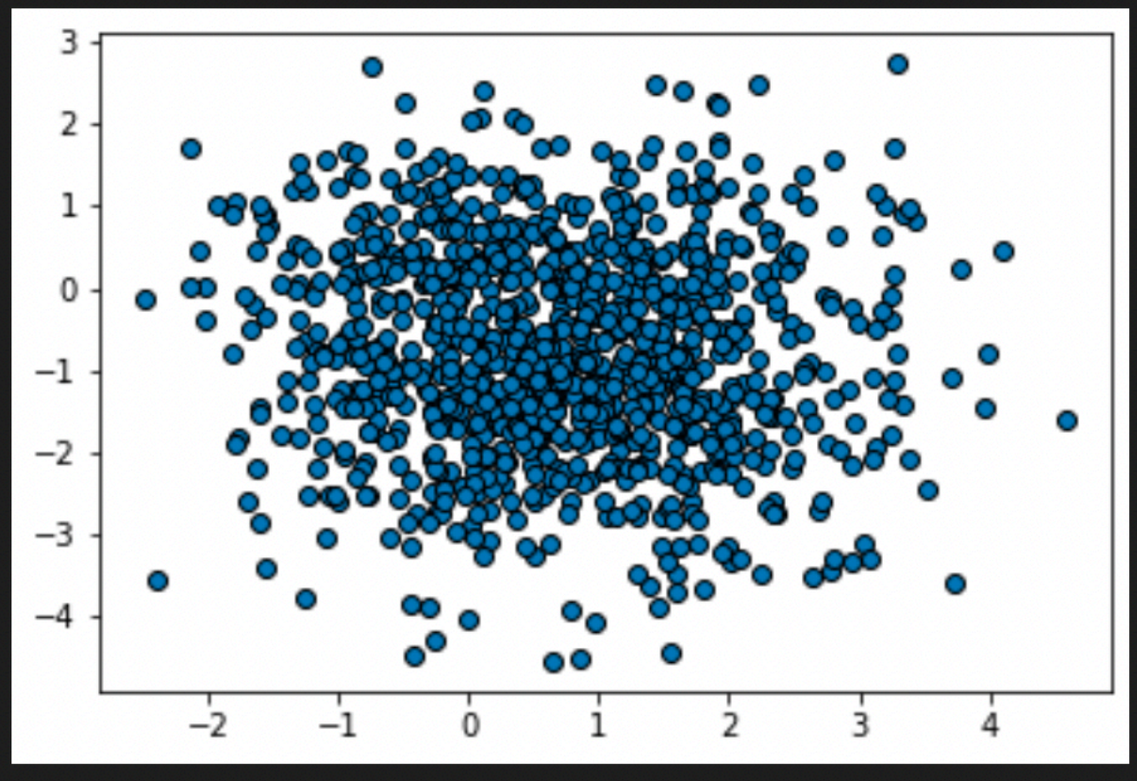
# Exercise 3

- 利用 sklearn.datasets.make_classification 造 classification dataset
  - n_samples=1000, n_features=2, n_informative=2, n_redundant=0, n_clusters_per_class=1, class_sep=0.8, flip_y=0, weights=[0.95,0.05], random_state=100
  - 其餘用default
- 分別用3-nn、Gaussian NB 與 logistic regression (C=1)
  - 其中準確率, precision, recall 與 f1 score 都要看
  - logistic regression 須考慮不同權重 1~10
- 評估方式為資料分成訓練集70%與測試集30%

1. Accuracy
2. Precision
3. Recall
4. F1 score

```python
X,y = make_classification(n_samples=1000,n_features=2,n_informative=2,n_
plt.scatter(X[:,0],X[:,1],edgecolors='black')
plt.show()
```

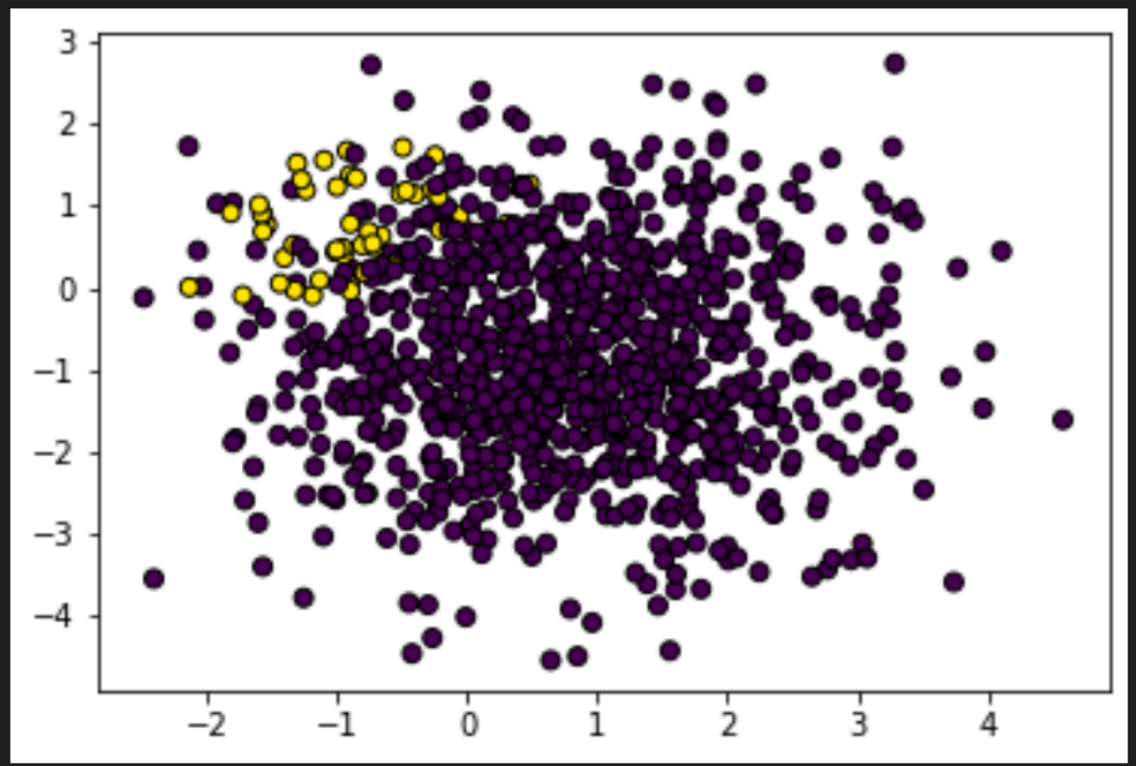
Python



```python
from sklearn.datasets import make_classification
import matplotlib.pyplot as plt
X,y = make_classification(n_samples=1000,n_features=2,n_informative=2,n_
plt.scatter(X[:,0],X[:,1],edgecolors='black', c=y)
plt.show()
```

✓ 0.1s

加入color看出類別分佈



```python
knn_accuracy = []
knn_precision = []
knn_recall = []
knn_f1 = []
for i in range(1,11):
    knn = KNeighborsClassifier(n_neighbors=3)
    knn.fit(Xtr,ytr)
    yhat = knn.predict(Xte)
    knn_accuracy.append(accuracy_score(yte,yhat))
    knn_precision.append(precision_score(yte,yhat))
    knn_recall.append(recall_score(yte,yhat))
    knn_f1.append(f1_score(yte,yhat))
```

Python

少了重複計算的時間

```python
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(Xtr, ytr)
ypr = knn.predict(Xte)
knn_Acc = accuracy_score(yte,ypr)
knn_Precision = precision_score(yte,ypr)
knn_Recall = recall_score(yte,ypr)
knn_F1 = f1_score(yte,ypr)
knn_Acc = [knn_Acc] * 10
print(knn_Acc)
```

✓ 0.4s Python

```
[0.9566666666666667, 0.9566666666666667, 0.9566666666666667,
0.9566666666666667, 0.9566666666666667, 0.9566666666666667,
0.9566666666666667, 0.9566666666666667, 0.9566666666666667,
0.9566666666666667]
```

# Exercise 4

- Airfoil Self-Noise Data Set
- https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise
- 設計一個實驗，決定用哪個演算法比較適合(包含要用什麼樣的超參數(hyper parameter))
  - knn regressor
  - Linear Regression
  - Ridge Regression

## Data Set Information:

The NASA data set comprises different size NACA 0012 airfoils at various wind tunnel speeds and angles of attack. The span of the airfoil and the observer position were the same in all of the experiments.

## Attribute Information:

This problem has the following inputs:
1. Frequency, in Hertzs.
2. Angle of attack, in degrees.
3. Chord length, in meters.
4. Free-stream velocity, in meters per second.
5. Suction side displacement thickness, in meters.

The only output is:
6. Scaled sound pressure level, in decibels.

這是target

```
#因為資料是continuous的，不是label，所以不適合用knn，只用其他兩個
```

CodingNote.cc

# 機器學習 第5篇：knn回歸

2020 年 11 月 2 日　筆記　機器學習

基於最鄰近演算法的分類，本質上是對離散的數據標籤進行預測，實際上，最鄰近演算法也可以用於對連續的數據標籤進行預測，這種方法叫做基於最鄰近數據的回歸，預測的值（即數據的標籤）是連續值，通過計算數據點最臨近數據點平均值而獲得預測值。

## 一，sklearn的knn回歸

scikit-learn實現了兩個不同的最鄰近回歸模型：

- KNeighborsRegressor：根據每個查詢點的最鄰近的k個數據點的均值作為預測值，其中，k是用戶指定的整數。
- RadiusNeighborsRegressor：基於查詢點的固定半徑內的數據點的均值作為預測值，其中r是用戶指定的浮點值。

回歸模擬器的定義如下，該定義只列出最重要的參數，詳細參數請參考sicikit-learn 官網：

```
sklearn.neighbors.KNeighborsRegressor(n_neighbors=5, weights='uniform', algorithm='a
uto', metric='minkowski',...)
sklearn.neighbors.RadiusNeighborsRegressor(radius=1.0, weights='uniform', algorithm=
'auto', metric='minkowski',...)
```

https://codingnote.cc/zh-tw/p/251182/