資料科學導論 HW1

a. 程式說明

1.引用套件:

pandas 對 csv 做應用 numpy 做數值處理 time 用來設亂數種子
從 sklearn 引入 labelEncoder 對 csv 檔的文字內容作編碼
train_test_split 分割資料集 切成訓練集與驗證集 檢查訓練情形
linear_model 引入寫好的 model
引入 Normalizer 對資料做標準化

```
import pandas as pd
import numpy as np
import time as time
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.preprocessing import Normalizer
```

2.讀入資料&資料預處理：

讀入 csv 開始做資料預處理

```
#==========Data_Preprocessing==========#
trainData = pd.read_csv('train.csv')
tesrData = pd.read_csv('test.csv')
train_df = pd.DataFrame(trainData)
test = pd.DataFrame(tesrData)
```

把殘缺值清掉

```
#濾掉慘缺值
drop_train = train_df.dropna()
```

把第一個 attribute(yyyy-mm-dd)處理為只有月份
attribute 裡的值如果不是數字就編碼為相對數值

```
#只取月份
for i in range(0,drop_train.shape[0]):
    drop_train.iloc[i,0]=int(drop_train.iloc[i,0].split('-')[1])
for i in range(0,test.shape[0]):
    test.iloc[i,0]=int(test.iloc[i,0].split('-')[1])

#value都編碼為數字
for attribute in drop_train.columns:
    value = drop_train.loc[754,attribute]
    if not str(value).isdigit():
        drop_train[attribute] = LabelEncoder().fit_transform(drop_train[attribute])
for attribute in test.columns:
    value = test.loc[0,attribute]
    if not str(value).isdigit():
        test[attribute] = LabelEncoder().fit_transform(test[attribute])
```

把訓練集分為訓練資料(x)對應 label(y)
將 x,y 切成用來訓練模型的 set 與驗證模型訓練情況的 set
然後對訓練資料做 normalize

```
#split trainData to train 15% to validate
x = drop_train.drop(['Attribute23'],axis=1)
y = drop_train['Attribute23'].values

x_train, x_validation ,y_train, y_validation = train_test_split(x, y,test_size=0.118, random_state = int(time.time()), shuffle=True)

x_train_norm = Normalizer().fit_transform(x_train)
print(x_train_norm.shape , y_train.shape , x_validation.shape , y_validation.shape )
```

```
(5930, 22) (5930,) (794, 22) (794,)
```

3.訓練模型：

建立模型 模型選擇 logistic regression
比對訓練的精準度 與 驗證的精準度

```
# 建立模型
logistic_regr = linear_model.LogisticRegression(C = 1 , solver='newton-cg' , warm_start = True)
logistic_regr.fit(x_train_norm, y_train)

Accuracy = logistic_regr.score(x_train_norm, y_train)
print(Accuracy)
```

```
0.8632377740303542
```

```
# 驗證模型
predictions = logistic_regr.predict(x_validation)
Accuracy = logistic_regr.score(x_validation, y_validation)
print(Accuracy)
```

```
0.8110831234256927
```

4.結果:

對 test 做預測 再按照繳交格式合併 輸出 result.csv

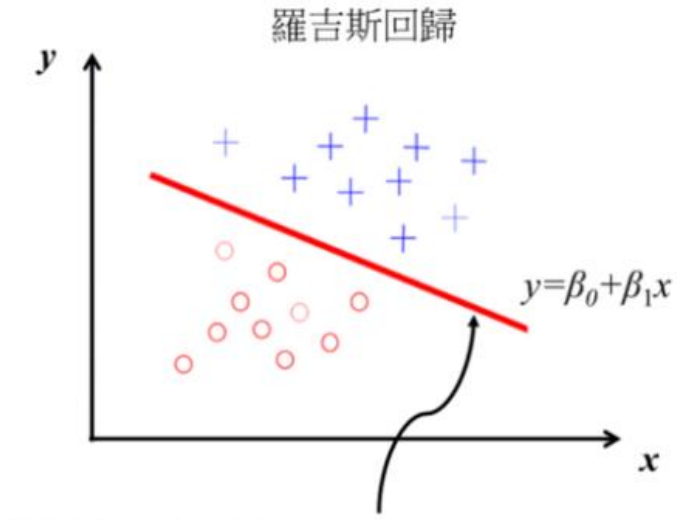
```
predictions = logistic_regr.predict(test)
id_list = list(range(len(predictions)))
for i in id_list:
    id_list[i] = str(id_list[i]) + '.0'
result = pd.DataFrame(list(zip(id_list, predictions)), columns = ['id', 'ans'])
result.to_csv('result.csv',index = False)
result
```

	id	ans
0	0.0	0
1	1.0	0
2	2.0	0
3	3.0	1
4	4.0	0
...	...	...
793	793.0	1
794	794.0	1
795	795.0	1
796	796.0	1
797	797.0	0

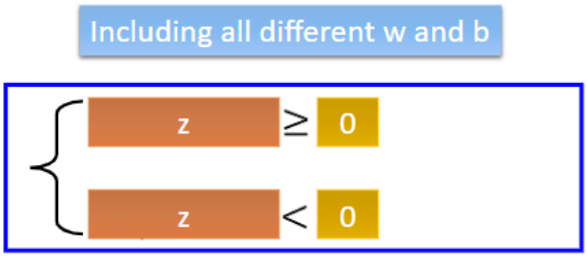
798 rows x 2 columns

b.演算法介紹

我用的模型是 logistic regression 將資料分為兩類
資料輸入後會計算每個 attribute 間的關係 特徵 距離 來得到權重
透過迭代做 Gradient Descent 來收斂函式 更改 β
函式的值代入 sigmoid 可以得到它的機率
機率>=0.5 就為 1(隔天會下雨) <0.5 為 0(隔天不會下雨)
會選用 logistic regression 的原因是題目輸出值只有 0 跟 1
所以我直覺就是要用這個模型 分為兩類的特性
看完資料內容 資料數 跟 task 目標 覺得一定適合 logistic regression



羅吉斯回歸希望
「找到那條紅線，讓資料可以區隔開來」



$$P_{w,b}(C_1|x) = \sigma(z)$$

$$z = w \cdot x + b = \sum_i w_i x_i + b$$

$$\sigma(z) = \frac{1}{1+exp(-z)}$$