# Hw2 Report

學號：r05921034
姓名：吳明憲

我的 python 版本是 3.5.2 若是使用 python2 可能會有不相容問題。

一開始我先將 data 去 shuffle,再利用 train_test_split(),將 traindata 跟 testdata 去分成 8:2 的資料量，利用 cross validation 去找出各個的最佳的參數。

Regression 我利用 grid search 去搜尋 C=[1,10,100,1000,10000]，參數 c 是約在 1 時有最佳值。
而其他參數是使用預設值，似乎差別不大。
而資料若有使用 feature scaling 去 normalize 似乎會使得 score 有些許下降，我的猜測是因為若將 data 標準化後，會使得 data 的 range 變小，使得 data 再 regression 不容易去區分。
以下的數值是我跑十次去做正確率的平均 比較 nonstandard 跟 standard

| Nonstandard | standard |
|---|---|
| 0.9163952226 | 0.9185667752 |
| 0.9348534202 | 0.9120521173 |
| 0.9131378936 | 0.9207383279 |
| 0.9250814332 | 0.9218241042 |
| 0.9207383279 | 0.9294245385 |
| 0.9381107492 | 0.904451683 |
| 0.9381107492 | 0.9087947883 |
| 0.9185667752 | 0.9272529859 |
| 0.9218241042 | 0.9348534202 |
| 0.9185667752 | 0.9283387622 |

R mean

| | |
|---|---|
| 0.9245385451 | 0.9206297503 |

Decision tree 我單純利用預設值的話，大約就有 90%左右的正確率， 一樣若是利用 feature scale 會比較差，而這裡的效果更加明顯，我想也許是再計算 gini 的時候會使得數值有較小的變動使得分類比較不好。

| nonstandard | standard |
|---|---|
| 0.9153094463 | 0.8371335505 |
| 0.8903365907 | 0.8935939197 |
| 0.9087947883 | 0.8718783931 |
| 0.9163952226 | 0.878393051 |
| 0.8859934853 | 0.7404994571 |
| 0.9142236699 | 0.8859934853 |
| 0.9077090119 | 0.8773072747 |
| 0.8925081433 | 0.8436482085 |
| 0.9218241042 | 0.8469055375 |
| 0.891422367 | 0.891422367 |

D mean

| | |
|---|---|
| 0.904451683 | 0.8566775244 |

svm 也是單純使用預設值就可以達到 80%以上，再加上 feature scale 之後，可以達到 90%以上的效果。

我嘗試去改動 kernel，不過會使得計算時間大幅提昇，若是使用 linear 跟 poly 似乎需要十分鐘以上且效果似乎沒有比較好。

| Nonstandard | standard |
|---|---|
| 0.8262757872 | 0.9305103149 |
| 0.8382193268 | 0.9370249729 |
| 0.8469055375 | 0.9305103149 |
| 0.8501628664 | 0.9229098806 |
| 0.8317046688 | 0.9185667752 |
| 0.8219326819 | 0.9305103149 |
| 0.8251900109 | 0.9381107492 |
| 0.8393051031 | 0.9239956569 |
| 0.8458197611 | 0.9153094463 |
| 0.8197611292 | 0.9305103149 |

S mean

| | |
|---|---|
| 0.8345276873 | 0.927795874 |

最後 nn 我是參考這篇 http://homepages.cae.wisc.edu/~ece539/project/s16/Edstrom_rpt.pdf
裡面使用三層 hidden layer 是 50,50 200
而我使用 hidden layer 是 100,100,200，maxiter=2000，iter 大於一千之後就進展不大，應該是已經在最佳點附近的震盪了。
nn 做了 feature scale 之後效果有大幅提昇，下表一樣是十次的結果做平均。

| Nonstandard | standard |
|---|---|
| 0.9120521173 | 0.9402823018 |
| 0.8371335505 | 0.9457111835 |
| 0.8675352877 | 0.9435396308 |
| 0.9022801303 | 0.9467969598 |
| 0.8023887079 | 0.9457111835 |
| 0.8762214984 | 0.9348534202 |
| 0.9077090119 | 0.9359391965 |
| 0.884907709 | 0.9283387622 |
| 0.8773072747 | 0.9424538545 |
| 0.9022801303 | 0.9424538545 |

nn mean

| | |
|---|---|
| 0.8769815418 | 0.9406080347 |

最後我嘗試使用了 random forest tree 去做，效果會比 decision tree 好，大概 3~4%。

結論:
在 svm 跟 nn 中使用了 feature scale 效果會好很多，而 regression 跟 decision 會有反效果。

我最後提交上去的 decision tree 是用 random forest 去實做
所以若是使用 python3 classification.py D train.csv test.csv
是使用 random forest
若使要用 基本的 decision
請使用 python3 classification.py T train.csv test.csv