一、 Introduction

Dataset: [Letter Recognition Data Set]

實驗目的：根據每個字母的長、寬、像素等等屬性建立辨識目標字母的模型。

實驗方法：kNN、SVM、J48、Naive Baye

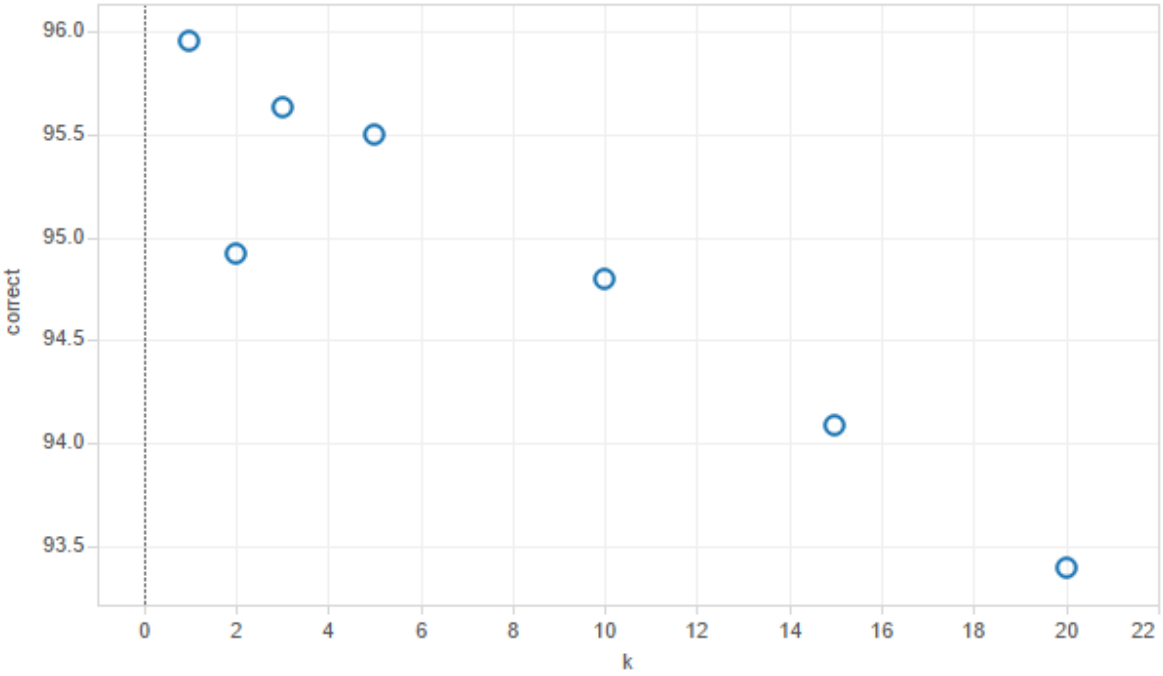
資料參數： lettr - A~Z 的大寫字母

x-box, y-box, width, high, onpix, x-bar, y-bar, x2bar, y2bar, xybar, x2ybr, xy2br, x-ege, xegvy, y-ege, yegvx：皆為整數型態的參數

二、 Experiments

1. kNN (K = 3 / 5 / 8 / 10 / 15 / 20)

下圖為在不同 K 值情況下，KNN 的準確率，K=1 沒什麼意義，因此選擇除 K=1 外正確率最高的 K = 3 做為 K 值。



2. Decision Tree （J48 ）

速度稍快，但正確率不高。

Correctly Classified Instances	17584	87.92 %
Incorrectly Classified Instances	2416	12.08 %

3. naive Bayes

此方法雖然速度夠快，但正確率太差。

Correctly Classified Instances	12802	64.01 %
Incorrectly Classified Instances	7198	35.99 %

4. Random Forest

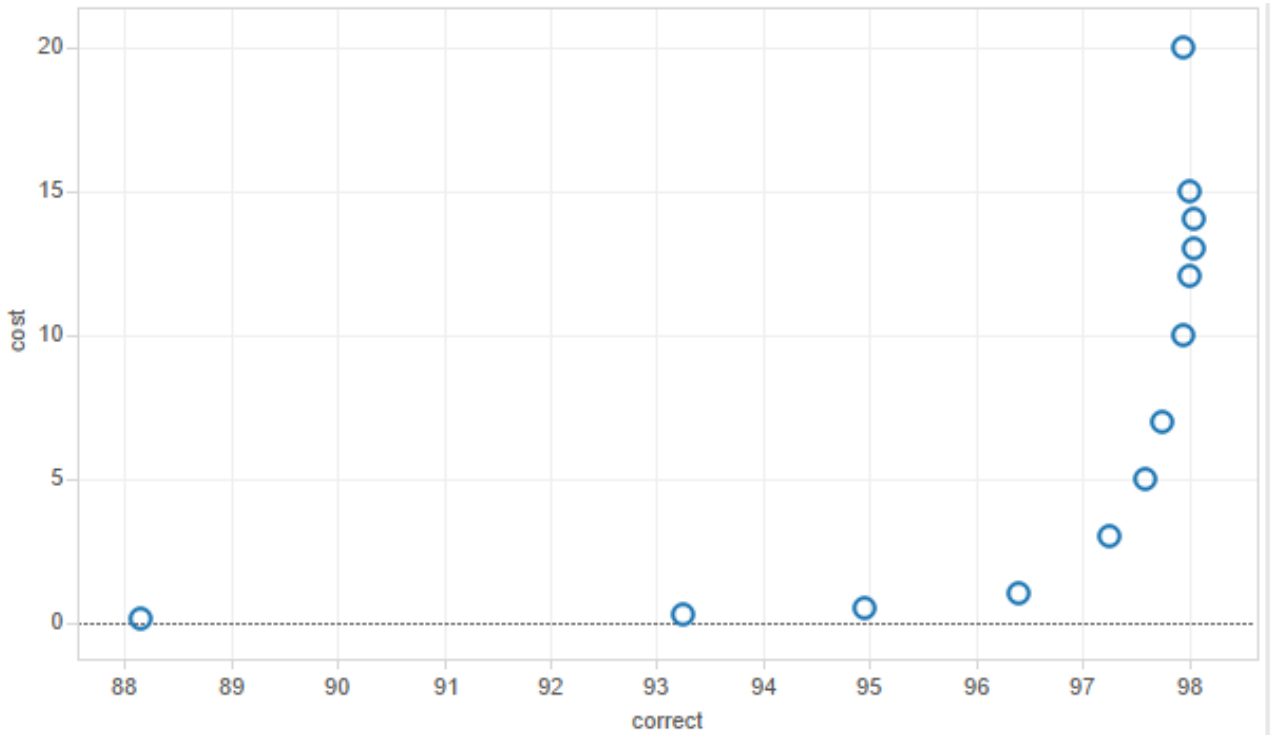
速度尚可，但正確率差 SVM 一點。

Correctly Classified Instances	19260	96.3 %
Incorrectly Classified Instances	740	3.7 %

5. SVM (cost = 10 / 5 / 1 / 0.5 / 0.1 , gamma = 0.1 / 0.3 / 0.5 / 0.8 / 1)

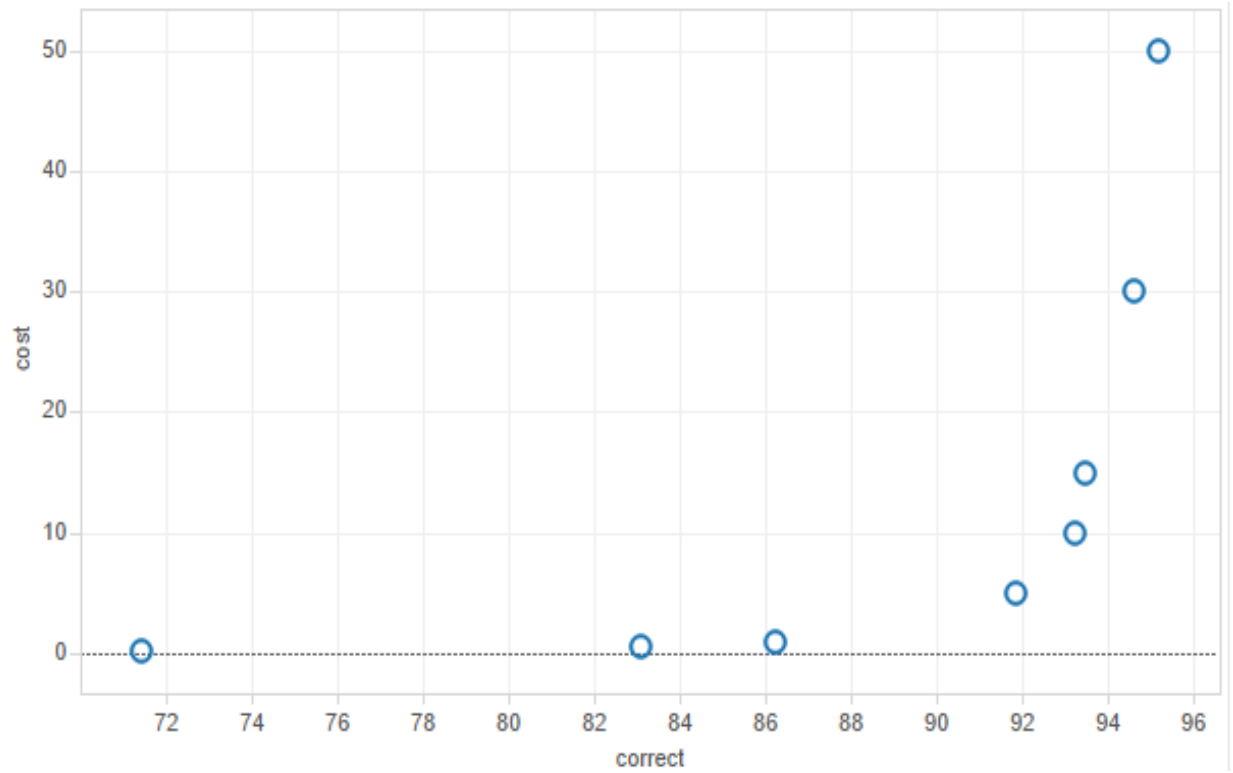
(1) gamma = 0.1 時，改變 cost 的正確率變化

在圖中可以清楚的看到正確率在 cost = 10~15 之間是最大值，大約 98%左右，cost 再往上增加的話會稍微下降一些，但仍然穩定在 97.5 上下。



(2) gamma = 0.01 時，改變 cost 的正確率變化

這張圖上在 cost = 30 時會有 95%左右的正確率，cost 再往上增加的效果有限，且需要加到非常大，因此認定 cost = 30 為極限，再往上增加，則可能造成 overfitting。



(3) overall staistics　　(各英文字母間的詳細數據)

```
Overall Statistics

               Accuracy : 0.9475
                 95% CI : (0.9401, 0.9542)
    No Information Rate : 0.042
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9454
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E Class: F Class: G Class: H Class: I Class: J Class: K
Sensitivity            0.9762  0.97163   0.9429  0.94340  0.92121  0.95732  0.92903  0.86624   0.9630  0.96711  0.90972
Specificity            0.9995  0.99430   0.9995  0.99740  0.99713  0.99739  0.99506  0.99558   0.9995  0.99922  0.99741
Pos Pred Value         0.9880  0.86164   0.9851  0.93750  0.93252  0.94012  0.88344  0.88889   0.9873  0.98000  0.92908
Neg Pred Value         0.9990  0.99896   0.9979  0.99766  0.99661  0.99817  0.99713  0.99454   0.9984  0.99870  0.99663
Prevalence             0.0420  0.03525   0.0350  0.03975  0.04125  0.04100  0.03875  0.03925   0.0405  0.03800  0.03600
Detection Rate         0.0410  0.03425   0.0330  0.03750  0.03800  0.03925  0.03600  0.03400   0.0390  0.03675  0.03275
Detection Prevalence   0.0415  0.03975   0.0335  0.04000  0.04075  0.04175  0.04075  0.03825   0.0395  0.03750  0.03525
Balanced Accuracy      0.9878  0.98297   0.9712  0.97040  0.95917  0.97736  0.96205  0.93091   0.9812  0.98316  0.95356
                     Class: L Class: M Class: N Class: O Class: P Class: Q Class: R Class: S Class: T Class: U Class: V
Sensitivity            0.92754  0.95652  0.96226  0.92157  0.94194  0.97436  0.92157  0.94545  0.96970   0.9452  0.92073
Specificity            0.99922  0.99740  0.99870  0.99792  0.99948  0.99662  0.99454  0.99765  0.99922   0.9990  0.99896
Pos Pred Value         0.97710  0.93902  0.96835  0.94631  0.98649  0.92121  0.87037  0.94545  0.98160   0.9718  0.97419
Neg Pred Value         0.99742  0.99818  0.99844  0.99688  0.99766  0.99896  0.99687  0.99765  0.99870   0.9979  0.99662
Prevalence             0.03450  0.04025  0.03975  0.03825  0.03875  0.03900  0.03825  0.04125  0.04125   0.0365  0.04100
Detection Rate         0.03200  0.03850  0.03825  0.03525  0.03650  0.03800  0.03525  0.03900  0.04000   0.0345  0.03775
Detection Prevalence   0.03275  0.04100  0.03950  0.03725  0.03700  0.04125  0.04050  0.04125  0.04075   0.0355  0.03875
Balanced Accuracy      0.96338  0.97696  0.98048  0.95974  0.97071  0.98549  0.95805  0.97155  0.98446   0.9721  0.95984
                     Class: W Class: X Class: Y Class: Z
Sensitivity            0.96711  0.96528  0.98693  0.98450
Specificity            0.99870  0.99818  0.99844  0.99948
Pos Pred Value         0.96711  0.95205  0.96178  0.98450
Neg Pred Value         0.99870  0.99870  0.99948  0.99948
Prevalence             0.03800  0.03600  0.03825  0.03225
Detection Rate         0.03675  0.03475  0.03775  0.03175
Detection Prevalence   0.03800  0.03650  0.03925  0.03225
Balanced Accuracy      0.98290  0.98173  0.99268  0.99199
```

(4) confusion matrix of SVM

```
Confusion Matrix and Statistics


pred   A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W   X   Y   Z
   A 164   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0   0   0   0   0   0
   B   0 137   0   2   1   1   0   4   0   0   0   0   3   1   0   0   0   2   1   1   0   5   1   0   0   0
   C   1   0 132   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   D   0   0   0 150   0   1   3   2   0   0   3   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
   E   0   0   4   0 152   0   0   0   0   2   0   3   0   0   0   1   0   0   1   0   0   0   0   0   0   0
   F   0   0   0   0   0 157   1   0   0   0   0   0   0   0   0   5   0   0   3   0   0   1   0   0   0   0
   G   0   1   1   0   8   0 144   3   0   0   0   3   0   0   1   1   0   0   0   1   0   0   0   0   0   0
   H   0   0   1   3   0   1   0 136   0   0   3   1   2   1   0   0   0   2   1   1   1   0   0   0   0   0
   I   0   0   0   0   0   0   0   0 156   1   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
   J   0   0   0   0   0   0   0   0   3 147   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   K   0   0   2   0   0   0   0   2   0   0 131   0   0   0   0   0   0   3   0   0   0   0   0   3   0   0
   L   1   0   0   0   0   0   1   0   0   0   0 128   0   0   0   0   0   0   1   0   0   0   0   0   0   0
   M   1   0   0   0   0   0   1   1   0   0   0   0 154   1   0   0   0   0   0   0   3   1   2   0   0   0
   N   0   0   0   0   0   0   0   0   0   0   0   0   2 153   0   0   0   2   0   0   0   1   0   0   0   0
   O   0   0   0   3   0   0   0   0   0   0   0   0   0   1 141   2   2   0   0   0   0   0   0   0   0   0
   P   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0 146   0   0   0   0   0   0   0   0   0   0
   Q   0   0   0   0   0   0   2   1   0   0   0   0   0   0   6   0 152   1   0   0   0   0   0   2   0   1
   R   0   2   0   1   0   0   2   7   0   0   6   0   0   2   1   0   0 141   0   0   0   0   0   0   0   0
   S   0   0   0   0   1   0   0   0   2   1   0   3   0   0   0   0   0   0 156   1   0   0   0   0   0   1
   T   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0 160   0   0   0   0   1   0
   U   0   0   0   0   0   0   0   1   0   0   0   0   0   0   2   0   0   0   0   0 138   0   1   0   0   0
   V   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2 151   1   0   1   0
   W   0   0   0   0   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   2   1 147   0   0   0
   X   0   1   0   0   0   0   0   0   1   1   1   0   0   0   0   0   0   1   1   1   0   0   0 139   0   0
   Y   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   4   0   0 151   0
   Z   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 127
```

三、 Discussion

1. Model Effectiveness

使用 R 內建函數 Tune 測試最佳 cost 與 gamma 值，歷經三十幾個小時的計算，結果如下圖：

```
> tuned <- tune.svm(lettr ~ ., data=data , gamma = 2^(-5:3),cost = 2^(-5:5))
> summary(tuned)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
 0.125    8

- best performance: 0.02165

- Detailed performance results:
     gamma      cost    error  dispersion
1  0.03125  0.03125 0.29235 0.010159970
2  0.06250  0.03125 0.24435 0.009162272
3  0.12500  0.03125 0.20965 0.008406512
4  0.25000  0.03125 0.19640 0.008906302
5  0.50000  0.03125 0.28530 0.022529240
6  1.00000  0.03125 0.85465 0.022370678
7  2.00000  0.03125 0.95845 0.002929259
                  =
                  =
                  =
                  =
73 0.03125  8.00000 0.04015 0.002838720
74 0.06250  8.00000 0.02750 0.003527668
75 0.12500  8.00000 0.02165 0.003504362
76 0.25000  8.00000 0.02290 0.003089408
77 0.50000  8.00000 0.02795 0.003370213
                  =
                  =
                  =
```

error最低，Best performance

以上圖之結果代入 SVM 做測試，準確率最高的為 SVM(cost=8, gamma=0.125) ，有 98.2%，下圖的函數中 spilt = 0.1 表示取 10%做為 Test data。

```
> my_svm(data=data,spilt=0.1,cost=8,gamma=0.125)
[1] 98.2
```

2. Model Complexity // time complexity

演算法	速度	準確度
kNN	2 分鐘以內	約 96%
naive Bayes	20 秒以內	約 64%
Random Forest	4 分鐘以上	約 96%
SVM	9 分鐘以內/4 分鐘以內	約 98% / 約 94%
ANN	30 分鐘以上	約 87%

3. Comparison to others

這裡主要比較 kNN 和 naive Bayes。這兩種演算法的速度都非常快，其中 kNN 的準確率非常高，達到 95.635%，而 naive Bayes 雖然是速度最快的，但準確率只有 64.01%，遠小於其他的演算法。

另外，雖然 KNN 看似正確率高，但 KNN 的 Run Time 過長是一個很大的缺點，如果希望做出一個 model 是可以持續使用的，那麼 kNN 就不是個好選擇。

而其餘像是 Decision Tree，做出來的效果稍差一些，因此我們繼續嘗試其他更高的可能性。至於 Random Forest 也能做到 95%以上的正確率，速度也算是蠻快的，不過正確率還是差 SVM 一些。

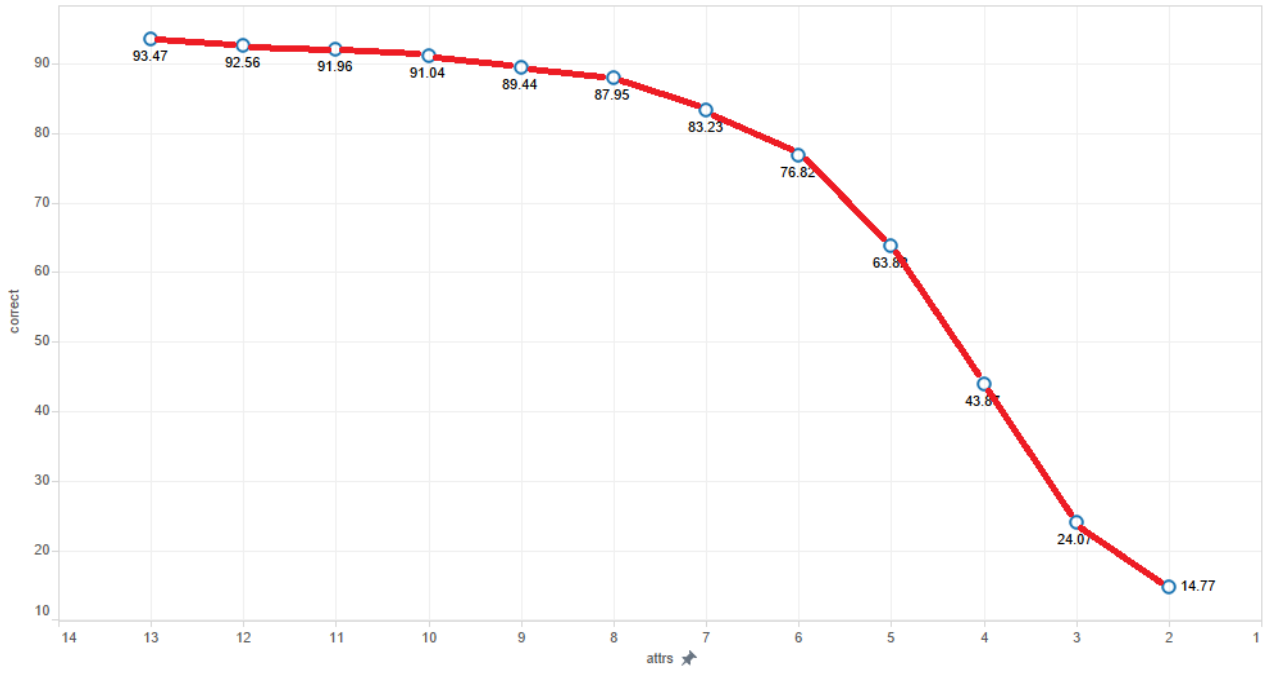
4. Creative ideas

嘗試著先在 dataset 上面做 Preprocessing，使用 PCA 來做 dimension reduction，因為不管是哪個演算法，如果 dataset 的 attribute 變少的話，都能夠或多或少的提升運算速度。

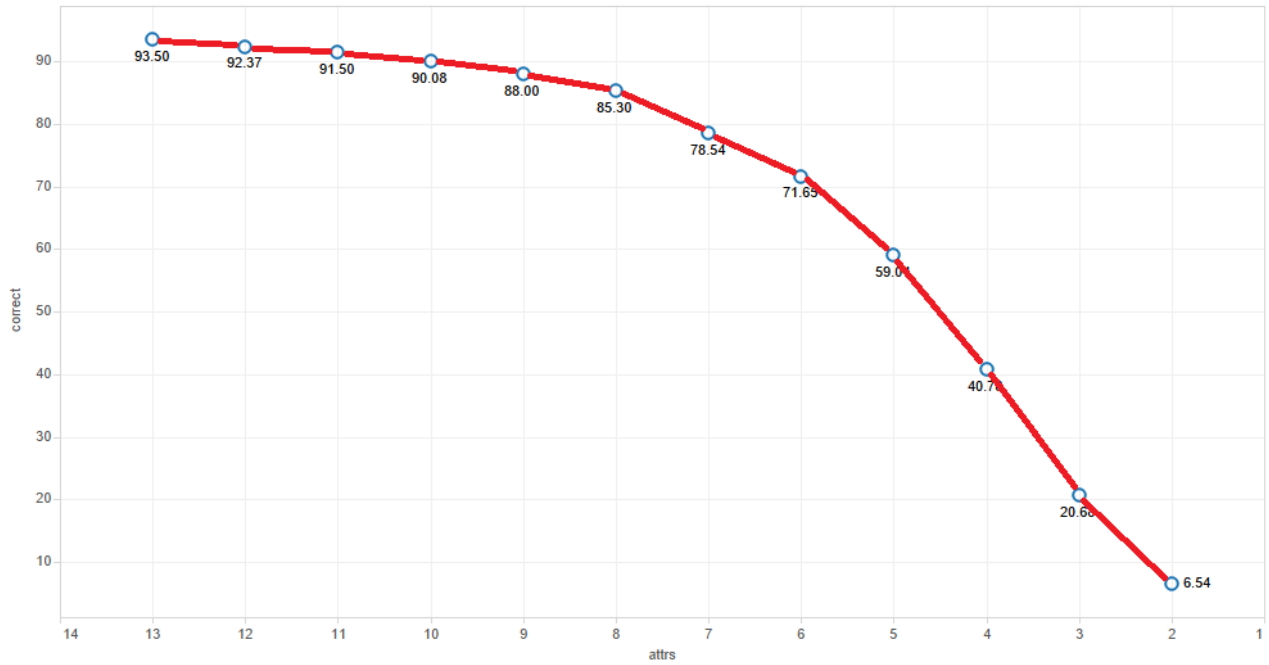
原始資料在 PCA 之後參數量從 16 降低到 13，實測結果速度提升一倍以上，準確率雖有降低但卻在容忍範圍內，總的來說做 PCA 是值得的。

而下圖中的實驗，目的是測試若在 PCA 之後再繼續降低參數量的話，正確率會如何變化，結果顯示繼續降低參數量只會讓準確率變低，因此 PCA 之後就不宜再繼續降低參數量了！

(1) Random Forest PCA 之後，繼續刪除參數的正確率變化



(2) SVM 進行 PCA 之後，繼續刪除參數的正確率變化 (持續降低正確率)



四、References

1. R 語言上的 e1071 套件 (SVM)
http://www.inside-r.org/node/57517
2. Support Vector Machines ∗ The Interface to libsvm in package e1071
https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf
3. 應用 R 語言於資料分析 – 從機器學習、資料探勘到巨量資料
2015 年 6 月 松崗出版社 – 李仁鐘 著