NCKU Data Science 2018

Competition 1

Kobe Bryant's Shot Type Prediction

Group25

醫工 107	F94036089	吳旻昇
藥學 107	I84031111	何明洋

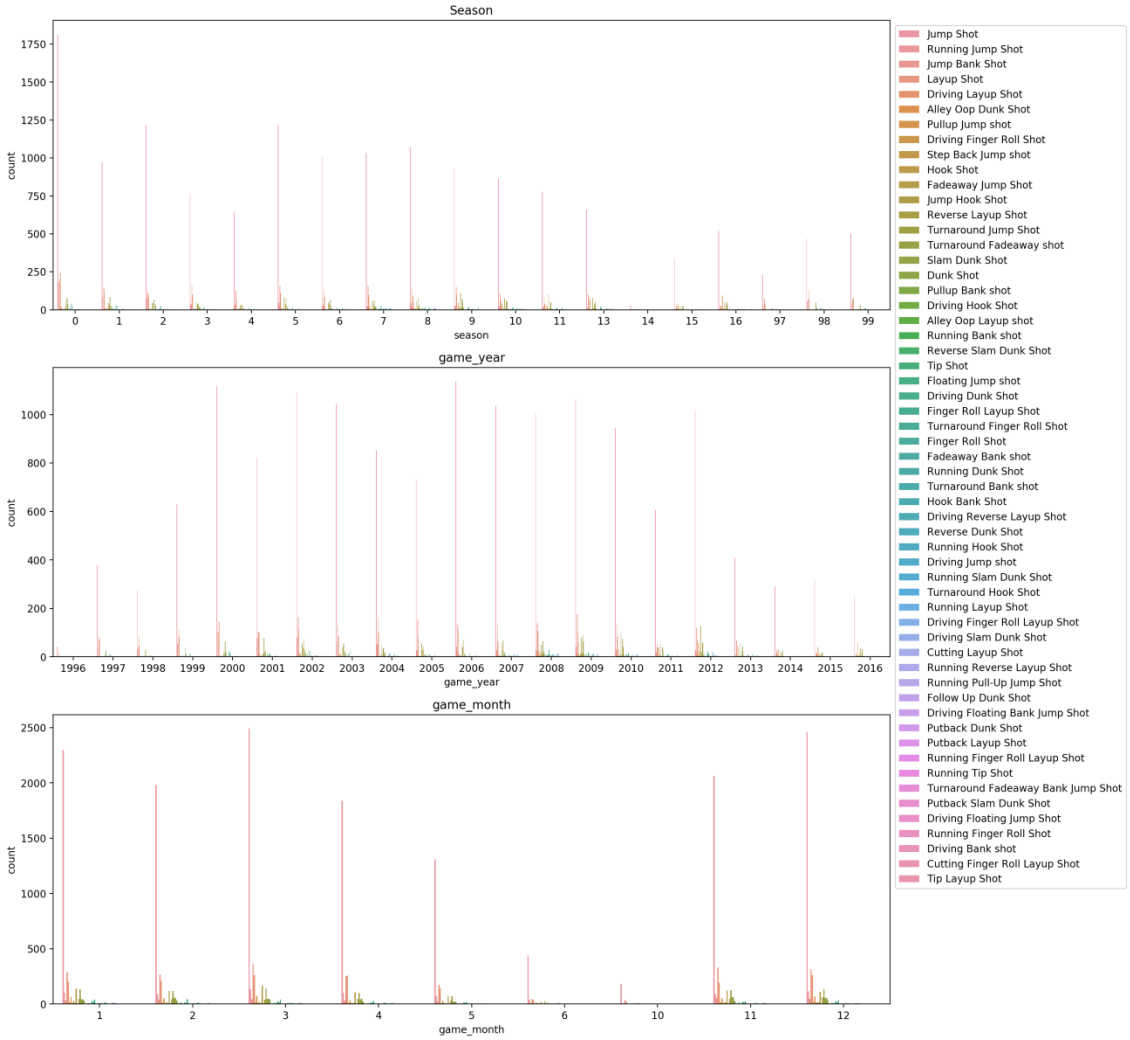
Description

我們將整個預測過程分為兩部分進行，首先是**資料預處理**，在其中我們透過繪製一些視覺化圖表判斷哪些是適合作為預測模型之特徵，又哪些可能導致偏差，並根據這些資訊整理出最後用來 train 之 data。**第二部分**則是使用 **XGBoost**(Extreme Gradient Boosting) 監督式學習演算法來根據 test 資料預測結果。

資料預處理部分，在 import data 後我們先對不同的 action type 統計每一種在資料中出現的總數，並曾嘗試去掉一些數量過少的 action type 看是否能提升預測精準度，惜最後成效不彰。思索 action type 可能會與投射之位置最有關係，但 16 種特徵資料中其中 lat、lon、loc_x、loc_y、shot_distance 這 5 個描述位置資料似乎彼此之間有重複性，我們先對這 5 個特徵互相作圖進行比對，的確發現 lon 與 loc_x 完全正相關，而 lat 與 loc_y 亦是(如**圖一**)，因此為避免重複考慮位置因素，這兩者應該取其一即可。

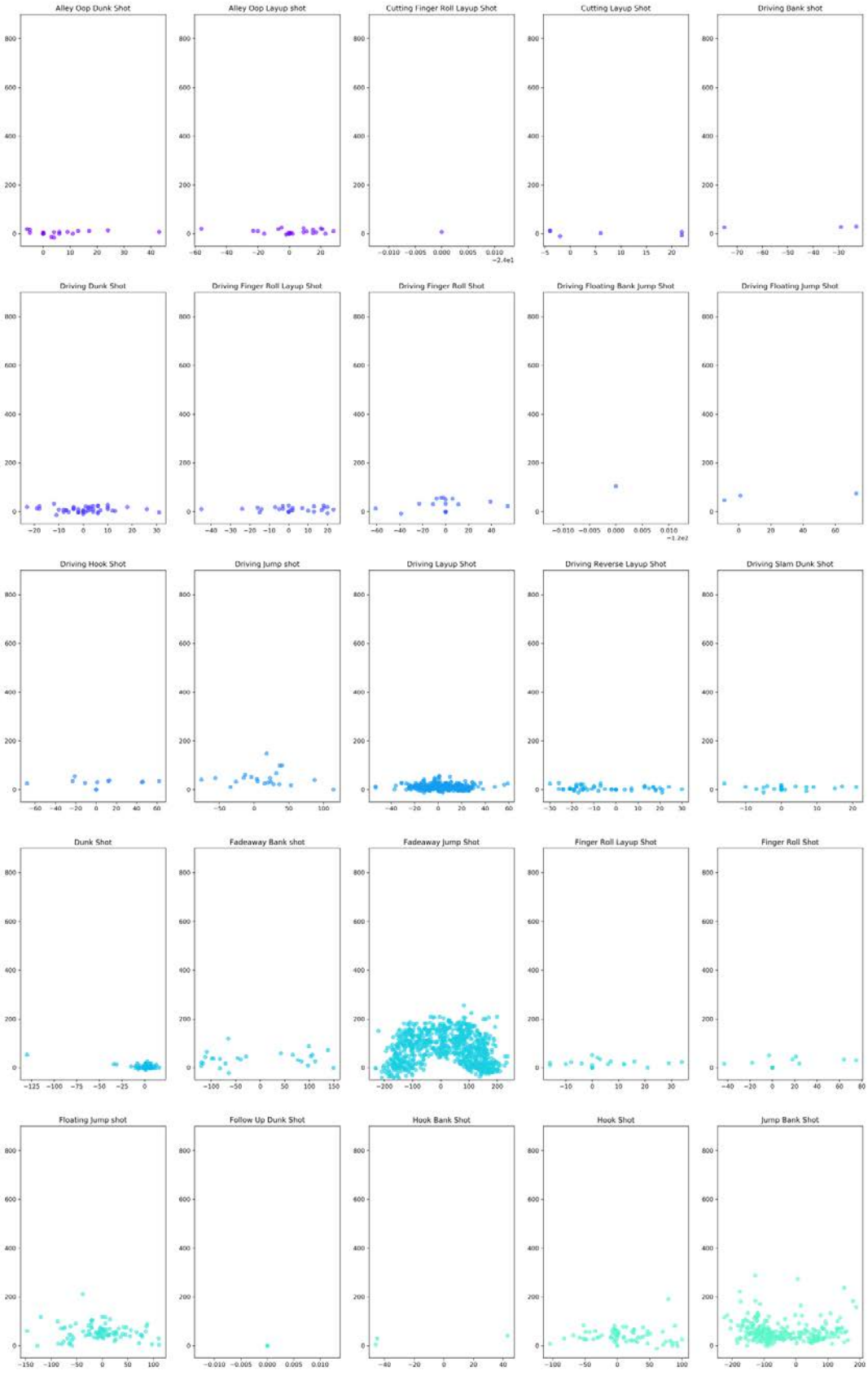
▲圖一

接著我們處理 season 部分，因其中資訊混亂，同時包含文字資料與數字，我們將在"-"後的數字取出後將之改成正整數之數據。此外也對剩餘時間進行處理，因原本給的 data 中剩餘時間的分與秒是分開的，我們將兩者合併改成整體所剩秒數，另外存成 remaining_time。而比賽時間部分則是將比賽年分(game_year)與月份(game_month)再分開儲存。處理完以上資料後我們針對 season、game_year 、game_month、period、playoffs、opponent、shot_zone_area、shot_made_flag 對不同的 action type 作圖，以判斷這些特徵是否適合作為訓練的資料(如**圖二**，僅節錄部分)。



▲圖二

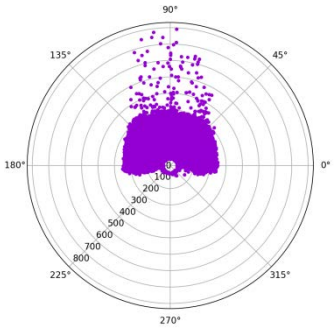
接著我們想確認投射位置對於不同的 action type 是否真的具很大的特異性，因此再對兩者作圖檢視(如**圖三**，僅節錄部分)。


Alley Oop Dunk Shot
Alley Oop Layup shot
Cutting Finger Roll Layup Shot
Cutting Layup Shot
Driving Bank shot
Driving Dunk Shot
Driving Finger Roll Layup Shot
Driving Finger Roll Shot
Driving Floating Bank Jump Shot
Driving Floating Jump Shot
Driving Hook Shot
Driving Jump shot
Driving Layup Shot
Driving Reverse Layup Shot
Driving Slam Dunk Shot
Dunk Shot
Fadeaway Bank shot
Fadeaway Jump Shot
Finger Roll Layup Shot
Finger Roll Shot
Floating Jump shot
Follow Up Dunk Shot
Hook Bank Shot
Hook Shot
Jump Bank Shot

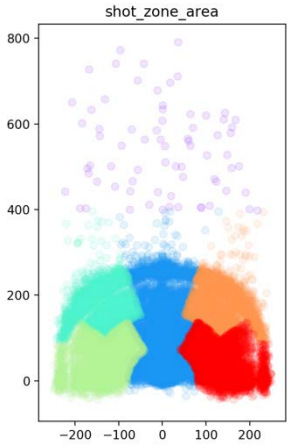
▲圖三

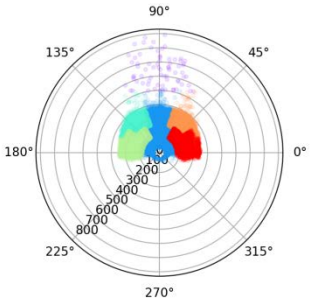
至此，我們認為若把直角坐標系轉換為極座標，直接考慮與籃板的投射距離與角度或許更貼近籃球員在使用不同 action type 所考慮之因素，因此將資料進行極座標轉換(如**圖四**)

。



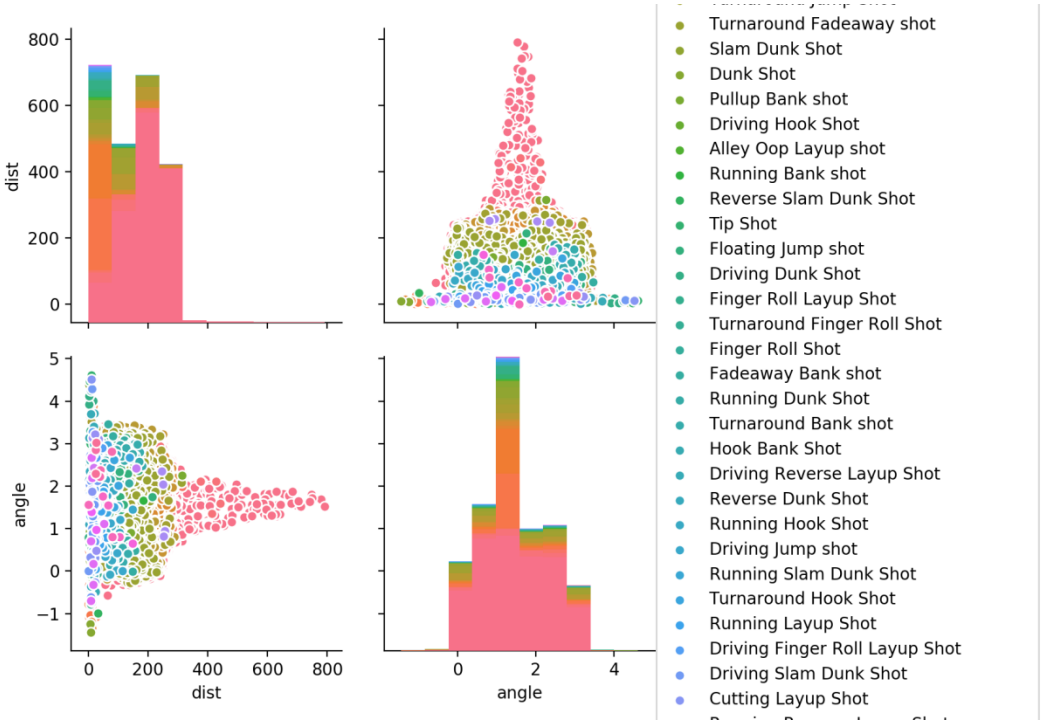
▲圖四



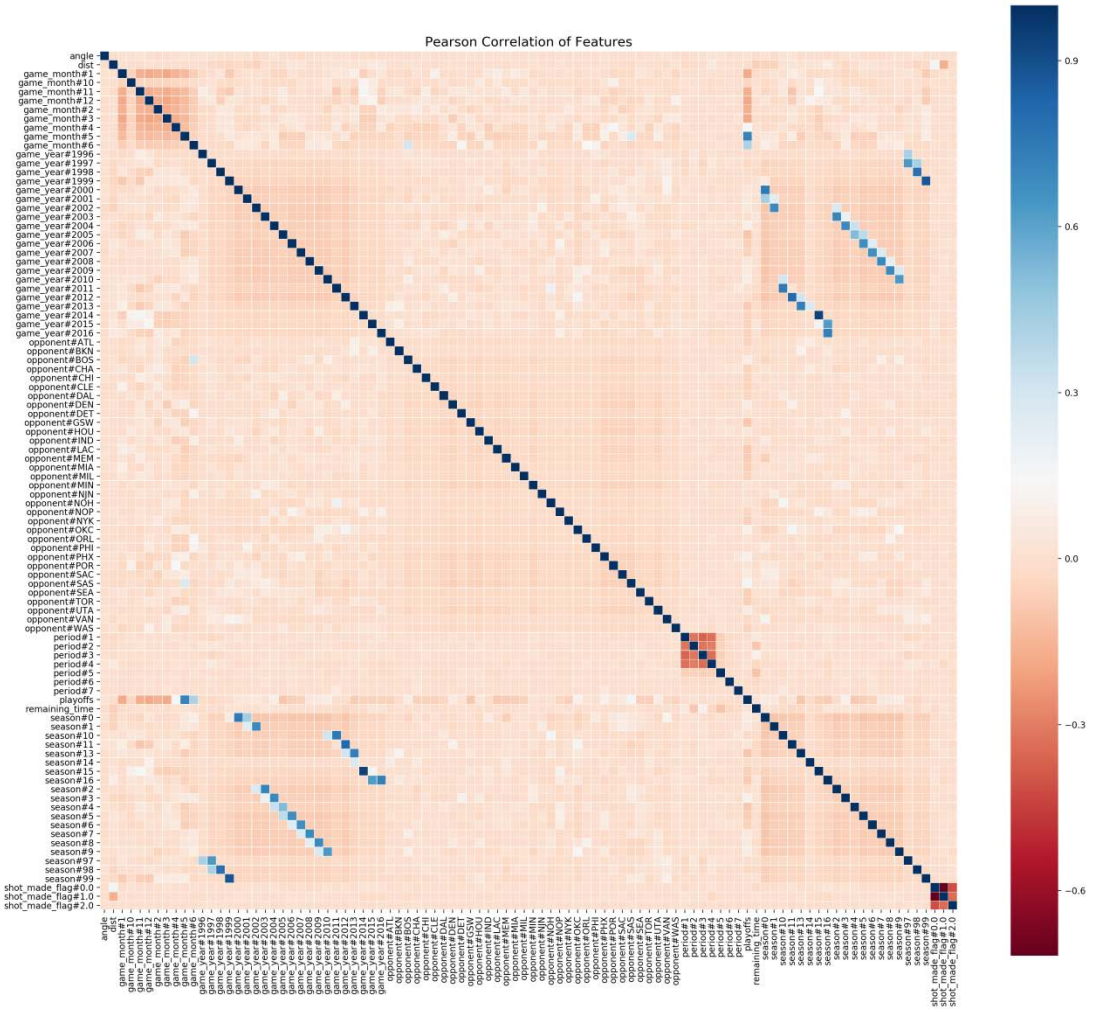


▲圖五

使用轉換過極座標之 distance 比對原先 data 中投射 distance 資料，可以發現其是一致的(如**圖五**)，也因為有了與籃板的角度、距離更細部的位置資訊可供模型去學，我們可去除 shot_zone_area 這項冗餘特徵。最後再確認不同 action_type 在極座標中分布確實是有特異性的(如**圖六**)。



▲圖六



▲圖七

另外因為 shot_made_flag 裡有缺失一些數據，我們將這些空缺先填成 2。根據以上分析，我們將用不到的特徵剃除，並把一些類別資料做 one-hot encoding 成啞變數以利之後的模型訓練。接著我們使用 heatmap(如**圖七**)去看資料特徵彼此間的 Pearson Correlation Coefficient，根據此結果，我們使用 PCA 將具備高度相關的特徵合併，合併後亦經過測試其 variance 提高許多。

而最終拿去進行模型訓練的特徵資料有：playoffs、remaining_time、dist、angle、period、season、game_year、game_month、opponent、shot_made_flag 轉啞變數後之經 PCA 互相降維之特徵，將這些存成 pickle 檔以利接下來使用。

第二部分我們使用 XGBoost 作為本次之模型訓練演算法，會以這個方式的原因是在 Kaggle 的很多比賽中，許多獲勝者皆使用 XGBoost，而其成果也非常卓越，在理解之中演算法之原理後，我們最後也採取 XGBoost 作為本次競賽最後的模型

建立方式。

資料分組部份我們 test_size 設置為 0.20，以 train data 比上 test data 為 4:1 的比例進行模型訓練，此外其他參數設置部分為 max_depth=4 (避免 over-fitting)、n_estimators=160 (tree 的個數不宜太多但也不宜太少) 、learning_rate=0.05 (若太小則收斂速度會太慢且容易陷入局部最優點，若太大則會在收斂時震盪過劇烈)，最終把要預測的資料置入進行預測，將結果另存成檔案丟入比賽的網站中，所得 accuracy 為 0.7213。

Analysis

在這次比賽中我們能提高 accuracy 原因：

1.使用 Xgboost

Xgboost 是在 GBDT(Gradient Boosting Decision Tree)基礎上改進 boosting 演算法，其強勢之處在於形成一棵決策樹後，下一棵樹是從上一次的預測基礎中取最優的一步進行分裂/建樹，最後加總所有決策樹的預測結果得到答案，在這樣最優化的狀況下預測精準度較其他演算法高，也就成為競賽中之翹楚。

除了使用 Xgboost 外，我們也使用了 SVM、Random Forest、Lightgbm 等演算法測試，然效果都沒有 Xgboost 好。

在參數調整的部分，我們也測試了許多種組合，最後從之中選了一個 accuracy 較佳的作為使用。

NO.	Max_depth	N_estimators	Learning_rate	刪除次數過小樣本	Test_accuracy
1	3	300	0.05	X	**0.7090**
2	3	500	0.05	X	**0.7057**
3	3	100	0.05	V	**0.7093**
4	3	100	0.05	X	**0.7111**
5	3	150	0.05	X	**0.7116**
6	3	200	0.05	X	**0.7121**
7	3	250	0.05	X	**0.7103**
8	3	200	0.05	X	**0.7199**
9	3	150	0.05	X	**0.7208**
10	3	100	0.05	X	**0.7189**

11	3	180	0.05	X	**0.7202**

2.無剔除樣本過小的 action type

原先在觀察資料時有發現許多 action type 只有三組甚至一兩組的 data，在考慮樣本太小可能導致最後模型建構有偏差的狀況下，我們一開始剔除了樣本數小於 3 的 action type，然而最後 accuracy 並沒有提升。之後也測試了剔除 50 以下、100 以下等樣本，而 accuracy 亦無提升。

後來觀察了數據後認為整體而言有三大 action type 數量上千可能導致訓練結果有極大偏差，因此測試了一次剔除 1000 以下的 action type，在本身 test 資料的測試中 accuracy 來到 90%以上，然而最後產生預測結果傳回比賽網站中後依然也只能得到 0.71 左右的 accuracy，合理推測是因為沒考慮其他情形而 overfitting，因剔除樣本數過少的方式並無有效提升 accuracy，因此我們最後選擇不剔除。

3.直角坐標系轉換成極座標系

正常投籃應是考量與籃板間之距離與角度，我們將直角坐標系轉成極座標，而原先資料中有提供 distance 這組數據，以及將投籃位置分區的數據，我們轉成極座標候用這兩組數據進行驗證，得到完全的正相關，因此可證明轉成極座標並無太大問題，此外此法亦可將 x、y、dist 三者構成三維資料變成二維，在模型建構時參數權重亦可下降。實際上亦經測試如使用直角坐標，accuracy 的確比較差。

4.剔除彼此間正相關之特徵

如 loc_x、loc_y 與經度緯度四者倆倆匹配，經過我們一開始作圖分析後發現其實具完全正相關，因此只要考慮其中一者即可，避免參數權重過重，然而我們後來使用極座標後就直接避免掉這問題。

5.使用 PCA 合成正相關之特徵並降維

一開始我們並沒有使用 PCA 對一些相似度過多的特徵降維，預測精準度一直停留在 0.70 附近，而後對所有特徵進行 PCA 分析後也看不出 PCA 的好處。

但在思索許久後，我們以最後選定要做為模型訓練之特徵以 heatmap 互相比較，可以看見其實還有許多資料之間具極大正相關，可能也會導致前述的特徵參數權重過重，比起直接對全部的資料做 PCA 這樣的方法才能有效處理資料，因此透過 heatmap 選取其中藍色者(正相關大者)，以 PCA 組合降成一維資料，並作為最後建構模型使用，最後確實有效提升 accuracy 至 0.72。

6.大量用圖表觀察特徵間關係

我們能夠判斷哪些特徵具有特異性是因為在數據前處理時有使用大量的圖表來判讀，如觀察 action type 在不同特徵其中不同數據分布，最後選出來的特徵都是對不同的 action type 有特異性的。

7. 不剔除 shot_made_flag 特徵

一開始我們認為 shot_made_flag 這個特徵並不重要，因此在 Accuracy 上一直停留在 0.70 附近，然而最後加入這個特徵進行模型建構在透過上述的 PCA 一起降維，才將 accuracy 提升至 0.72 以上，由此可知在一開始資料判讀還是具有很大的重要性，究竟那些特徵適合作為模型建構使用是一大重點。

References

本次我們使用 xgboost 演算法作為模型訓練使用，其所用的方式是決策樹(回歸樹)整體安裝與使用 Xgboost document 參考以及網路上其他人使用的資料
http://xgboost.readthedocs.io/en/latest/