

# NBA勝隊預測

組別:A組

組員:植微四 馬詠芝、植微四 王靖瞳、心理四 徐聖璇、心理四 李彥廷、生醫電資所博三 王子毅

## 一、主題:為什麼要做這題目?

我們這組的組員對於 NBA 比賽非常有興趣且有一定程度的研究,也時常會去各種 NBA 網站瀏覽相關資訊,因次常常各種 NBA 賽事預測的消息。我們很想知道這些預測到底是怎麼做出來的,需要使用什麼樣的參數,預測結果的可靠性有多高,而這些預測方法是否有什麼改善空間。因此,我們透過本課程期末報告的機會深入探討「NBA 比賽勝負預測」這個熱門的題目,利用現有的 NBA 球隊和球員相關數據來預測兩隊比賽的勝負,看我們是否能用課程中所學到的工具與知識來做出更好、更合理的預測模型。

## 二、建模、調整方式與成果

### (一)資料整理:

透過 kaggle 等網站取得過去所有 NBA 比賽的數據、歷年球員得獎紀錄、歷年全明星賽名單等資料之後,我們進行了以下處理:

#### 1. 資料篩選:

在球隊的比賽數據方面,我們選擇使用傳統數據來做分析。由於年代較久遠的資料不齊全且沒有太大的參考價值(比賽規則與現在 NBA 不同),因此我們第一步先篩選出 game\_date 大於等於 "1990-11-01" (1990-91 賽季以後) 的資料,並刪除其中資料不齊全的場次。

在球隊的球員資料方面,我們選擇用「入選當年明星賽球員數」、「過去三年入選 NBA 年度前三隊球員數」、「過去三年名列年度 MVP 排行榜球員分數」等資料作為球隊陣容實力的指標。

#### 2. 資料前處理:

##### (1) 定義隊伍的簡稱:

由於 1990~ 2023 期間有些球隊改名或搬遷到其他城市,這些隊伍的 ID 及全名相同,但卻有不同的簡寫代號,因此我們用字典整理這些隊伍的 ID 與簡寫代號之間的對應關係,並將同一球隊不同的簡稱改成一個統一的簡稱。

##### (2) 資料代換

將 wl\_home 欄位中的 'W' 和 'L' 值替換為數值 1 和 0。

##### (3) 定義新參數

透過畫圖分析,我們發現主客隊資料相減所得的差值,比起原始的數字更能夠呈現各項資料與比賽勝負的關係,故我們將主客隊的相同特徵值兩兩相減,以此創造衍生欄位 "dev" 來代表這些差值。舉例來說,我們會將原始數據中的 fta\_home (主隊罰球次數) 減去 fta\_away (客隊罰球次數),以 "fta\_dev\_home" 代表所得到的差值 (即主客隊罰球次數差)。

##### (4) 資料重組:

在查閱過去的文獻時,我們發現以球隊「近期數據」來預測比賽勝負的效果比較好,因此我們要計算某支球隊在特定場次之前 N 場的平均數據值。由於原本資料形式的一個 entry 包含了主隊與客隊兩個隊伍的資料,故需先將主隊與客隊資料切開並往下併,變成長表格(long\_table),並按照 season\_id、game\_id、game\_date 排序。

(5) 計算某球隊前 N 場比賽的平均值：

如上，由於我們再做預測時，需要用在某一球季中，某一球隊在擬預測之比賽前 N 場的資料平均為 feature，所以我們以 season\_id 及隊伍名稱分組，用 rolling 函式對 long\_table 中的資料進行滾動計算平均值(rate)。其中，我們有測試並觀察不同的 N 值(7~10)對於預測的準確率是否有影響，但因我們發現影響不大，所以後面固定設 N =10。

(6)合併回寬表格並輸出：

我們使用 pd.merge 函式將主隊和客隊資料集根據 season\_id、game\_id、team\_name 和 order 欄位進行合併，並檢查合併後的資料集不包含重複的欄位。

(7)併入球員數據：

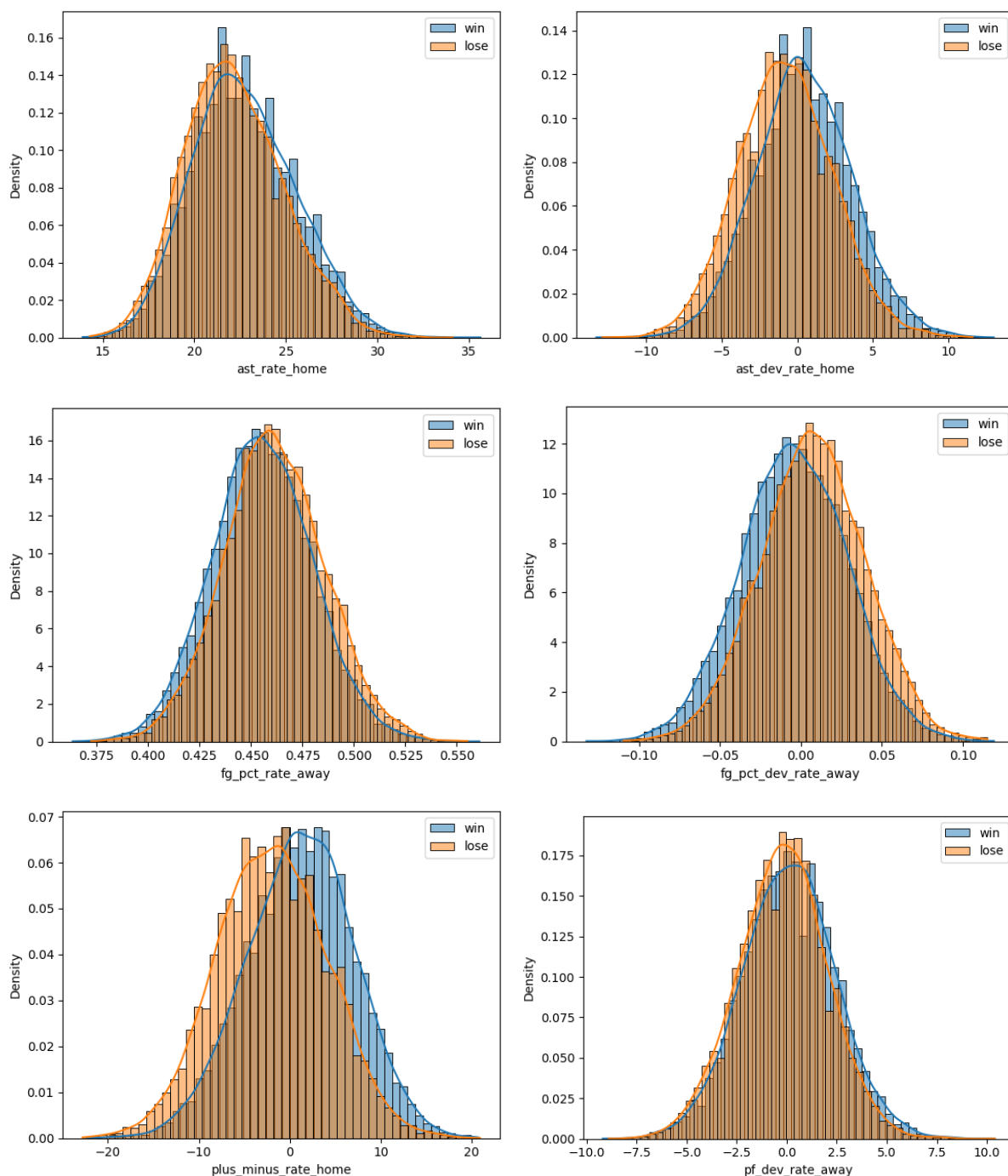
我們將球隊比賽數據和球員數據以 game\_id 作為合併的 key，與上面的資料表 merge 成最終的表格。最後將資料存成 CSV 檔案輸出，以用於訓練資料。

### 3. 製作用於預測的資料表格

由於我們所製作的使用者介面，要讓使用者選擇欲預測之比賽日期與主客隊名稱，因此我們另外寫了能夠按條件抓出主客隊資料的程式。

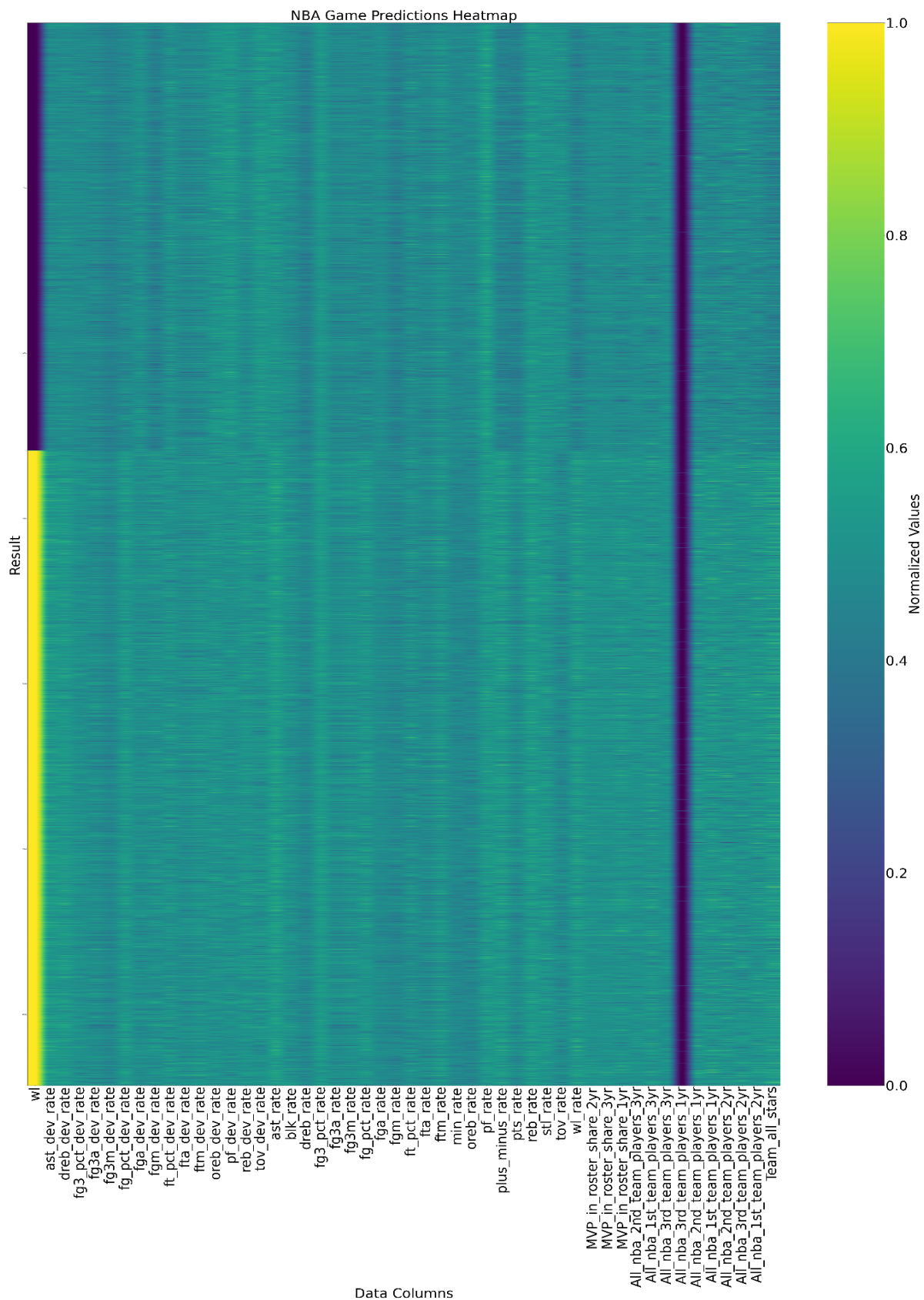
由於我們在預測時只需要計算球隊前 10 場平均數據，而不須去計算球員的數據 (假定陣容不變)，因此我們製作了包含所有場次前 10 場平均的長表格，和各場次球員資料的長表格，並寫成了一個 module 'get\_data'，只要利用其中的 function 'get\_data(欲預測之比賽日期, 主隊名, 客隊名)'，我們就能自動產生對應的csv，供模型預測結果。

## (二) 特徵值分析



圖一、特徵值分析

我們利用特徵值分析來決定針對各項原始特徵值的處理方式。繪製輸贏分布的機率分布圖進行分析後，我們發現某些特徵值 (如助攻數 `ast`、罰球命中率 `fg_pct`) 原先的輸贏分群差異不明顯，但是計算兩者的差值之後，輸贏的分群就變得很明顯，可幫助模型更準確地預測結果；有些特徵值原本輸贏分群就很明顯 (如正負值 `plus_minus`)，有些則是即便計算主客隊差值分群仍然不明顯 (犯規次數 `pf`) (圖一)



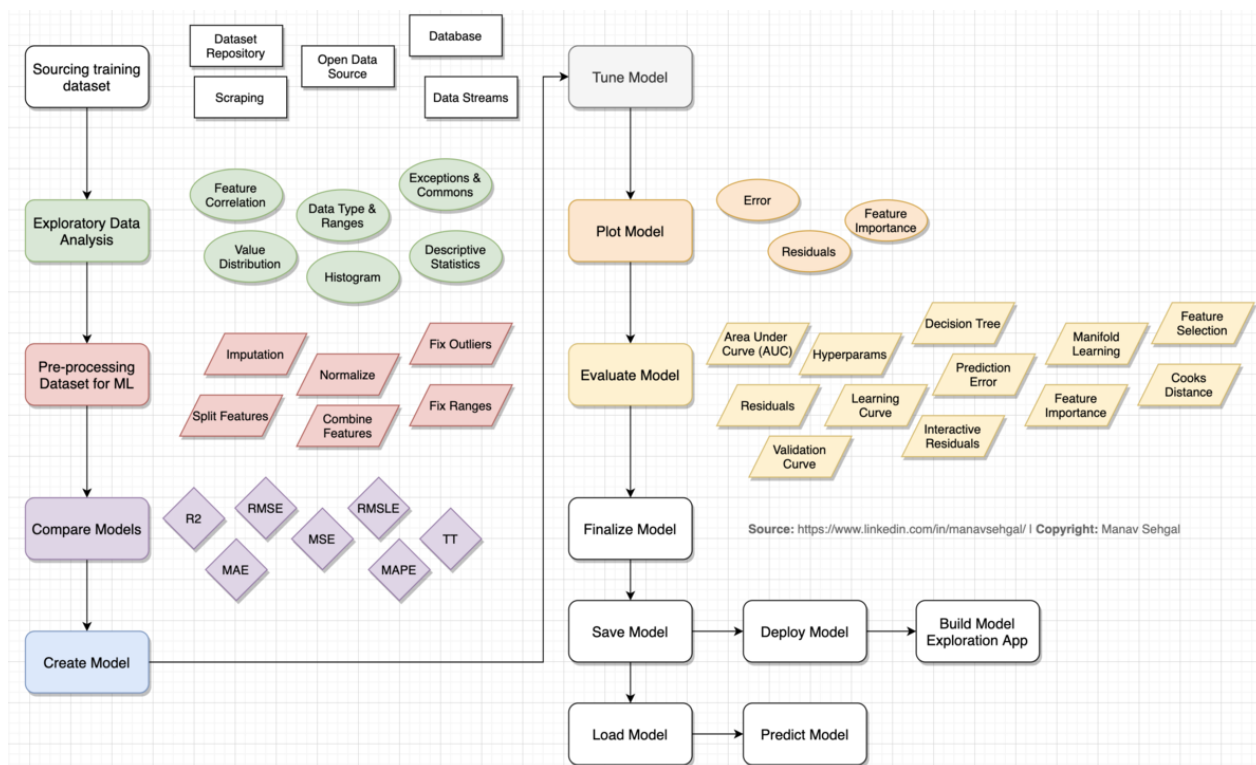
圖二、各項特徵值之熱力圖

各項特徵值的熱力圖，其中每一橫列為不同的場次，每一直條則代表主客隊各特徵值相減後標準化的數值。wl欄中黃色代表主隊贏的場次，藍色為主隊輸之場次。我們可以透過這張圖，觀察不同特徵值與輸贏的關係之強弱，其中有較明顯分界的特徵值為對於比賽勝負影響較大的參數，和前面長條圖分群明顯者吻合(圖二)。

### (三) 模型建置

#### 1. PyCaret

PyCarat源自R的Caret(Classification and Regression Training), 是一個開源、低程式碼的自動機器學習工具, 由於經簡易的流程就能完成自動機器學習, 並生成多種解釋圖表, 還能利用 dashboard指令生成互動式頁面, 將參數調整後的預測結果直接以視覺化的圖表呈現, 因此相當適合素人資料科學家使用。圖三及圖四為PyCarat的分析流程及模型種類, 囊括大部分常見的機器學習模型。



圖三、PyCaret分析流程

ID	Name	Reference	Turbo
lr	Logistic Regression	sklearn.linear_model._logistic.LogisticRegression	True
knn	K Neighbors Classifier	sklearn.neighbors._classification.KNeighborsCl...	True
nb	Naive Bayes	sklearn.naive_bayes.GaussianNB	True
dt	Decision Tree Classifier	sklearn.tree._classes.DecisionTreeClassifier	True
svm	SVM - Linear Kernel	sklearn.linear_model._stochastic_gradient.SGDC...	True
rbfsvm	SVM - Radial Kernel	sklearn.svm._classes.SVC	False
gpc	Gaussian Process Classifier	sklearn.gaussian_process._gpc.GaussianProcessC...	False
mlp	MLP Classifier	sklearn.neural_network._multilayer_perceptron....	False
ridge	Ridge Classifier	sklearn.linear_model._ridge.RidgeClassifier	True
rf	Random Forest Classifier	sklearn.ensemble._forest.RandomForestClassifier	True
qda	Quadratic Discriminant Analysis	sklearn.discriminant_analysis.QuadraticDiscrim...	True
ada	Ada Boost Classifier	sklearn.ensemble._weight_boosting.AdaBoostClas...	True
gbc	Gradient Boosting Classifier	sklearn.ensemble._gb.GradientBoostingClassifier	True
lda	Linear Discriminant Analysis	sklearn.discriminant_analysis.LinearDiscrimina...	True
et	Extra Trees Classifier	sklearn.ensemble._forest.ExtraTreesClassifier	True
xgboost	Extreme Gradient Boosting	xgboost.sklearn.XGBClassifier	True
lightgbm	Light Gradient Boosting Machine	lightgbm.sklearn.LGBMClassifier	True
dummy	Dummy Classifier	sklearn.dummy.DummyClassifier	True

圖四、PyCaret涵蓋之模型列表

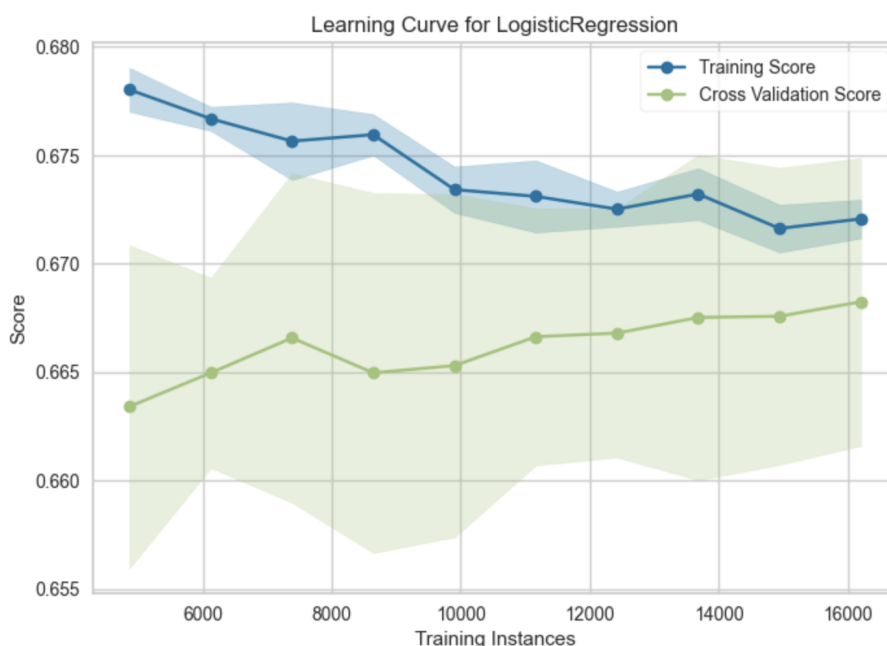
由於本程式旨在預測NBA的比賽結果，我們依時間順序將較早期80%的資料作為訓練資料，較晚期20%的資料作為測試資料，並刪除'team\_name\_home'，'team\_name\_away'，'season\_id'，'game\_id'，'wl\_away'，'video\_available\_rate\_home'，'video\_available\_rate\_away'等經主觀判斷後認為不影響輸贏的欄位。

PyCaret先選擇的最優模型為LogisticRegression，再利用tune\_model指令來測試最佳參數。最終的參數如圖五。

Parameters	
C	0.179
class_weight	{}
dual	False
fit_intercept	True
intercept_scaling	1
l1_ratio	None
max_iter	1000
multi_class	auto
n_jobs	None
penalty	l2
random_state	2023
solver	lbfgs
tol	0.0001
verbose	0
warm_start	False

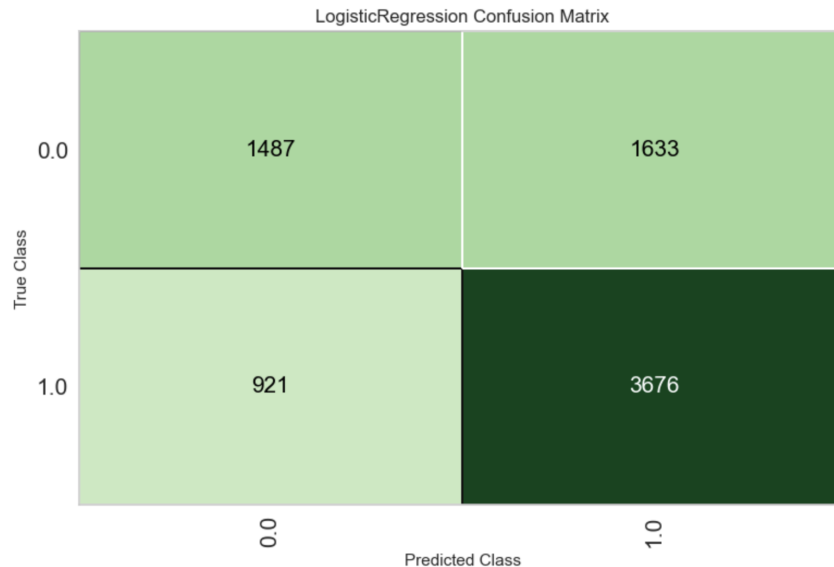
圖五、PyCaret選擇之最佳參數值

針對訓練資料，本模型的Accuracy rate為66.83%，再利用此模型預測較晚期的測試資料，Accuracy rate約69.85%。接著再利用PyCaret的指令evaluate\_model，即可取得多種訓練模型時的資訊，且皆為視覺化的圖表。如圖六的Learning Curve，可看出Training Score和Cross Validation Score兩條線近乎收斂，毋須藉由增加資料量提高模型的準確率。而混淆矩陣(圖七)則是準確率、精確率、召回率等的計算依據。除此之外，PyCaret還提供deepcheck函數，提供模型的更多細節資訊，如測試資料和訓練資料的重疊性、各項Feature的關聯性、各項Feature對於預測的貢獻度、Feature本身的歧異度等。



圖六、Learning Curve for Logistic Regression





圖七、LogisticRegression Confusion Matrix

## 2. TPOT

TPOT是一個AutoML工具，它是Tree-based Pipeline Optimization Tool的縮寫。TPOT的目標是自動化機器學習的過程，包括特徵選擇、模型選擇、超參數調整等，以找到最佳的機器學習流程。TPOT是利用Genetic Programming的方式來對可能的機器學習流程進行篩選和修改，可以自動探索不同的特徵轉換方法、模型組合和超參數設置，最終可以演化出在這個問題上最適合的流程。

TPOT的函數可以自訂世代的數量與每個世代的人口數量，加大這兩個參數會讓模型在搜尋時更透徹，但也會花很多時間。不過TPOT有提供自訂時間上限的參數設定，還有讓他可以在停止之後接續訓練的參數，因此可以訓練到一半看成果再決定是否繼續訓練。

我們在一開始資料處理時先刪除‘video\_available\_rate\_home’，‘video\_available\_rate\_away’的欄位，因為這兩個欄位與輸贏沒有太大關係。之後我們將資料用依序的方式切成80%的訓練資料與測試資料，並放入TPOT進行訓練。TPOT訓練好之後可以匯出一個.py檔案，裡面會有訓練的pipeline，也就是除了機器學習模型本身以外，也會決定特徵選取的方式、還有一些數值的轉換等等。提供匯出的py檔很方便，可以直接用這個模版來訓練模型。

實際匯出的py檔如圖八：

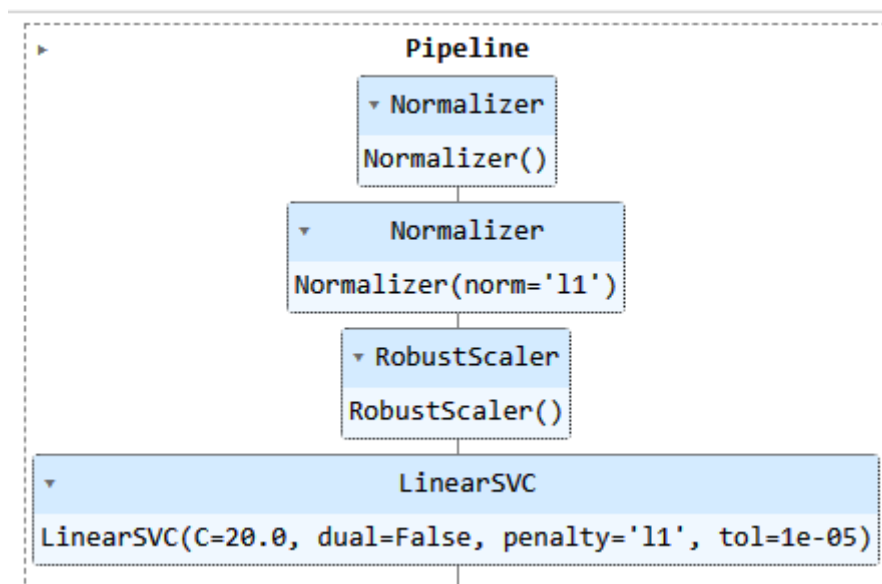
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import Normalizer, RobustScaler
from sklearn.svm import LinearSVC

# NOTE: Make sure that the outcome column is labeled 'target' in the data file
tpot_data = pd.read_csv('PATH/TO/DATA/FILE', sep='COLUMN_SEPARATOR', dtype=np.float64)
features = tpot_data.drop('target', axis=1)
training_features, testing_features, training_target, testing_target = \
    train_test_split(features, tpot_data['target'], random_state=None)

# Average CV score on the training set was: 0.6783561074176807
exported_pipeline = make_pipeline(
    Normalizer(norm="l2"),
    Normalizer(norm="l1"),
    RobustScaler(),
    LinearSVC(C=20.0, dual=False, loss="squared_hinge", penalty="l1", tol=1e-05)
)

exported_pipeline.fit(training_features, training_target)
results = exported_pipeline.predict(testing_features)
```

圖八、TPOT訓練過程程式碼



圖九、模型產生過程之pipeline

訓練好之後，我們利用joblib這個套件來將這個pipeline還有訓練好的模型儲存起來(圖九)，供後續使用。最終我們獲得的測試資料正確率約為67%。

### 3. Auto-sklearn

AutoSklearn是一個是基於Sklearn(Scikit-learn)庫開發的自動化的機器學習工具，會自動選擇模型並測試超參數，簡化、縮短了機器學習流程的複雜性與時間。使用者輸入各模型測試時間與總時長，最後會輸出一個集成模型，也就是多個模型分別預測結果後，按照權重決定最終預測的答案。

一開始的資料庫只有整個隊的資料，去除掉與比賽表現無關的欄位項目(order、video available、season\_id... 等等)後訓練模型，發現訓練資料與測試資料的準確率都只有66%，不論是訓練時間短(6分鐘)或長(兩小時)，或是刪除大部分的欄位只剩下wl\_rate、plus\_minus\_rate、pts\_rate(看著heatmap選分界最明顯的)，準確率都維持66%。畫資料分析圖時發現如果將主客隊資料相減後的數值，在圖上可看出與輸贏更顯著的關係，所以後來將各項資料主客隊相減、增添dev欄位，又增加了多欄有關MVP、All star等球員相關數據，結果訓練資料準確率提升為71%，測試資料提升為67.9%。

下表為最終集成模型的內容：

Rank	Ensemble_weight	Model	Cost
1	0.02	SGD	0.3293029
2	0.32	Gradient_boosting	0.3297694
3	0.42	LDA	0.3308192
4	0.1	Gradient_boosting	0.3354457
5	0.12	QDA	0.3433381
6	0.02	Gradient_boosting	0.4008009

※ SGD: Stochastic Gradient Descent; LDA: Linear Discriminant Analysis; QDA: Quadratic Discriminant Analysis

最後將模型存成joblib檔並將預測用的程式碼整理成function 'wl\_scikitlearn\_v2'。



#### (四) 成果

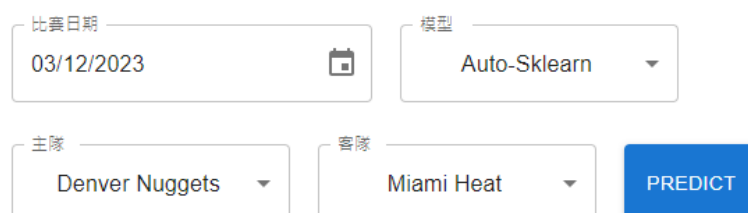
##### 1. 結果呈現

如期中報告所述，我們已將預測模型部署到網頁上，使用者可以在下圖的頁面中選擇對戰日期、對戰的主客隊和三種工具 (Auto-sklearn/TPOT/Pycaret) 所生成的模型，按下 predict 之後，網頁下方會顯示預測的勝隊。

網址和程式碼如下：

網址：<http://140.112.211.104:5001/>

程式碼：<https://github.com/lytt925/NBA-Flask>

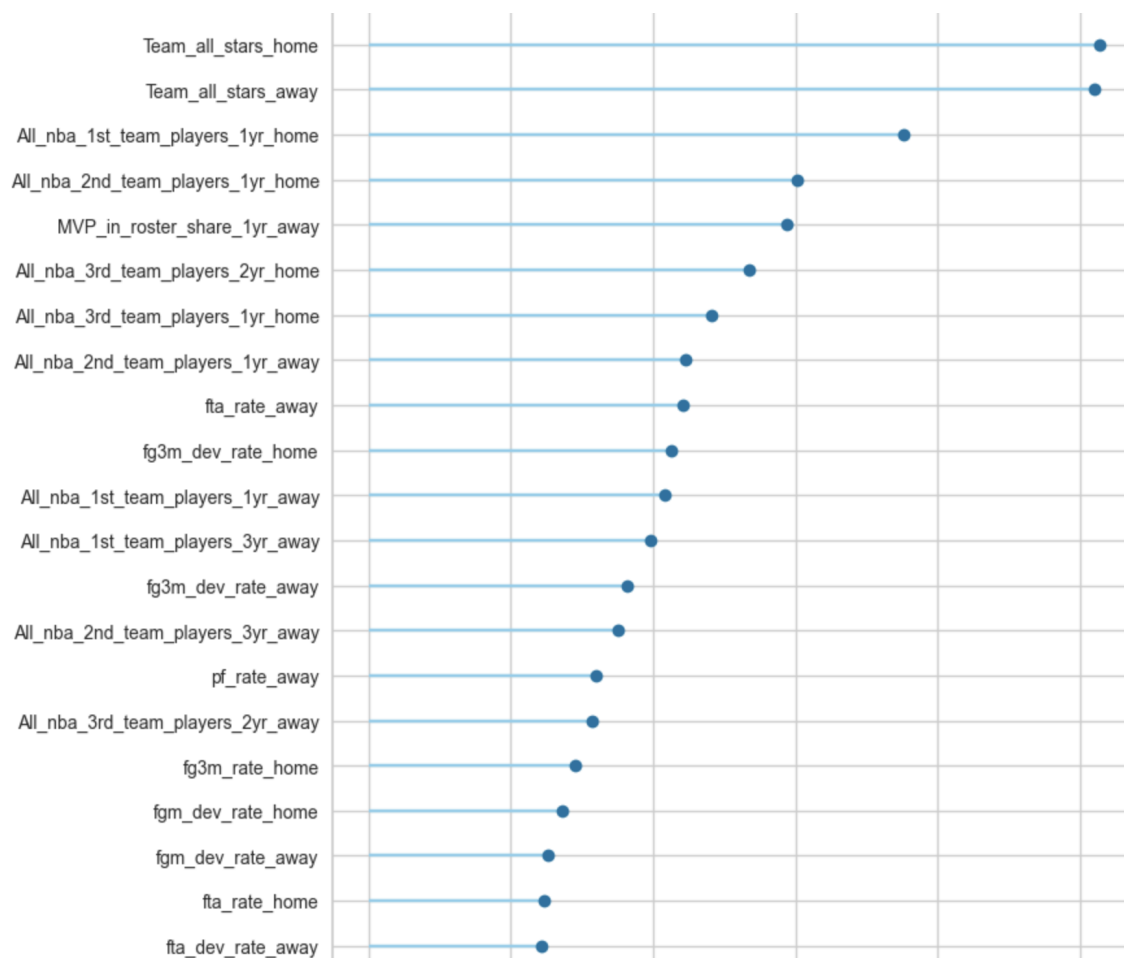


**The predicted winner is Denver Nuggets.**

圖十、NBA勝隊預測網頁

## 2. Feature Importance 分析

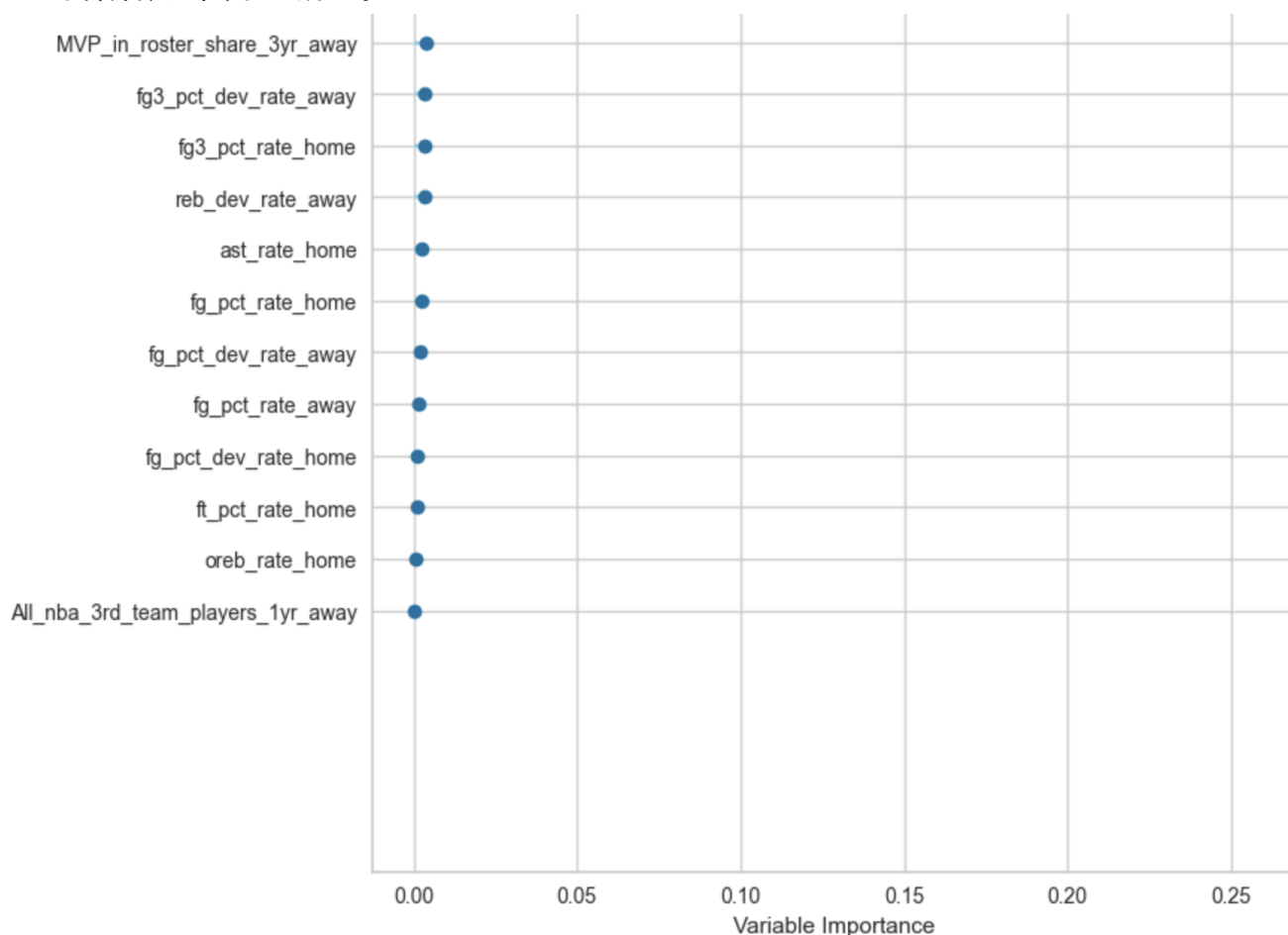
利用PyCaret的 `evaluate_model` 指令，我們可以得到模型的 Feature Importance Plot，圖十一列出前 20 重要的 feature：



圖十一、前20重要的特徵

透過這張圖，我們發現「球員」的重要性明顯大於「球隊平均」——前十重要的 feature 當中，第 1~8 名都是與球員獎項相關的特徵值，主要為主客隊當年入選全明星賽人數 (Team\_all\_stars)、前一年獲選年度前三隊人數 (All\_nba\_1st/2nd/3rd\_team\_players\_1yr)，和前一年列入 MVP 排行榜之球員的票數；而球隊數據中，最重要則依序為罰球數、三分命中數、犯規數和 fieldgoal 命中數。

而fieldgoal/三分/罰球命中率、進攻防守籃板數和助攻數等傳統數據對於勝負的影響都不顯著，如圖十二所示。



圖十二、對預測結果不重要之特徵

### 三、結論：困難、解決方法與心得

在本期末專題當中，我們成功做出了 NBA 比賽勝負預測的模型，並將成果部署到一個網頁的使用者介面上，供使用者選擇兩支球隊以預測其對戰勝負。在參數選擇和處理方面，我們針對過去他人研究中沒有做好的部分進行了修正，比方說：

#### 1. 球隊數據：

過去有些研究會使用球隊整季的平均數據和當季戰績來預測該季某一場比賽勝負（比如，用 2022-23 球季整季的戰績來預測 2022 年某隊第一場比賽的勝負），這種做法實際上是錯誤的，因為它會用未來的數據來預測現在的比賽。因此在本專題中，我們使用的是球隊「過去十場比賽的平均數據」，除了更能反映球隊近況之外，也能確保我們用的是「已知」的數據來預測現在的比賽，這樣做法比較合理。

#### 2. 球員資料：

使用球員資料的目的，是為了在預測比賽勝負中考量各隊球員整體的實力。大部分前人的研究，都是以球隊中每一位上場球員過去一個月或過去十場比賽的數據來預測該場比賽的勝負。然而，我們認為這樣的作法其實有盲點，因為球員個人的數據與個人的實力並不一定成正比。舉例來說，幾位超級球星組成一隊時，他們的個人數據會因為彼此之間球權受壓縮而下降，甚至可能不如一些中下游球隊中的主力球員，但這並不表示他們真正的實力比其他球員弱。

因此在討論之後，我們決定以球隊中「當年入選明星賽的球員數」、「過去三年曾入選NBA年度前三隊的人次數」和「過去三年的 MVP 排行榜總得分」作為球員資料中的特徵值，也作為球隊實力的指標。

經過上述的修正後，我們使用了 PyCaret、TPOT 和 Auto-sklearn 這三個工具來進行自動機器學習以建置模型，得出「只用球隊數據」進行預測和利用「球隊數據+球員資料」進行預測的準確度，並用內建的功能來分析參數 importance 的重要資訊。

根據我們的研究成果，我們用 Pycaret 建置的模型對於訓練資料的 Accuracy rate 為 66.83%，對測試資料的 Accuracy rate 則為 69.85%；用 TPOT 建置的模型對於訓練資料的 Accuracy rate 為 67.8%，對測試資料則為 67.3 %；在 Auto-scikitlearn 方面，只用球隊數據建置的模型對於訓練資料和測試資料的 Accuracy rate 皆為 66.6%，而用球隊+球員資料建置的模型對於訓練資料的 Accuracy rate 則上升到 71.0%，對測試資料則上升到 67.9%。前人的研究表明，NBA 比賽中的 upset rate (即弱隊打贏強隊的機率) 約為 30%，故前人的 NBA 比賽預測模型準確率多半在 66%~70%左右。因此，我們本次專題成果與前人研究相符且合理。

另外，由於我們所查的文獻中並沒有太多關於 feature importance 的資訊，因此我們有特別做 feature importance 的分析。根據分析結果，我們發現「球員資料」的重要性明顯大於「球隊平均」——球隊中入選當年全明星賽的球員數、前一年獲選年度前三隊人數，和前一年列入 MVP 排行榜之球員的票數，很大程度上決定了比賽的勝負。單看球隊數據的話，我們從熱力圖得知防守籃板、助攻數、投籃命中、三分命中、三分嘗試、三分命中率、罰球命中率、籃板數、正負值等等皆是重要的特徵值。這些發現也讓我們對 NBA 比賽各項數據的重要性有了更深刻的理解。

除了資料整理和機器學習本身以外，我們還遇到了許多其他技術問題：

### 1. 硬體限制

除了自動機器學習本身外，許多檢查模型預估能力、解釋資料的指令，都耗時相當久。未來在修相關課程之前，或許應先確保自己有準備夠好的配備。

### 2. 開發環境維護不易

因為要測試很多模型，常常遇到版本的衝突跟不相容的情形，例如auto sklearn要求的scikit learn版本較舊，但pycaret和tpot要求的較新。再來就是pycaret在apple silicon的電腦上面會裝不成功，需要額外下載東西才行。解決辦法就是用conda建立很多虛擬環境然後測試最相容但也不會要一直更換的條件。

### 3. 程式與資料版本控制

因為完全沒有使用github之類的版控工具，所以大家的code跟整理好的資料沒辦法有很好的追蹤，都是用日期編號放在google drive。而且因為環境問題，某個套件或程式可能可以在某人的電腦上跑，但換台電腦又不行了。下次在從事類似的合作 project 之前，我們要先熟悉一般常用的版控工具，以免再遇到類似的麻煩。

有了這次的經驗，相信我們未來在從事相關研究時，將能更有效率的完成各自的研究工作。

總而言之，在這次的研究中，我們針對前人研究不足之處做出了修正，並成功的建置了準確度符合前人研究成果的勝負預測模型。希望我們研究的方法、工具和發現，可以幫助到未來要做 NBA 或其他運動賽事勝負預測的同學和研究員，以實現更準確合理的預測成果。