*系級：四資工四乙 學號： B10615020 姓名： 林哲旭*

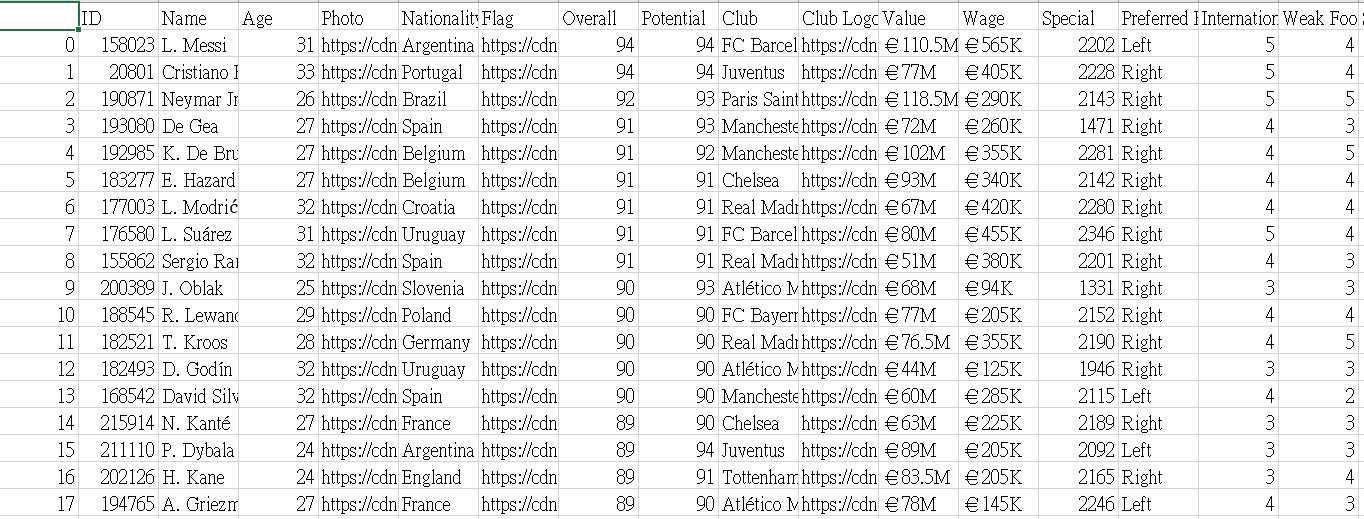
期末Python程式：FIFA19選手能力評鑑

### 一、設計此程式的動機

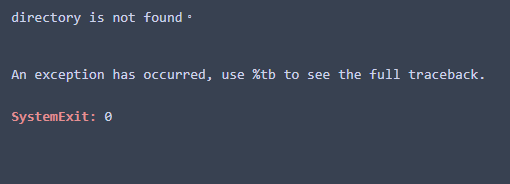
身為一個電玩遊戲的愛好者，能夠精打細算的從遊戲資訊中挑取最高價值的角色是必要的技能，如今在大數據的課程學會使用Python製作簡易程式，不妨自行設計一個能夠跟自己興趣相符的程式，除了增加程式撰寫能力的基礎，還可以讓身在遊戲中的我獲得一份成就感，因此上Kaggle網站尋找到FIFA19足球遊戲的選手資料集，藉此來分析選手評分。

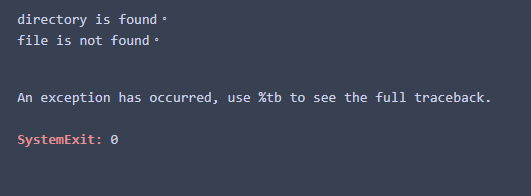
### 二、程式設計目標

撰寫Python程式，讀取FIFA19的選手相關資料，總共有88種角色屬性，包含人名、年紀、隊名、各種能力值等等，根據所有角色屬性把不重要的屬性篩選掉，例如：照片，國旗，背號等等。



執行程式的時候，會偵測目的路徑中是否有指定的選手資料集(.csv)，如果系統沒有偵測到則會直接結束程式，如下圖所示，





程式會自動在「D:\FIFA19」的工作目錄中，產生選手評分的預測結果(.csv)以及比較圖(.png)，最後可透過預測結果(.csv)來選取最高評分的角色。



### 三、本程式所使用的套件功能介紹

本程式主要運用3個套件，其功能說明如下：

* pandas套件

pandas套件為資料處理套件，方便使用者在處理文書檔案(.csv, .json, .xls)非常的方便，無論是讀取、撰寫、修改，都能夠輕而易舉。

程式中使用以下幾種語法：

|  |  |
| --- | --- |
| 語法 | 功能 |
| read\_csv(“路徑”) | 讀取本地端的檔案，或是網路上的來源檔，其中路徑的斜線與windows表示法會相反，需要多加注意。 |
| to\_csv(“路徑”) | 用於輸出Dataframe為(.csv)檔，其中路徑的部分斜線與windows表示法不同只能有一個正斜線。 |
| Dataframe | 一種資料格式，用於檔案的存取，也可以用於操作新增、刪除、修改、更新，是非常方便管理資料的格式。 |

* os套件

Python提供os.path套件來進行檔案的操作，可以取得檔案路徑、檔案大小、建立目錄、刪除目錄、刪除檔案與執行命令…等操作。

程式中使用以下幾種語法：

|  |  |
| --- | --- |
| 語法 | 功能 |
| os.path.isdir(“路徑”) | 確認路經中的資料夾是否存在。 |
| os.path.isfile (“路徑”) | 確認路經中的檔案是否存在。 |

* sys套件

Python提供sys套件用於系統上的執行指令，通常sys套件可以處理例外情況，或是強制性的執行某些關鍵指令，sys中的部分指令具有不可回朔性，使用上必須多加小心。

程式中使用以下幾種語法：

|  |  |
| --- | --- |
| 語法 | 功能 |
| sys.exit(0) | 強制讓程式結束執行，並且視為正常結束。 |

* sklearn套件

sklearn提供許多預測模型，提供使用者只需要預處理資料集，就可以通過預測模型達到想要預測的效果，其中模型的實作相當複雜，因此開發者透過簡單的function窗口取代原先困難的內容，每個模型都有提供多樣的參數讓使用者根據不同的情形去調整模型，模型在訓練的過程中會不斷的反正修正，因此給予適當的資料集是相當重要的工作。

程式中使用以下幾種語法：

|  |  |
| --- | --- |
| 語法 | 功能 |
| train\_test\_split(資料集,答案集,訓練與預測資料分配比例) | 此功能相當的強大，能夠隨機的把整體資料集切割成訓練集以及預測集，並且使用者可以隨意決定切割的比例。 |
| tree.DecisionTreeClassifier() | 建立決策樹的分類器，括弧內有許多參數可以使用，此篇不再贅述。 |
| KNeighborsClassifier(n\_neighbors=k) | 建立KNN的分類器，括弧內有許多參數可以使用，此篇不再贅述。 |
| 分類器變數.fit(訓練集,訓練集解答) | 此功能用於訓練模型，給予訓練集以及解答，讓模型自行修正。 |
| 分類器變數.predict(預測集) | 此功能用於已經訓練好的模型，接著使用想要預測的資料集讓模型預測，即可得知預測結果。 |

* matplotlib套件

matplotlib套件提供使用者用簡易的函數就可以畫出直線、圖形，圖表，用於實驗結果表示是非常方便的套件，畫出圖表後還可以儲存結果以便攜帶。

程式中使用以下幾種語法：

|  |  |
| --- | --- |
| 語法 | 功能 |
| plt.figure(figsize=(x,y)) | 控制圖片大小。 |
| plt.title("title",fontsize=x) | 加入圖表標題。 |
| plt.ylabel("y",fontsize=x)  plt.xlabel("x",fontsize=x) | 加入X，Y軸標題。 |
| plt.xticks(fontsize=15)  plt.yticks(fontsize=15) | 控制X，Y軸刻度大小。 |
| plt.plot(X軸資料,Y軸資料) | 加入單一折線。 |
| plt.legend() | 加入圖例。 |
| plt.savefig("路徑") | 儲存圖表。 |
| plt.show() | 顯示圖表。 |

### 四、參考程式碼與程式碼解說

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | import pandas as pd  import matplotlib.pyplot as plt  from sklearn import tree  from sklearn.model\_selection import train\_test\_split  from sklearn import metrics  from sklearn.neighbors import KNeighborsClassifier  from sklearn.svm import SVC  import os  import sys |
| 此區塊主要是加入程式中需要的所有套件。 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #open file  #user can change path on his/her own  folderpath = 'D:/FIFA19'  if os.path.isdir(folderpath):  print("directory is found。")  else:  print("directory is not found。")  sys.exit(0)    filepath = folderpath + '/data.csv'  if os.path.isfile(filepath):  print("file is found。")  else:  print("file is not found。")  sys.exit(0)  df = pd.read\_csv("D:/FIFA19/data.csv") |
| 此區塊主要是讀取選手資料集，當沒有路徑上的資料夾與檔案不存在時，將會跳出訊息告訴使用者需要檢查路徑。 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87 | #drop non-useful attributes  drop\_list = ["ID","Name","Photo","Nationality","Flag","Club","Club Logo","Special","Work Rate", "Real Face","Jersey Number","Joined","Loaned From","Contract Valid Until", "Crossing","Finishing","HeadingAccuracy","ShortPassing","Volleys","Dribbling",  "Curve","FKAccuracy","LongPassing","BallControl","Acceleration","SprintSpeed",  "Agility","Reactions","Balance","ShotPower","Jumping","Stamina","Strength",  "LongShots","Aggression","Interceptions","Positioning","Vision","Penalties", "Composure","Marking","StandingTackle","SlidingTackle","GKDiving","GKHandling", "GKKicking","GKPositioning","GKReflexes","LS","ST","RS","LW","LF","CF","RF","RW",  "LAM","CAM","RAM","LM","LCM","CM","RCM","RM","LWB","LDM","CDM","RDM","RWB","LB","LCB","CB","RCB","RB","Overall","Position"]  fifa\_data = df.drop(drop\_list,axis = 1)  #fill nan value to 0  fifa\_data = fifa\_data.fillna("0")  #handle "€","M","K" in value,wage and release clause attributes  for i in range(len(fifa\_data["Value"])):  temp\_str = fifa\_data["Value"][i]  temp\_str = temp\_str.replace("€","")  if "M" in temp\_str:  temp\_str = temp\_str.replace("M","")  temp\_str = str(int(float(temp\_str)\*1000000))  elif "K" in temp\_str:  temp\_str = temp\_str.replace("K","")  temp\_str = str(int(float(temp\_str)\*1000))  fifa\_data.loc[i,"Value"] = temp\_str  for i in range(len(fifa\_data["Wage"])):  temp\_str = fifa\_data["Wage"][i]  temp\_str = temp\_str.replace("€","")  if "M" in temp\_str:  temp\_str = temp\_str.replace("M","")  temp\_str = str(int(float(temp\_str)\*1000000))  elif "K" in temp\_str:  temp\_str = temp\_str.replace("K","")  temp\_str = str(int(float(temp\_str)\*1000))  fifa\_data.loc[i,"Wage"] = temp\_str    for i in range(len(fifa\_data["Release Clause"])):  temp\_str = fifa\_data["Release Clause"][i]  temp\_str = temp\_str.replace("€","")  if "M" in temp\_str:  temp\_str = temp\_str.replace("M","")  temp\_str = str(int(float(temp\_str)\*1000000))  elif "K" in temp\_str:  temp\_str = temp\_str.replace("K","")  temp\_str = str(int(float(temp\_str)\*1000))  fifa\_data.loc[i,"Release Clause"] = temp\_str    #handle Preferred Foot by translation in 0(Left) and 1(Right)  for i in range(len(fifa\_data["Preferred Foot"])):  if "Left" in fifa\_data["Preferred Foot"][i]:  fifa\_data.loc[i,"Preferred Foot"] = "0"  else:  fifa\_data.loc[i,"Preferred Foot"] = "1"    #handle body type by quantizing  monsters = ['C. Ronaldo','Messi','Neymar','Akinfenwa','Shaqiri','Courtois']  for i in range(len(fifa\_data["Body Type"])):  if fifa\_data["Body Type"][i] in monsters:  fifa\_data.loc[i,"Body Type"] = "4"  elif fifa\_data["Body Type"][i] == "Stocky":  fifa\_data.loc[i,"Body Type"] = "3"  elif fifa\_data["Body Type"][i] == "Normal":  fifa\_data.loc[i,"Body Type"] = "2"  elif fifa\_data["Body Type"][i] == "Lean":  fifa\_data.loc[i,"Body Type"] = "1"  else:  fifa\_data.loc[i,"Body Type"] = "0"    #handle Height by translating in CM  for i in range(len(fifa\_data["Height"])):  temp\_str = fifa\_data["Height"][i]  temp\_str = temp\_str.replace("\'",".")  inch = float(temp\_str) % 1 \* 100  if inch > 12:  inch/=10  temp\_str = str(float("{:.2f}".format((float(temp\_str) // 1 \* 12 + inch) \* 2.54)))  fifa\_data.loc[i,"Height"] = temp\_str    #handle Weight by translating in KG  for i in range(len(fifa\_data["Weight"])):  temp\_str = fifa\_data["Weight"][i]  temp\_str = temp\_str.replace("lbs","")  temp\_str = str(float("{:.2f}".format(float(temp\_str) \* 0.453)))  fifa\_data.loc[i,"Weight"] = temp\_str |
| 此區塊主要負責資料預處理，首先把不需要的column drop，接著用fillna補上資料空值，因為模型需要數值化的資料因此需要把文字轉換成純數字，17-49行用於處理資料中具有” €,K,M”文字的值；51-56行用於處理轉換”Left,Right”為0,1；58-70行用於處理選手各種不同的體態字詞；72-80行用於處理英呎轉成公分；82-87用於處理磅轉為公斤。前述有提到預處理對於模型訓練非常重要，因此對於drop\_list的挑選必須慎選。 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | #take overall as standard picking rule  fifa\_ovr = df[["Overall","Name","Position"]]  #split fifa\_data into trainning data and  train\_data, test\_data, train\_ovr, test\_ovr = train\_test\_split(fifa\_data,fifa\_ovr,test\_size = 0.5)  test\_name = pd.DataFrame({"Name":test\_ovr["Name"]})  test\_pos = pd.DataFrame({"Position":test\_ovr["Position"]})  train\_ovr = train\_ovr.drop(["Name","Position"],axis = 1)  test\_ovr = test\_ovr.drop(["Name","Position"],axis = 1)  #copy data  train\_Ddata = train\_data  train\_Dovr = train\_ovr  test\_Ddata = test\_data  test\_Dovr = test\_ovr  test\_Dname = test\_name  test\_Dpos = test\_pos  #build "DecisionTree" classifier  clf = tree.DecisionTreeClassifier()  fifa\_clf = clf.fit(train\_Ddata, train\_Dovr)  #predict  fifa\_Dpredited = fifa\_clf.predict(test\_Ddata)  #rearrange test standard overalls' index  test\_Dovr.index = range(len(test\_Dovr))  test\_Dname.index = range(len(test\_Dname))  test\_Dpos.index = range(len(test\_Dpos))  #accuracy  accuracy = metrics.accuracy\_score(test\_Dovr,fifa\_Dpredited)  print(accuracy)  #concate predicted answer and standard overall  df\_Dpredicted = pd.DataFrame({'Score':fifa\_Dpredited})  Dresult = {  'Name': [],  'Position': [],  'Score': [],  'Overall': []  }  df\_Dresult = pd.DataFrame(Dresult)  for i in range(len(test\_Dovr)):  df\_Dresult = df\_Dresult.append({  'Name': test\_Dname['Name'][i],  'Position': test\_Dpos['Position'][i],  'Score': df\_Dpredicted['Score'][i],  'Overall': test\_Dovr['Overall'][i]  },ignore\_index=True)  df\_Dresult.sort\_values(by = 'Score',inplace=True,ascending=False,ignore\_index=True)  print(df\_Dresult)  #output result to csv  output\_PATH = 'D:/FIFA19/fifa\_D\_result.csv'  df\_Dresult.to\_csv(output\_PATH,index=0) |
| 此區塊主要是使用決策數(DecisionTree)作為模型去預測資料，首先第2行先取出答案樣本，第5行切割資料為訓練集以及預測集，12-17行為了保護資料而複製一份一樣的資料，接著第20行建立模型雛形，第21行訓練模型，再來交給第24行預測結果，因為切割訓練集以及預測集時為隨機切割，因此index不是從1開始所以27-29行用於重新定義index，第32行計算預測結果跟答案樣本的相似度(準確性)，最後35-56行把分散的資料合併成一個dataframe且輸出至(.csv)檔。 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59 | #copy data  train\_Kdata = train\_data  train\_Kovr = train\_ovr  test\_Kdata = test\_Ddata  test\_Kovr = test\_ovr  test\_Kname = test\_name  test\_Kpos = test\_pos  train\_Kovr = train\_Kovr.values.ravel()  #Because we don't know how to choose K index in KNN algorithm,  #so we give a test to bulid many different K classifier.  accuracy = []  for k in range(1, 100):  knn = KNeighborsClassifier(n\_neighbors=k)  knn.fit(train\_Kdata, train\_Kovr)  y\_pred = knn.predict(test\_Kdata)  accuracy.append(metrics.accuracy\_score(test\_Kovr, y\_pred))  k\_range = range(1,100)  plt.plot(k\_range, accuracy)  plt.savefig("D:/FIFA19/K\_selection.png")  plt.show()  #build "KNN" classifier  knn = KNeighborsClassifier(n\_neighbors=1)  knn = knn.fit(train\_Kdata, train\_Kovr)  #predict  fifa\_Kpredited = knn.predict(test\_Kdata)  #test standard overall rearrange index  test\_Kovr.index = range(len(test\_Kovr))  test\_Kname.index = range(len(test\_Kname))  test\_Kpos.index = range(len(test\_Kpos))  #accuracy  accuracy = metrics.accuracy\_score(test\_Kovr,fifa\_Kpredited)  #concate predicted answer and standard overall  df\_Kpredicted = pd.DataFrame({'Score':fifa\_Kpredited})  Kresult = {  'Name': [],  'Position': [],  'Score': [],  'Overall': []  }  df\_Kresult = pd.DataFrame(Kresult)  for i in range(len(test\_Kovr)):  df\_Kresult = df\_Kresult.append({  'Name': test\_Kname['Name'][i],  'Position': test\_Kpos['Position'][i],  'Score': df\_Kpredicted['Score'][i],  'Overall': test\_Kovr['Overall'][i]  },ignore\_index=True)  df\_Kresult.sort\_values(by = 'Score',inplace=True,ascending=False,ignore\_index=True)  print(df\_Kresult)  #output result to csv  output\_PATH = 'D:/FIFA19/fifa\_K\_result.csv'  df\_Kresult.to\_csv(output\_PATH,index=0) |
| 此區塊主要是使用K-近鄰演算法(KNN)作為模型去預測資料， 1-8行為複製與前區塊一樣，因為KNN其中有一個參數K為鄰近數量，這個參數將會影響預測結果，必須謹慎選擇K值，因此12-21行為K值訓練，用K=1到K=100的方式逐次建立模型去訓練，藉此決定最後K取多少最為合理，此K值訓練的成果將會輸出成折線圖存至指定路徑，23-59行與前區塊19-56行雷同，在此不多贅述。 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | #comparasion between DecisionTree and KNN  x\_axis = []  D\_line = []  K\_line = []  O\_line = []  for i in range(0,len(test\_data),500):  x\_axis.append(i)  D\_line.append(fifa\_Dpredited[i])  K\_line.append(fifa\_Kpredited[i])  O\_line.append(test\_ovr['Overall'][i])  plt.figure(figsize=(10,10))  plt.title("Comparasion of Original, DicisionTree, KNN",fontsize=15) # title  plt.ylabel("Accuracy",fontsize=15) # y label  plt.xlabel("Serial number of data",fontsize=15) # x label  plt.xticks(fontsize=15)  plt.yticks(fontsize=15)  plt.plot(x\_axis,O\_line,linewidth = 5,label = 'Original')  plt.plot(x\_axis,D\_line,'--',color = 'red',label = 'DecisionTree')  plt.plot(x\_axis,K\_line,'--',color = 'green',label = 'KNN')  plt.legend()  plt.savefig("D:/FIFA19/Comparasion.png")  plt.show() |
| 此區塊為最後總整理，這次的程式使用兩種預測模型，決策樹與K-近鄰演算法，為了比較這兩種演算法的好壞與標準答案的相似度，使用折線圖表示並存在指定路徑中，經結果顯示決策樹的預測較為貼近標準答案，因為決策樹善於處理大量的資料且容易被解釋，KNN則是懶散學習法，速度快但受限於K值，因此結果較為不理想。 | |

### 五、心得

經由這次作業不但了解資料處理的重要性，且對於一些訓練模型更加的了解，某業界人士說：「現在的時代人人都可以做AI，只要有一台電腦跟足夠的資料」，這句話我非常的認同，因為整份程式用於AI的部分只有模型訓練，但模型訓練只有少少幾行，資料的預處理則是多上好幾倍，對於模型而言好的資料可以帶給他非常優秀的成果，而垃圾資料只會吐出垃圾資訊，就是人口中所戲稱的Garbage in Garbage out(GIGO)。身為資工系的學生，大多時間都是在專研演算法，熟悉的語言頂多C與C++，Python則是很少碰觸到，而在現在的趨勢下Python逐漸變成人人必備的程式語言，為了與時代接軌，在大數據的課程中學到Python各種相關知識之外，也學會不少套件的應用，相信這門課程對於未來研究所或是找工作會非常有幫助。