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。

	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	Club	Club Logo	Value	Wage	Special	Preferred I	International	Weak Foo
0	158023	L. Messi	31	https://cdn	Argentina	https://cdn	94	94	FC Barcel	https://cdn	€110.5M	€565K	2202	Left	5	4
1	20801	Cristiano I	33	https://cdn	Portugal	https://cdn	94	94	Juventus	https://cdn	€77M	€405K	2228	Right	5	4
2	190871	Neymar Jr	26	https://cdn	Brazil	https://cdn	92	93	Paris Saint	https://cdn	€118.5M	€290K	2143	Right	5	5
3	193080	De Gea	27	https://cdn	Spain	https://cdn	91	93	Mancheste	https://cdn	€72M	€260K	1471	Right	4	3
4	192985	K. De Bru	27	https://cdn	Belgium	https://cdn	91	92	Mancheste	https://cdn	€102M	€355K	2281	Right	4	5
5	183277	E. Hazard	27	https://cdn	Belgium	https://cdn	91	91	Chelsea	https://cdn	€93M	€340K	2142	Right	4	4
6	177003	L. Modrić	32	https://cdn	Croatia	https://cdn	91	91	Real Madi	https://cdn	€67M	€420K	2280	Right	4	4
7	176580	L. Suárez	31	https://cdn	Uruguay	https://cdn	91	91	FC Barcel	https://cdn	€80M	€455K	2346	Right	5	4
8	155862	Sergio Ra	32	https://cdn	Spain	https://cdn	91	91	Real Madi	https://cdn	€51M	€380K	2201	Right	4	3
9	200389	J. Oblak	25	https://cdn	Slovenia	https://cdn	90	93	Atlético M	https://cdn	€68M	€94K	1331	Right	3	3
10	188545	R. Lewan	29	https://cdn	Poland	https://cdn	90	90	FC Bayern	https://cdn	€77M	€205K	2152	Right	4	4
11	182521	T. Kroos	28	https://cdn	Germany	https://cdn	90	90	Real Madi	https://cdn	€76.5M	€355K	2190	Right	4	5
12	182493	D. Godín	32	https://cdn	Uruguay	https://cdn	90	90	Atlético M	https://cdn	€44M	€125K	1946	Right	3	3
13	168542	David Silv	32	https://cdn	Spain	https://cdn	90	90	Mancheste	https://cdn	€60M	€285K	2115	Left	4	2
14	215914	N. Kanté	27	https://cdn	France	https://cdn	89	90	Chelsea	https://cdn	€63M	€225K	2189	Right	3	3
15	211110	P. Dybala	24	https://cdn	Argentina	https://cdn	89	94	Juventus	https://cdn	€89M	€205K	2092	Left	3	3
16	202126	H. Kane	24	https://cdn	England	https://cdn	89	91	Tottenham	https://cdn	€83.5M	€205K	2165	Right	3	4
17	194765	A. Griezm	27	https://cdn	France	https://cdn	89	90	Atlético M	https://cdn	€78M	€145K	2246	Left	4	3

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，

```
directory is not found。


An exception has occurred, use %tb to see the full traceback.

SystemExit: 0
```

```
directory is found。
file is not found。


An exception has occurred, use %tb to see the full traceback.

SystemExit: 0
```

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。

H. Kane	ST	89	89
A. Griezm	CAM	89	89
M. Neuer	GK	89	89
T. Courtoi	GK	89	89
E. Cavani	LS	89	89
Coutinho	LW	88	88
P. Aubam	LM	88	88
S. Handan	GK	88	88
C. Eriksen	CAM	88	88
H. Lloris	GK	88	88

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  import pandas as pd
2  import matplotlib.pyplot as plt
3  from sklearn import tree
4  from sklearn.model_selection import train_test_split
5  from sklearn import metrics
6  from sklearn.neighbors import KNeighborsClassifier
7  from sklearn.svm import SVC
8  import os
9  import sys
```

此區塊主要是加入程式中需要的所有套件。

```
#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。

```
#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。

```
#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': [],
```

```
39        'Position': [],
40        'Score': [],
41        'Overall': []
42  }
43  df_Dresult = pd.DataFrame(Dresult)
44  for i in range(len(test_Dovr)):
45        df_Dresult = df_Dresult.append({
46              'Name': test_Dname['Name'][i],
47              'Position': test_Dpos['Position'][i],
48              'Score': df_Dpredicted['Score'][i],
49              'Overall': test_Dovr['Overall'][i]
50        },ignore_index=True)
51  df_Dresult.sort_values(by = 'Score',inplace=True,ascending=False,ignore_index=True)
52  print(df_Dresult)
53
54  #output result to csv
55  output_PATH = 'D:/FIFA19/fifa_D_result.csv'
56  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。

```
#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。

```
#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
```

```
13 plt.ylabel("Accuracy",fontsize=15) # y label
14 plt.xlabel("Serial number of data",fontsize=15) # x label
15 plt.xticks(fontsize=15)
16 plt.yticks(fontsize=15)
17 plt.plot(x_axis,O_line,linewidth = 5,label = 'Original')
18 plt.plot(x_axis,D_line,'--',color = 'red',label = 'DecisionTree')
19 plt.plot(x_axis,K_line,'--',color = 'green',label = 'KNN')
20 plt.legend()
21 plt.savefig("D:/FIFA19/Comparasion.png")
22 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。