Progetto di Statistica II: Seconda Relazione

Corso di Laurea Magistrale in
Artificial Intelligence and Data Engineering

Simone Landi
A.A. 2022-2023

Indice

Introduzione

Lo scopo della presente analisi è quello di costruire un modello di classificazione per poter predire se un giocatore NBA potrà rimanere all'interno della lega date le attuali statistiche per partita del giocatore stesso. L'analisi può essere utilizzata dalle squadre NBA per valutare se l'andamento corrente del giocatore possa consentire di mantenere il giocatore in squadra. Si può ipotizzare quindi che l'analisi sia stata commissionata dalla dirigenza di una franchigia NBA.

Dataset

La tabella di dati è stata ottenuta tramite l'unione di quattro tabelle: due relative alle statistiche della stagione 2021-22 e due relative a quelle della stagione 2022-23.

https://www.nba.com/stats/players/traditional?Season=2021-22
https://www.nba.com/stats/players/advanced?Season=2021-22
https://www.nba.com/stats/players/traditional?Season=2022-23
https://www.nba.com/stats/players/advanced?Season=2022-23

Per ottenere le tabelle relative alle statistiche è stato utilizzato uno script Python che sarà analizzato nel dettaglio nell'appendice di questa relazione.
L'analisi è stata effettuata prendendo in considerazione le statistiche dei giocatori della stagione 2021-22; per quanto riguarda le tabelle delle statistiche della stagione corrente (2022-23), queste sono state utilizzate con il solo scopo di confrontare quanti giocatori della passata stagione sono ancora all'interno della lega. I passaggi effettuati per ottenere la colonna relativa alla presenza del giocatore sono riportati nell'appendice.

Contenuto del Dataset

La tabella derivata contiene 14 colonne e 605 osservazioni. Le colonne sono le seguenti:

- GP: partite giocate dal giocatore nel corso della stagione
- MIN: minuti giocati per partita
- PTS: punti segnati per partita
- FGA_PG: tiri tentati per partita
- FG3A: tiri da 3 punti tentati per partita
- FTA: tiri liberi tentati per partita
- REB: rimbalzi totali per partita
- AST: assist realizzati per partita
- TOV: palle perse per partita
- STL: palle rubate per partita
- BLK: stoppate effettuate per patita
- OFF_RATING: numero di punti segnati da un giocatore per 100 possessi di gioco
- DEF_RATING: numero di punti concessi da un giocatore per 100 possessi di gioco
- CONFIRMED: la classe modello. Vale 1 se il giocatore gioca in NBA anche nella stagione 2022-23, 0 altrimenti

Manipolazione delle tabelle

Per la tabella relativa alle statistiche dei giocatori della stagione 2021-22 sono state mantenute solamente le colonne significative (nome del giocatore, partite giocate, minuti giocati, punti segnati, tiri tentati, tiri da 3 punti tentati, tiri liberi tentati, rimbalzi totali, assist, palle perse, palle rubate, stoppate, rating offensivo, rating difensivo e la classe modello).
Per la tabella relativa alla stagione 2022-23 sono state mantenute solo le colonne PLAYERN_NAME e CONFIRMED.

La tabella finale è stata ottenuta usando il comando merge con una operazione di join sinistro in base al valore della colonna PLAYER_NAME tra la tabella 2021-22 e 2022-23.
Successivamente è stato rimosso il nome del giocatore dalla tabella finale.

```
data = data1[c("GP", "MIN", "PTS", "FGA_PG", "FG3A", "FTA", "REB",
"AST", "TOV", "STL", "BLK", "OFF_RATING", "DEF_RATING", "CONFIRMED")]
```

Analisi

Analisi preliminare

Distribuzione della classe

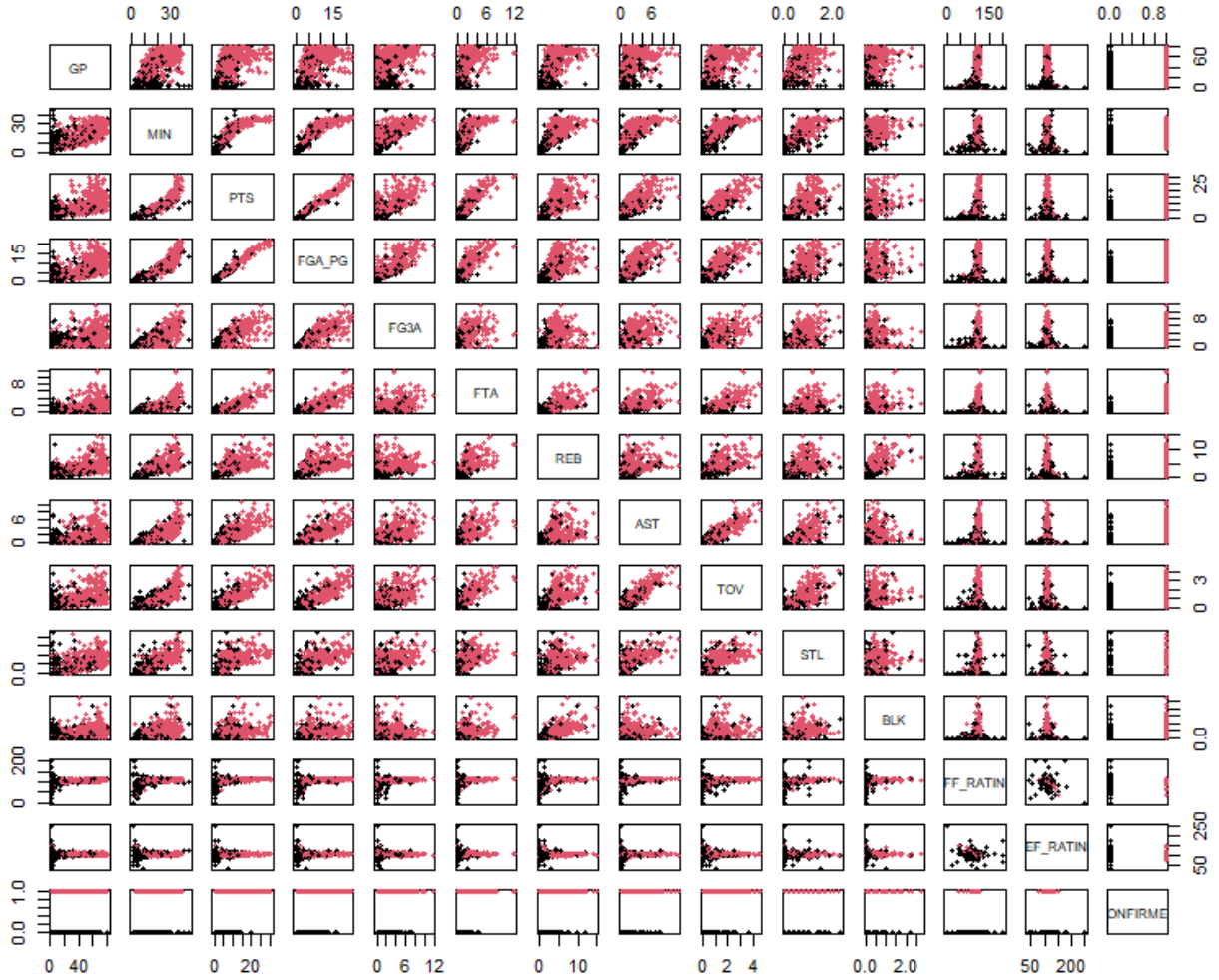
Per prima cosa possiamo notare che la classe non è equamente distribuita:

```
CONFIRMED
  0   1
210 395
```

I giocatori che al termine della stagione sono stati confermati nella lega (395) sono leggermente inferiori al doppio dei giocatori non confermati (210). Il nostro obiettivo è trovare tutti i giocatori da confermare, in quanto possiamo affermare che è preferibile segnalare come un possibile giocatore da mantenere uno che non lo è, rispetto che scartare alcuni giocatori che devono essere mantenuti in squadra. Cercheremo quindi un metodo che massimizzi la sensibilità, ovvero il tasso di positivi corretti, anche a discapito della specificità (tasso di negativi corretti).

Diagramma di dispersione

Dal diagramma di dispersione si può osservare come, sebbene per alcune variabili la distinzione tra giocatori confermati e non confermati sia più netta, ci siano molte osservazioni appartenenti a classi diverse che si sovrappongono, rendendo il problema di trovare tutti i giocatori confermati non banale.



Metodi di classificazione

Dopo la preparazione dei dati, sono state analizzate le performance di tre modelli di classificazione: Regressione Logistica, Discriminante Lineare e Discriminante Quadratico. Le accuratezze ottenute di seguito sono state calcolate senza effettuare divisioni tra training set e test set e pertanto risultano leggermente superiori a quelle effettive.

Regressione logistica

Per prima cosa costruiamo il modello di classificazione mediante regressione logistica valutando CONFIRMED come la variabile di uscita e usando tutti gli altri fattori come parametri di ingresso.
L'analisi porta ad un'accuratezza dell'82,15% classificando correttamente 497 osservazioni, valore ottimo, con la seguente matrice di confusione:

	actual 1	actual 0
predicted 1	349	62
predicted 0	46	148

Come possiamo notare, la sensibilità (88,35%) risulta maggiore della specificità (70,48%). Risultato incoraggiante per la nostra analisi.

Analisi discriminante lineare

Nel caso dell'analisi discriminante lineare l'accuratezza è pari all'81,16% classificando correttamente 491 osservazioni, valore di poco più basso, ma comunque alto. La matrice di confusione è la seguente:

```
            actual 1 actual 0
predicted 1      339       58
predicted 0       56      152
```

Ancora una volta la sensibilità (85,82%) risulta maggiore della specificità (72,38%).

Analisi discriminante quadratica

Applichiamo infine il modello di analisi discriminante quadratica. In questo caso otteniamo un risultato inferiore all'80%. L'accuratezza infatti è pari al 79,50% classificando correttamente 481 osservazioni, il valore più basso ottenuto fino ad ora. La matrice di confusione è la seguente:

```
            actual 1 actual 0
predicted 1      347       76
predicted 0       48      134
```

Come possiamo notare, la sensibilità (87,85%) risulta comunque maggiore della specificità (63,81%).

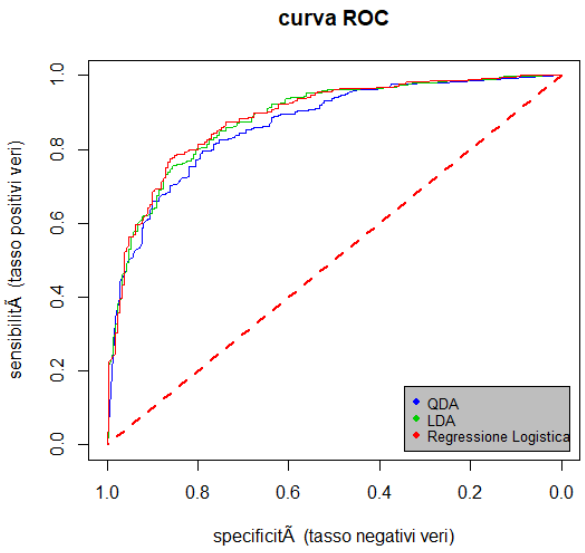
Confronto curve ROC

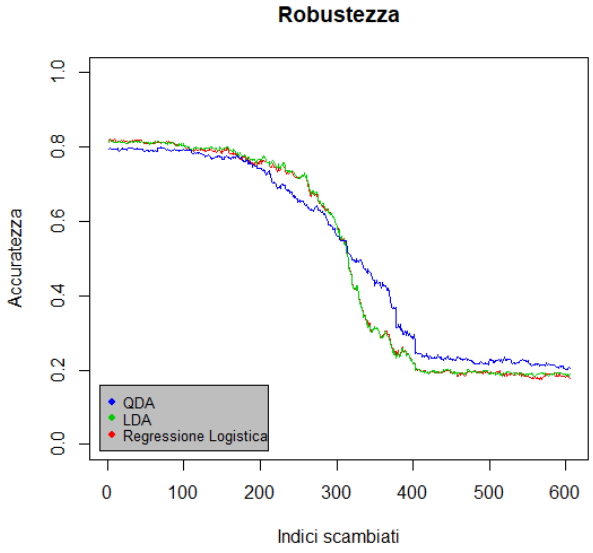
I modelli costruiti restituiscono risultati molto simili tra loro, risulta inevitabile quindi l'utilizzo di metodi di confronto più elaborati per stabilire quale sia il miglior classificatore.
Il confronto delle curve ROC conferma che i tre modelli sono simili, con la regressione logistica ed analisi discriminante lineare di poco migliori. Infatti, valutando i valori dell'area sotto la curva (AUC), possiamo notare che quelli relativi alla regressione logistica ed analisi discriminante lineare siano i più alti. In generale comunque, un valore di AUC superiore a 0,9 indica che il modello è altamente accurato, nel nostro caso il modello con regressione logistica e di analisi discriminante lineare sono leggermente al di sotto di questa soglia.

```
> #qda
> s2_auc(data.qda.roc)
[1] 0.8655046
> #lda
> s2_auc(data.lda.roc)
[1] 0.8815283
> #regressione logistica
> s2_auc(data.glm.roc)
[1] 0.8853015
```

Confronto robustezza modelli

Nel test sulla robustezza dei modelli, che consiste nell'introdurre informazioni false per studiare l'andamento dell'accuratezza, si può notare come i modelli siano accurati fino a che i valori sbagliati non superano la metà dei valori totali. Da quel punto in poi l'accuratezza cala a picco, il che significa che i modelli sono accurati, ma in senso opposto. Questo andamento dell'accuratezza denota un'elevata robustezza dei modelli.



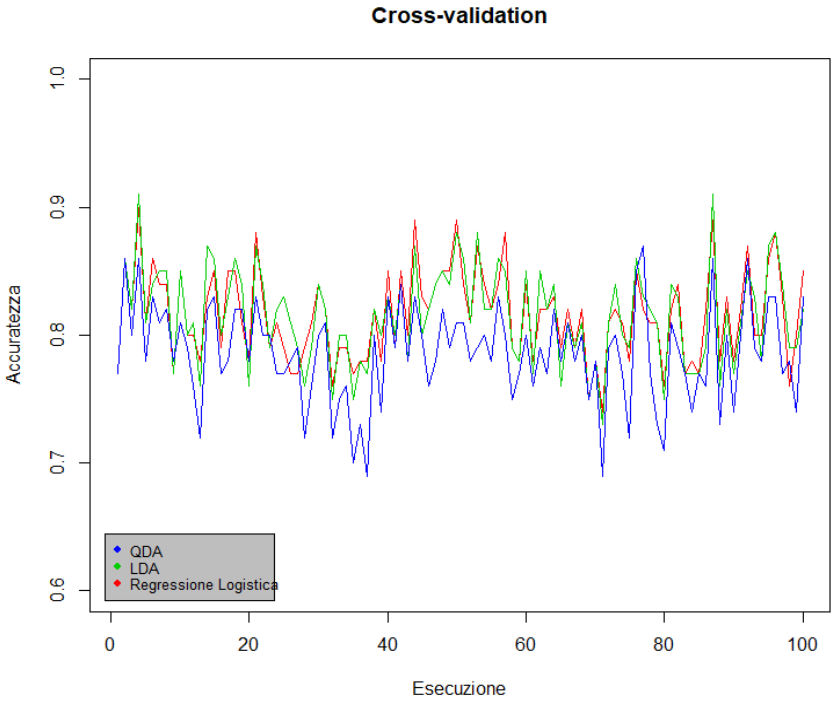


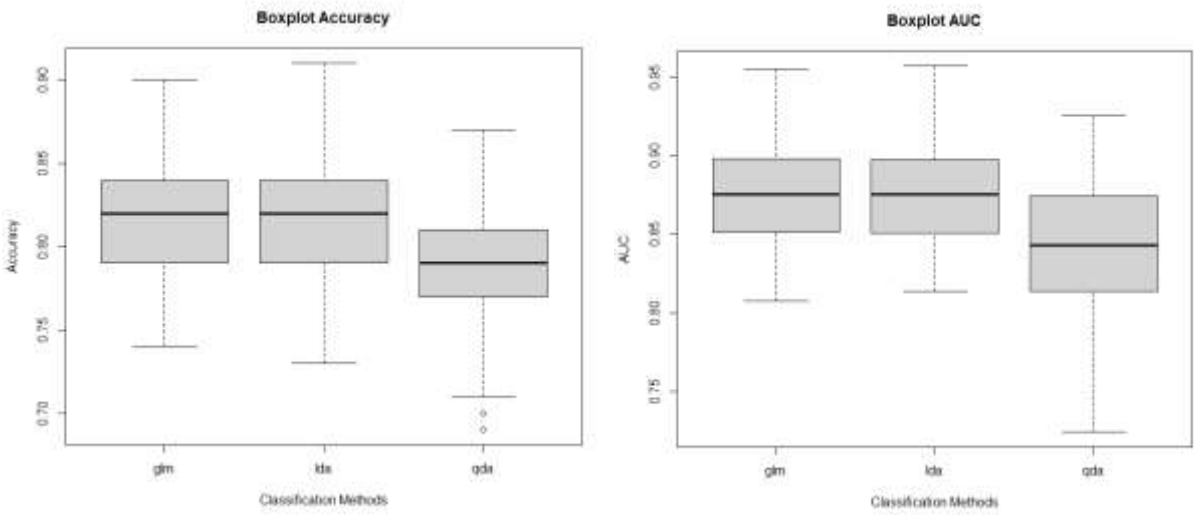
A seguito dei vari test risulta che il modello di regressione logistica e di analisi discriminante lineare siano quelli preferibili. Proviamo adesso ad effettuare un'autovalutazione con cross-validation per decretare definitivamente il modello migliore e per verificare di non avere problemi di overfitting.

Predizione e autovalutazione

Suddividiamo le osservazioni originali in un sottoinsieme di 100 osservazioni, il test set, e un altro sottoinsieme di 505 osservazioni, il training set e ripetiamo l'esperimento 100 volte. Sono stati confrontati i valori medi e le deviazioni standard dell'accuratezza e dell'AUC ottenuti per ogni modello. Come preannunciato, le accuratezze calano lievemente rispetto a quelle ottenute senza la divisione del dataset in training test e test set.

```
> # regressione logistica
> mean(acc[,1])
[1] 0.8156
> sd(acc[,1])
[1] 0.03540017
> mean(auc[,1])
[1] 0.8751718
> sd(auc[,1])
[1] 0.03164039
> # lda
> mean(acc[,2])
[1] 0.8153
> sd(acc[,2])
[1] 0.03793815
> mean(auc[,2])
[1] 0.8759353
> sd(auc[,2])
[1] 0.03200119
> # qda
> mean(acc[,3])
[1] 0.7859
> sd(acc[,3])
[1] 0.03829959
> mean(auc[,3])
[1] 0.8426418
> sd(auc[,3])
[1] 0.03927887
```





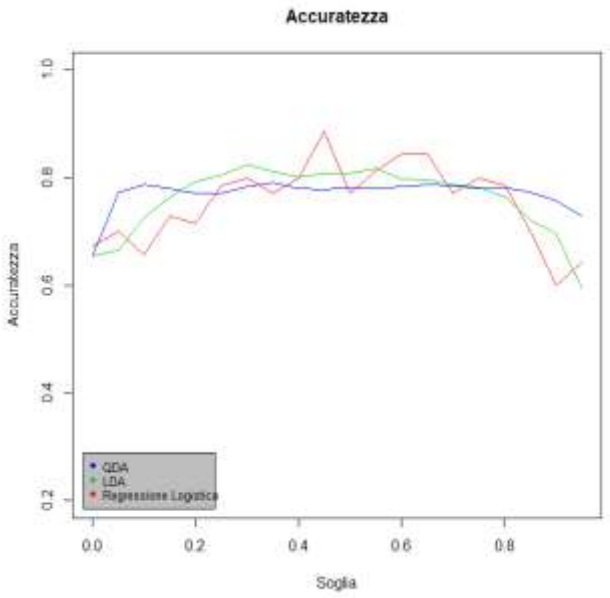
Nuovamente la regressione logistica e l'analisi discriminante lineare si rivelano essere le più affidabili, con accuratezze e valori di AUC superiori. Per questo motivo è stato ritenuto necessario effettuare un test per valutare se la lieve differenza di accuratezza sia o meno statisticamente significativa. È stato utilizzato il test Wilcoxon in quanto è un test non parametrico, dato che non abbiamo informazioni sulla distribuzione dell'accuratezza. Dato il valore estremamente elevato del p-value, che ci impedisce di rigettare l'ipotesi nulla, si può concludere che non vi sia differenza statistica tra le accuratezze registrate per i modelli di regressione logistica e di analisi discriminante lineare e che quindi i risultati forniti da essi siano statisticamente equivalenti.

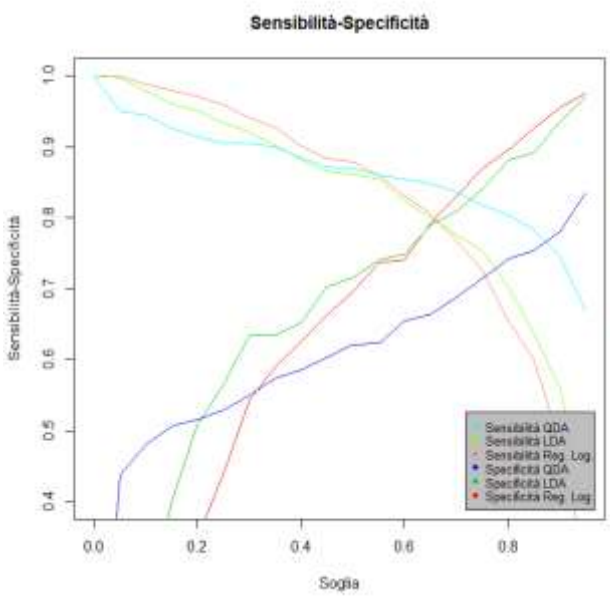
```
        Wilcoxon rank sum test with continuity correction

data:  acc[, 1] and acc[, 2]
W = 4988, p-value = 0.9775
alternative hypothesis: true location shift is not equal to 0
```

Sensibilità

Come già detto, l'obiettivo è quello di massimizzare la sensibilità, potrebbe essere necessario quindi effettuare un cambio di soglia di probabilità per i vari modelli. Allo stesso tempo è anche importante valutare come diminuisce la specificità, poiché, nonostante la sensibilità sia molto più importante, non è auspicabile che ci sia un alto numero di giocatori confermati che dovrebbero essere tagliati fuori dalla squadra. Valutiamo dunque come variano i valori di accuratezza, sensibilità e specificità al variare della soglia.
Come possiamo vedere dai grafici seguenti a livello di sensibilità-specificità otteniamo risultati migliori con i modelli di regressione logistica e analisi discriminante lineare. Diminuendo la soglia otteniamo un valore di sensibilità maggiore ma come detto precedentemente dobbiamo comunque trovare un tradeoff tra sensibilità e specificità.
Il miglior risultato sembra verificarsi con il modello di regressione logistica, utilizzando una soglia di probabilità pari al 45%. Infatti per quel valore abbiamo un'accuratezza dell'89%, una sensibilità dell'88% e una specificità del 66%. Con il modello analisi discriminante lineare otteniamo risultati simili se non per l'accuratezza pari all'81%.





Conclusioni

I vari test effettuati per confrontare i modelli hanno restituito esiti migliori per i modelli di regressione logistica ed analisi discriminante lineare, inoltre, tra i suddetti modelli, è stata registrata una forte somiglianza che non ha permesso di preferirne uno piuttosto che l'altro. L'accuratezza ottenuta è soddisfacente e difficilmente migliorabile: non è stato possibile ottenere un'accuratezza più elevata sia costruendo i modelli tramite una riduzione di fattori, sia considerando combinazioni di fattori di ingresso differenti da quelli utilizzati, sia costruendo i modelli logaritmici (in quanto la natura dei dati analizzati è lineare, come emerso dalle conclusioni della prima relazione).

Un'ipotesi sull'impossibilità di incrementare l'accuratezza e sulla somiglianza tra i risultati ottenuti con i tre modelli analizzati è che essi presentino le stesse difficoltà a classificare le medesime osservazioni. Ciò può essere dovuto al fatto che i contratti dei giocatori molto spesso sono su base pluriennale. Questo significa che un giocatore può ottenere statistiche non ottime ma avere ancora anni di contratto e di conseguenza essere confermato. Sarebbe possibile ottenere un'accuratezza migliore realizzando un modello di classificazione basato solamente sui giocatori all'ultimo anno di contratto.

In definitiva comunque i modelli scelti hanno un'accuratezza migliore, un comportamento più affidabile e sono molto più efficaci nel classificare i giocatori da confermare, obiettivo primario della classificazione.

Per concludere, un modello statistico di questo tipo, che permetta di classificare i giocatori è utile per le varie franchigie NBA in quanto il fattore di uscita permette di determinare i giocatori da mantenere in base alle loro statistiche, in modo da poter investire su di essi.

Appendice

Codice Manipolazioni

Lo script utilizzato per ottenere le due tabelle utilizzate nell'analisi, fa uso delle nba-api ufficiali per Python. La funzione get_data() scarica i dati sotto forma di liste di dictonaries utilizzando la funzione leaguedashtplayerstatsLeagueDashPlayerStats delle nba-api. Specificando il valore "Base" del parametro measure_type_detailed_defense sarà possibile scaricare le informazioni della tabella Players General Traditional; specificando invece il valore "Advanced" le informazioni saranno quelle della tabella Players General Advanced. Una volta ottenute entrambe le liste di dictionaries, get_data() le unisce usando il codice riportato in seguito e le aggiunge al dataframe da restituire. Dal dataframe ottenuto tramite get_data(), vengono rimosse tutte le informazioni che non sono presenti nelle tabelle ai link riportati in precedenza. In particolare vengono rimosse le colonne contenenti i ranking relativi ai vari aspetti del gioco (PTS RANK, AST RANK...). il dataframe viene salvato in formato .csv con il nome "stat1.csv" per la stagione 2021-22 e "stat2.csv" per la stagione 2022-23.

Ai dataframe ottenuti vengono aggiunte le colonne CONFIRMED per determinare tramite un join sinistro i giocatori che dopo la stagione 2021-22 hanno proseguito il loro percorso nella NBA.

Infine, il dataframe viene salvato in formato .csv con il nome "tabella.csv".

```
pip install nba_api
import pandas as pd
from nba_api.stats.endpoints import leaguedashplayerstats
from collections import defaultdict
import time
customHeaders = {
    'Host': 'stats.nba.com',
    'Connection': 'keep-alive',
    'Cache-Control': 'max-age=0',
    'Upgrade-Insecure-Requests': '1',
    'User-Agent':   'Mozilla/5.0   (Windows   NT   6.1;   Win64;   x64)
AppleWebKit/537.36    (KHTML,    like    Gecko)    Chrome/61.0.3163.100
Safari/537.36',
    'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/
apng,*/*;q=0.8,application/signed-exchange;v=b3',
    'Accept-Encoding': 'gzip, deflate, br',
    'Accept-Language': 'en-US,en;q=0.9',
    'Referer': 'https://stats.nba.com/',
    'x-nba-stats-origin': 'stats',
    'x-nba-stats-token': 'true',
}
# Create a list of the seasons to analyze
seasons = ['2021-22']
seasons2 = ['2022-23']
# Scrapes stats.nba.com to get season average statistics using nba-api,
# creates Pandas dataframe from statistics
def get_data(season_list):
    time.sleep(1)
    stats = pd.DataFrame()

    for season in season_list:
        # Get Players Traditional Info
        allPlayersTraditionalInfo                                          =
leaguedashplayerstats.LeagueDashPlayerStats(per_mode_detailed='PerGame',

season=season,
```

```
measure_type_detailed_defense='Base',

headers=customHeaders,

timeout=120)
        allPlayersTraditionalDict                                       =
allPlayersTraditionalInfo.get_normalized_dict()
        allPlayersTraditionalList                                       =
allPlayersTraditionalDict['LeagueDashPlayerStats']

        # Get Players Advanced Info
        allPlayersAdvancedInfo                                          =
leaguedashplayerstats.LeagueDashPlayerStats(per_mode_detailed='PerGame',

season=season,

measure_type_detailed_defense='Advanced',

headers=customHeaders,

timeout=120)
        allPlayersAdvancedDict                                          =
allPlayersAdvancedInfo.get_normalized_dict()
        allPlayersAdvancedList                                          =
allPlayersAdvancedDict['LeagueDashPlayerStats']

        # Merge the two lists of dicts (Traditional Info and Advanced
Info)
        d = defaultdict(dict)
        for l in (allPlayersTraditionalList, allPlayersAdvancedList):
            for elem in l:
                d[elem['PLAYER_ID']].update(elem)
        result = d.values()

        # Add the merged dict to the DataFrame
        df = pd.DataFrame(result)

        # Add season information
        df.insert(0, 'SEASON', season)

        # Concat with previous years stats
        stats = pd.concat([stats, df], ignore_index=True)

    return stats
df = get_data(seasons)
# Find the columns that do not appear on the NBA official website
indexes_to_drop = []
i = 0
for column in df.columns:
    if "RANK" in column:
        indexes_to_drop.append(i)

    if column == "SEASON" or column == "TEAM_ID" or column == "CFPARAMS"
or column == "CFID" or column == "PACE_PER40" or column == "PLAYER_ID" or
column == "NICKNAME" or column == "W_PCT" or column == "FGM" or column ==
"FGA" or column == "BLKA" or column == "PFD" or column == "NBA_FANTASY_PTS"
or  column  ==  "WNBA_FANTASY_PTS"  or  column  ==  "sp_work_OFF_RATING"  or
```

```
column == "sp_work_DEF_RATING" or column == "sp_work_NET_RATING" or column
== "sp_work_PACE":
        indexes_to_drop.append(i)

    if column.startswith("E_"):
        indexes_to_drop.append(i)

    i += 1
# Drop the columns
df.drop(df.columns[indexes_to_drop], axis=1, inplace=True)
# Add the new column
df['CONFIRMED'] = 0
df
# Save dataframe in order to perform R analysis
df.to_csv('stat1.csv', sep=";", index=False)
df2 = get_data(seasons2)
# Find the columns that do not appear on the NBA official website
indexes_to_drop = []
i = 0
for column in df2.columns:
    if "RANK" in column:
        indexes_to_drop.append(i)

    if column == "SEASON" or column == "TEAM_ID" or column == "CFPARAMS"
or column == "CFID" or column == "PACE_PER40" or column == "PLAYER_ID" or
column == "NICKNAME" or column == "W_PCT" or column == "FGM" or column ==
"FGA" or column == "BLKA" or column == "PFD" or column == "NBA_FANTASY_PTS"
or column == "WNBA_FANTASY_PTS" or column == "sp_work_OFF_RATING" or
column == "sp_work_DEF_RATING" or column == "sp_work_NET_RATING" or column
== "sp_work_PACE" or column == "TEAM_ABBREVIATION" or column == "AGE" or
column == "GP" or column == "W" or column == "L" or column == "MIN" or
column == "FG_PCT" or column == "FG3M" or column == "FG3A" or column ==
"FG3_PCT" or column == "FTM" or column == "FTA" or column == "FT_PCT" or
column == "OREB" or column == "DREB" or column == "REB" or column == "AST"
or column == "TOV" or column == "STL" or column == "BLK" or column == "PF"
or column == "PTS" or column == "PLUS_MINUS" or column == "DD2" or column
== "TD3" or column == "OFF_RATING" or column == "DEF_RATING" or column ==
"NET_RATING" or column == "AST_PCT" or column == "AST_TO" or column ==
"AST_RATIO" or column == "OREB_PCT" or column == "DREB_PCT" or column ==
"REB_PCT" or column == "TM_TOV_PCT" or column == "EFG_PCT" or column ==
"EFG_PC" or column == "USG_PCT" or column == "PACE" or column == "PIE" or
column == "POSS" or column == "FGM_PG" or column == "FGA_PG" or column ==
'TS_PCT':
        indexes_to_drop.append(i)

    if column.startswith("E_"):
        indexes_to_drop.append(i)

    i += 1
# Drop the columns
df2.drop(df2.columns[indexes_to_drop], axis=1, inplace=True)
# Add the new column
df2['CONFIRMED'] = 1
df2
# Save dataframe in order to perform R analysis
df2.to_csv('stat2.csv', sep=";", index=False)
tabella = pd.merge(df, df2, how = 'left', on='PLAYER_NAME')
tabella['CONFIRMED_y'] = tabella['CONFIRMED_y'].fillna(0)
tabella.drop('CONFIRMED_x', axis=1, inplace=True)
```

```
tabella.rename(columns = {'CONFIRMED_y':'CONFIRMED'}, inplace = True)
tabella
tabella = pd.merge(df, df2, how = 'left', on='PLAYER_NAME')
tabella['CONFIRMED_y'] = tabella['CONFIRMED_y'].fillna(0)
tabella.drop('CONFIRMED_x', axis=1, inplace=True)
tabella.rename(columns = {'CONFIRMED_y':'CONFIRMED'}, inplace = True)
tabella
# Save dataframe in order to perform R analysis
tabella.to_csv('tabella.csv', sep=";", index=False)
```

Codice Analisi

```
library(corrplot)
library(caret)
library(MASS)
require(multiROC)
source("s2_helper.r")
library(readr)

# caricamento dei dati dal file csv
data1 = read.csv('tabella.csv', header = TRUE, sep = ";")

#tabella finale
data = data1[c("GP", "MIN", "PTS", "FGA_PG", "FG3A", "FTA", "REB", "AST",
"TOV", "STL", "BLK", "OFF_RATING", "DEF_RATING", "CONFIRMED")]

#salvo la tabella
final = data

#distribuzione classe
with(data, table(CONFIRMED))

#diagramma di dispersione
plot(data,pch=20,col=1+data$CONFIRMED)

################ Regressione logisitca ################

data = final

data.glm=glm(CONFIRMED~.,data=data,family=binomial)

data.glm.p=predict(data.glm,type="response")
#accuratezza
sum((data.glm.p>0.5)==(data$CONFIRMED>0.5))/length(data$CONFIRMED)
s2_confusion(data$CONFIRMED,data.glm.p)
data.glm.roc = s2_roc(data$CONFIRMED,data.glm.p)


################ Analisi Discriminante Lineare ################

data.lda = lda(CONFIRMED~., data=data,CV=F)

data.lda.p=predict(data.lda)
data.lda.post=data.lda.p$posterior[,2]
sum((data.lda.post>0.5)==(data$CONFIRMED>0.5))/length(data$CONFIRMED)

s2_confusion(data$CONFIRMED,data.lda.post)

data.lda.roc=s2_roc(data$CONFIRMED,data.lda.post)

################ Analisi Discriminante Quadratica ################

data.qda = qda(CONFIRMED~., data=data,CV=F)

data.qda.p=predict(data.qda)
data.qda.post=data.qda.p$posterior[,2]
```

```
sum((data.qda.post>0.5)==(data$CONFIRMED>0.5))/length(data$CONFIRMED)

s2_confusion(data$CONFIRMED,data.qda.post)

data.qda.roc=s2_roc(data$CONFIRMED,data.qda.post)

################ Confronto ROC e AUC ################

s2_roc.plot(data.qda.roc,col="blue")
s2_roc.lines(data.lda.roc,col="green3")
s2_roc.lines(data.glm.roc,col="red")

legend("bottomright",
        inset = 0.02,
        c("QDA", "LDA", "Regressione Logistica"),
        col = c("blue", "green3", "red"),
        pch = 19,
        bg = "gray",
        cex=0.8)

#qda
s2_auc(data.qda.roc)
#lda
s2_auc(data.lda.roc)
#regressione logistica
s2_auc(data.glm.roc)

################ Robustezza ###############

idx=sample(605,605)
acc=matrix(0,605, 3)
for(i in 1:605){
  dataf=data
  dataf$CONFIRMED[idx[1:i]]=as.numeric(!dataf$CONFIRMED[idx[1:i]])

  data.glm=glm(CONFIRMED~.,data=dataf,family=binomial)
  data.glm.p=predict(data.glm,type="response")
  acc[i,1]=sum((data.glm.p>0.5)==(data$CONFIRMED>0.5))/length(data$CONFIRMED)

  data.lda = lda(CONFIRMED~., data=dataf,CV=F)
  data.lda.p=predict(data.lda)
  data.lda.post=data.lda.p$posterior[,2]

acc[i,2]=sum((data.lda.post>0.5)==(data$CONFIRMED>0.5))/length(data$CONFIRMED)

  data.qda = qda(CONFIRMED~., data=dataf,CV=F)
  data.qda.p=predict(data.qda)
  data.qda.post=data.qda.p$posterior[,2]

acc[i,3]=sum((data.qda.post>0.5)==(data$CONFIRMED>0.5))/length(data$CONFIRMED)
}

#regressione logistica
plot(acc[,1], type='l', col="red", ylim=c(0,1), xlab="Indici scambiati",
ylab="Accuratezza", main="Robustezza")
#lda
lines(acc[,2], type='l', col="green3")
#qda
lines(acc[,3], type='l', col="blue")

legend("bottomleft",
        inset = 0.02,
        c("QDA", "LDA", "Regressione Logistica"),
        col = c("blue", "green3", "red"),
        pch = 19,
```

```
        bg = "gray",
        cex=0.8)

################ Autovalutazione ################

library(pROC)
l=length(data$CONFIRMED)
acc=matrix(0,100,3)
auc=matrix(0,100,3)
for(i in 1:100){

  idx=sample(l,100)
  datatrain=data[-idx,]

  data.glm=glm(CONFIRMED~.,family=binomial,data=datatrain)
  data.glm.p=predict(data.glm,data[idx,],type="response")
  acc[i,1]=sum((data.glm.p>0.5)==(data$CONFIRMED[idx]>0.5))/100
  auc[i,1]=roc(data$CONFIRMED[idx], data.glm.p)$auc

  data.lda=lda(CONFIRMED~.,data=datatrain)
  data.lda.p=predict(data.lda,data[idx,])$posterior[,2]
  acc[i,2]=sum((data.lda.p>0.5)==(data$CONFIRMED[idx]>0.5))/100
  auc[i,2]=roc(data$CONFIRMED[idx], data.lda.p)$auc

  data.qda=qda(CONFIRMED~.,data=datatrain)
  data.qda.p=predict(data.qda,data[idx,])$posterior[,2]
  acc[i,3]=sum((data.qda.p>0.5)==(data$CONFIRMED[idx]>0.5))/100
  auc[i,3]=roc(data$CONFIRMED[idx], data.qda.p)$auc

}

# regressione logistica
mean(acc[,1])
sd(acc[,1])

mean(auc[,1])
sd(auc[,1])

# lda
mean(acc[,2])
sd(acc[,2])

mean(auc[,2])
sd(auc[,2])

# qda
mean(acc[,3])
sd(acc[,3])

mean(auc[,3])
sd(auc[,3])

# risultati
boxplot(acc, main="Boxplot Accuracy", xlab="Classification Methods",
ylab="Accuracy", xaxt="n")
axis(1,at=1:3,labels=c("glm", "lda", "qda"))

boxplot(auc, main="Boxplot AUC", xlab="Classification Methods", ylab="AUC",
xaxt="n")
axis(1,at=1:3,labels=c("glm", "lda", "qda"))

wilcox.test(acc[,1], acc[,2])

#regressione logistica
```

```
plot(acc[,1], type='l', col="red", ylim=c(0.6,1), xlab="Esecuzione",
ylab="Accuratezza", main="Cross-validation")
#lda
lines(acc[,2], type='l', col="green3")
#qda
lines(acc[,3], type='l', col="blue")

legend("bottomleft",
        inset = 0.02,
        c("QDA", "LDA", "Regressione Logistica"),
        col = c("blue", "green3", "red"),
        pch = 19,
        bg = "gray",
        cex=0.8)

################ Sensibilità ################

l=length(data$CONFIRMED)

meanAcc=matrix(0,20,3)
meanSens=matrix(0,20,3)
meanSpec=matrix(0,20,3)
prob=rep(0,20)

for(j in 1:20){

  acc=matrix(0,70,3)
  sens=matrix(0,70,3)
  spec=matrix(0,70,3)

  p = 0 + ((j-1)*0.05)
  prob[j]=p

  for(i in 1:70){

    idx=sample(l,70)
    datatrain=data[-idx,]

    data.glm=glm(CONFIRMED~.,family=binomial,data=datatrain)
    data.glm.p=predict(data.glm,data[idx,],type="response")

    acc[i,1]=sum((data.glm.p>p)==(data$CONFIRMED[idx]>p))/70
    tp=sum((data.glm.p>p)&(data$CONFIRMED[idx]==1))
    fn=sum((data.glm.p<p)&(data$CONFIRMED[idx]==1))
    sens[i,1]=tp/(tp+fn)

    tn=sum((data.glm.p<p)&(data$CONFIRMED[idx]==0))
    fp=sum((data.glm.p>p)&(data$CONFIRMED[idx]==0))
    spec[i,1]=tn/(tn+fp)

    data.lda=lda(CONFIRMED~.,data=datatrain)
    data.lda.p=predict(data.lda,data[idx,])$posterior[,2]
    acc[i,2]=sum((data.lda.p>p)==(data$CONFIRMED[idx]>p))/70

    tp=sum((data.lda.p>p)&(data$CONFIRMED[idx]==1))
    fn=sum((data.lda.p<p)&(data$CONFIRMED[idx]==1))
    sens[i,2]=tp/(tp+fn)

    tn=sum((data.lda.p<p)&(data$CONFIRMED[idx]==0))
    fp=sum((data.lda.p>p)&(data$CONFIRMED[idx]==0))
    spec[i,2]=tn/(tn+fp)

    data.qda=qda(CONFIRMED~.,data=datatrain)
    data.qda.p=predict(data.qda,data[idx,])$posterior[,2]
    acc[i,3]=sum((data.qda.p>p)==(data$CONFIRMED[idx]>p))/70
```

```
    tp=sum((data.qda.p>p)&(data$CONFIRMED[idx]==1))
    fn=sum((data.qda.p<p)&(data$CONFIRMED[idx]==1))
    sens[i,3]=tp/(tp+fn)

    tn=sum((data.qda.p<p)&(data$CONFIRMED[idx]==0))
    fp=sum((data.qda.p>p)&(data$CONFIRMED[idx]==0))
    spec[i,3]=tn/(tn+fp)
  }

  #logistica
  meanAcc[j,1]=mean(acc[i,1])
  meanAcc[j,2]=mean(acc[,2])
  meanAcc[j,3]=mean(acc[,3])

  #lda
  meanSens[j,1]=mean(sens[,1])
  meanSens[j,2]=mean(sens[,2])
  meanSens[j,3]=mean(sens[,3])

  #qda
  meanSpec[j,1]=mean(spec[,1])
  meanSpec[j,2]=mean(spec[,2])
  meanSpec[j,3]=mean(spec[,3])

}


#regressione logistica
plot(prob, meanAcc[,1], type='l', col="red", ylim=c(0.2,1), xlab="Soglia",
ylab="Accuratezza", main="Accuratezza")
#lda
lines(prob, meanAcc[,2], type='l', col="green3")
#qda
lines(prob, meanAcc[,3], type='l', col="blue")

legend("bottomleft",
        inset = 0.02,
        c("QDA", "LDA", "Regressione Logistica"),
        col = c("blue", "green3", "red"),
        pch = 19,
        bg = "gray",
        cex=0.8)

#regressione logistica
plot(prob, meanSens[,1],  type='l', col="coral", ylim=c(0.4,1), xlab="Soglia",
ylab="Sensibilità-Specificità", main="Sensibilità-Specificità")
#lda
lines(prob, meanSens[,2], type='l', col="chartreuse")
#qda
lines(prob, meanSens[,3], type='l', col="cyan")
#regressione logistica
lines(prob, meanSpec[,1], type='l', col="red")
#lda
lines(prob, meanSpec[,2], type='l', col="green3")
#qda
lines(prob, meanSpec[,3], type='l', col="blue")

legend("bottomright",
        inset = 0.02,
        c("Sensibilità QDA", "Sensibilità LDA", "Sensibilità Reg. Log.",
"Specificità QDA", "Specificità LDA", "Specificità Reg. Log."),
        col = c("cyan", "chartreuse", "coral","blue", "green3", "red"),
        pch = 19,
        bg = "gray",
```

```
cex=0.8)
```