Progetto di Statistica II: Prima 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 regressione lineare multivariata per poter predire il rating che un giocatore NBA avrà al termine di una stagione date le attuali statistiche per partita del giocatore stesso.
L'analisi può essere utilizzata dalla 2K Sports, azienda statunitense produttrice di videogiochi, con l'obiettivo di poter determinare il possibile rating dei giocatori per la nuova edizione del videogioco. Il modello può anche risultare particolarmente utile per prevedere i futuri talenti della NBA. Si può ipotizzare quindi che l'analisi sia stata commissionata per il mercato dei videogiochi oppure per aziende che intendono firmare una sponsorship con i giocatori.

Dataset

La tabella di dati è stata ottenuta tramite l'unione di due tabelle: una relativa alle statistiche ed una relativa ai rating dei giocatori.

https://www.nba.com/stats/players/traditional?Season=2021-22
https://www.nba.com/stats/players/advanced?Season=2021-22
https://hoopshype.com/nba2k/players/

Per ottenere le tabelle relative alle statistiche è stato utilizzato uno script Python che sarà analizzato nel dettaglio nell'appendice di questa relazione.
Per ottenere i valori relativi ai rating, non essendo disponibile una libreria Python, è stato necessario importare i dati manualmente in formato csv.
L'analisi è stata effettuata prendendo in considerazione le statistiche dei giocatori della stagione 2021-22; per quanto riguarda il rating, è stato utilizzato quello assegnato dalla 2K Sports al momento dell'uscita del videogioco NBA2K23 (stagione 2022-23).

Contenuto del Dataset

La tabella derivata contiene 14 colonne e 416 osservazioni. Le colonne sono le seguenti:

- RANK: rating del giocatore nel videogioco NBA2K23
- 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

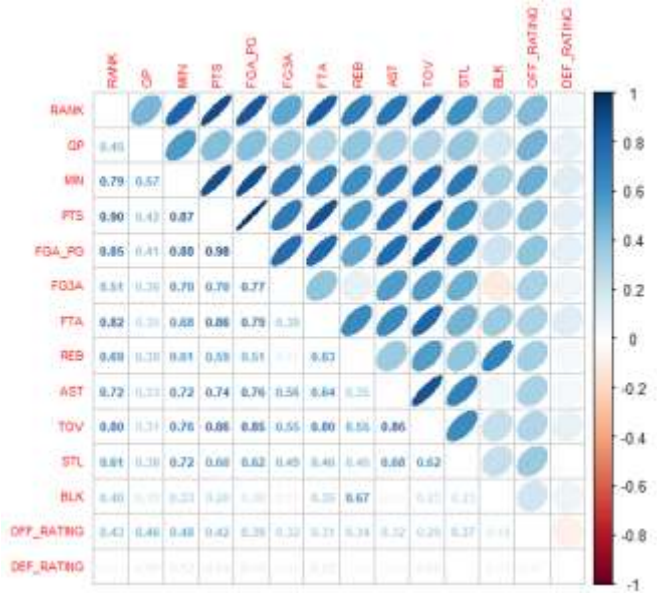
Manipolazione delle tabelle

Per la tabella relativa alle statistiche dei giocatori 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 e rating difensivo).
La tabella finale è stata ottenuta usando il comando merge in base al valore della colonna PLAYER_NAME
Successivamente è stato rimosso il nome del giocatore dalla tabella finale.

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

Il valore finale di osservazioni inferiore sia a quello della tabella relativa alle statistiche che quella relativa ai ratings è dovuto dal fatto che ogni anno un certo numero di giocatori termina/inizia la sua esperienza nella NBA.

Considerazioni preliminari



Come è possibile osservare dal grafico riassuntivo delle correlazioni, i ratings risultano essere nettamente correlati (>|0.6|) con gli indicatori dei minuti giocati, punti segnati, tiri tentati, tiri liberi tentati, rimbalzi, assist, palle perse e palle rubate. La correlazione positiva con il numero di palle perse può essere dovuta dal fatto che i giocatori con più palle perse molto spesso sono coloro che giocano più minuti e di conseguenza i più forti della squadra. I tiri tentati da 3pt (|0.51|), le stoppate (|0.40|) e l'OFF_RATING (|0.43|) hanno invece delle correlazioni più lievi. Non vi sono ulteriori correlazioni significative.

Analisi

Per effettuare l'analisi sono stati confrontati un modello di regressione lineare e un modello di regressione non lineare (logaritmico). Dato il consistente numero di fattori di ingresso, per entrambi i modelli, è stata operata una riduzione valutando di volta in volta, i valori di varianza spiegata, varianza spiegata corretta e i p-value dei fattori di ingresso.

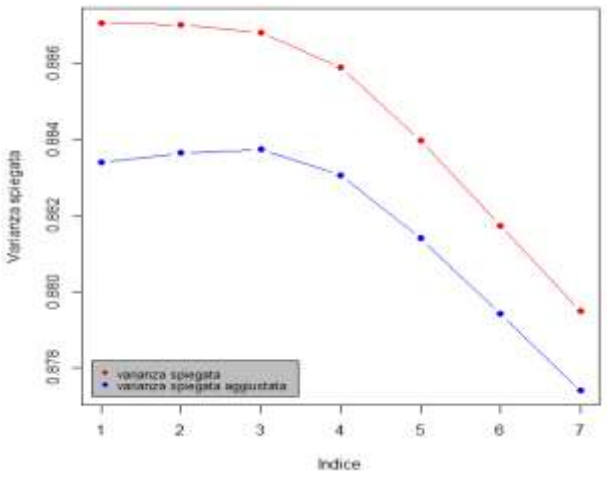
Modello di regressione lineare

Per prima cosa costruiamo il modello di regressione lineare valutando il RANK dei giocatori come variabile di uscita e usando tutti gli altri attributi come parametri di ingresso.
La varianza spiegata è superiore all'88%, valore più che soddisfacente. Valutando i p-value dei vari fattori possiamo notare che alcuni di essi contribuiscono in maniera poco significativa alla determinazione dei fattori di uscita.
Applichiamo quindi una riduzione del modello.

```
Call:
lm(formula = RANK ~ ., data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-7.0138 -1.1034  0.0068  1.0539 15.6610

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 73.025149   2.710299  26.944  < 2e-16 ***
GP           0.022663   0.006219   3.644 0.000304 ***
MIN         -0.149695   0.033499  -4.469 1.02e-05 ***
PTS          1.360315   0.129700  10.488  < 2e-16 ***
FGA_PG      -0.727697   0.148623  -4.896 1.42e-06 ***
FG3A        -0.074523   0.090048  -0.828 0.408392
FTA         -0.329175   0.163528  -2.013 0.044785 *
REB          0.460124   0.082188   5.598 4.01e-08 ***
AST          0.792456   0.120028   6.602 1.28e-10 ***
TOV         -0.689602   0.340671  -2.024 0.043606 *
STL          1.221178   0.399139   3.060 0.002365 **
BLK          1.084439   0.376232   2.882 0.004159 **
OFF_RATING  -0.007295   0.017315  -0.421 0.673762
DEF_RATING  -0.044105   0.016961  -2.600 0.009656 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.996 on 402 degrees of freedom
Multiple R-squared:  0.8871,    Adjusted R-squared:  0.8834
F-statistic: 242.9 on 13 and 402 DF,  p-value: < 2.2e-16
```

Riduzione del modello

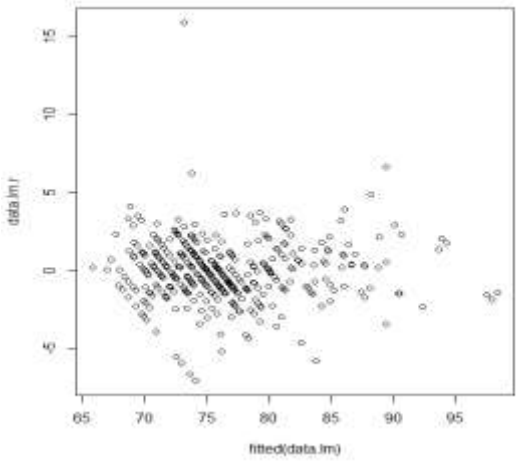
```
Call:
lm(formula = RANK ~ ., data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-7.0637 -1.1166  0.0148  1.0677 15.8654

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 72.186378   1.827905  39.491  < 2e-16 ***
GP           0.022396   0.006041   3.707 0.000239 ***
MIN         -0.145869   0.031577  -4.619 5.18e-06 ***
PTS          1.183410   0.092576  12.783  < 2e-16 ***
FGA_PG      -0.611819   0.124409  -4.918 1.27e-06 ***
REB          0.469462   0.074867   6.271 9.23e-10 ***
AST          0.805389   0.118670   6.787 4.09e-11 ***
TOV         -0.851074   0.326182  -2.609 0.009411 **
STL          1.204986   0.398935   3.021 0.002683 **
BLK          1.135475   0.373656   3.039 0.002529 **
DEF_RATING  -0.043618   0.016737  -2.606 0.009498 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.998 on 405 degrees of freedom
Multiple R-squared:  0.8859,    Adjusted R-squared:  0.8831
F-statistic: 314.4 on 10 and 405 DF,  p-value: < 2.2e-16
```

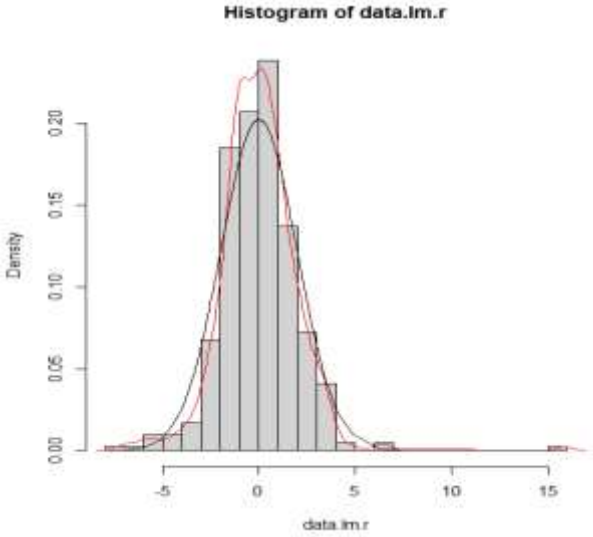


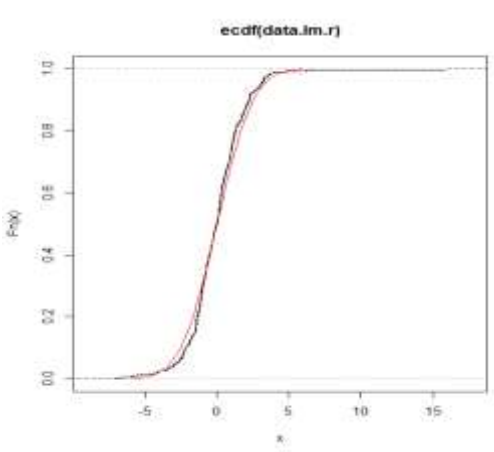
Eliminando i fattori d’ingresso in ordine di p-value decrescente possiamo ottenere un grafico che delinea i cambiamenti di varianza spiegata e varianza spiegata aggiustata ogni volta che un fattore viene rimosso dal modello. Si può notare un netto calo di entrambe le varianze tra i punti di ascissa 4 e 5. Ne consegue che la quarta versione è la migliore. Possiamo notare che la varianza spiegata rimane comunque superiore all’80% e i p-value dei fattori sono quasi nulli. Ne consegue che il modello di regressione ottenuto riesce a catturare buona parte del problema ed è statisticamente significativo.

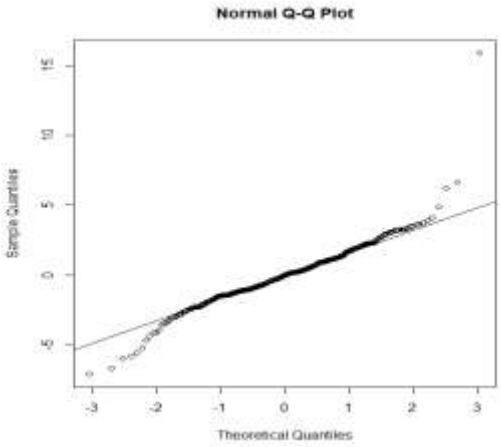
Analisi dei residui

Procediamo dunque con l’analisi dei residui. L’obiettivo è ottenere dei residui che siano disposti uniformemente intorno allo zero e che abbiano una distribuzione Gaussiana.









Dal diagramma di dispersione possiamo notare che i residui non sono dispersi in modo del tutto uniforme intorno allo zero, possiamo infatti notare la presenza di outlier, in particolare nella parte superiore-sinistra del grafico. Sempre dal diagramma di dispersione possiamo vedere che i residui tendono ad aver un certo pattern a linee oblique parallele nella parte sinistra del grafico. Analizzando la densità empirica (l'istogramma) possiamo vedere che la distribuzione dei residui non si discosta eccessivamente da una distribuzione Gaussiana (linea nera), ad eccezione che per alcuni valori nella parte centrale e nella coda destra. Nel Q-Q plot è evidente che i residui seguono un andamento lineare per buona parte dei valori ad eccezione della coda superiore e inferiore, che si discosta dalla retta teorica. Usiamo il Test di Shapiro-Wilk per la verifica della normalità.

```
        Shapiro-Wilk normality test

data:  data.lm.r
W = 0.92128, p-value = 6.028e-14
```

Se il p-value è superiore a 0.05 si può supporre che i dati abbiano una distribuzione normale, cosa che non accade in questo caso. Anche i valori di skewness e kurtosi suggeriscono che la distribuzione dei residui non sia Gaussiana.

```
> #Skewness
> mean(((data.lm.r-mean(data.lm.r))/sd(data.lm.r))^3)
[1] 1.060999
> #kurtosi
> mean(((data.lm.r-mean(data.lm.r))/sd(data.lm.r))^4) - 3
[1] 10.2257
```

Un valore vicino a 1 di skewness indica una distribuzione asimmetrica dei valori, mentre un valore positivo di kurtosi indica una curva più "appuntita" di una normale. La mia ipotesi, dato che la distribuzione è centrata in zero e ricorda l'andamento di una Gaussiana, è che ci siano alcuni valori di residui eccessivamente elevati o inferiori rispetto alla media (come abbiamo anche visto nel grafico QQ-plot). Rimuovendo queste osservazioni il valore del p-value del test Shapiro-Wilk aumenta, superando la soglia di 0.05

```
        Shapiro-Wilk normality test

data:  data.lm.r
W = 0.99437, p-value = 0.1422
```

Purtroppo dal punto di vista statistico non è corretto rimuovere questi 11 outlier, poiché rappresentano una piccola parte del problema che non riusciamo a catturare. Questo significa che al nostro modello mancano dei fattori d'ingresso che riescano a spiegare queste poche osservazioni.

Modello di Regressione Logaritmico

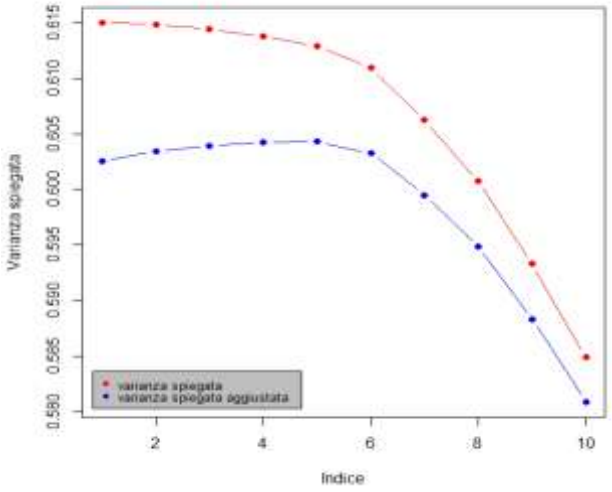
Costruiamo un modello di regressione non lineare per vedere se possiamo migliorare i risultati ottenuti con il modello lineare. Alcune osservazioni hanno un valore nullo, quindi prima di trasformare il modello sostituiamole con 0.001 per non avere valori infiniti nel modello logaritmico.

```
ldata = replace(ldata, ldata == 0, 0.001)
```

Riduzione del modello

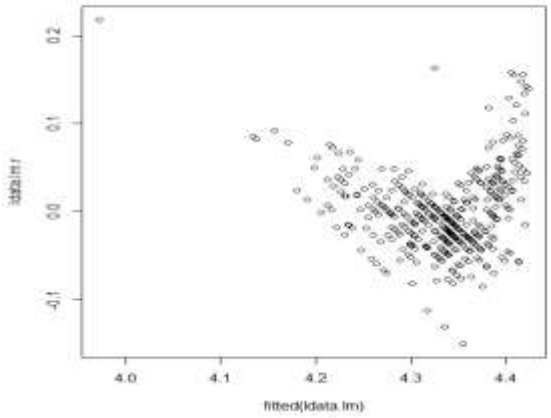
Seguendo lo stesso ragionamento procediamo con la riduzione del modello logaritmico.

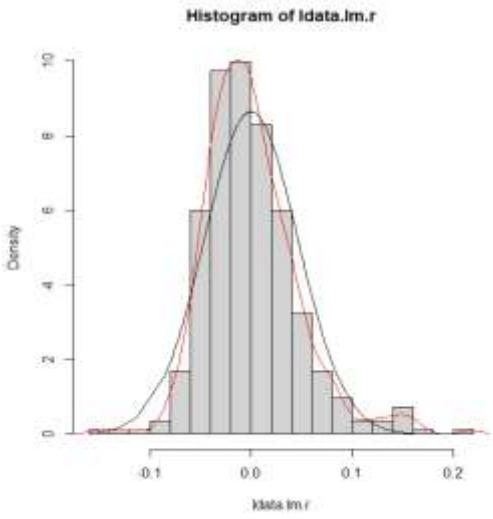
```
Call:
lm(formula = RANK ~ ., data = ldata)

Residuals:
      Min        1Q    Median        3Q       Max
-0.150692 -0.029925 -0.005311  0.023809  0.217856

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.979675   0.279677  17.805  < 2e-16 ***
MIN          0.075367   0.010296   7.320 1.33e-12 ***
FGA_PG       0.025171   0.008198   3.070 0.002281 **
FG3A        -0.005007   0.001352  -3.705 0.000241 ***
FTA          0.012300   0.003116   3.947 9.30e-05 ***
TOV          0.010187   0.004178   2.439 0.015170 *
STL         -0.014846   0.003069  -4.837 1.87e-06 ***
OFF_RATING  -0.090255   0.040812  -2.212 0.027556 *
DEF_RATING  -0.107503   0.041211  -2.609 0.009426 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.04659 on 407 degrees of freedom
Multiple R-squared:  0.6109,    Adjusted R-squared:  0.6033
F-statistic: 79.88 on 8 and 407 DF,  p-value: < 2.2e-16
```

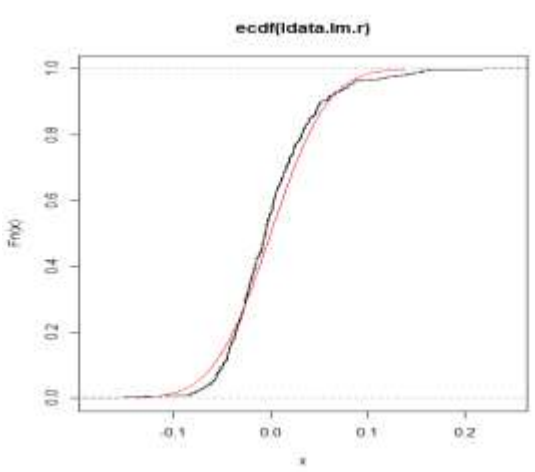


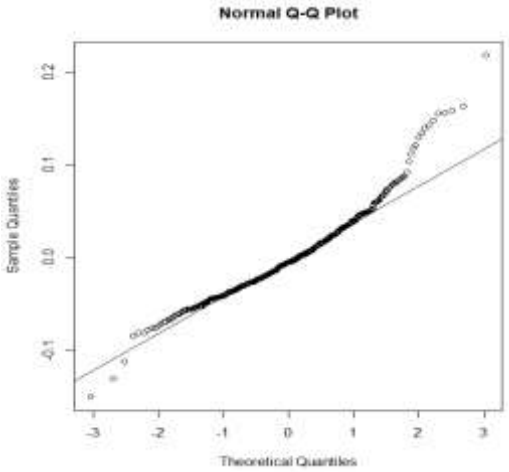
Dei dieci modelli ottenuti tramite riduzione, questa volta è stato selezionato il sesto. Rispetto al modello lineare, la proporzione di varianza spiegata e varianza spiegata aggiustata calano di più del 27%. La varianza spiegata in questo caso equivale al 61%, valore nettamente inferiore all’80%. Possiamo notare come alcune delle variabili più significative per il modello sono cambiate rispetto a prima.

Analisi dei residui









In questo caso, il grafico di dispersione dei valori residui e dei valori stimati evidenzia una relazione particolare tra i residui: la presenza di un pattern evidente nella distribuzione dei residui suggerisce che il modello logaritmico possa non essere il più adeguato all'analisi del fenomeno. Analizzando la densità empirica possiamo vedere che la distribuzione dei residui tende a discostarsi maggiormente da una distribuzione Gaussiana rispetto al modello precedente. Inoltre possiamo vedere come i residui seguono un andamento lineare per buona parte dei valori, ma si discostano dalla retta teorica nella parte superiore.

```
        Shapiro-Wilk normality test

data:  ldata.lm.r
W = 0.94751, p-value = 5.579e-11
```

Il p-value del test di Shapiro-Wilk è nettamente inferiore alla soglia di 0.05. Proviamo anche qui a rimuovere i residui estremi.

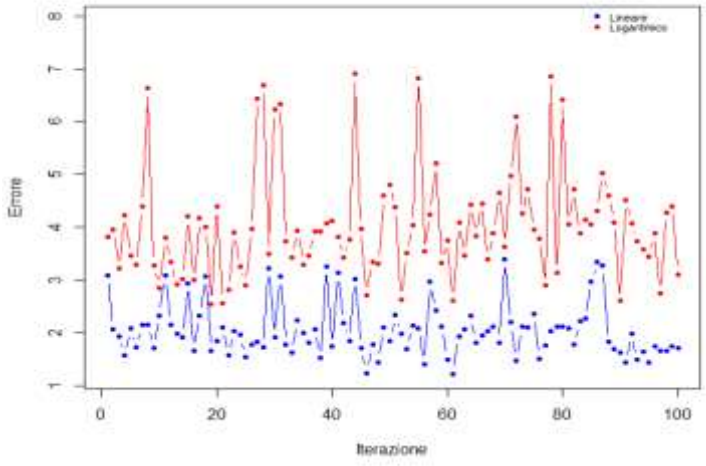
```
        Shapiro-Wilk normality test

data:  ldata.lm.r
W = 0.9879, p-value = 0.002099
```

Il test Shapiro Wilk restituisce un p-value nettamente migliorato, però non permette ancora di rigettare l'ipotesi nulla di normalità. L'analisi dei residui sul modello non lineare, dopo la rimozione degli outliers, suggerisce nuovamente che siamo comunque lontani dalla normalità dei residui.

Autovalutazione e previsione

Per concludere l'analisi ed esprimere un giudizio definitivo compariamo i due modelli in fase di previsione. Suddividiamo le osservazioni originali in un sottoinsieme di 40 osservazioni, il test set, e un sottoinsieme di 376 osservazioni, il training set. Questo procedimento è stato ripetuto per un totale di 100 iterazioni in modo da garantire una certa rilevanza statistica. I risultati delle 100 iterazioni sono riportati nel seguente grafico

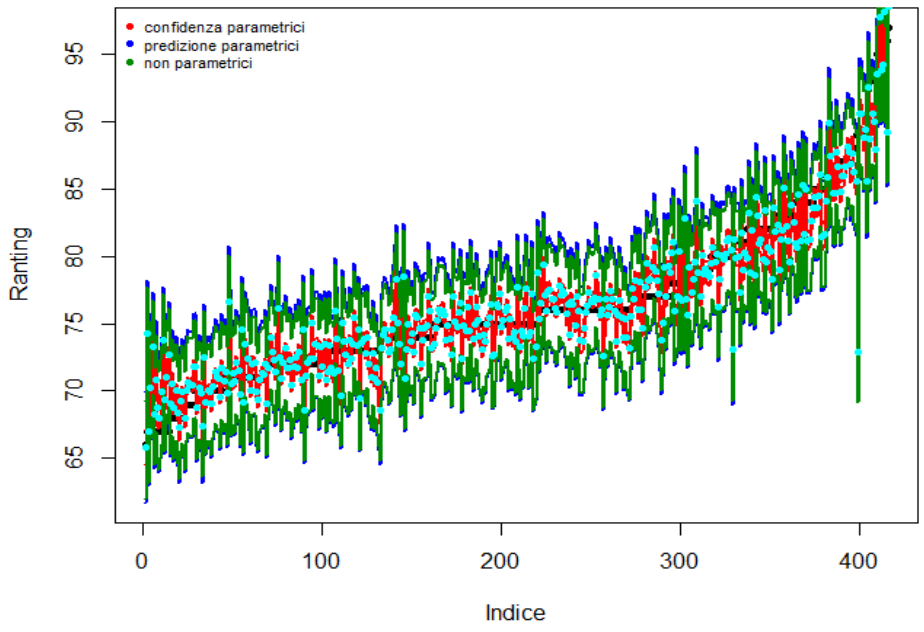
```
> #media errori
> mean(s_err)
[1] 2.057959
> mean(l_err)
[1] 4.041644
> #deviazione standard errori
> sd(s_err)
[1] 0.5069273
> sd(l_err)
[1] 1.025357
```



Possiamo dunque notare come il modello lineare abbia risultati migliori, con una media di errore ed una deviazione standard inferiore, cosa che lo rende più affidabile.

Stima delle incertezze

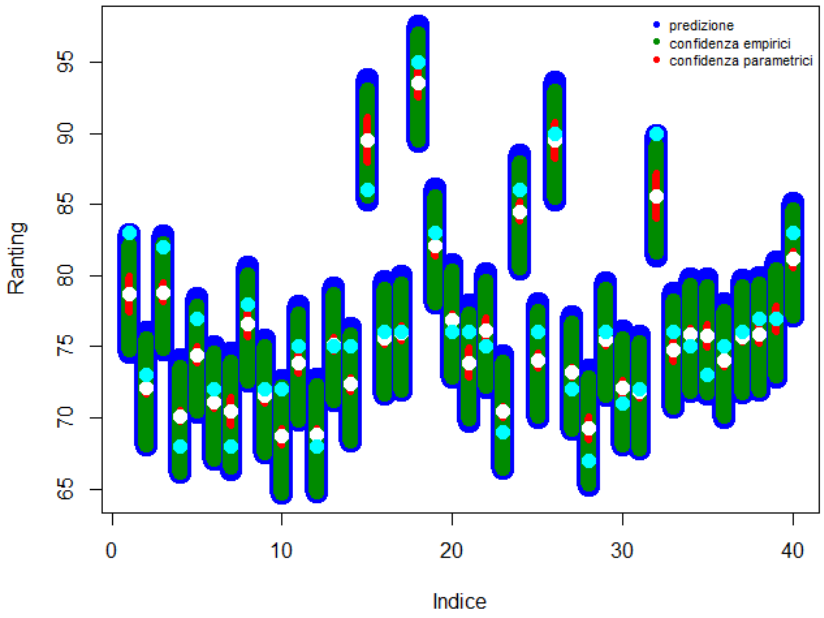
Per fornire un'analisi dell'incertezza della previsione, è stato necessario studiare le incertezze parametriche e non parametriche nella stima, e le incertezze nella previsione.
Come è possibile vedere dalla figura riportata, i punti celesti sono previsioni basate sul modello, ma nelle quali il valore osservato relativo alla previsione non è entrato nella stima del modello; mentre i punti neri sono i valori effettivi. A livello di limite inferiore e superiore i livelli di previsione parametrici e non parametrici seguono un andamento molto simile, con un lieve distaccamento più accentuato per il limite superiore.



Stima delle incertezze nella previsione

Ancora una volta, andando ad analizzare solo i nuovi dati, abbiamo effettuato una stima delle incertezze nella previsione. I pallini bianchi rappresentano le stime, i pallini celesti i valori osservati. I colori degli intervalli

sono gli stessi di prima: rosso intervallo di confidenza parametrico, blu intervallo di previsione e verde intervallo di confidenza empirico. Abbiamo una situazione analoga a quella precedente. Non è detto che l'osservazione cada nell'intervallo rosso, ma in quasi tutti i casi cade nell'intervallo blu (che contiene praticamente sempre l'intervallo verde).



Conclusioni

Dopo la conferma delle nostre previsioni, il modello lineare si è rivelato migliore in fase di predizione, con un errore medio inferiore ed un comportamento più affidabile rispetto a quello logaritmico. Per quanto riguarda il risultato, un errore medio di circa 2 unità per una variabile di uscita che varia tra 0 e 99 può essere considerato accettabile. Per concludere, un modello statistico di questo tipo, che permetta di predire il rating dei giocatori, è utile in quanto:

- il fattore di uscita permette alle aziende in ricerca di talenti di predire i giocatori con maggiore rating per offrire loro possibilità di sponsorship;
- la riduzione del modello permette alla 2K Sports di determinare quali siano i fattori che concorrono maggiormente all'aumento del rating di un giocatore, per lo sviluppo delle future versioni del videogioco;

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 "stat.csv".
Al dataframe ottenuto vengono aggiunti i rating dei giocatori presenti nel file "rankingNetto.csv" in base al campo "PLAYER_NAME".
Alcuni valori relativi ai nomi dei giocatori in "rankingNetto.csv" sono stati manipolati manualmente a causa della dicitura discorde tra i due file (Ex: OG Anunoby → O.G. Anunoby)
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']
# 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)
# Save dataframe in order to perform R analysis
df.to_csv('stat.csv', sep=";", index=False)
rating = pd.read_csv('rankingNetto.csv', sep=";")
stat = pd.read_csv('stat.csv', sep=";")
tabella = pd.merge(left=stat, right=rating, on='PLAYER_NAME')
# Save dataframe in order to perform R analysis
tabella.to_csv('tabella.csv', sep=";", index=False)
```

Codice Analisi

```
#LIBRERIE

library(corrplot)

library(readr)


#MANIPOLAZIONE TABELLE

#caricamento dei dati dai file csv

data1 = read.csv("tabella.csv", header = TRUE, sep = ";")


#rimuovo il nome del giocatore dalla tabella

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


# dati utilizzati per l'analisi

summary(data)

str(data)

plot(data)


# standardizzazione tabella

st_data = data.frame(scale(data))


# la standardizzazione non varia il grafico di dispersione

# e non è stata quindi considerata nel seguito dell'analisi

plot(st_data)


# grafico correlazioni

corrplot.mixed(cor(data), lower = "number", upper = "ellipse", number.cex=0.70,
tl.pos = "lt", tl.cex=0.70)
```

```
# MODELLO LINEARE

#modello lineare
data.lm = lm(RANK~., data=data)
summary(data.lm)

# riduzione del modello
r=matrix(ncol = 2, nrow = 7)
r[1,] = c(summary(data.lm)$r.squared, summary(data.lm)$adj.r.squared)

data.lm2 = lm(RANK~.-OFF_RATING, data=data)
r[2,] = c(summary(data.lm2)$r.squared, summary(data.lm2)$adj.r.squared)
summary(data.lm2)

data.lm3 = lm(RANK~.-OFF_RATING-FG3A, data=data)
r[3,] = c(summary(data.lm3)$r.squared, summary(data.lm3)$adj.r.squared)
summary(data.lm3)

data.lm4 = lm(RANK~.-OFF_RATING-FG3A-FTA, data=data)
r[4,] = c(summary(data.lm4)$r.squared, summary(data.lm4)$adj.r.squared)
summary(data.lm4)

data.lm5 = lm(RANK~.-OFF_RATING-FG3A-FTA-DEF_RATING, data=data)
r[5,] = c(summary(data.lm5)$r.squared, summary(data.lm5)$adj.r.squared)
summary(data.lm5)

data.lm6 = lm(RANK~.-OFF_RATING-FG3A-FTA-DEF_RATING-TOV, data=data)
r[6,] = c(summary(data.lm6)$r.squared, summary(data.lm6)$adj.r.squared)
summary(data.lm6)

data.lm7 = lm(RANK~.-OFF_RATING-FG3A-FTA-DEF_RATING-TOV-BLK, data=data)
r[7,] = c(summary(data.lm7)$r.squared, summary(data.lm7)$adj.r.squared)
summary(data.lm7)

ymin=min(r)
ymax=max(r)
```

```
plot(r[,1],pch=19,type="b",col="red",ylim=c(ymin,ymax), ylab="Varianza
spiegata", xlab = "Indice")

lines(r[,2],pch=19,type="b",col="blue")

legend("bottomleft",inset=0.02,c("varianza spiegata","varianza spiegata
aggiustata"),col=c("red","blue"),pch=c(19,19),bg="gray",cex=.8)


#versione ridotta (migliore versione 4)

data = data[c("RANK", "GP", "MIN", "PTS", "FGA_PG", "REB", "AST", "TOV", "STL",
"BLK", "DEF_RATING")]


#modello lineare

data.lm = lm(RANK~., data=data)

summary(data.lm)


#analisi dei residui

data.lm.r = residuals(data.lm)

plot(fitted(data.lm), data.lm.r)


#densità empirica

hist(data.lm.r, 20, freq=F)

lines(density(data.lm.r), col="red")

m = mean(data.lm.r)

s = sd(data.lm.r)

lines(sort(data.lm.r), dnorm(sort(data.lm.r),m,s))


#funzione cumulativa di probabilità

plot(ecdf(data.lm.r), pch=".")

y=seq(m-3*s,m+3*s,6*s/100)

lines(y,pnorm(y,m,s),col="red")


#distribuzione dei quantili

qqnorm(data.lm.r)

qqline(data.lm.r)


#Shapiro-Wilk

shapiro.test(data.lm.r)


#Skewness

mean(((data.lm.r-mean(data.lm.r))/sd(data.lm.r))^3)
```

```
#kurtosi
mean(((data.lm.r-mean(data.lm.r))/sd(data.lm.r))^4) - 3


# salvo il modello lineare per la predizione
linear_model = data


#rimuovo outlier
boxplot(data.lm.r, main="Boxplot residui iniziali",
col=(c("gold","darkgreen")), outcol="red")
Residuals_outliers <- boxplot(data.lm.r, plot=FALSE)$out
Residuals_outliers <- rev(sort(Residuals_outliers))
data = data[c(-which(data.lm.r %in%
Residuals_outliers[1:length(Residuals_outliers)])),]


data.lm = lm(RANK~., data=data)
summary(data.lm)


#residui
data.lm.r = residuals(data.lm)


#Shapiro-Wilk senza outlier
shapiro.test(data.lm.r)


#MODELLO LOGARITMICO


ldata = data1


#rimuovo il nome del giocatore dalla tabella
ldata = ldata[c("RANK", "GP", "MIN", "PTS", "FGA_PG", "FG3A", "FTA", "REB",
"AST", "TOV", "STL", "BLK", "OFF_RATING", "DEF_RATING")]


#metto 0.001 al posto di 0 per non avere infinito
ldata = replace(ldata, ldata == 0, 0.001)
ldata=log(ldata)
ldata.lm = lm(RANK~., data=ldata)
summary(ldata.lm)


#riduzione del modello
```

```
lr=matrix(ncol = 2, nrow = 10)
lr[1,] = c(summary(ldata.lm)$r.squared, summary(ldata.lm)$adj.r.squared)

ldata.lm2 = lm(RANK~.-REB, data=ldata)
lr[2,] = c(summary(ldata.lm2)$r.squared, summary(ldata.lm2)$adj.r.squared)
summary(ldata.lm2)

ldata.lm3 = lm(RANK~.-REB-GP, data=ldata)
lr[3,] = c(summary(ldata.lm3)$r.squared, summary(ldata.lm3)$adj.r.squared)
summary(ldata.lm3)

ldata.lm4 = lm(RANK~.-REB-GP-PTS, data=ldata)
lr[4,] = c(summary(ldata.lm4)$r.squared, summary(ldata.lm4)$adj.r.squared)
summary(ldata.lm4)

ldata.lm5 = lm(RANK~.-REB-GP-PTS-BLK, data=ldata)
lr[5,] = c(summary(ldata.lm5)$r.squared, summary(ldata.lm5)$adj.r.squared)
summary(ldata.lm5)

ldata.lm6 = lm(RANK~.-REB-GP-PTS-BLK-AST, data=ldata)
lr[6,] = c(summary(ldata.lm6)$r.squared, summary(ldata.lm6)$adj.r.squared)
summary(ldata.lm6)

ldata.lm7 = lm(RANK~.-REB-GP-PTS-BLK-AST-OFF_RATING, data=ldata)
lr[7,] = c(summary(ldata.lm7)$r.squared, summary(ldata.lm7)$adj.r.squared)
summary(ldata.lm7)

ldata.lm8 = lm(RANK~.-REB-GP-PTS-BLK-AST-OFF_RATING-FGA_PG, data=ldata)
lr[8,] = c(summary(ldata.lm8)$r.squared, summary(ldata.lm8)$adj.r.squared)
summary(ldata.lm8)

ldata.lm9 = lm(RANK~.-REB-GP-PTS-BLK-AST-OFF_RATING-FGA_PG-FG3A, data=ldata)
lr[9,] = c(summary(ldata.lm9)$r.squared, summary(ldata.lm9)$adj.r.squared)
summary(ldata.lm9)

ldata.lm10 = lm(RANK~.-REB-GP-PTS-BLK-AST-OFF_RATING-FGA_PG-FG3A-DEF_RATING,
data=ldata)
lr[10,] = c(summary(ldata.lm10)$r.squared, summary(ldata.lm10)$adj.r.squared)
```

```
summary(ldata.lm10)


ymin=min(lr)

ymax=max(lr)

plot(lr[,1],pch=19,type="b",col="red",ylim=c(ymin,ymax), ylab="Varianza
spiegata", xlab = "Indice")

lines(lr[,2],pch=19,type="b",col="blue")

legend("bottomleft",inset=0.02,c("varianza spiegata","varianza spiegata
aggiustata"),col=c("red","blue"),pch=c(19,19),bg="gray",cex=.8)


#modello ridotto (6)

ldata = ldata[c("RANK", "MIN", "FGA_PG", "FG3A", "FTA", "TOV", "STL",
"OFF_RATING", "DEF_RATING")]


#modello lineare

ldata.lm = lm(RANK~., data=ldata)

summary(ldata.lm)


#residui

ldata.lm.r = residuals(ldata.lm)

plot(fitted(ldata.lm), ldata.lm.r)


#densità empirica

hist(ldata.lm.r, 20, freq=F)

lines(density(ldata.lm.r), col="red")

m = mean(ldata.lm.r)

s = sd(ldata.lm.r)

lines(sort(ldata.lm.r), dnorm(sort(ldata.lm.r),m,s))


#funzione cumulativa di probabilità

plot(ecdf(ldata.lm.r), pch=".")

y=seq(m-3*s,m+3*s,6*s/100)

lines(y,pnorm(y,m,s),col="red")


#distribuzione dei quantili

qqnorm(ldata.lm.r)

qqline(ldata.lm.r)


#Shapiro-Wilk
```

```
shapiro.test(ldata.lm.r)


# salvo modello logaritmico per predizione
log_model = ldata


# elimino outlier
boxplot(ldata.lm.r, main="Boxplot residui iniziali",
col=(c("gold","darkgreen")), outcol="red")
Residuals_outliers <- boxplot(ldata.lm.r, plot=FALSE)$out
Residuals_outliers <- rev(sort(Residuals_outliers))
ldata = ldata[c(-which(ldata.lm.r %in%
Residuals_outliers[1:length(Residuals_outliers)])),]


ldata.lm = lm(RANK~., data=ldata)
summary(ldata.lm)
ldata.lm.r = residuals(ldata.lm)


#Shapiro-Wilk senza outlier
shapiro.test(ldata.lm.r)


#VALIDAZIONE


data=linear_model
ldata=log_model
l=nrow(data)


m=100 # numero iterazioni
s_err=rep(0,m)
l_err=rep(0,m)


for(j in 1:m){
  # selezione del test set (uguale nelle due stime per aiutare il confronto)
  idx = sample(l,40)
  # modello lineare
  strain = data[-idx,]
  stest = data[idx,]
  st.lm=lm(RANK~.,data=strain)
  st.p=predict(st.lm,stest)
  # modello nonlineare
```

```
  ltrain = ldata[-idx,]
  ltest = ldata[idx,]
  lt.lm=lm(RANK~.,data=ltrain)
  lt.p=predict(lt.lm,ltest)
  # errore medio
  s_err[j] = sqrt(mean((st.p - stest$RANK)^2))
  l_err[j] = sqrt(mean((exp(lt.p) - exp(ltest$RANK))^2))
}

#media errori
mean(s_err)
mean(l_err)

#deviazione standard errori
sd(s_err)
sd(l_err)

# rappresentazione grafica errori dei due modelli
par(mfrow=c(1,1))
gmin=min(s_err,l_err)
gmax=max(s_err,l_err)
plot(s_err,type="b", pch=20, col="blue", ylim=c(gmin, gmax+1),
     ylab="Errore", xlab="Iterazione")
points(l_err,type="b", pch=20, col="red")
legend("topright", inset = c(0, 0), c("Lineare", "Logaritmico"),
       col = c("blue", "red"), pch = c(19,19), cex=0.7, bty="n")

#STIMA DELLE INCERTEZZE
# previsione modello lineare
data<-data[order(data$RANK),]
data.lm=lm(RANK~.,data=data)
alpha=0.95

#utilizzo ogni dato come nuovo dato per la stima degli intervalli
n=nrow(data)
data.cil=matrix(rep(0,3*n),n,3) # per gli intervalli di confidenza parametrici
data.pil=matrix(rep(0,3*n),n,3) # per gli intervalli di previsione parametrici
data.nil=matrix(rep(0,3*n),n,3) # per gli intervalli non parametrici
```

```
for(i in 1:n){
  tr=data[-i,]
  nd=data[i,]
  alm=lm(RANK~.,data=tr)
  data.cil[i,]=predict(alm,nd,interval="confidence",level=alpha)
  data.pil[i,]=predict(alm,nd,interval="prediction",level=alpha)
  data.nil[i,]=predict(alm,nd)+c(0,quantile(resid(alm),(1-
alpha)/2),quantile(resid(alm),(1+alpha)/2))
}


ymin=min(data.pil)
ymax=max(data$RANK)
plot(data$RANK,pch=20,ylim=c(ymin,ymax),ylab = "Ranting",xlab = "Indice") #
valori veri
# intervalli di confidenza parametrici
lines(data.cil[,2],col="red",lwd=2) # stima inferiore
lines(data.cil[,3],col="red",lwd=2) # stima superiore
# intervalli di predizione parametrici
lines(data.pil[,2],col="blue",lwd=2) # stima inferiore
lines(data.pil[,3],col="blue",lwd=2) # stima superiore
# intervalli non parametrici
lines(data.nil[,2],col="green4",lwd=2)
lines(data.nil[,3],col="green4",lwd=2)
points(data.nil[,1],col="cyan", pch=20)
legend("topleft", inset = c(0, 0), c("confidenza parametrici", "predizione
parametrici", "non parametrici"),
       col = c("red", "blue", "green4"), pch = c(19,19), cex=0.7, bty="n")


#stima delle incertezze nella previsione
alpha=0.95
u=sample(416,40)
dataa=data[-u,]
datab=data[u,]
dataa.lm=lm(RANK~.,data=dataa)


#ricavo intervalli
# intervalli di confidenza
datab.ci=predict(dataa.lm,datab,interval="confidence",level=alpha)
# intervalli di predizione
```

```
datab.pi=predict(dataa.lm,datab,interval="prediction",level=alpha)
# intervalli di confidenza empirici
dataa.r=resid(dataa.lm)
qi=quantile(dataa.r,(1-alpha)/2)
qs=quantile(dataa.r,(1+alpha)/2)

#rappresento graficamente il risultato
##
ymin=min(c(datab.pi[,2],datab.ci[,2]))
ymax=max(c(datab.pi[,3],datab.ci[,3]))
plot(datab$RANK,ylim=c(ymin,ymax),ylab = "Ranting",xlab = "Indice")
x=1:40
###
# intervalli di predizione
segments(x,datab.pi[,2],x,datab.pi[,3],col="blue",lwd=18)
# intervalli di confidenza empirici
segments(x,datab.pi[,1]+qi,x,datab.pi[,1]+qs,col="green4",lwd=12)
# intervalli di confidenza parametrici
segments(x,datab.ci[,2],x,datab.ci[,3],col="red",lwd=6)
# valori stimati
points(x,datab.pi[,1],pch=19,col="white",cex=1.5)
points(datab$RANK,pch=19,col="cyan",cex=1.5)
legend("topright", inset = c(0, 0), c("predizione", "confidenza empirici",
"confidenza parametrici"),
       col = c("blue", "green4", "red"), pch = c(19,19), cex=0.7, bty="n")
```