Progetto Metodologie Statistiche per i Big Data

Analisi dataset: Mobile Price Classification

Questo dataset è stato creato da una compagnia di telefonia mobile raccogliendo i dati di vendita dei cellulari di varie aziende al fine di poter classificare le fasce di prezzo dei telefoni basandosi sulle loro caratteristiche e poter quindi scegliere la fascia di mercato a cui proporre i prodotti da loro creati.

Iniziamo importando il nostro dataset in R e, dopo aver fatto le prime analisi, vediamo che:

- Sono presenti 2000 osservazioni a fronte di 21 variabili;
- Delle 21 variabili presenti 14 sono numeriche mentre le altre 7 qualitative;
- Non sono presenti valori duplicati ne valori nulli;

Legenda delle variabili:

battery_power = potenza della batteria (mAh)
blue = ha o meno il bluetooth
clock_speed = velocità con la quale il microprocessore esegue le istruzioni (GHz)
dual_sim = presenza o meno della doppia sim
fc = camera frontale (Mega Pixels)
four_g = presenza o meno del 4G
int_memory = memoria interna (Gigabytes)
m_dep = spessore del telefono (cm)
mobile_wt = peso del telefono (grammi)
n_cores = numero dei core del processore
pc = camera principale (Mega Pixels)
px_height = risoluzione pixel dello schermo in altezza
px_width = risoluzione pixel dello schermo in larghezza
ram = RAM (Megabytes)
sc_h = altezza schermo (cm)
sc_w = spessore schermo (cm)
talk_time = durata di una carica di batteria se si è in chiamata (h)
three_g = presenza o meno del 3G
touch_screen = presenza o meno del touch screen
wifi = presenza o meno del modulo wi-fi
price_range = fascia di prezzo (0-1-2-3)

Procediamo trasformando le variabili qualitative in fattori per poi fare un plot delle correlazioni (FIG 1) tra le variabili numeriche.

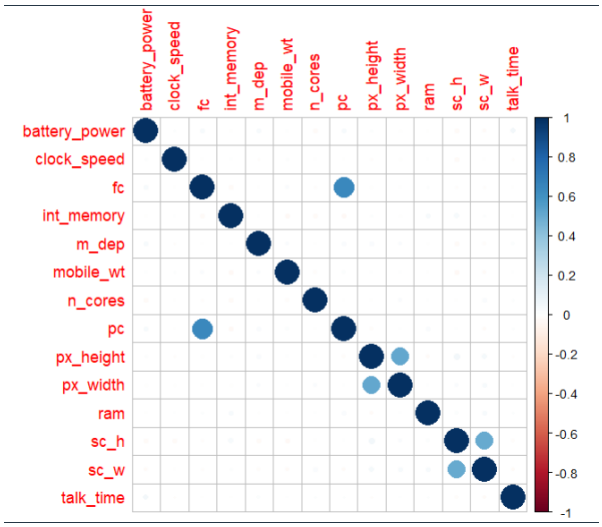
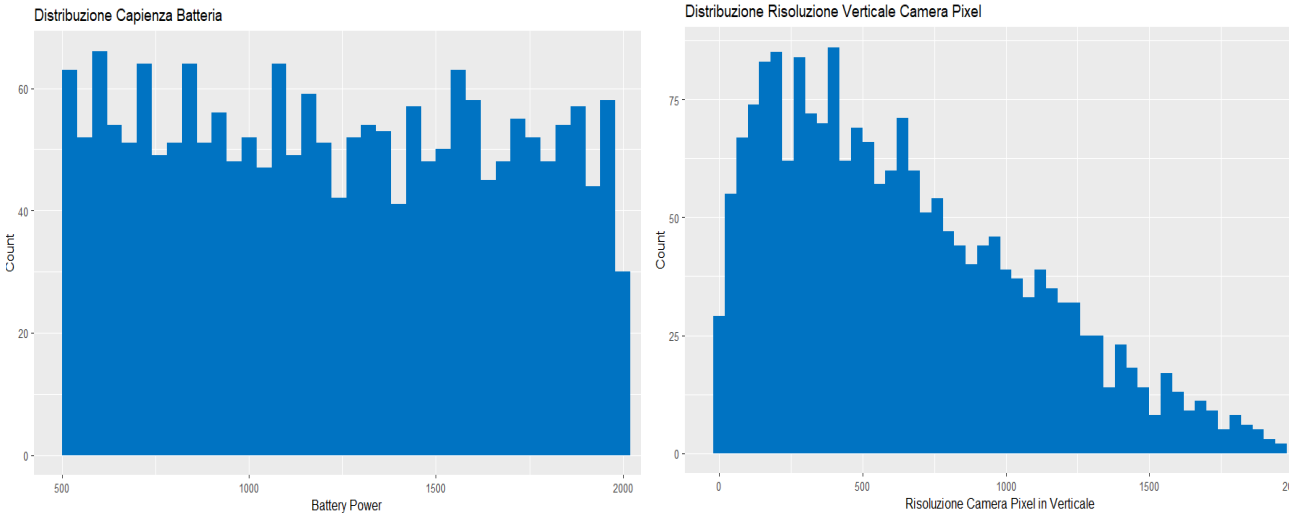


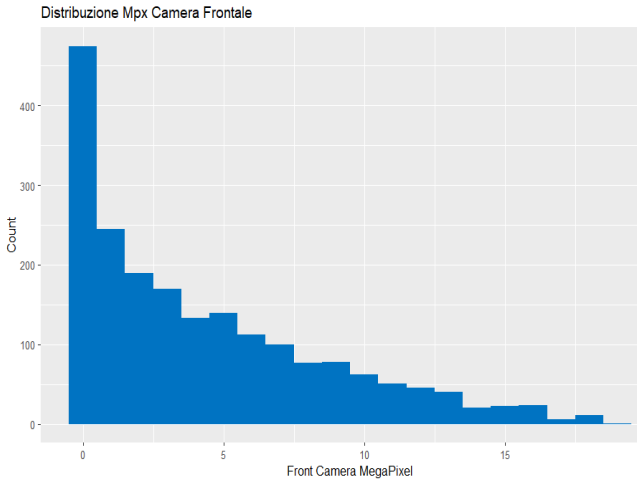
FIG. 1

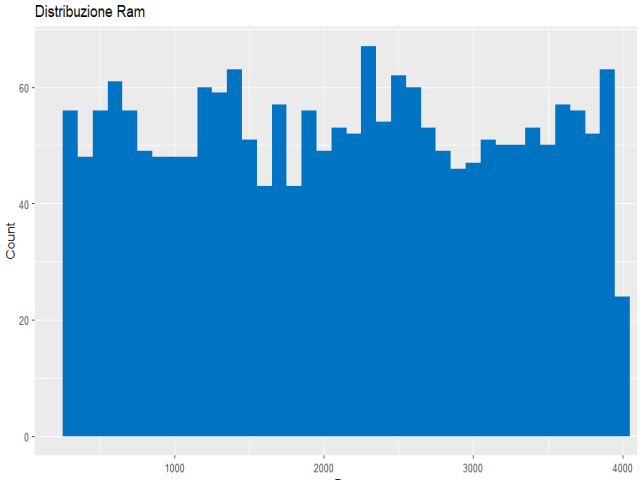
vediamo correlazione tra:
fotocamera interna - fotocamera principale (0.64)
risoluzione pixel display in altezza - risoluzione pixel display in larghezza (0.51)
altezza schermo - spessore schermo (0.51)

Il nostro lavoro prosegue eseguendo i plot delle distribuzioni di tutte le variabili al fine di capire meglio come le osservazioni si distribuiscono all'interno del dataset, qua sotto (FIG.2) ne sono presenti alcuni per aiutare anche il lettore a comprendere.

FIG. 2







Procediamo adesso standardizzando le variabili in nostro possesso al fine di ottenere dei risultati meno influenzati dalla grandezza delle variabili in gioco. La standardizzazione ci aiuta anche per attenuare l'asimmetria di qualche variabile. Standardizziamo sottraendo ad ogni variabile la propria media e dividendo per la deviazione standard.

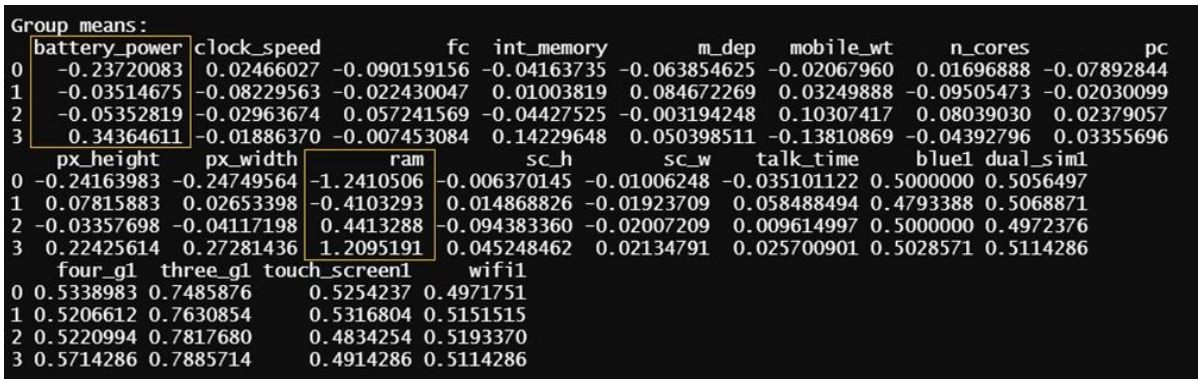
Continuiamo dividendo il nostro dataset in due parti, una, che chiameremo "train_data" dove appunto traineremo, alleneremo, i nostri modelli, l'altra, "test_data", dove proveremo la forza dei nostri modelli comparandoli con dati sui quali non si sono allenati.
Confrontando i test e il train vediamo che essi si distribuiscono, perlomeno sulla variabile che ci interessa (Price_Range) in maniera omogenea (FIG. 4).

```
> set.seed(123)
> sample_data<- sample.split(data,SplitRatio = 0.75)
> sample_data
 [1]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
[17]  TRUE  TRUE  TRUE FALSE FALSE
> train_data<-subset(data, sample_data==TRUE)
> dim(train_data)
[1] 1429   21
> test_data<-subset(data, sample_data==FALSE)
> dim(test_data)
[1] 571  21
> prop.table(table(train_data$price_range))

        0         1         2         3
0.2477257 0.2540238 0.2533240 0.2449265
> prop.table(table(test_data$price_range))

        0         1         2         3
0.2556918 0.2399299 0.2416813 0.2626970
```

FIG. 4

Analisi Discriminante Lineare

Il primo strumento utilizzato è quello dell'analisi discriminante (LDA). Dopo aver inserito le frequenze a priori utilizziamo l'analisi discriminante sul training set.
L'analisi discriminante proietta le osservazioni su una nuova dimensione in modo tale da massimizzare la distanza tra i centroidi delle classi e minimizzare la distanza delle osservazioni che appartengono alla stessa classe. Dalla FIG.5 possiamo osservare che tra le nostre variabili le uniche che hanno delle medie ben separate tra le quattro classi sono la batteria e la RAM.



Group means:

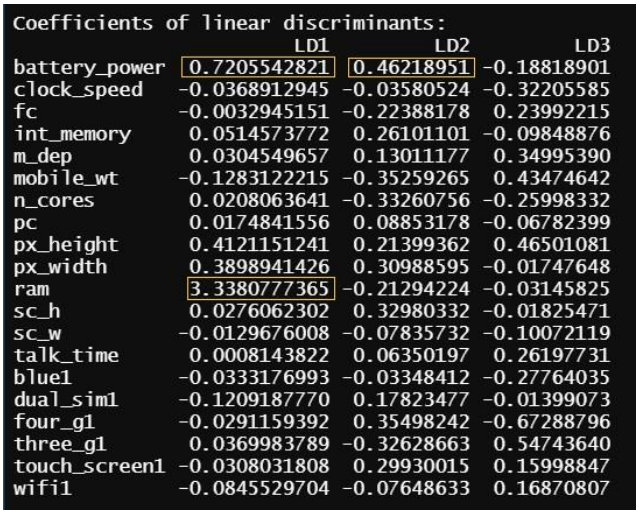
	battery_power	clock_speed	fc	int_memory	m_dep	mobile_wt	n_cores	pc
0	-0.23720083	0.02466027	-0.090159156	-0.04163735	-0.063854625	-0.02067960	0.01696888	-0.07892844
1	-0.03514675	-0.08229563	-0.022430047	0.01003819	0.084672269	0.03249888	-0.09505473	-0.02030099
2	-0.05352819	-0.02963674	0.057241569	-0.04427525	-0.003194248	0.10307417	0.08039030	0.02379057
3	0.34364611	-0.01886370	-0.007453084	0.14229648	0.050398511	-0.13810869	-0.04392796	0.03355696

	px_height	px_width	ram	sc_h	sc_w	talk_time	blue1	dual_sim1
0	-0.24163983	-0.24749564	-1.2410506	-0.006370145	-0.01006248	-0.035101122	0.5000000	0.5056497
1	0.07815883	0.02653398	-0.4103293	0.014868826	-0.01923709	0.058488494	0.4793388	0.5068871
2	-0.03357698	-0.04117198	0.4413288	-0.094383360	-0.02007209	0.009614997	0.5000000	0.4972376
3	0.22425614	0.27281436	1.2095191	0.045248462	0.02134791	0.025700901	0.5028571	0.5114286

	four_g1	three_g1	touch_screen1	wifi1
0	0.5338983	0.7485876	0.5254237	0.4971751
1	0.5206612	0.7630854	0.5316804	0.5151515
2	0.5220994	0.7817680	0.4834254	0.5193370
3	0.5714286	0.7885714	0.4914286	0.5114286

FIG. 5

Quelli in FIG. 6 sono i coefficienti che trasformano le osservazioni nelle funzioni discriminanti. Come possiamo osservare la battery power e soprattutto la RAM hanno un peso decisamente superiore alle altre variabili nella "costruzione" delle nuove dimensioni.



Coefficients of linear discriminants:

	LD1	LD2	LD3
battery_power	0.7205542821	0.46218951	-0.18818901
clock_speed	-0.0368912945	-0.03580524	-0.32205585
fc	-0.0032945151	-0.22388178	0.23992215
int_memory	0.0514573772	0.26101101	-0.09848876
m_dep	0.0304549657	0.13011177	0.34995390
mobile_wt	-0.1283122215	-0.35259265	0.43474642
n_cores	0.0208063641	-0.33260756	-0.25998332
pc	0.0174841556	0.08853178	-0.06782399
px_height	0.4121151241	0.21399362	0.46501081
px_width	0.3898941426	0.30988595	-0.01747648
ram	3.3380777365	-0.21294224	-0.03145825
sc_h	0.0276062302	0.32980332	-0.01825471
sc_w	-0.0129676008	-0.07835732	-0.10072119
talk_time	0.0008143822	0.06350197	0.26197731
blue1	-0.0333176993	-0.03348412	-0.27764035
dual_sim1	-0.1209187770	0.17823477	-0.01399073
four_g1	-0.0291159392	0.35498242	-0.67288796
three_g1	0.0369983789	-0.32628663	0.54743640
touch_screen1	-0.0308031808	0.29930015	0.15998847
wifi1	-0.0845529704	-0.07648633	0.16870807

FIG. 6

Dalla FIG.7 è evidente che la dimensione 1 è nettamente migliore rispetto alle altre 2 nel riuscire a separare le osservazioni tra le classi di appartenenza, vediamo infatti che fa il 99.62% del lavoro (FIG.8).

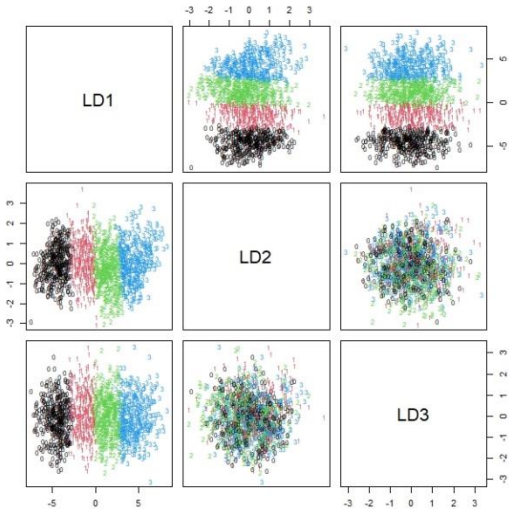


FIG. 7

```
Proportion of trace:
   LD1    LD2    LD3
0.9962 0.0028 0.0011
```

FIG. 8

Applicando i risultati al test set otteniamo degli ottimi risultati. L'accuracy è del 95.44% circa, con solo 26 osservazioni classificate male.

Albero Decisionale

Il nostro lavoro prosegue con il Decision Tree, questo algoritmo ci permette, dopo aver scelto il nostro Complexity Parameter per avere un albero quanto più parsimonioso possibile, di strutturare il suddetto albero (FIG. 9).

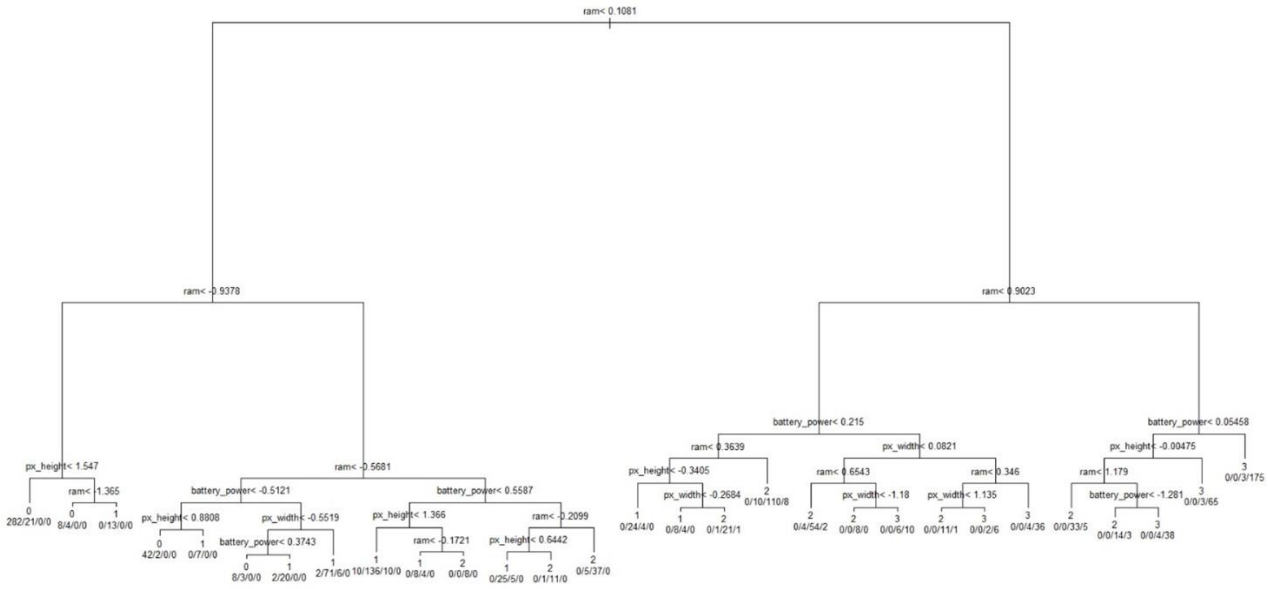


FIG. 9

Analizzando il risultato vediamo come per la creazione dell'albero sono state selezionate solamente quattro variabili: Battery_Power, Px_height, Px_Width e RAM.
Possiamo osservare come per i primi split sia stata usata solamente la RAM il che la rende probabilmente il componente più costoso e che meglio descrive la differenza tra le fasce di prezzo dei cellulari.

L’accuracy del nostro modello è del 88.92%, inferiore a quella dell’analisi discriminante ma comunque molto elevata.

Random Forest

Il prossimo algoritmo utilizzato è quello della Random Forest, questo algoritmo ci permette di prendere diversi alberi decisionali incorrelati (in questo caso 500) e, prendendo le previsioni di ognuno di essi, di ottenere una previsione complessiva. Il grafico (FIG.10) mostra le varie curve ottenute facendo previsioni sulle singole classi e ci mostra come il sistema scelga l’errore più “appropriato” facendo una media tra queste.

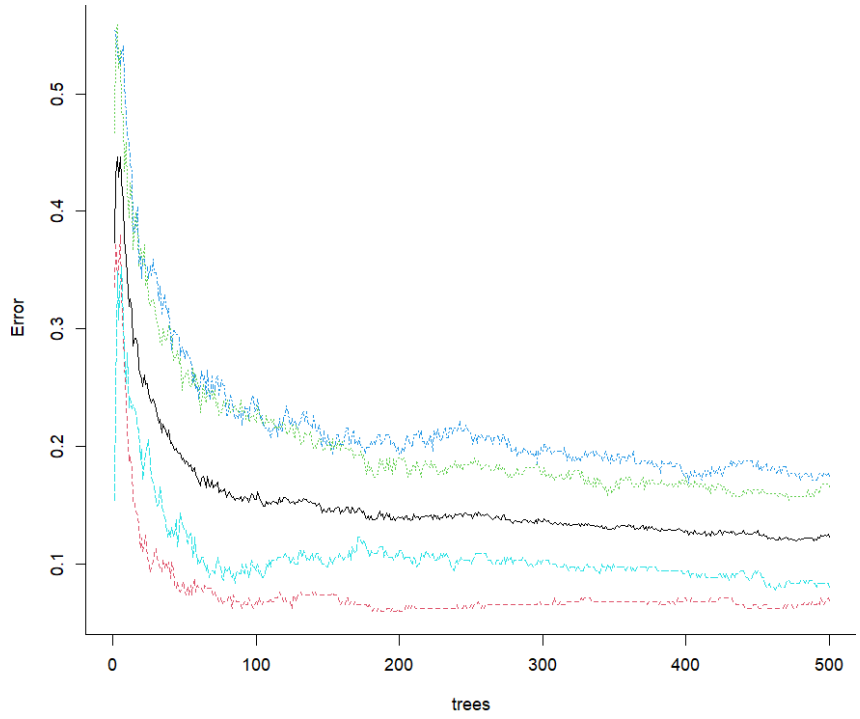


FIG. 10

Ovviamente con questo algoritmo non abbiamo la possibilità di plottare un grafico come quello degli alberi decisionali proprio perché di alberi ne vengono usati 500; tuttavia, possiamo vedere quali sono le variabili che più hanno influito nella predizione finale e ancora una volta osserviamo come la RAM sia al primo posto (FIG.11).

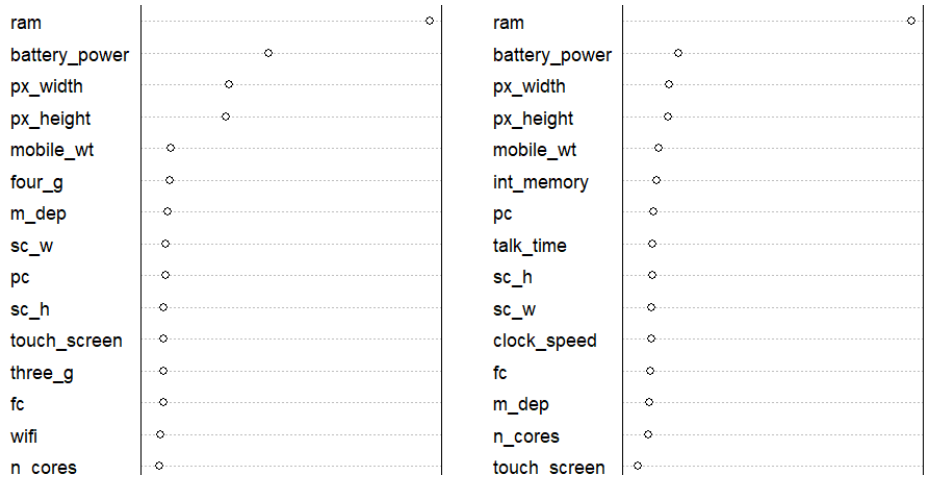


FIG. 11

A seguito di una confusion matrix notiamo come, ancora una volta nel caso di Supervised Machine Learning l'accuracy si attesti su un 89,84%.

Nearest Neighbor (KNN)

Decidiamo quindi di provare il comportamento del Nearest Neighbor, questo algoritmo si basa sulle distanze tra le varie osservazioni del dataset assegnando ad ogni nuova osservazione testata una classe scelta sulla base appunto delle distanze dalle osservazioni del train_data. Settiamo il parametro K=20, quindi prendiamo come riferimento le 20 osservazioni più vicine a quella da testare, come risultato abbiamo un'accuracy del 89.84% (FIG. 12)

```
> cl = as.factor(train_data$price_range)
> pr_knn2 = knn(train_data, test_data, cl , k = 20)
> tab4.2=xtabs(~pr_knn2+test_data$price_range)
> valori_beccati4.2 = diag(tab4.2)
> accuracy_knn2 = sum(valori_beccati4.2)/length(test_data$price_range)
> tab4.2
       test_data$price_range
pr_knn2   0   1   2   3
      0 130   4   0   0
      1  16 122   5   0
      2   0  11 124  13
      3   0   0   9 137
> accuracy_knn2
[1] 0.8984238
```

FIG. 12

Non eravamo tuttavia soddisfatti di questo risultato poiché sappiamo la potenza di questo algoritmo e immaginavamo che i risultati potessero essere migliori, abbiamo dunque aumentato i K passando a 250. Questo, specie nei problemi di classificazione, potrebbe portare a degli errori portando il sistema a classificare l'osservazione nel gruppo più numeroso. Fortunatamente la struttura del nostro dataset (come possibile vedere dalla FIG.4) fa si che la variabile Price_range sia equamente distribuita con il 25% circa di osservazioni per classe portandoci ad alzare il K con meno paura di rischiare una classificazione sbagliata. Vediamo infine come, all'aumentare dei K, aumenti significativamente anche l'accuratezza (FIG. 13).

```
> cl = as.factor(train_data$price_range)
> pr_knn2 = knn(train_data, test_data, cl , k = 250)
> tab4.2=xtabs(~pr_knn2+test_data$price_range)
> valori_beccati4.2 = diag(tab4.2)
> accuracy_knn2 = sum(valori_beccati4.2)/length(test_data$price_range)
> tab4.2
       test_data$price_range
pr_knn2   0   1   2   3
      0 144   5   0   0
      1   2 125   6   0
      2   0   7 132  18
      3   0   0   0 132
> accuracy_knn2
[1] 0.9334501
```

FIG. 13

È ormai lapalissiano che gli algoritmi supervisionati funzionano egregiamente e, dopo aver rivisto i precedenti risultati siamo arrivati alla conclusione che qualche variabile stesse classificando da sola i risultati. Plottando le variabili usate dal **decision tree** (Battery_Power, Px_height, Px_Width e RAM) e dalla **random forest** con la variabile che stavamo studiando, ovvero il price range, ci siamo accorti di come la RAM stesse semplificando gran parte delle classificazioni poiché facendo girare gli stessi algoritmi escludendo questa variabile l'accuracy crollava vertiginosamente.

Principal Component Analysis (PCA)

Dopo aver effettuato le predizioni con tecniche di machine learning supervisionato, vediamo se utilizzando quelle non supervisionate siamo comunque in grado di ottenere dei buoni risultati. La variabile che discrimina nel miglior modo le classi come abbiamo potuto vedere è la RAM, proviamo a costruire le componenti principali e vediamo quanto la RAM influisce sulla costruzione di queste.

```
Importance of components:
                          PC1      PC2      PC3      PC4      PC5      PC6     PC7     PC8
Standard deviation     1084.7779 538.4416 439.5203 305.99036 35.40763 18.12040 6.84271 5.46913
Proportion of Variance    0.6704   0.1652   0.1101   0.05335  0.00071  0.00019 0.00003 0.00002
Cumulative Proportion     0.6704   0.8356   0.9457   0.99903  0.99974  0.99993 0.99995 0.99997
                           PC9    PC10   PC11   PC12   PC13   PC14
Standard deviation     5.22756 3.02570 2.914 2.281 0.8153 0.2877
Proportion of Variance 0.00002 0.00001 0.000 0.000 0.0000 0.0000
Cumulative Proportion  0.99999 0.99999 1.000 1.000 1.0000 1.0000
```

FIG. 14

Nella FIG. 14 notiamo che le prime 4 componenti principali catturano cumulativamente il 99.903% della varianza totale del dataset.

```
> pcav$rotation
                       PC1          PC2          PC3          PC4
battery_power  3.471979e-04 -1.038840e-02  9.994425e-01 -3.171326e-02
clock_speed   -2.630029e-06  2.087359e-05  2.144006e-05 -1.396166e-05
fc            -6.055997e-05  6.733209e-05  3.293617e-04 -8.234845e-05
int_memory    -5.482375e-04 -6.847613e-05 -1.472883e-04  1.190878e-03
m_dep          2.534142e-06 -1.515199e-05  2.212233e-05 -6.542344e-07
mobile_wt      8.443485e-05 -3.911582e-05  1.500664e-04  8.096725e-05
n_cores       -1.035355e-05 -3.924499e-05 -1.588837e-04 -2.275485e-04
pc            -1.624202e-04  9.088395e-05  4.286954e-04 -4.638556e-04
px_height      9.903074e-03 -7.250362e-01  1.430758e-02  6.884899e-01
px_width      -9.647755e-04 -6.886014e-01 -3.014695e-02 -7.245119e-01
ram           -9.999503e-01 -6.519582e-03  5.178224e-04  7.505923e-03
sc_h          -6.123714e-05 -3.718628e-04 -2.846868e-04  5.401018e-04
sc_w          -1.422252e-04 -3.656035e-04 -2.157749e-04  1.337468e-04
talk_time     -5.468870e-05  2.090449e-05  6.481607e-04 -3.502860e-04
```

FIG. 15

In tutte le tecniche di machine learning supervisionato abbiamo visto che oltre alla RAM le uniche variabili che hanno giocato un ruolo importante nella predizione sono state “Battery_Power”,” Px_height”, e “Px_Width” (FIG. 6, 9, 11). Con gli autovettori delle prime quattro componenti principali abbiamo la controprova del fatto che sono le uniche variabili utili. La RAM ha in coefficiente più grande nel primo autovettore, mentre “Battery_Power”,” Px_height”, e “Px_Width” hanno i coefficienti più grandi nelle successivi tre.

Quanto sono in grado di separare le classi?

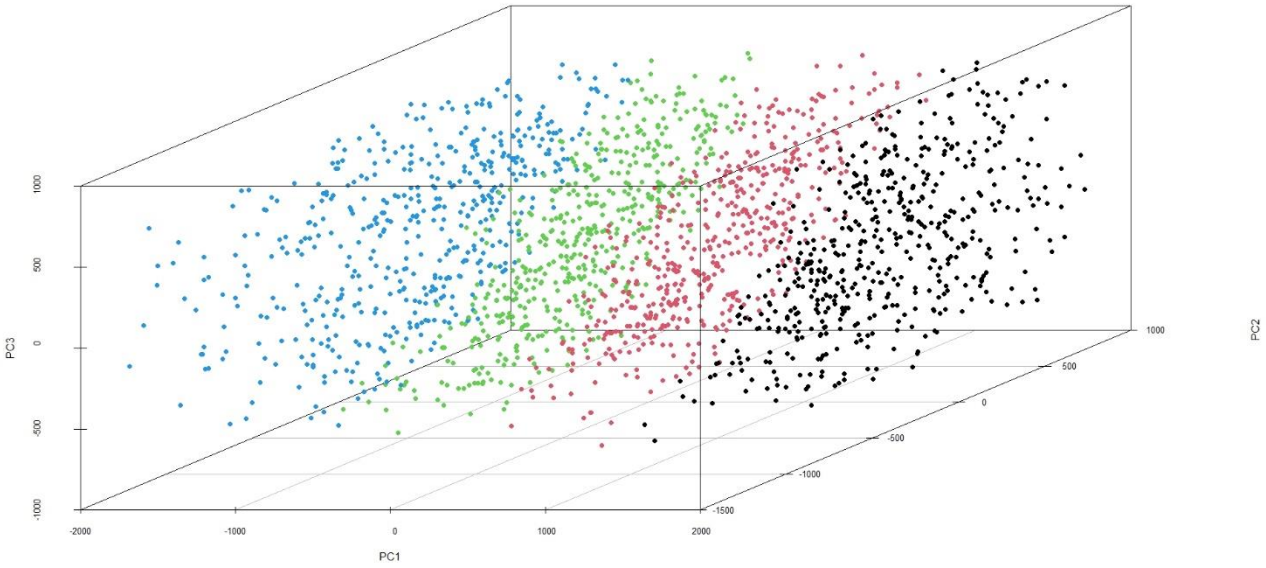


FIG. 16

Le prime 3 componenti principali sembrerebbero ordinare bene le classi, quindi vediamo come si comporta un algoritmo k-means nella classificazione delle osservazioni.

K-MEANS

Di seguito effettuiamo il k-means utilizzando prima le quattro componenti principali e poi solo la RAM per vedere in quale delle due situazioni le classificazioni sono più precise.

K-MEANS CON COMPONENTI PRINCIPALI

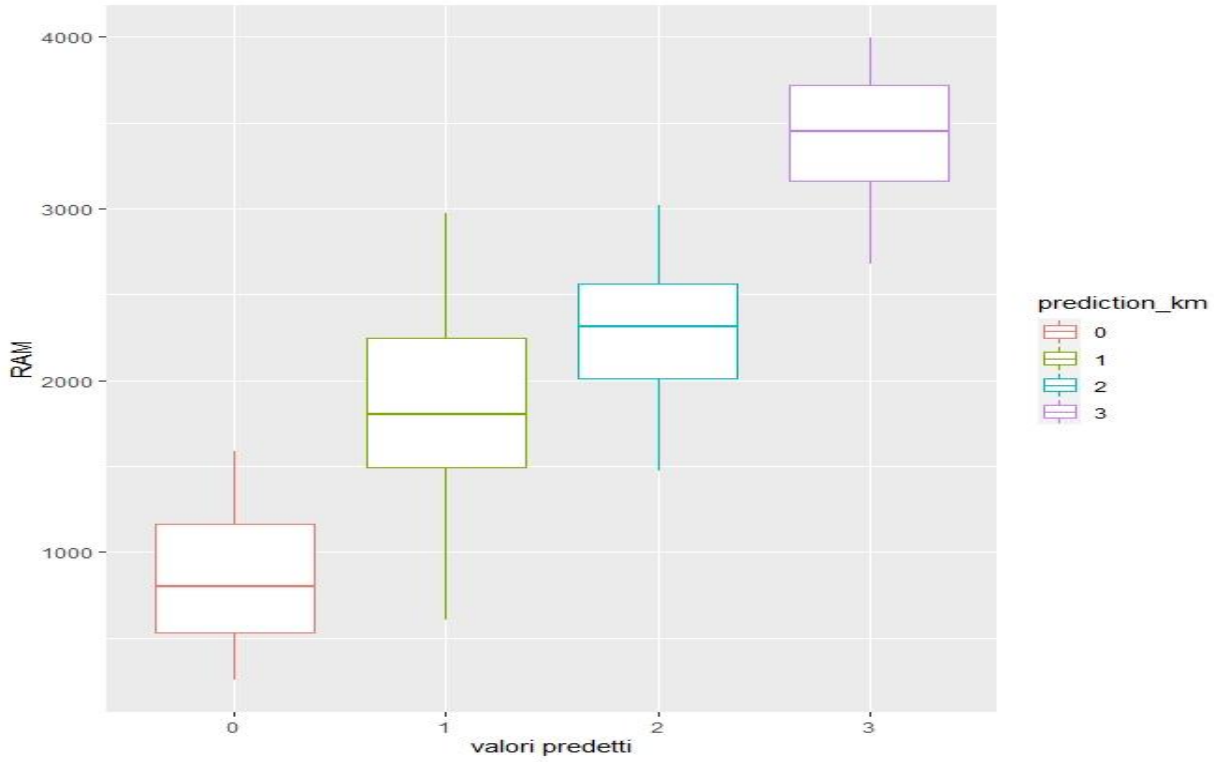


FIG. 17

```
          predicted
observed   0   1   2   3
       0 477   8  15   0
       1 125 156 219   0
       2   0 131 230 139
       3   0  31   8 461
> accuracy_km_pc
[1] 0.662
```

FIG. 18

Come possiamo vedere dalla FIG. 17 i boxplot della RAM rispetto alle classi predette si sovrappongono troppo, questo ci fa intuire che soprattutto le classi 1 e 2 non saranno ben classificate. La predizione con le componenti principali ha un accuracy del 66,2%.

K-MEANS CON RAM

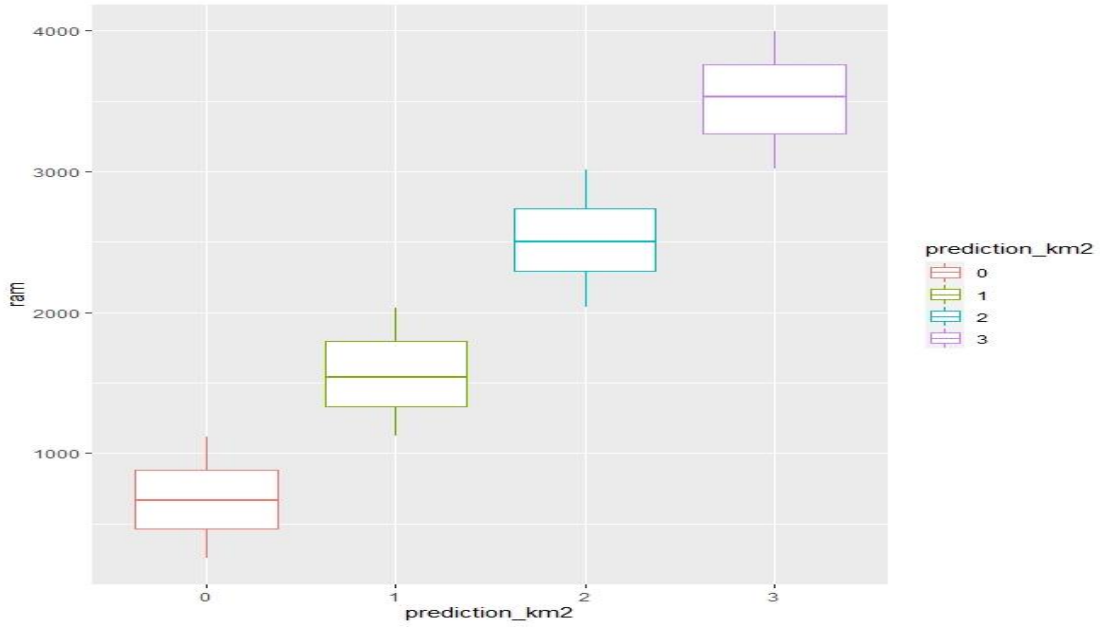


FIG. 19

```
          predicted
observed   0   1   2   3
       0 403  97   0   0
       1  54 321 125   0
       2   0  65 347  88
       3   0   0  69 431

> accuracy_km_ram
[1] 0.751
```

FIG. 20

Nella FIG. 19 i boxplot sono nettamente più divisi, ciò è dovuto dal fatto che il k-means è stato effettuato utilizzando solo la RAM e come possiamo vedere dalla FIG.20 sorprendentemente l'algoritmo riesce comunque a classificare meglio con la RAM piuttosto che con tutte e 4 le componenti principali. La precisione di classificazione ha raggiunto in questo caso il 75,1 %.

CONCLUSIONE

Per la compagnia di telefonia mobile l'utilizzo di tutte le variabili per la classificazione dei propri dispositivi è pressoché inutile, dato che come abbiamo potuto vedere dalle tecniche di machine-learning supervisionato le variabili che influenzano la classificazione sono principalmente "RAM", "Battery_Power"," Px_height", e "Px_Width". Tuttavia dai risultati ottenuti con il k-means potremmo anche pensare che la classificazione possa essere effettuata tenendo semplicemente in considerazione la RAM.