Il Problema dei Dati Mancanti

da "*Feature Engineering and Selection: A Practical Approach for Predictive Models*" di Max Kuhn and Kjell Johnson

Alberto Carlone 726894, Alice Ondei 826399, Davide Miori 813692

1. Origine del problema

Si parla di Missing Data quando vi è una mancanza di espressione del dato; si tratta di un errore che sfortunatamente è molto comune e la probabilità di incorrere in un missing value cresce al crescere dell'ampiezza del dato. Le ragioni per cui questo accade sono le più disparate, tra le tante vi sono eventi casuali, unione di dati provenienti da fonti differenti, errori di misurazione ecc.

La presenza di dati mancanti crea problemi in fase di costruzione dei modelli poichè molti di questi non possono essere applicati se il dataset considerato non presenta osservazioni complete. Altri modelli, invece, riescono a produrre stime nonostante la mancanza di informazione. Tali stime però potrebbero essere poco precise o insoddisfacenti.

Esistono quindi tecniche di feature engineering che permettono di affrontare il problema a priori, durante la fase di preprocessing dei dati. Di fondamentale importanza, laddove possibile, è la comprensione dell'origine della mancanza del dato, infatti l'assenza stessa potrebbe essere un'informazione utile al trattamento del problema in esame; la prima e più importante domanda da farsi quando si incontrano dati mancanti è proprio: "perchè mancano?". Si pensi ad un caso di indagine clinica: se durante uno studio un paziente decide di abbandonare il gruppo di ricerca, la sua scelta darà origine da quel momento a una serie di dati mancanti, perché le rilevazioni su tale paziente non potranno più essere effettuate. Tale lacuna però potrebbe essere trasformata in informazione utile.

Vediamo ora più nel dettaglio le tipologie di missing value esistenti. I tre meccanismi generatori più frequenti sono:

1. carenze strutturali nei dati
2. occorrenze casuali
3. cause specifiche

Si parla di carenza strutturale quando una componente di un predittore è stata omessa. Questo tipo di mancanza può essere gestita cercando di identificare la componente necessaria. Alcune volte la presenza o l'assenza di una caratteristica viene codificata in modo che si specifichi esclusivamente l'esistenza. In tal caso un valore mancante potrebbe semplicemente identificare l'assenza di tale proprietà: questo è il caso più facile da gestire e può essere risolto trovando l'encoding più adeguato al caso in esame, ad esempio aggiungendo come livello la "non presenza". Ipotizzando un dataset immobiliare, un dato mancante per la variabile "lunghezza vialetto" potrebbe indicare l'assenza del vialetto di ingresso per l'abitazione considerata; tale mancanza andrebbe quindi codificata in modo corretto.

Per quanto riguarda le occorrenze casuali e le cause specifiche si parla, più precisamente di:

- MCAR (missing completely at random) quando la probabilità di avere un dato mancante è uguale per tutti i punti, che siano osservati o no. Dal punto di vista statistico si dice che i valori mancanti sono indipendenti dai dati.
- MAR (missing at random) quando la probabilità di avere un missing value non è uguale per tutti i punti ma dipende dai dati osservati ed è indipendente dai dati non osservati.
- NMAR (not missing at random) quando esiste una causa specifica che genera l'assenza di informazione, è questo spesso il caso degli studi clinici. Questo tipo di dato mancante è il più difficile da gestire.

2. Esplorazione

```
library(caret)
library(tidyverse)
library(naniar)
library(visdat)
library(ComplexHeatmap)
library(psych)
library(rpart)
library(rpart.plot)
library(randomForest)
library(recipes)
library(ipred)
library(partykit)
library(C50)
library(knitr)
library(kknn)

theme_set(theme_bw())
```

```
data(scat)
load("chicago_raw_entries.RData")
load("stations.RData")
```

Scat dataset

Le analisi presenti nell'elaborato fanno riferimento a due dataset. Il primo, fornito direttamente da R, è lo 'scat' dataset contenente informazioni relative alle feci di tre animali quali coyote, bobcat e greyfox. Questo è composto da 110 osservazioni e 19 variabili le quali sono descritte in breve nella tabella 1.

Variabile	Unità	Descrizione
Species	/	Specie di appartenenza
Month	/	Mese di raccolta
Year	/	Anno di raccolta
Site	/	Luogo di ritrovamento
Location	/	Posizione rispetto al sentiero
Age	/	Età dell'animale
Number	intero	Numero di pezzi
Length	mm	Lunghezza del pezzo più lungo
Diameter	mm	Diametro
Taper	mm	Lunghezza della forma conica
TI	unitless	Rapporto diametro e taper

Variabile	Unità	Descrizione
Mass	grammi	Peso registrato dopo essicazione e cottura
d13C	unitless	impronta isotopica per studiare la dieta dell'animale
d15N	unitless	impronta isotopica per misurare l'impatto sul sedimento
CN	unitless	Rapporto tra atomi di carbonio e nitrogeno
Ropey	binaria	Appare in maniera scomposta?
Segmented	binaria	Mostra segmentazioni?
Flat	binaria	Ha una forma piatta?
Scrape	binaria	Presenta un segno/insenatura vicino all'escremento?

Table 1

Osservando la natura di queste variabili nello snippet sottostante si evince immediatamente che alcune di queste sono codificate in maniera errata: ad esempio la variabile 'flat' è posta come numerica nonostante assuma valore binario dipendente dalla forma dell'escremento. Questo tipo di variabili sono state convertite in factor.

```
str(scat)
```

```
## 'data.frame':    110 obs. of  19 variables:
##  $ Species  : Factor w/ 3 levels "bobcat","coyote",..: 2 2 1 2 2 2 1 1 1 1 ...
##  $ Month    : Factor w/ 9 levels "April","August",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ Year     : int  2012 2012 2012 2012 2012 2012 2012 2012 2012 2012 ...
##  $ Site     : Factor w/ 2 levels "ANNU","YOLA": 2 2 2 2 2 2 1 1 1 1 ...
##  $ Location : Factor w/ 3 levels "edge","middle",..: 1 1 2 2 1 1 3 3 3 2 ...
##  $ Age      : int  5 3 3 5 5 5 1 3 5 5 ...
##  $ Number   : int  2 2 2 2 4 3 5 7 2 1 ...
##  $ Length   : num  9.5 14 9 8.5 8 9 6 5.5 11 20.5 ...
##  $ Diameter : num  25.7 25.4 18.8 18.1 20.7 21.2 15.7 21.9 17.5 18 ...
##  $ Taper    : num  41.9 37.1 16.5 24.7 20.1 28.5 8.2 19.3 29.1 21.4 ...
##  $ TI       : num  1.63 1.46 0.88 1.36 0.97 1.34 0.52 0.88 1.66 1.19 ...
##  $ Mass     : num  15.9 17.6 8.4 7.4 25.4 ...
##  $ d13C     : num  -26.9 -29.6 -28.7 -20.1 -23.2 ...
##  $ d15N     : num  6.94 9.87 8.52 5.79 7.01 8.28 4.2 3.89 7.34 6.06 ...
##  $ CN       : num  8.5 11.3 8.1 11.5 10.6 9 5.4 5.6 5.8 7.7 ...
##  $ ropey    : int  0 0 1 1 0 1 1 0 0 1 ...
##  $ segmented: int  0 0 1 0 1 0 1 1 1 1 ...
##  $ flat     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ scrape   : int  0 0 1 0 0 0 1 0 0 0 ...
```

```
scat$Year <- as.factor(scat$Year)
scat$ropey<- as.factor(scat$ropey)
scat$segmented<- as.factor(scat$segmented)
scat$flat<- as.factor(scat$flat)
scat$scrape<- as.factor(scat$scrape)
```

La figura 1, invece, mostra un pairs plot delle variabili numeriche in cui viene evidenziato come tra di esse vi sia principalmente incorrelazione, salvo un paio di casi in cui il valore si aggira intorno allo 0.5.

```
scat_num <- scat[,c(6,7,8,9,10,11,12,13,14,15)]
pairs.panels(scat_num)
```

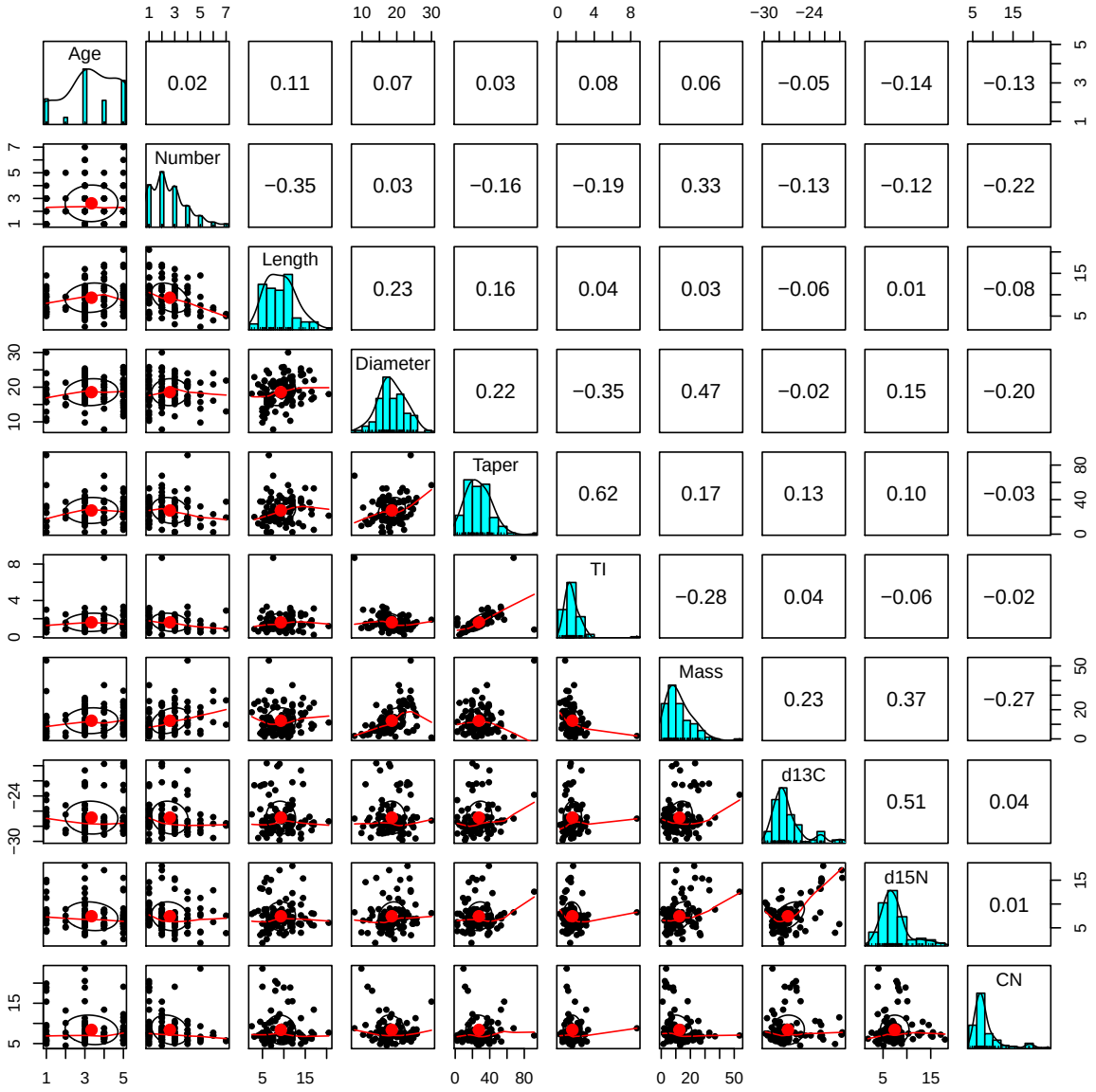

Figure 1: Pairs Plot per dataset scat

Infine, studiando il summary, è possibile notare che alcune variabili hanno valori mancanti, indicati come NA, la cui presenza va trattata con un livello di priorità maggiore rispetto ad altre analisi.

```
summary(scat)
```

```
##      Species         Month      Year      Site        Location       Age
##  bobcat  :57   November :17   2011:31   ANNU:92   edge    :38   Min.   :1.000
##  coyote  :28   January  :16   2012:55   YOLA:18   middle  :47   1st Qu.:3.000
##  gray_fox:25   April    :14   2013:24             off_edge:25   Median :3.000
##                September:14                                     Mean   :3.345
##                June     :13                                     3rd Qu.:5.000
##                October  :12                                     Max.   :5.000
##                (Other)  :24
##      Number          Length          Diameter         Taper
##  Min.   :1.000   Min.   : 2.500   Min.   : 7.80   Min.   : 2.30
##  1st Qu.:2.000   1st Qu.: 6.500   1st Qu.:16.07   1st Qu.:17.30
##  Median :2.000   Median : 9.000   Median :18.05   Median :25.80
##  Mean   :2.618   Mean   : 9.298   Mean   :18.56   Mean   :27.43
##  3rd Qu.:3.000   3rd Qu.:11.500   3rd Qu.:21.32   3rd Qu.:37.40
##  Max.   :7.000   Max.   :20.500   Max.   :30.00   Max.   :91.50
##                                   NA's   :6       NA's   :17
##        TI              Mass            d13C             d15N
##  Min.   :0.230   Min.   : 0.94   Min.   :-29.85   Min.   : 1.840
##  1st Qu.:0.990   1st Qu.: 5.66   1st Qu.:-28.08   1st Qu.: 5.620
##  Median :1.430   Median : 9.75   Median :-27.47   Median : 6.885
##  Mean   :1.602   Mean   :12.46   Mean   :-26.86   Mean   : 7.436
##  3rd Qu.:1.890   3rd Qu.:17.61   3rd Qu.:-26.45   3rd Qu.: 8.305
##  Max.   :8.680   Max.   :53.70   Max.   :-19.67   Max.   :18.000
##  NA's   :17      NA's   :1       NA's   :2        NA's   :2
##        CN         ropey  segmented flat    scrape
##  Min.   : 4.500   0:48   0:48      0:104   0:105
##  1st Qu.: 6.200   1:62   1:62      1:  6   1:  5
##  Median : 7.250
##  Mean   : 8.399
##  3rd Qu.: 8.650
##  Max.   :23.600
##  NA's   :2
```

Chicago trainship dataset

Il secondo dataset è stato raccolto dal Chicago City Portal e contiene informazioni in merito alle corse dei treni nella città di Chicago. Più precisamente sono stati utilizzati due dataset: il primo riguarda il numero di corse (espresso in migliaia) il cui valore è ottenuto incrociando le corse giornaliere (dal gennaio 2001 al settembre 2016) alle stazioni presenti a Chicago. La tabella 2 mostra la forma del dato.
Il secondo, visibile nella tabella 3, fa riferimento esclusivamente alle stazioni citate nel dataset precedente di cui vengono indicati il nome, la posizione geografica e la linea di appartenenza.

Table 2: Chicago raw entries

date	s_40010	s_40020	s_40030	s_40040	s_40050	s_40060	s_40070	s_40080	s_40090
2001-01-01	0.290	0.633	0.483	0.374	0.804	1.165	0.649	1.116	0.411
2001-01-02	1.240	2.950	1.230	7.737	3.199	4.046	5.777	3.854	1.823
2001-01-03	1.412	3.107	1.394	8.051	3.476	4.153	6.482	4.147	1.905
2001-01-04	1.388	3.259	1.370	8.027	3.540	4.362	6.766	4.202	2.008
2001-01-05	1.465	3.357	1.453	7.653	3.684	4.400	6.308	4.404	2.088
2001-01-06	0.613	1.569	0.839	0.844	2.467	2.231	1.798	2.545	1.024
2001-01-07	0.403	0.887	0.589	0.464	1.367	1.565	1.055	1.823	0.561
2001-01-08	1.463	3.222	1.500	8.371	3.544	4.599	7.341	4.527	2.095
2001-01-09	1.505	3.281	1.547	8.351	3.612	4.725	7.537	4.514	2.185
2001-01-10	1.519	3.303	1.521	8.359	3.660	4.684	7.782	4.588	2.166

Table 3: Chicago stations

lon	lat	name	description	station_id	ADA	Red	Blue	Brn
-87.77681	41.87085	Austin	Austin (Blue Line)	s_40010	0	0	1	0
-87.80318	41.88685	Harlem/Lake	Harlem/Lake (Green Line)	s_40020	1	0	0	0
-87.72540	41.88541	Pulaski	Pulaski (Green Line)	s_40030	1	0	0	0
-87.63374	41.87872	Quincy/Wells	Quincy/Wells (Brown, Orange, Purple & Pink Lines)	s_40040	0	0	0	0
-87.68354	42.04771	Davis	Davis (Purple Line)	s_40050	1	0	0	0
-87.71236	41.93813	Belmont	Belmont (Blue Line)	s_40060	0	0	1	0
-87.62930	41.87818	Jackson	Jackson (Blue Line)	s_40070	1	0	1	0
-87.65493	41.95378	Sheridan	Sheridan (Red Line)	s_40080	0	1	0	0
-87.67864	41.96629	Damen	Damen (Brown Line)	s_40090	1	0	0	1
-87.66591	42.00836	Morse	Morse (Red Line)	s_40100	0	1	0	0

Sia per il dataset scat sia per quello di Chicago viene mostrata nelle figure 2 la percentuale di missing value per colonna.

```
g1 =gg_miss_var(scat, show_pct = TRUE)+
  ggtitle('Scat')
g2 = raw_entries %>%
  select_if(any_na) %>%
  gg_miss_var( show_pct = TRUE) +
  ggtitle('Chicago stations')
gridExtra::grid.arrange(g1, g2, ncol=2)
```

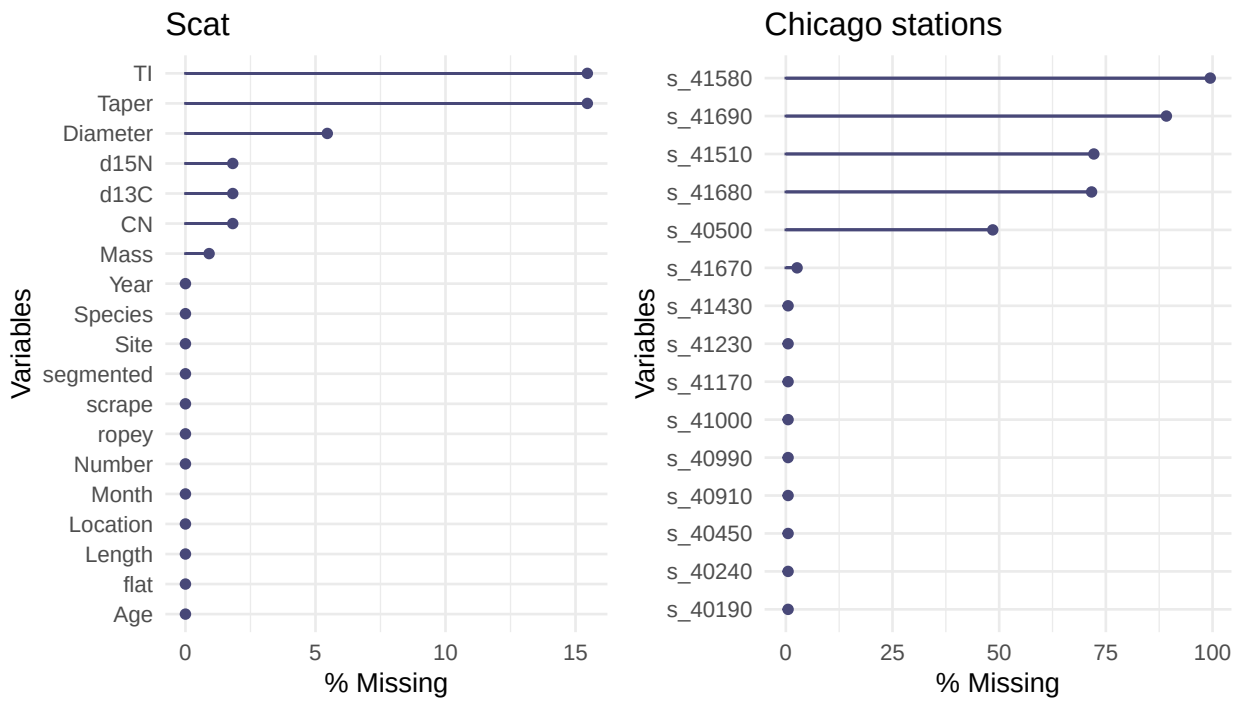

Figure 2: Missing per colonna

2.1 Visualizzazioni

La visualizzazione dei dati è uno strumento necessario a comprendere la natura e il comportamento degli stessi. Trova primario impiego nella fase esplorativa e descrittiva di un dataset, permettendo di avere una visione più completa di ciò su cui si lavora. Nel momento in cui si viene a conoscenza della presenza di valori mancanti internamente al dato è opportuno visualizzarli al fine di comprenderne l'entità e l'origine. Diversi sono i metodi grafici utilizzabili che differiscono in base alla grandezza del dato; se il dato avesse numerose osservazioni e parecchi predittori sarebbe necessario condensare l'informazione prima di visualizzarla.

Quando il dataset ha un numero di righe e colonne moderato lo strumento più utilizzato è la heatmap. La figura 3 mostra un esempio relativo al dataset scat in cui la presenza di valori mancanti è evidenziata dal colore più scuro. In particolare, questo metodo sfrutta il cluster gerarchico applicato sia ai predittori sia alle osservazioni. È facilmente visibile come la maggior parte delle variabili non presenti alcun valore mancante mentre Diameter, Taper e TI ne presentino un numero consistente. Inoltre, tale grafico mostra eventuali relazioni tra righe e colonne: i missing compaiono simultaneamente in Taper e TI, per fare un esempio.

```
convert_missing <- function(x) ifelse(is.na(x), 0, 1)
scat_missing <- apply(scat, 2, convert_missing)

Heatmap(
  scat_missing,
  name = "Missing",
  column_title = "Predittori", row_title = "Osservazioni",
  col = c("black","lightblue"),
  show_heatmap_legend = FALSE,
  row_names_gp = gpar(fontsize = 0)
)
```

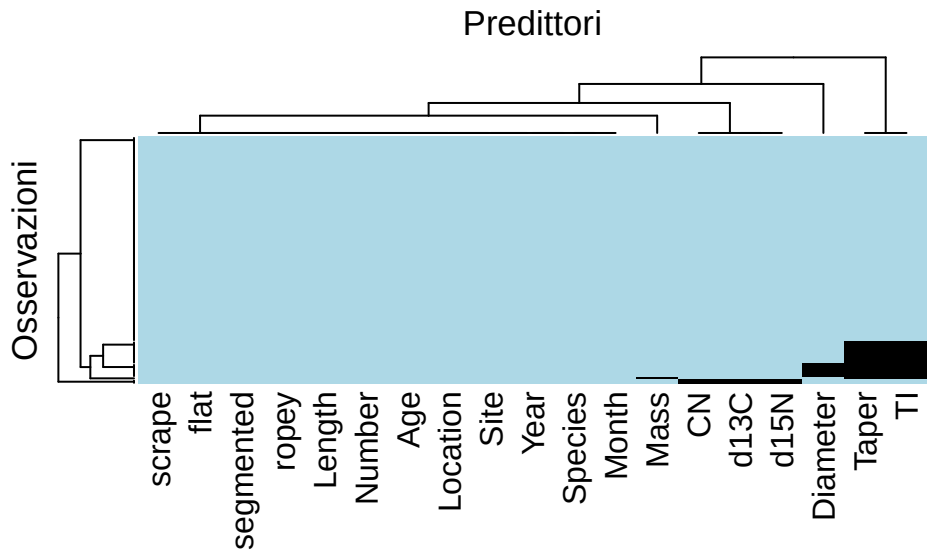

Figure 3: Heatmap gerarchica

```
gg_miss_upset(scat, nsets = 7)
```

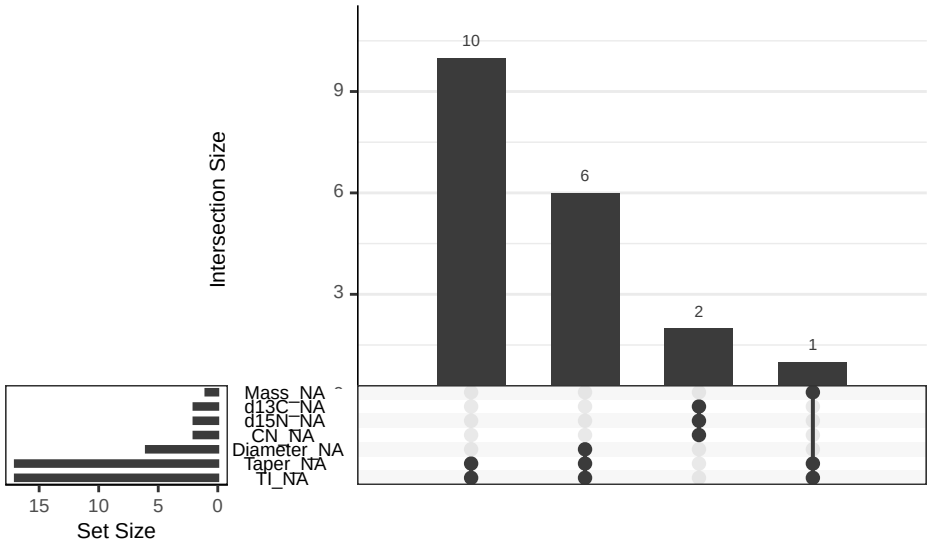

Figure 4: Co-occurency plot

In coppia alla heatmap è possibile ricorrere al co-occurrence plot, il quale rappresenta la frequenza di valori comunemente mancanti tra i predittori. Un esempio di tale grafico è mostrato nella figura 4, dove la relazione tra Taper e TI, già evidenziata in precedenza, appare ancora più marcata.

Oltre ai precedenti, l'utilizzo di grafici bivariati è di grande aiuto per comprendere l'origine dei valori mancanti. Il grafico 5 mostra uno scatterplot originato dall'incrocio delle variabili Diameter e Mass i cui punti sono colorati sulla base di una terza variabile Flat. I trattini posizionati sugli assi indicano le osservazioni che presentano un valore mancante per una delle due variabili. Ad esempio, i trattini azzurri sull'asse y sono osservazioni presentanti un missing nel Diameter. È curioso notare come solo questi siano classificati

come Flat, ovvero escrementi aventi una forma piatta. Ottenuta questa informazione è possibile attribuire la mancanza di valori per il diametro al fatto che un'osservazione è considerata flat. Questa relazione può probabilmente essere di natura strutturale, anche se non si esclude una randomicità dal momento che la scarsa numerosità del dato non permette di conoscere con buona confidenza la natura del valore mancante.

```
scat_flat <-
  scat %>%
  mutate(flat = ifelse(flat == 1, "yes", "no"))

ggplot(scat_flat, aes(col = flat)) +
  geom_point(aes(x = Diameter, y = Mass), alpha = .7, size=2) +
  geom_rug(data = scat_flat[is.na(scat_flat$Mass),],
           aes(x = Diameter),
           sides = "b",
           lwd = 1)+
  geom_rug(data = scat_flat[is.na(scat_flat$Diameter),],
           aes(y = Mass),
           sides = "l",
           lwd = 1) +
  theme(legend.position = "top")
```

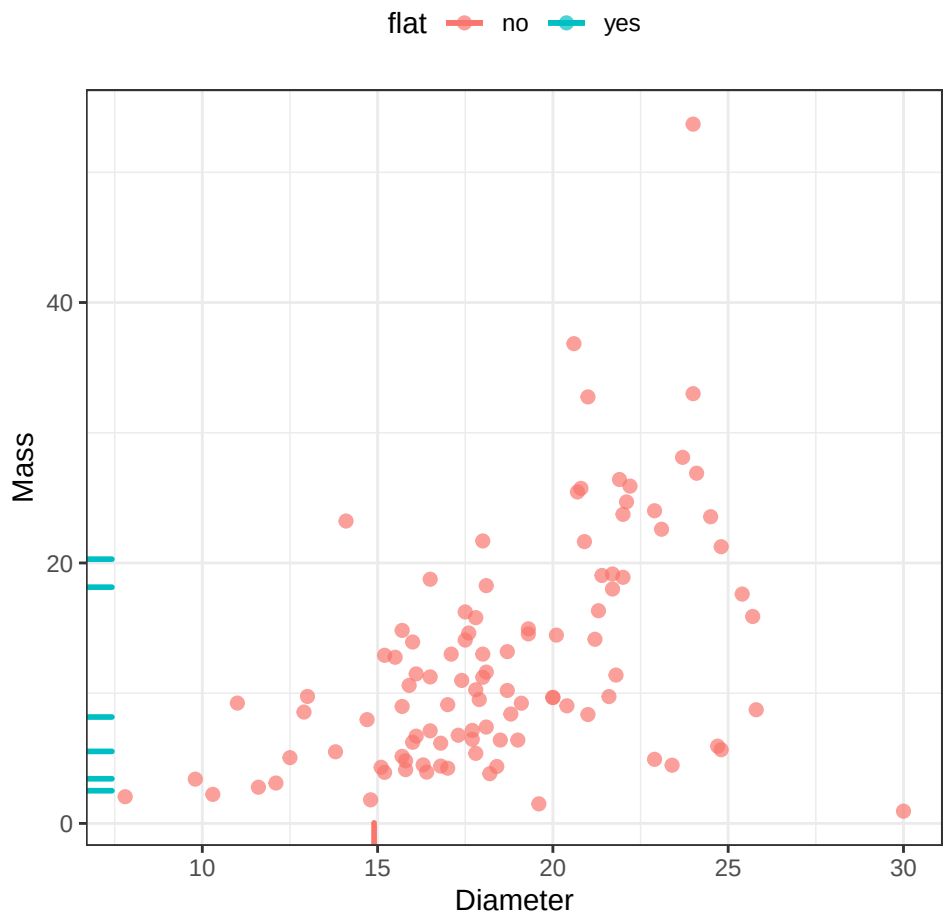

Figure 5: ScatterPlot valori mancanti condizionati

Quando si è di fronte ad un dato con una grande quantità di osservazioni e predittori, le tecniche sopra citate perdono di efficacia e chiarezza. Come anticipato, è possibile condensare l'informazione riuscendo a visualizzarne l'impatto. Per tale fine si ricorre alla Principal Component Analysis, meglio nota come PCA, utilizzata per ridurre la dimensionalità del dato senza che ne consegua una perdita sostanziale dell'informazione.

In particolare, la PCA applica una trasformazione lineare ai predittori introducendo un set nuovo, formato dalle componenti principali in numero minore o uguale al numero delle variabili originali.
In questo caso la matrice formata dalle variabili esplicative viene convertita in una matrice formata da 0 e 1, dove il primo rappresenta un valore non-missing mentre il secondo viene attribuito quando il valore è mancante. Siccome la PCA applica una trasformazione cercando di massimizzare la varianza spiegata, la dimensione iniziale cattura la variazione causata dalla presenza o assenza dei valori mancanti. Sinteticamente, osservazioni presentanti un missing saranno proiettate lontano dall'origine, mentre quelle che non ne presentano saranno vicine ad esso.

Di conseguenza è possibile rappresentare graficamente le prime due componenti (quelle che hanno una varianza spiegata maggiore) al fine di identificare la natura e l'entità dei valori mancanti. Qualora si volesse studiare questa natura rispetto ai predittori, sarebbe sufficiente calcolare la matrice trasposta prima di applicare la PCA; in questo modo verrà catturata la variazione causata dai missing tra le esplicative.

La figura 6 è uno scatterplot rappresentante le prime due componenti principali ottenute dal dataset Chicago ridership del quale sono state considerate le stazioni e le date in cui è avvenuta la corsa. La dimensione dei punti è data dalla quantità di stazioni mancanti; sono presenti solo otto punti perché sono stati considerati solo i valori unici corrispondenti alla proporzione di valori mancanti per data. Questi otto punti, dunque, rappresentano gli otto pattern che ricorrono all'interno del dataset. In più ogni giorno ha almeno un valore mancante.

```
only_rides <-
  raw_entries %>%
  dplyr::select(-date)

convert_missing <- function(x) ifelse(is.na(x), 0, 1)
date_missing <- apply(only_rides, 2, convert_missing)

date_missing_ppn <-
  apply(only_rides, MARGIN = 1, function(x)
    sum(is.na(x))) / ncol(only_rides)

pca_dates <- prcomp(date_missing)

pca_d <-
  data.frame(pca_dates$x) %>%
  dplyr::select(PC1, PC2) %>%
  mutate(Percentuale = date_missing_ppn * 100) %>%
  dplyr::distinct(PC1, PC2, Percentuale)

pca_d_rng <- extendrange(c(pca_d$PC1, pca_d$PC2))

ggplot(pca_d, aes(x = PC1, y = PC2, size = Percentuale)) +
  geom_point(alpha = .5, col = "#1B9E77") +
  xlim(pca_d_rng) +
  ylim(pca_d_rng) +
  scale_size(limits = c(0, 10), range = c(.5, 10))
```

Un'ulteriore via nel caso si avessero a disposizione dati di grandi dimensioni è l'utilizzo di tabelle summary. Queste permettono di sintetizzare tramite degli indicatori semplici la situazione in cui ci si trova.

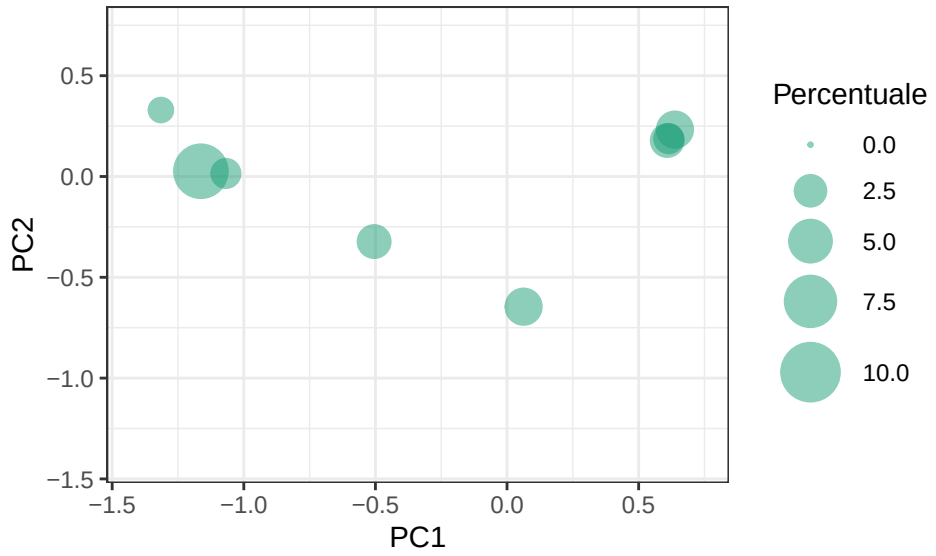

Figure 6: PCA

Si calcola solitamente per ogni variabile (o osservazione) la percentuale di valori mancanti relativo al numero di righe (o colonne), in modo tale da avere una visione completa e veloce.

3. Approccio al problema

L'esistenza di valori mancanti all'interno del dataset è un problema qualora si intenda utilizzare un algoritmo non in grado di tollerare tali mancanze, come ad esempio le Support Vector Machines (SVM), le reti neurali e i modelli lineari generalizzati (GLM). Fortunatamente, esistono diverse vie percorribili che permettono in ogni caso un'analisi sufficientemente accurata.

3.1 Modelli resistenti

Una prima possibilità sono i modelli resistenti alla presenza di valori mancanti.
Tra questi i modelli basati sugli alberi sono molto utilizzati, come ad esempio i CART (Classification And Regression Tree): come illustrato da Breiman nell'articolo Classification and Regression Trees (Breiman et al. 1984), il modello CART sfrutta l'idea dei surrogate splits, ossia la creazione di uno split aggiuntivo nel caso in cui sia presente un valore mancante nell'attributo originariamente considerato.

Prendendo in esame il dataset scat, e considerando l'attributo CN con uno split iniziale di 7, nel caso in cui quest'ultimo presenti un valore mancante il modello procede considerando uno split differente attraverso un attributo tale da contenere il più alto numero di elementi presenti nello split CN.

```
rpart_mod <- rpart(Species ~ ., data=scat)
rpart_party <- as.party(rpart_mod)
plot(rpart_party)
```

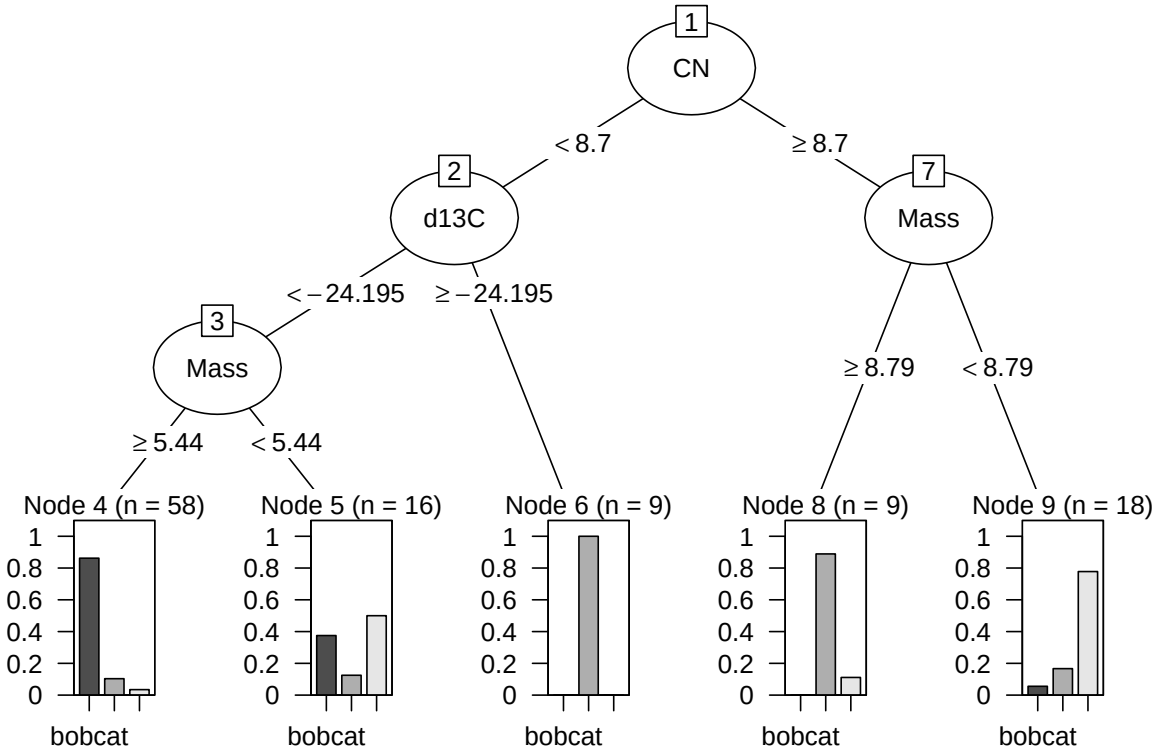

Figure 7: Albero CART

Un'ulteriore approccio, differente rispetto a CART, è il C5.0. Esso utilizza conteggi frazionari per i valori mancanti in ogni split successivo al primo. Se, ad esempio, il primo split basato sull'attributo $d13c > 24.55$ ne genera uno di training in cui tutti e 13 i samples sono coyote e per quell'attributo esiste un valore mancante, negli split successivi il conteggio delle specie sarà frazionario per riflettere la proporzione di valori mancanti.

3.2 Gestione Missing

La gestione dei missing value è un passo importante da compiere prima di procedere nell'analisi. Fatta eccezione per casi in cui la presenza di valori mancanti non rappresenta un problema (si veda 3.1) è importante adottare la strategia giusta nella gestione degli stessi. Tale strategia varia in base al contesto, a come si sono originati i missing e alla natura del dato in cui compaiono (numerico o categorico).

3.2.1 Eliminazione

Il metodo più semplice e veloce è sicuramente l'eliminazione della riga o colonna contenente il valore mancante. Chiaramente questa decisione, se presa senza criterio, può introdurre una distorsione in analisi future; è bene quindi ponderare attentamente possibili conseguenze.

Relativamente ai predittori, se questi sono inutili o poco adeguati all'analisi in questione è possibile rimuovere interamente la variabile contenente i valori mancanti. Se questa invece risulta una valida esplicativa è bene adottare una strategia diversa rispetto alla sua eliminazione.

Allo stesso modo, è importante pesare la decisione di rimuovere le osservazioni. Se il dataset è numeroso in termini di righe e tra queste il numero di missing non è elevato, è possibile rimuoverle. Qualora si avesse a che fare con un dato di dimensioni ridotte, è buona norma evitare di privarsene.

Solitamente, quando vi è la presenza di numerose righe, una volta rimossi i predittori considerati come poco esplicativi, si potrebbe optare per l'eliminazione delle righe se queste superano una determinata soglia di valori mancanti.

Dal punto di vista della natura del valore mancante, invece, se questo risulta essere generato completamente in maniera casuale è possibile rimuoverlo. Se al contrario si riscontra la presenza di un pattern, si ha la possibilità di ottenere un valore dalla mancanza in questione. La relazione che sussiste tra Diameter e Flat, citata nel paragrafo 2.1, è un esempio di questa eventualità.

Considerando nuovamente il dataset di Chicago, la figura 8 mostra solo le stazioni presentanti valori quantitativamente elevati di missing value. In questo caso, la grande quantità di valori mancanti comporterebbe l'eliminazione delle stazioni in oggetto nonostante tali mancanze non si dimostrino completamente casuali. A dar valore a questa eliminazione è soprattutto l'alta correlazione tra le varie stazioni che porterebbe, a parità di valore esplicativo, a preferirne una priva di valori mancanti.

```
miss_entries <-
  raw_entries %>%
  dplyr::select(-date) %>%
  is.na()
miss_num <- apply(miss_entries, 2, sum)

has_missing <- vapply(raw_entries[, -1], function(x) sum(is.na(x)) > 1, logical(1))
miss_station <- names(has_missing)[has_missing]


miss_data <-
  raw_entries[, miss_station] %>%
  is.na()
```

```
clst <- hclust(dist(t(miss_data)))
clst_stations <-
  tibble(
    station_id = colnames(miss_data),
    order = clst$order
  )

station_names <-
  stations %>%
  dplyr::select(name, station_id) %>%
  right_join(clst_stations, by = "station_id")

station_lvl <- station_names[["name"]][station_names$order]

miss_vert <-
  raw_entries %>%
  gather(station_id, raw_entries, -date) %>%
  filter(station_id %in% miss_station) %>%
  mutate(status = ifelse(is.na(raw_entries), "missing", "complete")) %>%
  full_join(station_names, by = "station_id") %>%
  mutate(
    name = factor(name, levels = station_lvl),
    stato = factor(status, levels = c("missing", "complete"))
  )

ggplot(miss_vert, aes(x = date, y = name, fill = stato)) +
  geom_tile() +
  ylab("") + xlab("") +
  scale_fill_manual(values = c("black", "lightgray"))
```

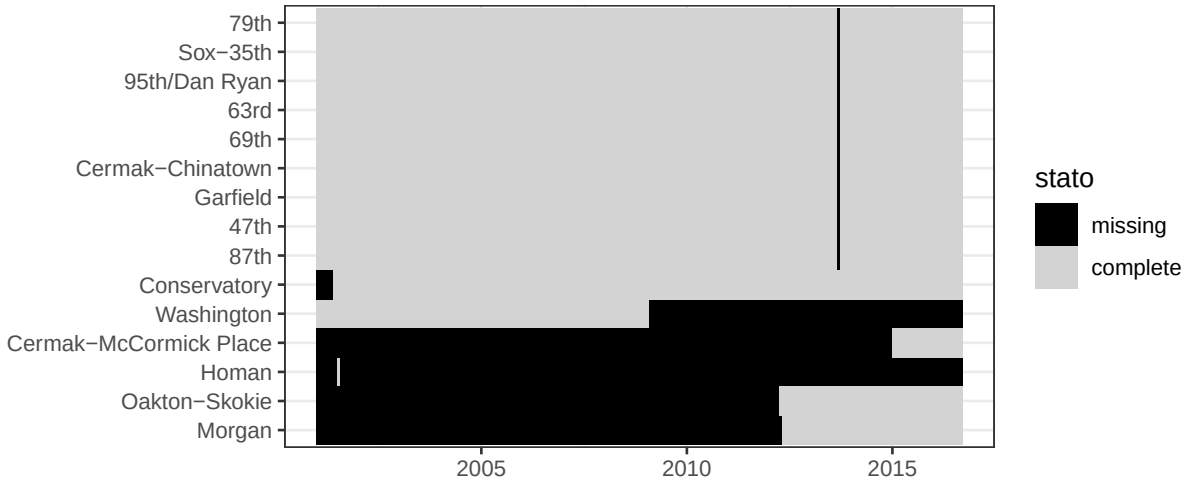

Figure 8: Missing per stazione

A titolo eseplificativo si riporta un possibile codice da utilizzare qualora si volessero eliminare le osservazioni aventi missing value:

```
for (i in 1:dim(raw_entries)[2]){
  if (sum(is.na(raw_entries[i]))!=0) {
    print(paste(colnames(raw_entries)[i], 'ha',sum(is.na(raw_entries[i])), 'missing'))
  }
}
```

```
## [1] "s_40190 ha 30 missing"
## [1] "s_40240 ha 30 missing"
## [1] "s_40450 ha 30 missing"
## [1] "s_40500 ha 2780 missing"
## [1] "s_40910 ha 30 missing"
## [1] "s_40990 ha 30 missing"
## [1] "s_41000 ha 30 missing"
## [1] "s_41170 ha 30 missing"
## [1] "s_41230 ha 30 missing"
## [1] "s_41430 ha 30 missing"
## [1] "s_41510 ha 4138 missing"
## [1] "s_41580 ha 5702 missing"
## [1] "s_41670 ha 151 missing"
## [1] "s_41680 ha 4108 missing"
## [1] "s_41690 ha 5113 missing"
```

Il primo snippet permette di eliminare tutte le colonne contenenti almeno un missing mentre il secondo di elimare solo le variabili con una percentuale maggiore del 40%.

```
raw_entries2 = raw_entries %>%
                  select_if(~ !any(is.na(.)))
```

```
threshold <- 0.4
raw_entries3 = raw_entries %>% select(where(~mean(is.na(.)) < threshold))
```

3.2.2 Encoding

Un secondo metodo di gestione dei missing value è l'encoding degli stessi. Questa tecnica può essere utilizzata quando l'attributo che si sta trattando è di tipo qualitativo; ai vari livelli ne verrà quindi aggiunto uno che andrà ad indicare la "missingness". Questo tipo di approccio trova applicazione specifica nei casi in cui l'assenza di una determinata caratteristica non viene registrata e, di conseguenza, produce un missing value (si veda l'esempio proposto nel capitolo di apertura relativo al contesto immobiliare).

La codifica dei valori mancanti può anche portare alla luce una relazione tra il fenomeno indagato e il fatto che per quell'attributo manchi il dato. Solitamente però è difficile riuscire a comprendere la ragione di questa associazione. Ad ogni modo, considerare questa soluzione come definitiva e quindi dichiarare una sorta di relazione causa - effetto può essere erroneo. Per questa ragione è necessario eseguire anche un successivo step, chiedendosi se l'operazione fatta ha valore anche dal punto di vista logico e di interpretazione del modello. Per concludere, è fondamentale capire come verrebbero interpretati i risultati ottenuti se l'informazione codificata diventasse importante per il modello.

3.2.3 Imputazione

Un successivo metodo di gestione dei dati mancanti è l'imputazione o la stima degli stessi. Per fare ciò si utilizza l'informazione fornita dagli altri predittori così da riempire il valore mancante. Le tecniche di imputazione vengono considerate in modo molto cauto, per via del loro potenziale impatto, talora molto forte, sull'analisi dei dati. Devono quindi garantire la rilevanza statistica dei risultati ottenuti, consentendo l'applicazione degli usuali test d'ipotesi.

Esistono tre metodi principali per l'imputazione dei dati:

- **Imputazione della media / mediana**: si inserisce la media / mediana calcolata a partire dai dati a disposizione. Questa procedura è utile e utilizzata specialmente in assenza di ogni altro tipo di informazione. Provoca una diminuzione della varianza dei dati, che implica una riduzione del coefficiente di correlazione.
- **Imputazione della media / mediana condizionata**: Si inserisce la media / mediana calcolata sulla base di gruppi specifici quindi condizionatamente a tali gruppi. La definizione dei gruppi, se non evidente dai dati, può essere determinata dalla conoscenza a priori del ricercatore.
- **Imputazione attraverso un metodo di regressione**: La stima del dato mancante è ottenuta stimando un modello di regressione la cui variabile con i casi mancanti è usata come variabile dipendente e le altre come esplicative. Il modello è stimato sulla base dei dati completi ed è utilizzato per prevedere i casi mancanti. Questo metodo è il più sofisticato ma può soffrire dei problemi legati alla qualità del modello di regressione utilizzato.

Quando si vogliono stimare i valori mancanti si tende a preferire modelli che riescono a produrre la previsione più accurata possibile del punto mancante, per questo vengono spesso preferiti modelli predittivi a modelli inferenziali.
Quando si effettua una procedura di imputazione è bene tener a mente che:

- imputazioni ripetute possono portare a costi computazionali alti e al sovraccarico;
- all'interno del campione a disposizione è possibile che più variabili abbiano missing value; per questo un buon metodo di imputazione dovrebbe tollerare la presenza di altri dati mancanti;
- nei dataset solitamente coesistono variabili qualitative e quantitative; invece di generare variabili fittizie per predittori qualitativi, un buon metodo di imputazione dovrebbe essere in grado di utilizzare predittori di diversi tipo come input;
- il metodo scelto dovrebbe essere relativamente stabile e poco influenzabile dalla possibile presenza di outlier;
- nel caso si decidesse, per ragioni legate all'analisi che si sta conducendo, di applicare trasformazioni sui dati, come la comune normalizzazione o standardizzazione del dato, è bene svolgere tali operazioni dopo aver imputato i dati mancanti. Quindi si è soliti considerare l'imputazione dei dati come primo passo della sequenza di pre-processing;
- prima di procedere con l'imputazione è necessario valutare l'ammontare dei dati mancanti; non esiste una vera e propria regola che stabilisca una soglia rigida oltre la quale risulta non essere opportuno procedere percorrendo questa via, per convenzione però si tende a considerare come limite il 20% del totale dei dati;
- si deve sempre tener a mente che l'imputazione migliora in modo fittizio i dati, spesso riduce la varianza poiché il valore stimato è probabilmente molto vicino alla media;
- il metodo è efficace se le variabili utilizzate per la stima sono dei buoni predittori per la variabile con i valori mancanti;

- le stime ottenute sono utilizzabili se assumono valori nel range dei valori assunti dalla variabile nei casi completi.

Generalmente ogni modello predittivo può essere utilizzato per imputare i dati mancanti, qui ne verranno esposti tre molto comuni.

K-Nearest Neighbors (KNN) Il KNN è un metodo che viene definito non parametrico, perchè non fa esplicite assunzioni circa la forma funzionale di f ma, al contrario, lascia che siano i dati a parlare per se stessi. Questa sua caratteristica permette di esplorare qualsiasi forma funzionale, a costo però di avere un gran numero di osservazioni a disposizione.

Supponiamo che y_1^* sia il valore che vogliamo prevedere, in quanto mancante. Per far ciò è necessario definire un intorno di x_1^*, quindi individuare un insieme di punti ad esso vicini. Il numero di vicini da considerare sarà indicato dal parametro k. Tale parametro viene definito di regolarizzazione (o di tuning). Più k è piccolo più si considerano intorni piccoli quindi la stima avrà varianza alta e stima più flessibile, all'aumentare di k invece la stima sarà meno flessibile. I due casi limite si verificano per $k = 1$, dove si considera esclusivamente il punto più vicino a quello considerato, oppure $k = n$, in tal caso la previsione y^* coincide con la media del training. Sebbene sia un parametro da ottimizzare, si predilige usare valori come $k = 5$ oppure $k = 10$.

Successivamente, viene definita una metrica che possa identificare la vicinanza o la lontananza di questi punti e possa permettere di definire l'intorno $N_k(x_1^*)$. Questa viene scelta in base alla natura del dato in questione. Se i predittori sono tutti quantitativi si utilizza la distanza euclidea, definita come:

$$\|x_i - x_1^*\| = \sqrt{(x_i - x_1^*)^T(x_i - x_1^*)}$$

Nel caso invece ci fossero anche predittori qualitativi è preferibile la distanza di Gower. Questa metrica utilizza due tecniche separate, una nel caso in cui l'attributo sia qualitativo, l'altra nel caso sia quantitativo. Per un predittore qualitativo esistono soltanto due possibilità: se il punto considerato e il predittore hanno lo stesso valore la distanza è 1, altrimenti è 0.

Per le variabili quantitative invece la distanza è definita come segue:

$$d(x_i, x_j) = 1 - \frac{|x_i - x_j|}{R_x}$$

Dove R_x è il range del predittore in esame. La distanza è calcolata per ogni predittore e come distanza generale si considera la loro media.

È buona norma standardizzare le variabili prima di procedere, così che il computo della distanza non risenta dell'ordine di grandezza delle variabili.

Questa tecnica viene utilizzata al fine di predire dati mancanti quando il training set è di dimensioni moderatamente piccole. Si procede nel seguente modo: dopo aver identificato i le osservazioni contenenti uno o più missing value, la misura di distanza e dopo aver assegnato un valore a k, si individuano i k esempi più vicini a quello considerato, scegliendo esclusivamente tra quelli che presentano osservazioni complete. Il missing value sarà quindi sostituito con la media dei valori assunti dal predittore di interesse nei k casi più vicini.

Applicazione
Abbiamo applicato KNN al fine di imputare i valori mancanti relativi alla variabile Diameter e Mass andando a confrontare l'imputazione con quella ottenuta nel libro. Per prima cosa abbiamo creato un dataframe in cui non compaiono missing values per le due variabili in questione. Questo corrisponderà al nostro training mentre le restanti righe contenenti i missing per Diameter e Mass formeranno il test set.

Successivamente, grazie alla CrossValidation (CV), è stato ottimizzato il parametro k sia per il KNN con Diameter come variabile dipendente sia per quello con Mass. I risultati sono mostrati dalle figure 9 e 10. Inoltre, è stato stampato l'errore reltivo al k migliore espresso in diverse forme. Il criterio di scelta è stato

RMSE.
Ottenuti i due k, sono stati stimati i modelli ottenendo per il test set le previsioni associate ai valori mancanti che conseguentemente sono state imputate nel dataset.

A causa della contemporanea mancanza di Taper e TI per le osservazioni di Diameter e Mass da imputare, la previsione di tali osservazioni risulta impossibile attraverso un modello che tiene in considerazione le prime due variabili. Per questo motivo, sono state escluse in fase di training.

```
scat_missing <-
  scat %>%
  mutate(
    was_missing = ifelse(is.na(Diameter)| is.na(Mass), "yes", "no"),
    was_missing = factor(was_missing, levels = c("yes", "no"))
  )
```

La funzione seguente viene utilizzata dall'autore del libro per imputare i valori mancanti di Diameter e Mass.

```
imp_knn <-
  recipe(Species ~ ., data = scat) %>%
  step_knnimpute(Diameter, Mass,
                 impute_with =
                   imp_vars(Month, Year, Site, Location,
                            Age, Number, Length, ropey,
                            segmented, scrape),
                 neighbors = 5) %>%
  prep(training = scat, retain = TRUE) %>%
  juice(Diameter, Mass) %>%
  set_names(c("diam_imp", "mass_imp")) %>%
  mutate(method = "5-Nearest Neighbors")

scat_knn_libro <- bind_cols(scat_missing, imp_knn)
```

```
scat_train_test <- subset(scat_missing, was_missing=='no')
scat_train_test$was_missing<-NULL
```

Il seguente metodo, invece, viene costruito manualmente come precedentemente illustrato. Prima per il diametro,

```
formula <- Diameter~Species+Month+Year+Site+Location+Age+Number+
  Length+Mass+d13C+d15N+CN+ropey+segmented+flat+scrape

set.seed(42)
cv <- trainControl(method = "cv", number = 10)
fit.knn <- train(formula, scat_train_test, method = "knn", trControl = cv,
                 tuneGrid = data.frame(k = 1:40), na.action = na.omit)
```

```
plot(fit.knn)
```

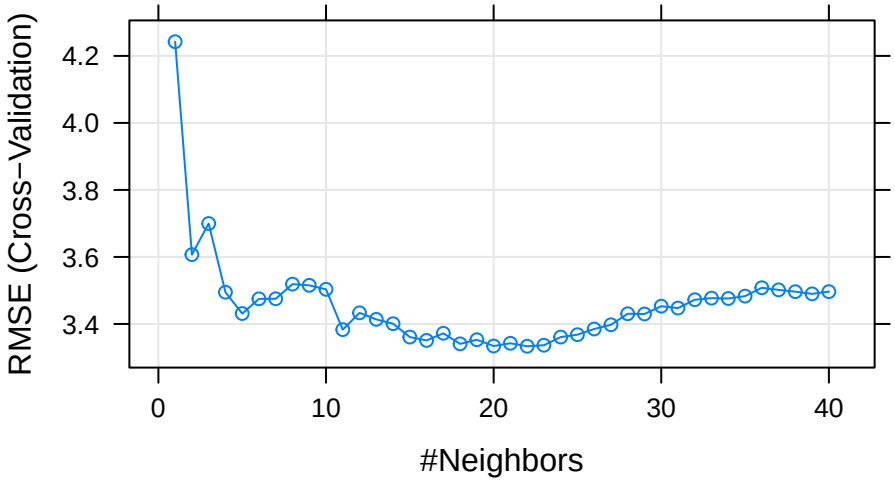

Figure 9: CV del modello Diameter

```
best_k = as.numeric(fit.knn$bestTune[1])
fit.knn$results[best_k,]
```

```
##     k     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 22 22 3.333424 0.3394407 2.552438 0.8609667  0.2227151 0.5237679
```

```
fit_imput <- kknn( formula, train=scat_train_test, test = scat[is.na(scat$Diameter),] ,
             distance = 2, kernel = 'rectangular',
             k=best_k)

yhat_imput <- fit_imput$fitted.values
```

poi per la massa.

```
formula2 <- Mass ~Species+Month+Year+Site+Location+Age+Number+Length+
  Diameter+d13C+d15N+CN+ropey+segmented+flat+scrape

cv2 <- trainControl(method = "cv", number = 10)
fit.knn2 <- train(formula2, scat_train_test, method = "knn", trControl = cv,
                tuneGrid = data.frame(k = 1:40), na.action = na.omit)
```

```
plot(fit.knn2)
```

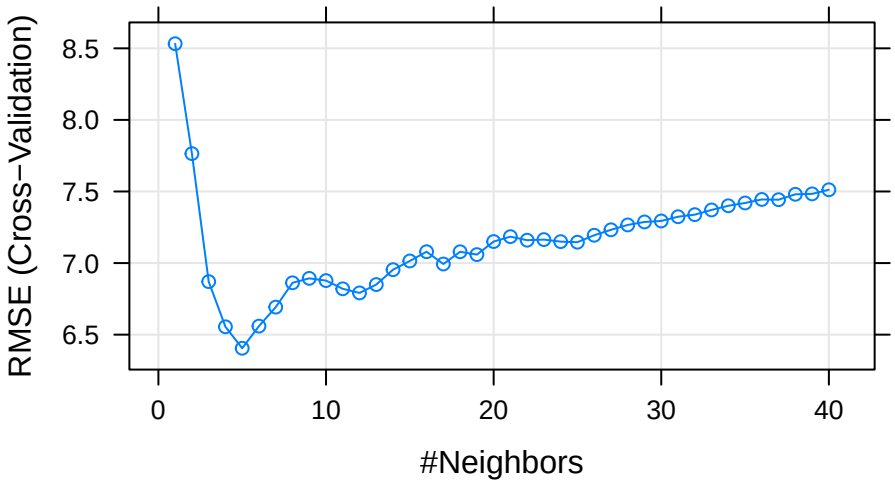

Figure 10: CV del modello Mass

```
best_k2 = as.numeric(fit.knn2$bestTune[1])
fit.knn2$results[best_k2,]
```

```
##   k     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 5 5 6.404961 0.5055969 4.972472 2.697029  0.2278451 1.796792
```

```
fit2_imput <- kknn( formula2, train=scat_train_test, test = scat[is.na(scat$Mass),],
              distance = 2, kernel = 'rectangular',
              k=best_k2)

yhat2_imput <- fit2_imput$fitted.values
```

```
scat_knn <-
  scat_missing %>%
  mutate(method = "KNN",
         diam_imp = Diameter, mass_imp = Mass)



scat_knn$diam_imp[is.na(scat_knn$Diameter)] <- yhat_imput
scat_knn$mass_imp[is.na(scat_knn$Mass)] <- yhat2_imput
```

La figura 11 mostra lo scatterplot in cui i punti rossi corrispondono alle imputazioni.

```
g3 = ggplot(scat_knn, aes(col = was_missing)) +
  geom_point(aes(x = diam_imp, y = mass_imp), alpha = .7, cex = 2) +
  geom_rug(data = scat_knn[is.na(scat_knn$Mass),],
```

```
            aes(x = Diameter),
            sides = "b",
            lwd = 1) +
  geom_rug(data = scat_knn[is.na(scat_knn$Diameter),],
            aes(y = Mass),
            sides = "l",
            lwd = 1) +
  theme(legend.position = "top") +
  xlab("Diameter") + ylab("Mass")+
  ggtitle('Knn')

g4 = ggplot(scat_knn_libro, aes(col = was_missing)) +
  geom_point(aes(x = diam_imp, y = mass_imp), alpha = .7, cex = 2) +
  geom_rug(data = scat_knn[is.na(scat_knn$Mass),],
            aes(x = Diameter),
            sides = "b",
            lwd = 1) +
  geom_rug(data = scat_knn[is.na(scat_knn$Diameter),],
            aes(y = Mass),
            sides = "l",
            lwd = 1) +
  theme(legend.position = "top") +
  xlab("Diameter") + ylab("Mass")+
  ggtitle('Knn del libro')

gridExtra::grid.arrange(g3, g4, ncol=2)
```

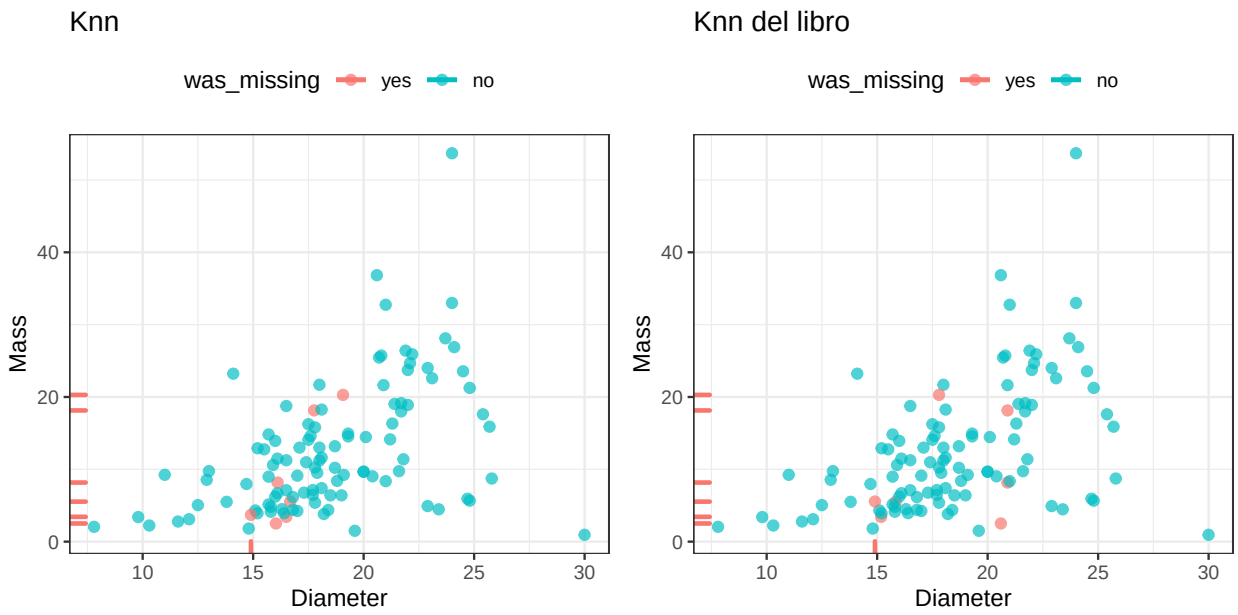

Figure 11: Valori imputati a confronto

TREES I modelli basati sugli alberi sono una buona scelta poiché possono essere utilizzati anche in presenza di valori mancanti (si veda 3.1), mantenendo buone performance in termini di accuratezza.

Il singolo albero (di classificazione o regressione) ha una elevata interpretabilità ma, di contro, vi è la tendenza ad avere modelli con errore basso e varianza elevata: una piccola variazione nei dati di partenza, infatti, può generare degli split diversi nelle ramificazioni. Inoltre cambiando uno split, anche quelli sottostanti verranno modificati, propagando la variabilità per tutto il modello.

```
scat_missing_tree <-
  scat %>%
  mutate(
    was_missing = ifelse(is.na(Diameter)| is.na(Mass), "yes", "no"),
    was_missing = factor(was_missing, levels = c("yes", "no"))
  )
```

Alberi singoli

```
d.tree <- rpart(Diameter ~ ., scat)
m.tree <- rpart(Mass ~ ., scat)

rpart.plot(d.tree, type = 0, extra = 1)
```

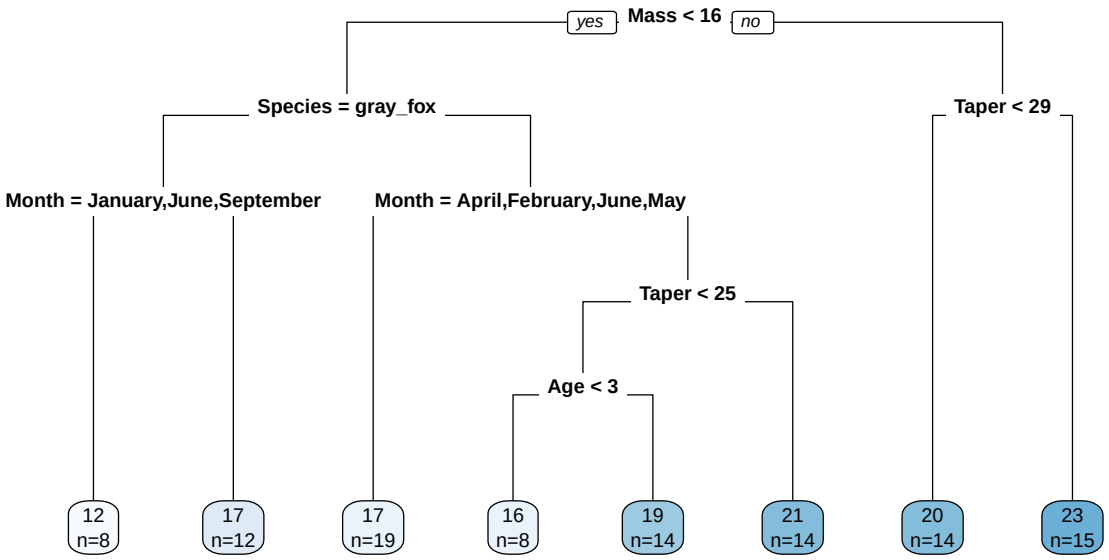

Figure 12: Albero CART per Diameter

```
rpart.plot(m.tree, type = 0, extra = 1)
```

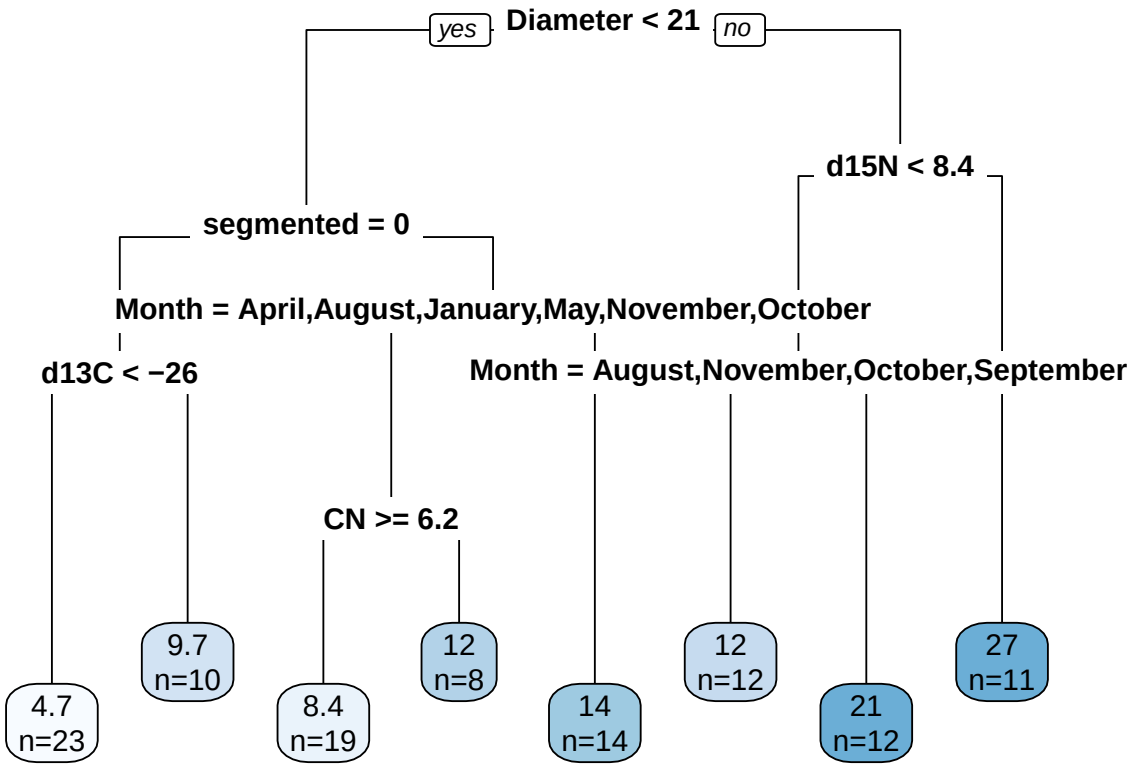

Figure 13: Albero CART per Mass

Al fine di contrastare questo fenomeno è possibile utilizzare metodi di ensemble learning. Si tratta di tecniche che derivano da modelli multipli, la cui combinazione ne restituisce uno complessivamente migliore in termini di performance. I singoli modelli che vengono combinati sono detti weak learners, ad indicare che risultano essere leggermente migliori rispetto alla scelta casuale. Quando però questi vengono combinati, vengono definiti strong learners, quindi in grado di restituire delle previsioni particolarmente precise e con una varianza inferiore, a discapito dell'interpretabilità e del costo computazionale.

Bagging

Alla base di queste tecniche vi è la tecnica del bootstrap, che prevede il ricampionamento con reimmissione delle n osservazioni appartenenti al dataset, anch'esso di dimensione n.

Accade quindi che non tutte le osservazioni appartenenti al dataset originale compaiano nel campione che, al contrario, conterrà anche osservazioni doppie. La probabilità di non includere una singola osservazione sarà pari a $1 - \frac{1}{n}$. All'aumentare di n, le osservazioni escluse (chiamate out-of-bag, OOB) tenderanno ad essere circa pari ad $\frac{1}{3}$.

Replicando B volte il processo di ricampionamento bootstrap si ottengono B dataset su cui allenare il modello, che avrà un errore medio di previsione pari a

$$\hat{Err}_{Boot} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{n}(y_i - \hat{f}^b(x_i))^2$$

con $\hat{f}^b$ e $\hat{f}^b(x_i)$ rispettivamente la funzione e la previsione dell'osservazione x_i del b-esimo dataset.

Nel caso specifico di bagged tree per problemi di regressione, per ciascuno dei B dataset ottenuti si allena un singolo albero. Le performance ottenute verranno poi aggregate con una media:

$$\bar{f}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

Per validare le performance del modello e avere una stima dell'errore su un dataset di test di cui non conosciamo i valori reali, è possibile sfruttare l'errore out-of-bag. Per ogni i-esima osservazione, viene effettuata la previsione utilizzando tutti i bagged trees che non la contengono: questo restituisce circa $\frac{B}{3}$ previsioni che vengono poi mediate.

Applicazione

Di seguito vengono messi a confronto due metodologie per ottenere i valori mancanti di Diameter e Mass attraverso i bagged tree. Il primo metodo consiste nell'iterazione dell'allenamento di un singolo albero su un campione estratto casualmente con reimmissione, prevedendo i valori mancanti sul dataset di test (in sostituzione dell'errore OOB in quanto non gestibile tramite ciclo "for") e calcolandone la media rispetto ai valori precedentemente generati. Vengono quindi calcolati i valori di R^2 e RMSE per ogni iterazione.

```
set.seed(3453)
trainIndex <- createDataPartition(scat_train_test$Species, p = .8,
                                  list = FALSE,
                                  times = 1)
scat_train <- scat_train_test[ trainIndex,]
scat_test <- scat_train_test[-trainIndex,]
```

```
n <- nrow(scat_train)

yhat<-numeric(nrow(scat_test))
yhat_incr<-numeric(nrow(scat_test))
rsq<-NULL
rmse <- NULL
B <- 100
for (b in 1:B) {
  use <- sample(n, size = n, rep = T)
  fit.tree <- rpart(Diameter ~ ., scat_train[use,], cp = 1e-04)
  yhat_incr <- yhat_incr+predict(fit.tree, newdata = scat_test)
  yhat <- (yhat_incr)/b
  rsq<-c(rsq, 1-((sum((scat_test$Diameter-yhat)^2))/
                   (sum((scat_test$Diameter-mean(scat_test$Diameter))^2))))
  rmse <- c(rmse, mean(sqrt((yhat-scat_test$Diameter)^2)))
}

print(paste('RMSE: ', rmse[B], 'R-squared: ', rsq[B]))
```

```
## [1] "RMSE:  2.25608470881227 R-squared:  0.339513091911509"
```

Dai plot degli errori possiamo vedere che l'R^2 e l'RMSE, all'aumentare delle iterazioni di bagging, si attestano su valori stabili.

```r
par(mfrow=c(2,1))

plot(1:B, rsq, xlab = "Bagging iterations", ylab = "R-squared",
     type = "l", main = "Bagging (bootstrap aggregation) - Diameter")

plot(1:B, rmse, xlab = "Bagging iterations", ylab = "RMSE",
     type = "l", main = "Bagging (bootstrap aggregation) - Diameter")
```

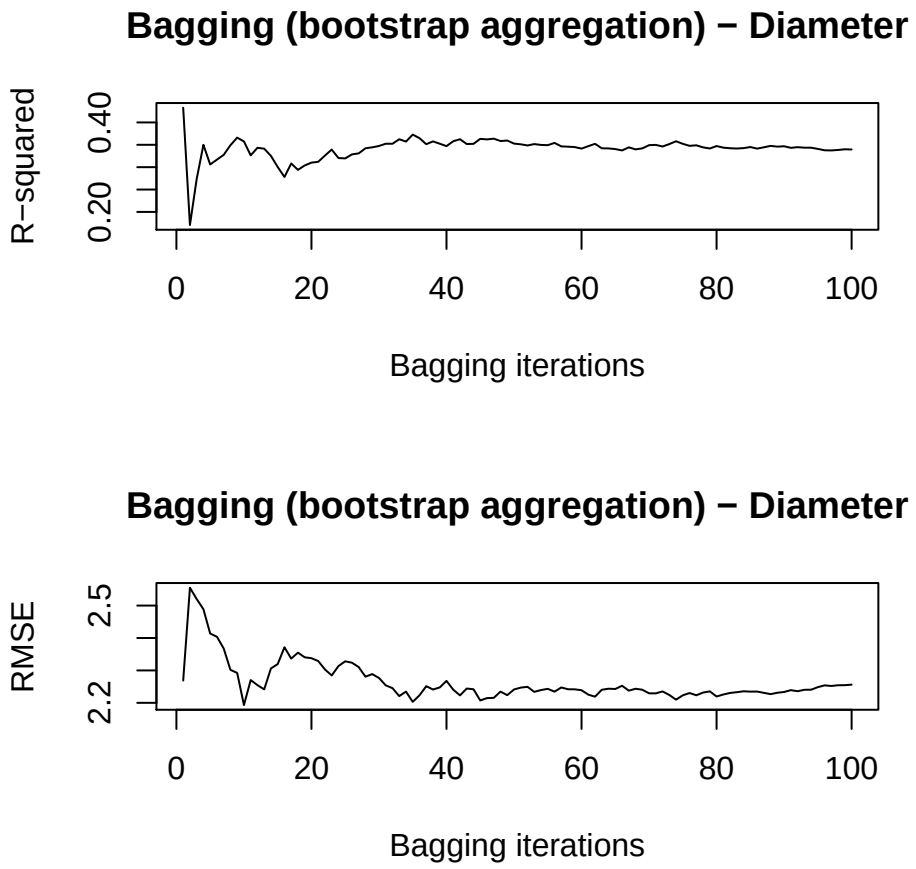

Figure 14: R-squared e RMSE per Diameter

```r
yhat<-numeric(nrow(scat_test))
yhat_incr<-numeric(nrow(scat_test))
rsq<-NULL
rmse <- NULL
B <- 100
for (b in 1:B) {
  use <- sample(n, size = n, rep = T)
  fit.tree <- rpart(Mass ~ ., scat_train[use,], cp = 1e-04)
  yhat_incr <- yhat_incr+predict(fit.tree, newdata = scat_test)
  yhat <- (yhat_incr)/b
  rsq<-c(rsq, 1-((sum((scat_test$Mass-yhat)^2))/
```

```
                    (sum((scat_test$Mass-mean(scat_test$Mass))^2))))
  rmse <- c(rmse, mean(sqrt((yhat-scat_test$Mass)^2)))
}

print(paste('RMSE: ', rmse[B], 'R-squared: ', rsq[B]))
```

```
## [1] "RMSE:  4.53924089080861 R-squared:  0.448062002405012"
```

Passiamo quindi all'allenamento su tutto il dataset (escluse le osservazioni con Diameter e Mass mancanti) e alla definitiva imputazione dei valori previsti.

```
scat_bag_rpart <-
  scat_missing_tree %>%
  mutate(method = "Bagged Tree Rpart",
         diam_imp = Diameter, mass_imp = Mass)

n=nrow(scat[!is.na(scat$Diameter),])

yhat<-numeric(nrow(scat[is.na(scat$Diameter),]))
yhat_incr<-numeric(nrow(scat[is.na(scat$Diameter),]))
B <- 100
for (b in 1:B) {
  use <- sample(n, size = n, rep = T)
  fit.tree <- rpart(Diameter ~ ., scat[!is.na(scat$Diameter),][use,], cp = 1e-04)
  yhat_incr <- yhat_incr+predict(fit.tree, newdata = scat[is.na(scat$Diameter),])
  yhat <- (yhat_incr)/b
}

scat_bag_rpart$diam_imp[is.na(scat_bag_rpart$Diameter)] <- yhat
```

```
n=nrow(scat[!is.na(scat$Mass),])

yhat<-numeric(nrow(scat[is.na(scat$Mass),]))
yhat_incr<-numeric(nrow(scat[is.na(scat$Mass),]))
B <- 100
for (b in 1:B) {
  use <- sample(n, size = n, rep = T)
  fit.tree <- rpart(Mass ~ ., scat[!is.na(scat$Mass),][use,], cp = 1e-04)
  yhat_incr <- yhat_incr+predict(fit.tree, newdata = scat[is.na(scat$Mass),])
  yhat <- (yhat_incr)/b
}

scat_bag_rpart$mass_imp[is.na(scat_bag_rpart$Mass)] <- yhat
```

Il secondo metodo, più diretto, consiste nell'utilizzo della funzione *bagging* che gestisce automaticamente il numero di iterazioni, restituendo inoltre il valore dell'errore OOB.

```
set.seed(3453)
diam_fit <- bagging(Diameter ~  ., data = scat,
                    nbagg = 100, coob = TRUE)
diam_res <- getModelInfo("treebag")[[1]]$oob(diam_fit)

print(diam_res)
```

```
##       RMSE   Rsquared     RMSESD RsquaredSD
##  4.1062754  0.1464446  0.5682396  0.1187055
```

```
set.seed(3453)
mass_fit <- bagging(Mass ~  ., data = scat,
                    nbagg = 100, coob = TRUE)
mass_res <- getModelInfo("treebag")[[1]]$oob(mass_fit)

print(mass_res)
```

```
##       RMSE   Rsquared     RMSESD RsquaredSD
##  8.6818461  0.2726447  1.7455547  0.1478428
```

```
scat_bag_ipred <-
  scat_missing_tree %>%
  mutate(method = "Bagged Tree ipred",
         diam_imp = Diameter, mass_imp = Mass)
scat_bag_ipred$diam_imp[is.na(scat_bag_ipred$Diameter)] <-
  predict(diam_fit, scat[is.na(scat$Diameter),])
scat_bag_ipred$mass_imp[is.na(scat_bag_ipred$Mass)] <-
  predict(mass_fit, scat[is.na(scat$Mass),])
```

Random Forest

Altro modello ensemble è il Random Forest: a differenza del bagging, oltre a variare le osservazioni utilizzate per la previsione dei singoli alberi, sceglie in modo aleatorio m (con $m < p$) predittori tra i p disponibili ogni volta che viene effettuato uno split: tipicamente per la regressione viene scelto $m = \frac{p}{3}$, se si impone invece $m = p$ si ottiene nuovamente il bagging.

La scelta casuale dei predittori fa sì che gli alberi non siano più correlati tra loro, portando ad una diminuzione della varianza:

$$Var(\bar{Z}) = \rho\sigma^2 + \frac{(1-\rho)}{B}\sigma^2$$

Il secondo addendo tende a zero all'aumentare delle iterazioni mentre il primo diminuisce all'aumentare dell'incorrelazione.

Inoltre, come per il bagging, anche Random Forest è in grado di sfruttare le osservazioni OOB per valutare l'errore di previsione su dati mai visti.

Applicazione

Per l'imputazione dei valori di Diameter e Mass viene utilizzata la funzione *randomForest*.

```
scat.rf<-randomForest(Diameter ~ ., data = scat_train_test, ntree=100,
                      mtry=18/3, nodesize=5,
                      importance = T, na.action = na.omit)
scat.rf
```

```
##
## Call:
##  randomForest(formula = Diameter ~ ., data = scat_train_test,      ntree = 100, mtry = 18/3, nodesiz
##                Type of random forest: regression
##                      Number of trees: 100
```

```
## No. of variables tried at each split: 6
##
##           Mean of squared residuals: 11.18675
##                     % Var explained: 26.92
```

```
print(paste('RMSE: ', sqrt(scat.rf$mse[100]), 'R-squared: ', scat.rf$rsq[100]))
```

```
## [1] "RMSE:  3.344659275178 R-squared:  0.269207338888976"
```

```
par(mfrow=c(1,1))
plot(sqrt(scat.rf$mse), xlab = "Trees", ylab = "RMSE",
     type = "l", main = "Random Forest OOB error - Diameter")
```

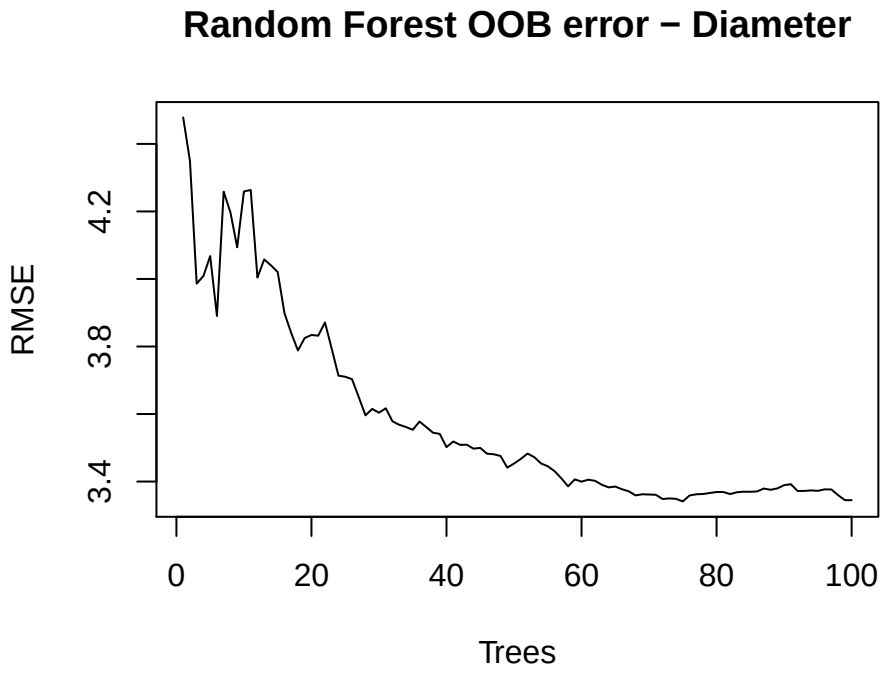

Figure 15: OOB RMSE per Diameter con tutti i predittori

Come detto precedentemente per KNN, anche in questo caso non è possibile considerare come predittori Taper e TI.

```
formula<-Diameter ~ Species+Month+Year+Site+Location+Age+Number+Length+Mass+
  d13C+d15N+CN+ropey+segmented+flat+scrape

scat.rf<-randomForest(formula, data = scat_train_test, ntree=100,
                      mtry=16/3, nodesize=5,
                      importance = T, na.action = na.omit)
scat.rf
```

```
##
```

```
## Call:
##  randomForest(formula = formula, data = scat_train_test, ntree = 100,      mtry = 16/3, nodesize = 5
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 12.34445
##                     % Var explained: 18.91
```

```
print(paste('RMSE: ', sqrt(scat.rf$mse[100]), 'R-squared: ', scat.rf$rsq[100]))
```

```
## [1] "RMSE:  3.51346765946756 R-squared:  0.189115317320918"
```

Si può vedere come l'errore OOB decresce all'aumentare degli alberi considerati, è plausibile quindi che sia presente lo stesso errore su valori mai visti prima.

```
par(mfrow=c(1,1))
plot(sqrt(scat.rf$mse), xlab = "Trees", ylab = "RMSE",
     type = "l", main = "Random Forest OOB error - Diameter")
```

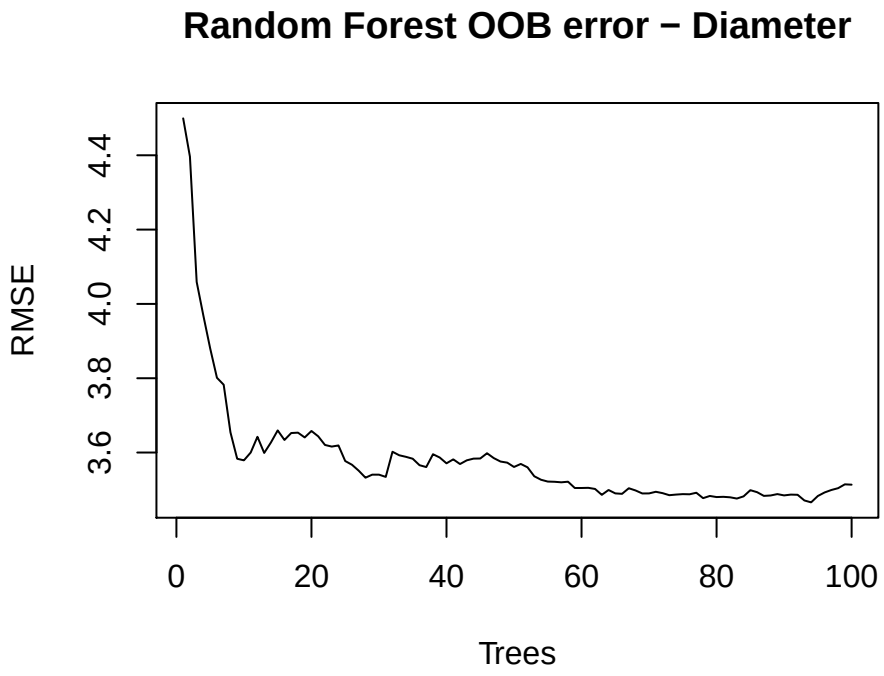

Figure 16: OOB RMSE per Diameter escludendo Taper e TI

```
scat_rf <-
  scat_missing %>%
  mutate(method = "Random Forest",
         diam_imp = Diameter, mass_imp = Mass)

scat_rf$diam_imp[is.na(scat_rf$Diameter)] <-
  predict(scat.rf, newdata = scat[is.na(scat$Diameter),])
```

```
formula<-Mass ~ Month+Year+Site+Location+Age+Number+Length+Diameter+
  d13C+d15N+CN+ropey+segmented+flat+scrape

scat.rf<-randomForest(formula, data = scat_train_test, ntree=100,
                      mtry=16/3, nodesize=5,
                      importance = T, na.action = na.omit)

scat.rf
```

```
##
## Call:
##  randomForest(formula = formula, data = scat_train_test, ntree = 100,      mtry = 16/3, nodesize = 5
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 49.67965
##                     % Var explained: 37.71
```

```
print(paste('RMSE: ', sqrt(scat.rf$mse[100]), 'R-squared: ', scat.rf$rsq[100]))
```

```
## [1] "RMSE:  7.04837902844366 R-squared:  0.377110048028451"
```

```
scat_rf$mass_imp[is.na(scat_rf$Mass)] <-
  predict(scat.rf, newdata = scat[is.na(scat$Mass),])
```

Vengono confrontati infine, nell'immagine 17, i risultati dei tre modelli ensemble utilizzati.

```
imputed_total<-bind_rows(scat_bag_ipred, scat_bag_rpart, scat_rf)

ggplot(imputed_total, aes(col = was_missing)) +
  geom_point(aes(x = diam_imp, y = mass_imp), alpha = .7, cex = 2) +
  geom_rug(data = imputed_total[is.na(imputed_total$Mass),],
           aes(x = Diameter),
           sides = "b",
           lwd = 1) +
  geom_rug(data = imputed_total[is.na(imputed_total$Diameter),],
           aes(y = Mass),
           sides = "l",
           lwd = 1) +
  theme(legend.position = "top") +
  xlab("Diameter") + ylab("Mass") +
  facet_wrap(~method)
```

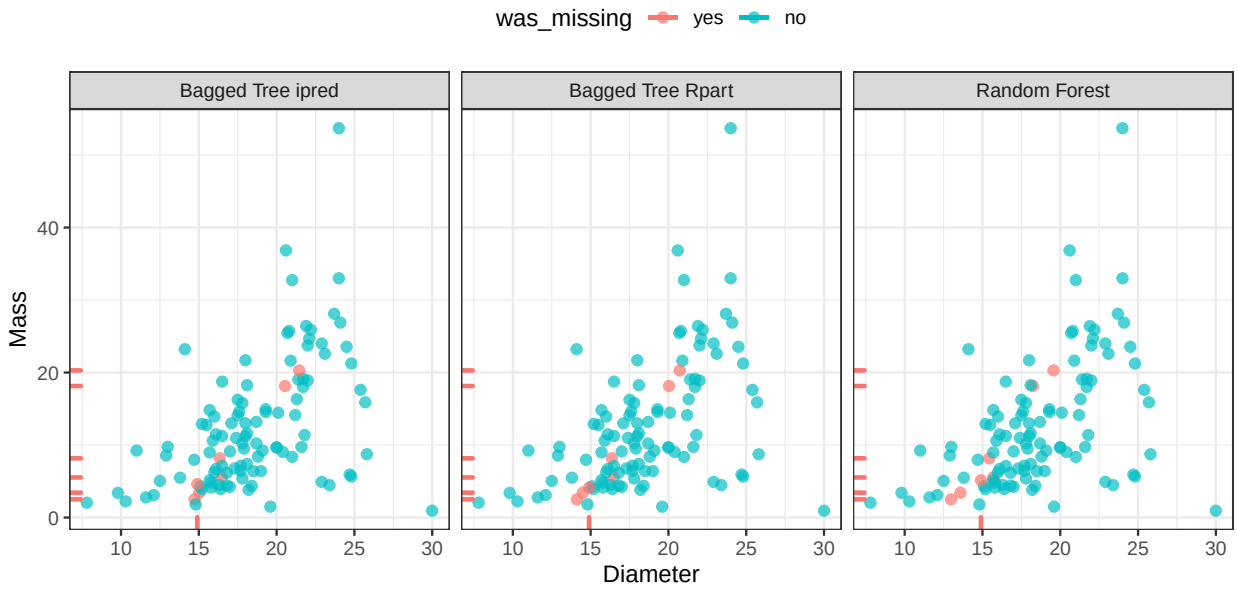

Figure 17: Valori imputati a confronto

I modelli (bagged tree e random forest) restituiscono valori non particolarmente performanti in termini di RMSE e R^2 ma, di contro, hanno risultati ragionevoli rispetto all'essere all'interno del range di valori del training set.

MODELLI LINEARI Un altro metodo di imputazione è il modello lineare. Questo può essere utilizzato quando un predittore senza missing mostra una relazione lineare con uno avente valori mancanti.
Se l'output è numerico la regressione lineare può essere un'ottima via, considerando la velocità computazionale che la caratterizza, mentre qualora l'output fosse categorico si può ricorrere alla regressione logistica.

La figura 18a relaziona le corse dei treni per la stazione di Halsted nel giorno corrente e dopo 14 giorni. A fini accademici, sono stati rimossi alcuni valori dal giorno corrente e sono stati utilizzati quelli di due settimane dopo per costruire un modello di regressione lineare. La scelta della linearità deriva dalla relazione che caratterizza le due variabili nel grafico. Nella figura 18b viene mostrato il risultato dell'imputazione a confronto con il valore originale. In quest'ultimo i valori previsti sono molto simili a quelli veri fatta eccezione per un singolo punto che corrisponde ad un giorno feriale (probabilmente l'aggiunta di una variabile dummy avrebbe migliorato la previsione).

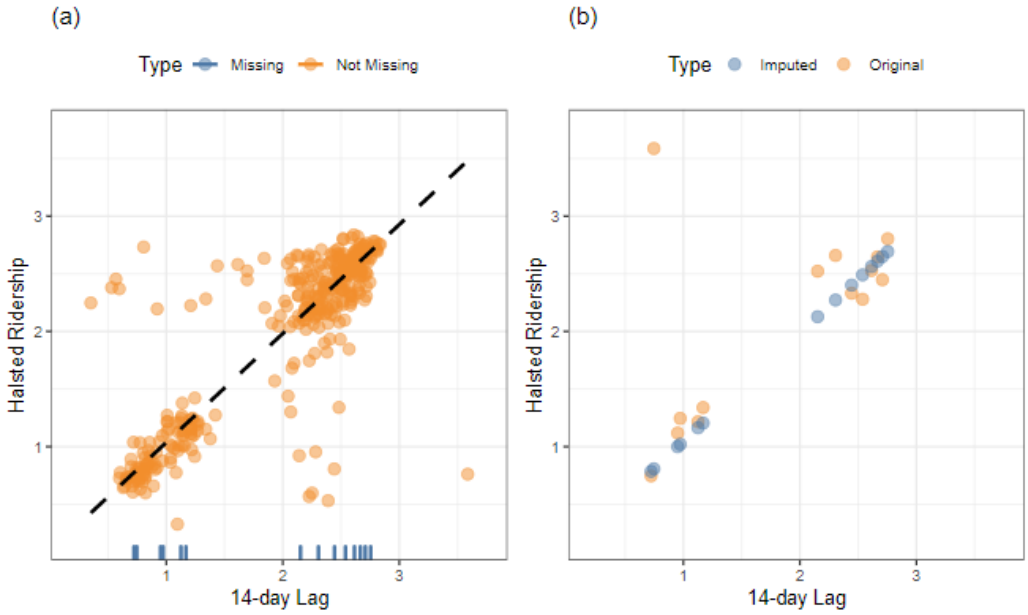

Figure 18: Applicazione del modello lineare

4. Casi Particolari

In certe situazioni non sono sufficienti gli approcci illustrati precedentemente poiché i dati oltre ad essere mancanti possono essere incompleti rispetto a certe informazioni relative al dominio di appartenenza del dataset. Ad esempio, quando si considerano eventi temporali, bisogna tener conto del tempo minimo tra un evento e l'altro.

Questi dati vengono detti censurati, in particolare censurati a destra se non conosciamo il valore finale e censurati a sinistra se non conosciamo il valore iniziale (ad esempio nelle misure di laboratorio in cui gli strumenti hanno una certa sensibilità nella rilevazione dei dati). Se chiamiamo X questo valore limite, all'interno del dataset i dati vengono catalogati come "$<$ X".

Se vogliamo considerare questi valori all'interno di un modello predittivo, è pratica comune quella di utilizzare il limite inferiore di X. Tuttavia, questo metodo è applicabile senza perdita di validità del modello all'interno di tree mentre per altri algoritmi influisce negativamente sulla variabilità del dato, sottostimando il valore reale. In questo caso, invece di utilizzare come unico valore il limite inferiore di X, si utilizza una distribuzione casuale tra zero ed X (o non casuale se si conoscono ulteriori informazioni relativamente all'attributo in esame).

Infine, se i dati hanno una forte componente temporale, è possibile utilizzare una media mobile per mantenere gli effetti di quest'ultima nel modello.

5. Conclusione

La gestione dei missing value è un punto fondamentale da svolgere in qualsiasi analisi. La presenza di valori mancanti rappresenta un problema sia da un punto di vista pratico, perchè alcuni modelli non possono essere applicati se l'algoritmo rileva l'assenza di informazione, sia da un punto di vista teorico, perché il modo in cui vengono trattati i dati mancanti può influire molto sui risultati dell'analisi. Per questa ragione dopo la scelta del metodo per il trattamento è molto importante verificarne gli effetti. Quando possibile, in genere, questa procedura si basa sul confronto delle analisi ottenute rispettivamente sui casi completi e sui casi incompleti o trattati, quindi paragonando i vari metodi di imputazione o encoding utilizzati.

Nell'eventualità vi siano forti differenze il metodo scelto potrebbe essere poco opportuno e condurre a risultati inaffidabili. In tal caso è necessario analizzare i motivi che hanno portato a tali differenze e valutare quale risultato si approssima di più alla realtà oppure riportare entrambi i risultati. La soluzione migliore è quella di avere la possibilità di un confronto con un esperto del settore (o ad un eventuale proprietario del dato) in modo da riuscire a dare l'interpretazione migliore possibile al problema.