Progetto statistica 2021

Alsina Gabriel Riccardo, Carboni Carlotta, Sancisi Federico

Gruppo Q

Contents

1 Descrizione dataset **2**

2 Caricare il dataset **2**

3 Pre-processing **2**

4 Visulizzazione **3**
4.1 Matrice di correlazione . . . 3
4.2 Pazienti per range di ragggio . . . 4
4.3 Barplot benigni maligni per range di raggio . . . 5
4.4 Correlazione raggio e punti concavi . . . 6
4.5 Boxplot simmetrie . . . 7
4.6 Correlazione compattezza e simmetria . . . 8
4.7 Box plot Fractal dimension . . . 9

5 Machine learning **10**
5.1 Splitting . . . 10
5.2 Funzioni che useremo per valutare il modello . . . 10
5.3 Addestriamo il modello . . . 10

6 Hyperparameter Tuning **13**
6.1 Lineare . . . 13
6.2 Polinomiale . . . 13
6.3 Radiale . . . 14
6.4 Spaghetti Plot . . . 15

7 Valutazione performance **19**

8 Interpretazione probabilistica **21**
8.1 Curva di ROC . . . 22

9 Studio statistico dei risultati **23**
9.1 SRS(k) . . . 23
9.2 Statistica descrittiva . . . 24
9.3 Statistica inferenziale . . . 28

10 Features Selection **34**

11 Codice completo **36**

1 Descrizione dataset

Il dataframe scaricato da https://www.kaggle.com/ è un dataset contenente 33 colonne e 569 osservazioni.
Il file "stat_cancer" contiene informazioni che descrivono le caratteristiche di una massa mammaria, calcolate da un'immagine digitalizzata di un ago aspirato (FNA), ovvero descrivono le caratteristiche dei nuclei cellulari presenti nelle 569 immagini studiate.
Per ogni cellula vengono presi in cosiderazione questi 10 valori:
1) **radius** :media delle distanze dal centro ai punti sul perimetro
2) **texture**:deviazione standard dei valori della scala di grigi
3) **perimeter**
4) **area**
5) **smoothness**:variazione locale nelle lunghezze del raggio
6) **compactness**:perimetro^2 / area - 1.0
7) **concavity**:gravità delle porzioni concave del contorno
8) **concave.points**:numero di porzioni concave del contorno
9) **symmetry**
10) **fractal_dimension**:"approssimazione della costa"

Le altre 23 colonne sono così composte: 1 colonna per il numero identificato, 1 colonna per la diagnosi, e una colonna "X"completamente nulla; in questo modo ci rimangono altre 20 colonne strettamente collegate ai 10 principali precedenti, infatti mettono in luce non la media dei valori ottenuta per ogni cellula, ma il valore peggiore e l'errore standard.
Ora che conosciamo il dataset possiamo iniziare a lavorarci.

2 Caricare il dataset

codice per importare un dataframe:

```
ds<-read.csv("data.csv")
```

3 Pre-processing

Succesivamente abbiamo ripulito il dataset da possibili valori nulli, ma prima, essendo la colonna "X" interamente nulla l'abbiamo eliminata e successivamente abbiamo verificato che la variabile "diagnosis" fosse categorica e non numerica:

```
ds$X<-NULL
ds<-na.omit(ds)
levels(ds$diagnosis)
```

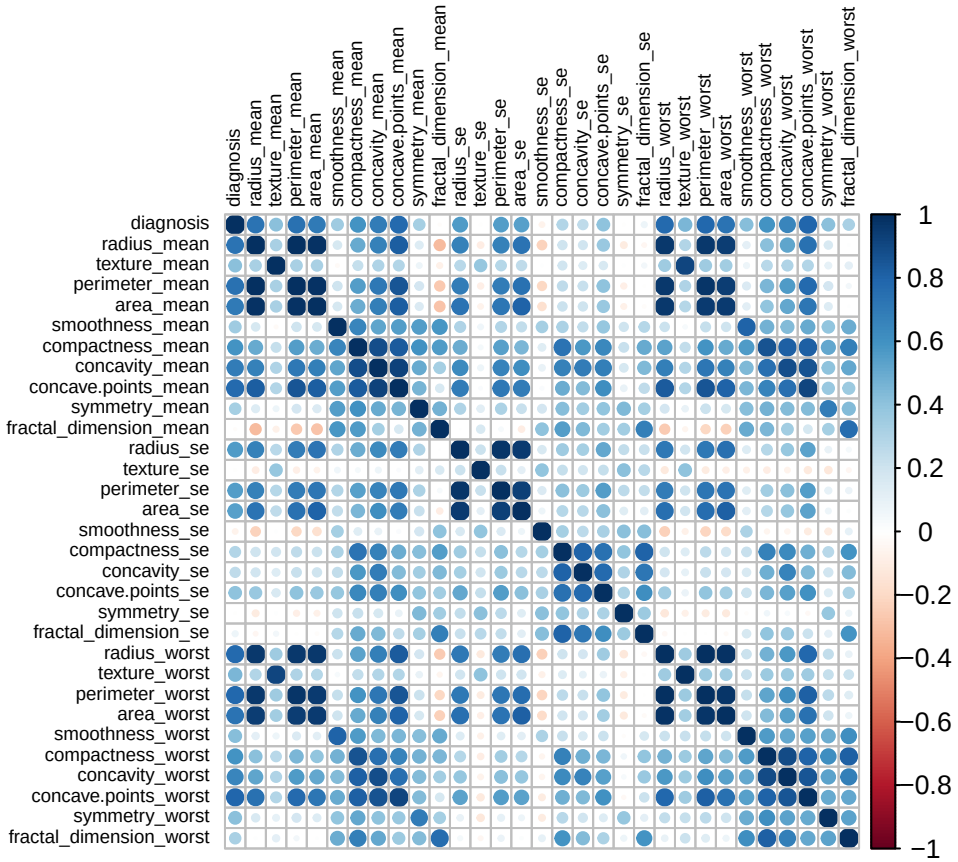
```
## [1] "B" "M"
```

Abbiamo copiato il dataframe pulito su un'altra variabile per poter mantenere invariato il dataset ripulito:

```
dsBU<-ds
```

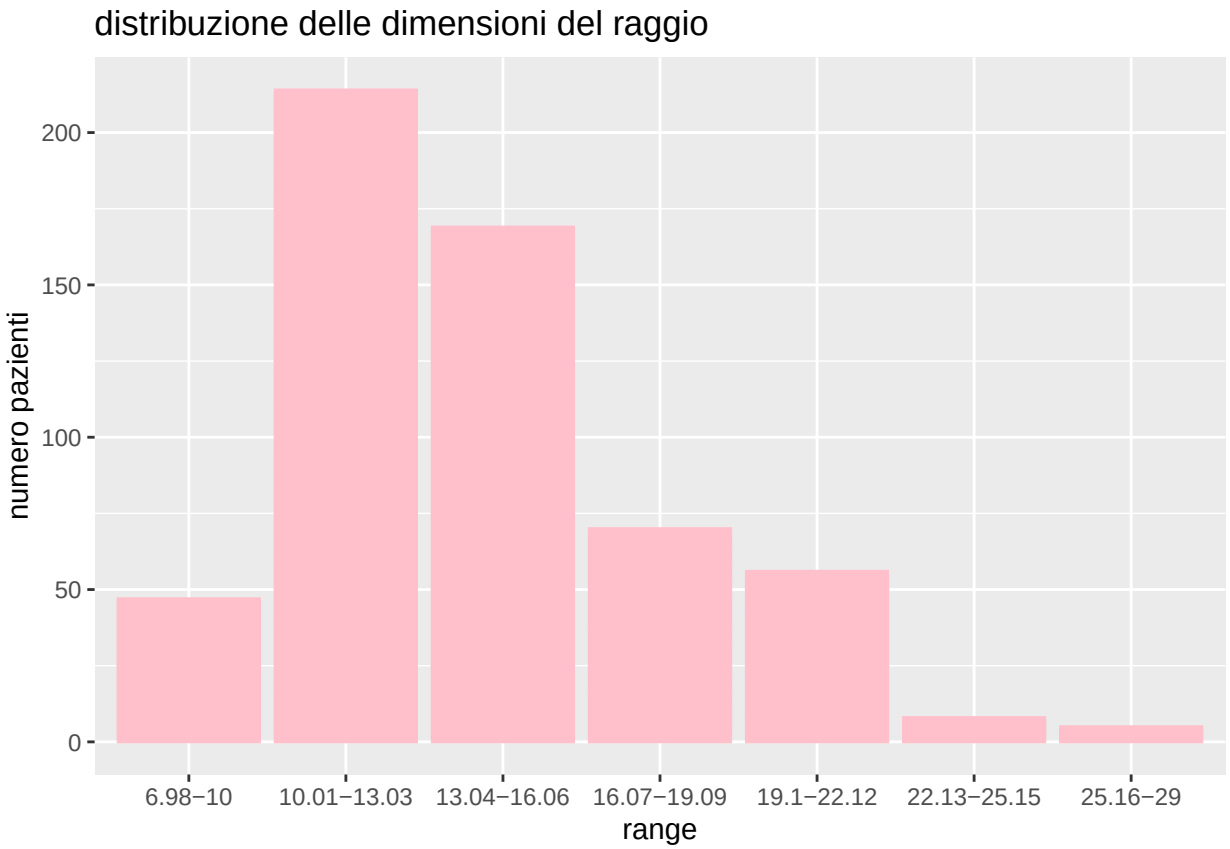
4 Visulizzazione

4.1 Matrice di correlazione



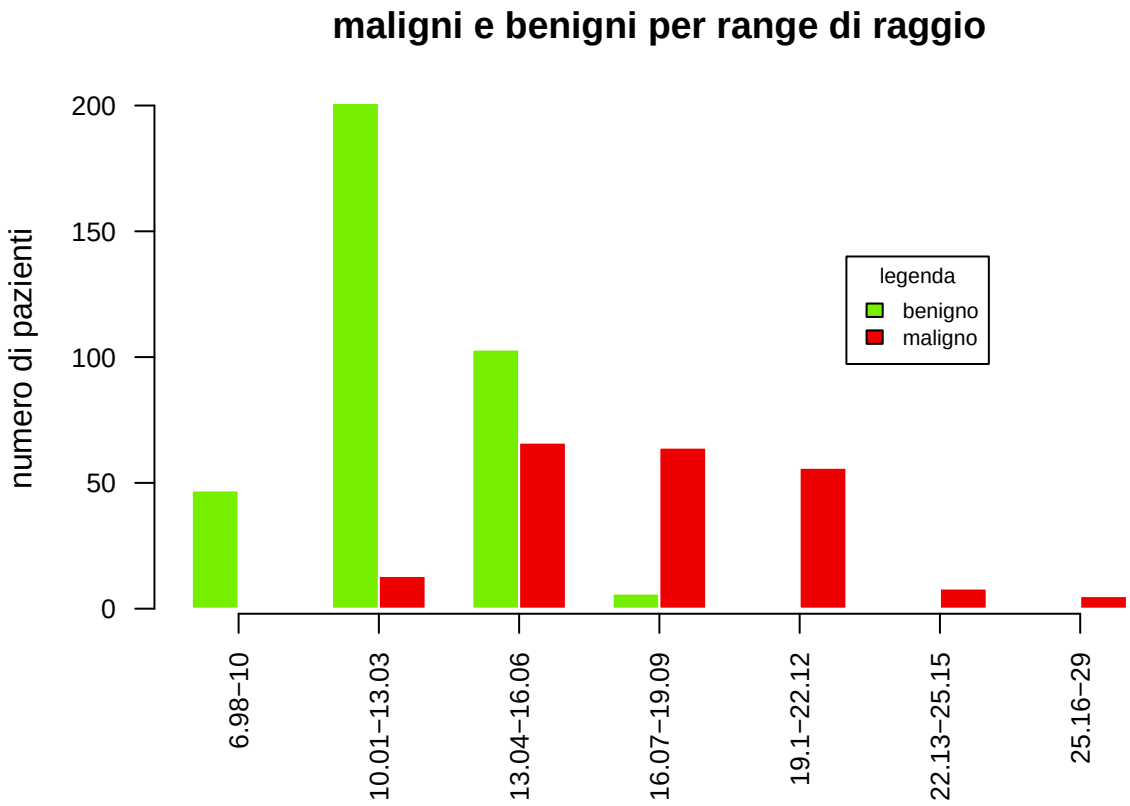
Questo grafico ci mostra come i dati del dataset siano correlati l'uno con l'altro. Per alcuni dati era ovvia la correlazione, pensiamo semplicemente alla misurazione del raggio e dell'area.
Questo grafico è interessante perché ci mostra in maniera molto chiara e intuitiva quali variabili influenzino maggiormente la diagnosi finale: infatti vediamo che il raggio, di conseguenza l'area e il perimetro, ma anche la compattezza e la concavità abbiano un'influenza molto forte; mentre la levigatezza, la simmetria, la tessitura e la fractal dimension non influenzino per niente la nostra diagnosi.
Quindi già da questo grafico possiamo sapere su quali dati è più importante condurre un'analisi.

4.2 Pazienti per range di ragggio



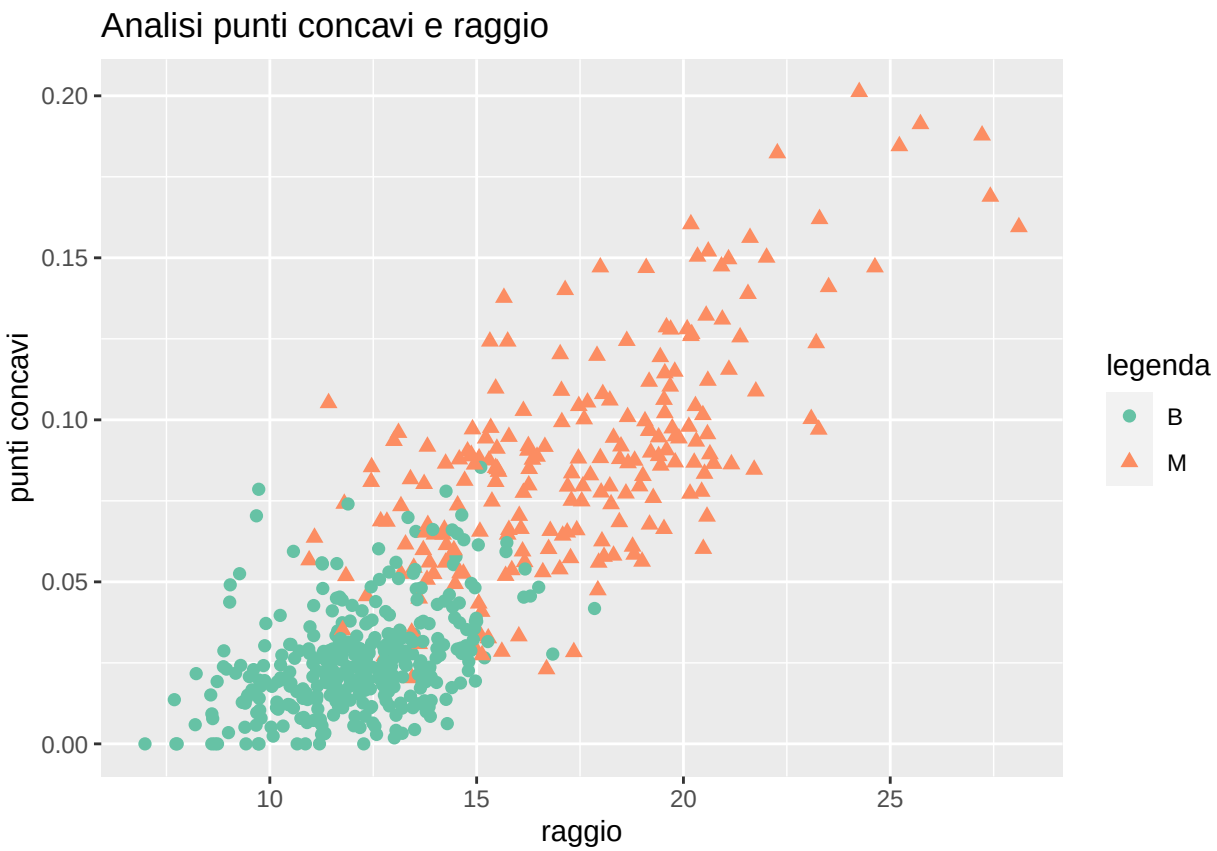
Questo grafico ci mostra per ogni range di dimensione del raggio quante cellulle sono state analizzate, la dimensione più frequente è quella tra 10.01 e 13.03.
Data questa osservazione sarebbe interessante andare a vedere quanti per ogni range siano maligni e benigni.

4.3 Barplot benigni maligni per range di raggio



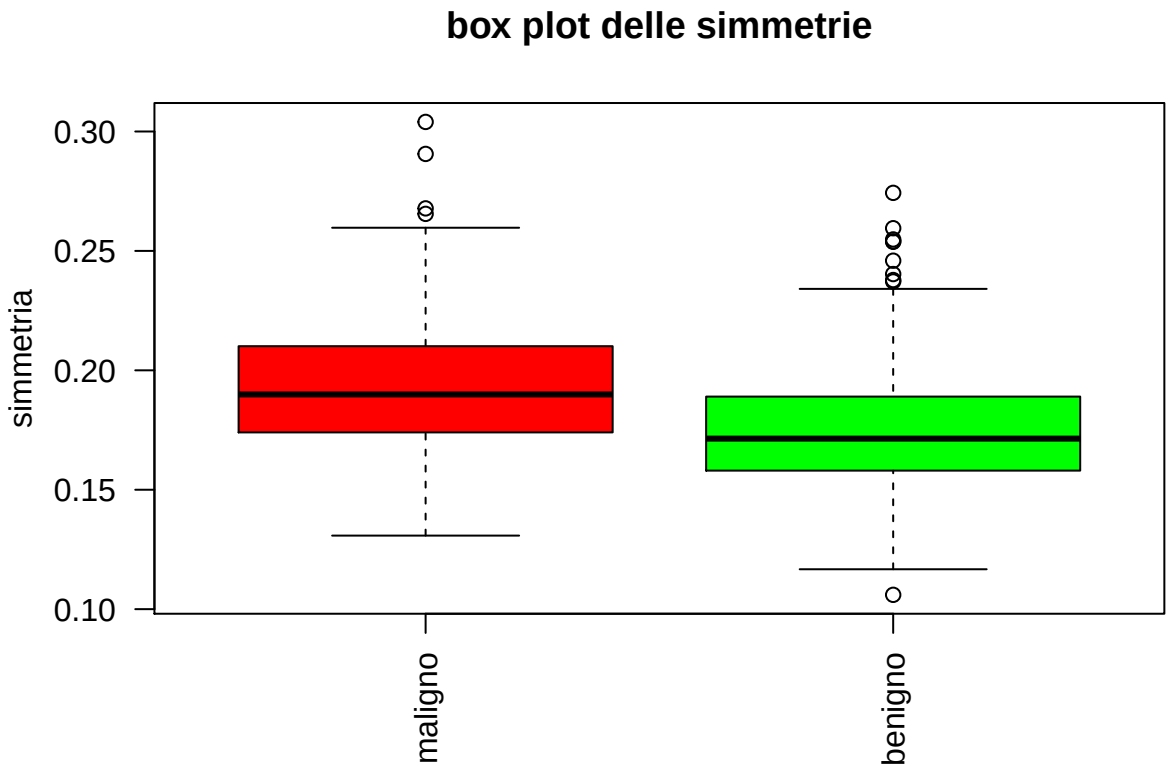
Grazie a questo grafico notiamo, perfortuna, che le cellule nel range 10.01-13.03, ovvero quello con più casi, nella maggior parte delle volte si rivelano cellule benigne. Il grafico ci mostra anche che più il raggio della cellula è piccolo più è probabile che il tumore sia benigno e che più il raggio aumenta più è probabile sia maligno, ma un'altra osservazione è che più il raggio aumenta, di conseguenza anche la cellula, minori sono i casi rilevati.
Possiamo dire, osservando il grafico che sopra al 13 iniziano a crescere le probabilità che sia maligno anche solo basandoci su questo dato e che il range più critico è quello che va dal 13.04 al 16.06; in questo range sarà necessario considerare anche altri dati.

4.4 Correlazione raggio e punti concavi



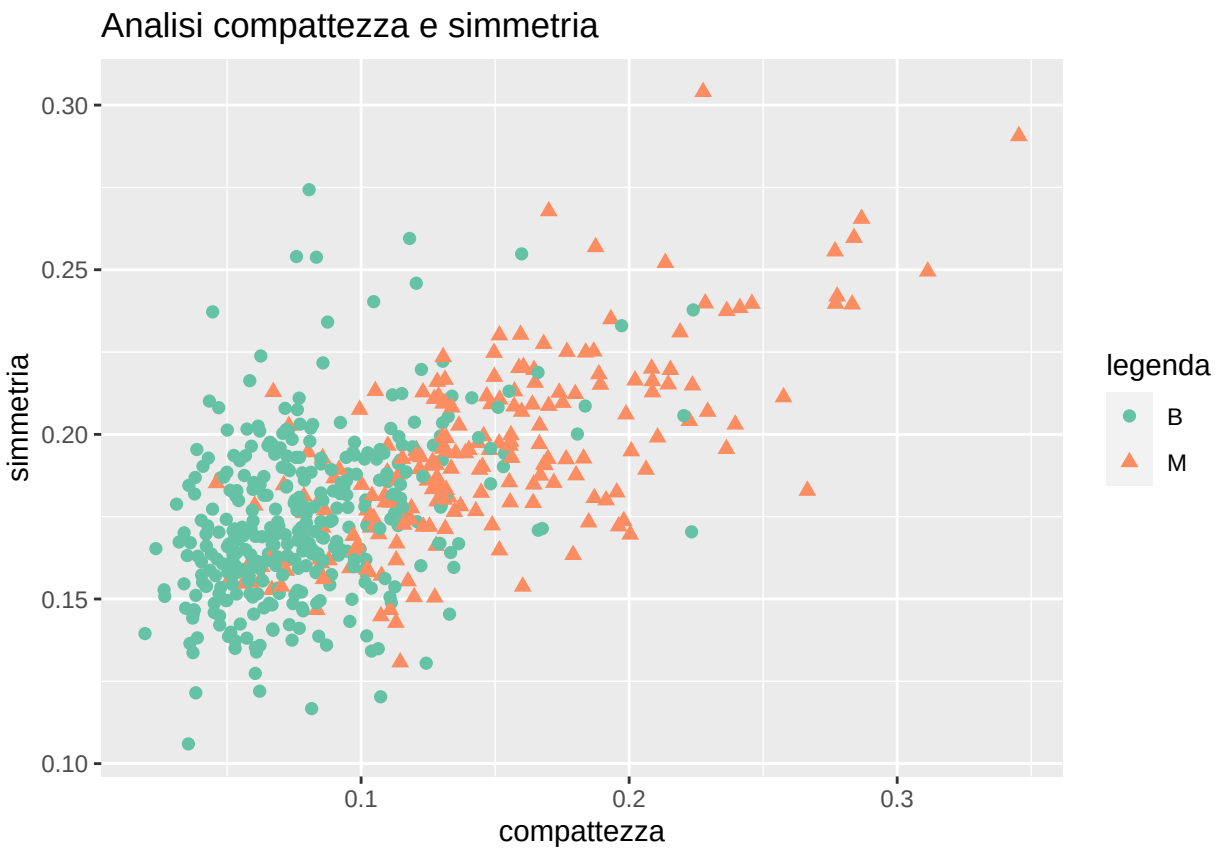
In questo grafico siamo andati ad analizzare il comportamento del raggio e dei punti concavi delle cellule, perché la diagnosi dipende soprattutto dal raggio e dalla quantità dei punti concavi, ma lo stesso raggio dipende dai punti concavi.
Come risultato abbiamo ottenuto quello che ci aspettavamo: ovvero che al crescere del raggio aumentano le porzioni concave e che all'aumentare di entrambi aumenta la probalità che la cellula sia maligna.

4.5 Boxplot simmetrie



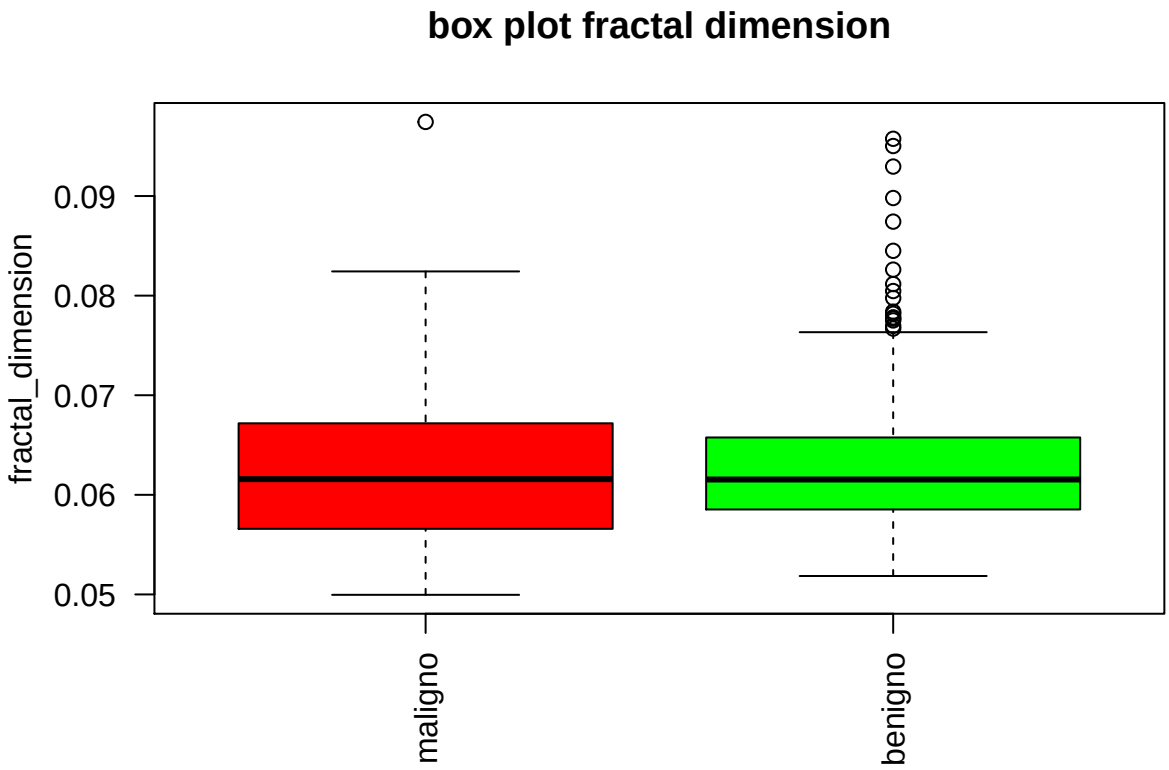
Nonostante con il primo grafico abbiamo osservato che la diagnosi non dipende in modo stretto dalla simmetria, quest'ultima incide invece sulla levigatezza, sulla compattezza e sulla concavità che a loro volta andavano a influire sulla diagnosi. Quindi può invece risultare interessante andare ad analizzare anche la simmetria.
Questo grafico ci mostra le distribuzioni delle simmetrie delle cellule analizzate differenziandole, come nei grafici precedenti in benigni e maligni. Entrambe le categorie hanno una distrubuzione quasi simmetrica, trovandosi la mediana in tutti e due i box quasi al centro.
Avendo le due scatole una di fianco all'altra possiamo notare che per quasi tutti i risultati la cellula potrebbe essere sia benigna che maligna: in questi casi si avrà bisogno di altri dati per identificare la natura della cellula. In ogni caso possiamo notare che più il valore della simmetria aumenta più è probabile sia maligna, ma i valori sono talmente tanto simili da non fornire alcuna sicurezza.
Può diventare interessante mostrare l'andamento della simmetria in correlazione alla compattezza dato che abbiamo detto si influenzino a vicenda.

4.6 Correlazione compattezza e simmetria



A differenza del grafico del raggio in correlazione ai punti concavi sta volta simmetria e compattezza non vanno a pari passo. Potevamo già sospettarlo dato che la diagnosi dipende più dalla compattezza che dalla simmetria, infatti come sottolinea questo grafico aumentando la simmetria non si sa se sarà benigno o maligno, mentre aumentando la compattezza sia ha quasi la certezza sarà maligno, però per valori più bassi con solo questi due dati non possiamo trarre conclusioni.

4.7 Box plot Fractal dimension



In questo grafico abbiamo voluto prendere in considerazione un dato che non influenzasse l'esito finale, proprio per mostrare anche con altri grafici che effettivamente solo questo dato non ci permette di identificare la natura della cellula.
Usando questo boxplot possiamo osservare che le distribuzioni della fractal dimension non sono molto differenti, possiamo solamente osservare che per valori più alti è più probabile sia maligno, ma non possiamo dire il contrario, ovvero che per valori più piccoli sia più probabile sia benigno, abbiamo quindi confermato che l'esito non dipenda da questo dato.

5 Machine learning

5.1 Splitting

Per poter analizzare il dataset lo dividiamo in tre parti: train 70%, validation 15% e test 15%(che rappresenterà i nostri dati futuri).

Noi non vogliamo che i primi 399 dati finiscano nel train set, perchè i dati potrebbero essere ordinati in qualche maniera e quindi rendere la nostra analisi non corretta. Per questo motivo mescoliamo i dati in modo da estrargli in maniera randomica grazie alla funzione *sample*:

```
train.sample<-sample(N, N.train)
ds.train<-ds[train.sample, ]
ds.test<-ds[-train.sample, ]

val.sample<-sample(N.test+N.val, N.val)
ds.val<-ds.test[val.sample, ]
ds.test<-ds.test[-val.sample, ]
```

Da questo momento in poi useremo ds.train, che per semplicità di lettura lo salviamo in una nuova variabile ds.

5.2 Funzioni che useremo per valutare il modello

Questa è la funzione che andremo ad utilizzare per valutare il valore dell'errore, quindi più sarà basso, più il modello sarà accurato:

```
MR<- function(y.pred, y.true){
  res<-mean(y.pred != y.true)
  return(res)
}
```

Questa è la funzione che andremo ad utilizzare per valutare l'accuratezza del modello, quindi più sarà alta, più il modello sarà accurato e si calcola sottraendo a 1 l'errore calcolato precedentemente:

```
Acc<-function(y.pred, y.true){
  res<-1-mean(y.pred!=y.true)
  return(res)
}
```

5.3 Addestriamo il modello

Sappiamo esserci 3 tipi di kernel: lineare, polinomiale e radiale; noi dobbiamo capire quale sia il miglior kernel per il nostro dataset. Per trovarlo abbiamo scelto di procedere con un approccio "Grid search" dove andiamo a provare tutte le combinazioni possibili con un costo e un grado che vanno dall'uno al dieci.

5.3.1 Lineare

```
MR.lin.total.Train<-1:10
for(c in 1:10){
  model.SVC<-svm(diagnosis~ ., ds, kernel="linear", cost=c)
  y.pred<-predict(model.SVC, ds)
  MR.lin<-MR(y.pred, ds$diagnosis)
  MR.lin.total.Train[c]<-MR.lin
}
MR.lin.total.Train
```

```
##  [1] 0.015037594 0.010025063 0.010025063 0.007518797 0.007518797 0.007518797
##  [7] 0.007518797 0.007518797 0.005012531 0.005012531
```

5.3.2 Polinomiale

```
MR.poly.total.Train<-matrix(ncol=10, nrow=10)
Error.poly.train<-c(0)
for(c in 1:10){
  for(d in 1:10){
    model.SVM<-svm(diagnosis~., ds, kernel="polynomial", cost=c, degree=d)
    y.pred<-predict(model.SVM, ds)
    MR.poly<-MR(y.pred, ds$diagnosis)
    MR.poly.total.Train[c,d]<-MR.poly
    Error.poly.train<-append(Error.poly.train, MR.poly)

  }
}
Error.poly.train<-Error.poly.train[-1]
MR.poly.total.Train
```

```
##             [,1]       [,2]       [,3]      [,4]      [,5]      [,6]      [,7]
##  [1,] 0.02506266 0.16541353 0.10025063 0.1729323 0.1553885 0.1829574 0.1754386
##  [2,] 0.02005013 0.15037594 0.06766917 0.1553885 0.1353383 0.1679198 0.1629073
##  [3,] 0.02005013 0.13283208 0.05263158 0.1428571 0.1278195 0.1578947 0.1578947
##  [4,] 0.02005013 0.12280702 0.04260652 0.1328321 0.1177945 0.1578947 0.1553885
##  [5,] 0.02005013 0.11779449 0.04260652 0.1303258 0.1127820 0.1553885 0.1478697
##  [6,] 0.02005013 0.10526316 0.03759398 0.1203008 0.1127820 0.1503759 0.1428571
##  [7,] 0.02005013 0.10025063 0.03007519 0.1152882 0.1077694 0.1428571 0.1378446
##  [8,] 0.02005013 0.09774436 0.03007519 0.1102757 0.1077694 0.1303258 0.1328321
##  [9,] 0.02005013 0.09523810 0.02756892 0.1102757 0.1027569 0.1278195 0.1303258
## [10,] 0.02005013 0.09022556 0.02756892 0.1077694 0.1002506 0.1203008 0.1278195
##            [,8]      [,9]     [,10]
##  [1,] 0.1904762 0.1879699 0.1954887
##  [2,] 0.1754386 0.1779449 0.1854637
##  [3,] 0.1679198 0.1704261 0.1779449
##  [4,] 0.1654135 0.1654135 0.1754386
##  [5,] 0.1654135 0.1654135 0.1679198
##  [6,] 0.1578947 0.1629073 0.1679198
##  [7,] 0.1578947 0.1578947 0.1654135
##  [8,] 0.1553885 0.1528822 0.1654135
##  [9,] 0.1528822 0.1528822 0.1629073
## [10,] 0.1503759 0.1528822 0.1604010
```

5.3.3 Radiale

```
MR.rad.total.Train<-matrix(ncol=10, nrow=10)
Error.rad.train<-c(0)
for(c in 1:10){
  for(g in 1:10){
    model.SVM<-svm(diagnosis ~ ., ds, kernel="radial", cost=c, gamma=g/100)
    y.pred<-predict(model.SVM, ds)
    MR.rad<-MR(y.pred, ds$diagnosis)
    MR.rad.total.Train[c,g]<-MR.rad
    Error.rad.train<-append(Error.rad.train, MR.rad)
  }
}
Error.rad.train<-Error.rad.train[-1]
MR.rad.total.Train
```

```
##             [,1]       [,2]       [,3]       [,4]        [,5]        [,6]
##  [1,] 0.02005013 0.02005013 0.01503759 0.01253133 0.012531328 0.015037594
##  [2,] 0.02005013 0.01754386 0.01503759 0.01503759 0.012531328 0.012531328
##  [3,] 0.02005013 0.01503759 0.01503759 0.01253133 0.012531328 0.012531328
##  [4,] 0.02005013 0.01503759 0.01253133 0.01253133 0.012531328 0.012531328
##  [5,] 0.02005013 0.01503759 0.01253133 0.01253133 0.012531328 0.010025063
##  [6,] 0.01754386 0.01503759 0.01253133 0.01253133 0.012531328 0.005012531
##  [7,] 0.01503759 0.01253133 0.01253133 0.01253133 0.007518797 0.005012531
##  [8,] 0.01503759 0.01253133 0.01253133 0.01253133 0.007518797 0.005012531
##  [9,] 0.01503759 0.01253133 0.01253133 0.01253133 0.005012531 0.005012531
## [10,] 0.01503759 0.01253133 0.01253133 0.01002506 0.005012531 0.002506266
##              [,7]        [,8]        [,9]       [,10]
##  [1,] 0.015037594 0.015037594 0.015037594 0.012531328
##  [2,] 0.012531328 0.012531328 0.010025063 0.007518797
##  [3,] 0.012531328 0.007518797 0.007518797 0.005012531
##  [4,] 0.007518797 0.005012531 0.005012531 0.005012531
##  [5,] 0.005012531 0.005012531 0.005012531 0.002506266
##  [6,] 0.005012531 0.005012531 0.002506266 0.002506266
##  [7,] 0.005012531 0.002506266 0.002506266 0.002506266
##  [8,] 0.002506266 0.002506266 0.002506266 0.000000000
##  [9,] 0.002506266 0.002506266 0.002506266 0.000000000
## [10,] 0.002506266 0.002506266 0.000000000 0.000000000
```

6 Hyperparameter Tuning

Per valutare la performance del modello applichiamo i nostri algoritmi non più sui dati di training, ma su validation:

6.1 Lineare

```
MR.lin.total.Val<-1:10
for(c in 1:10){
  model.SVC<-svm(diagnosis ~ ., ds, kernel="linear", cost=c)
  y.pred<-predict(model.SVC, ds.val)
  MR.lin<-MR(y.pred, ds.val$diagnosis)
  MR.lin.total.Val[c]<-MR.lin
}
MR.lin.total.Val
```

```
##  [1] 0.04705882 0.04705882 0.04705882 0.04705882 0.04705882 0.04705882
##  [7] 0.04705882 0.04705882 0.04705882 0.04705882
```

6.2 Polinomiale

```
MR.poly.total.Val<-matrix(ncol=10, nrow=10)
Error.poly.val<-c(0)
for(c in 1:10){
  for(d in 1:10){
    model.SVM<-svm(diagnosis~., ds, kernel="polynomial", cost=c, degree=d)
    y.pred<-predict(model.SVM, ds.val)
    MR.poly<-MR(y.pred, ds.val$diagnosis)
    MR.poly.total.Val[c,d]<-MR.poly
    Error.poly.val<-append(Error.poly.val, MR.poly)
  }
}
Error.poly.val<-Error.poly.val[-1]
MR.poly.total.Val
```

```
##             [,1]      [,2]       [,3]      [,4]       [,5]      [,6]      [,7]
##  [1,] 0.03529412 0.1647059 0.09411765 0.2000000 0.18823529 0.2352941 0.2117647
##  [2,] 0.02352941 0.1294118 0.09411765 0.1882353 0.15294118 0.2117647 0.2000000
##  [3,] 0.02352941 0.1176471 0.08235294 0.1529412 0.12941176 0.1882353 0.1882353
##  [4,] 0.02352941 0.1294118 0.08235294 0.1529412 0.12941176 0.1882353 0.1882353
##  [5,] 0.03529412 0.1294118 0.08235294 0.1411765 0.12941176 0.2000000 0.1882353
##  [6,] 0.03529412 0.1294118 0.08235294 0.1294118 0.12941176 0.2000000 0.1764706
##  [7,] 0.02352941 0.1176471 0.07058824 0.1411765 0.11764706 0.1882353 0.1764706
##  [8,] 0.02352941 0.1176471 0.05882353 0.1294118 0.11764706 0.1882353 0.1647059
##  [9,] 0.02352941 0.1176471 0.05882353 0.1411765 0.10588235 0.1764706 0.1647059
## [10,] 0.02352941 0.1176471 0.05882353 0.1411765 0.09411765 0.1764706 0.1529412
##            [,8]      [,9]     [,10]
##  [1,] 0.2352941 0.2352941 0.2352941
##  [2,] 0.2470588 0.2235294 0.2470588
##  [3,] 0.2352941 0.2235294 0.2588235
##  [4,] 0.2352941 0.2235294 0.2470588
##  [5,] 0.2235294 0.2117647 0.2470588
##  [6,] 0.2117647 0.2117647 0.2470588
##  [7,] 0.2117647 0.2000000 0.2352941
```

```
##  [8,] 0.2117647 0.2000000 0.2352941
##  [9,] 0.2000000 0.2000000 0.2352941
## [10,] 0.1882353 0.2000000 0.2352941
```

6.3 Radiale

```
MR.rad.total.Val<-matrix(ncol=10, nrow=10)
Error.rad.val<-c(0)
for(c in 1:10){
  for(g in 1:10){
    model.SVM<-svm(diagnosis~., ds, kernel="radial", cost=c, gamma=g/100)
    y.pred<-predict(model.SVM, ds.val)
    MR.rad<-MR(y.pred, ds.val$diagnosis)
    MR.rad.total.Val[c,g]<-MR.rad
    Error.rad.val<-append(Error.rad.val, MR.rad)
  }
}
Error.rad.val<-Error.rad.val[-1]
MR.rad.total.Val
```

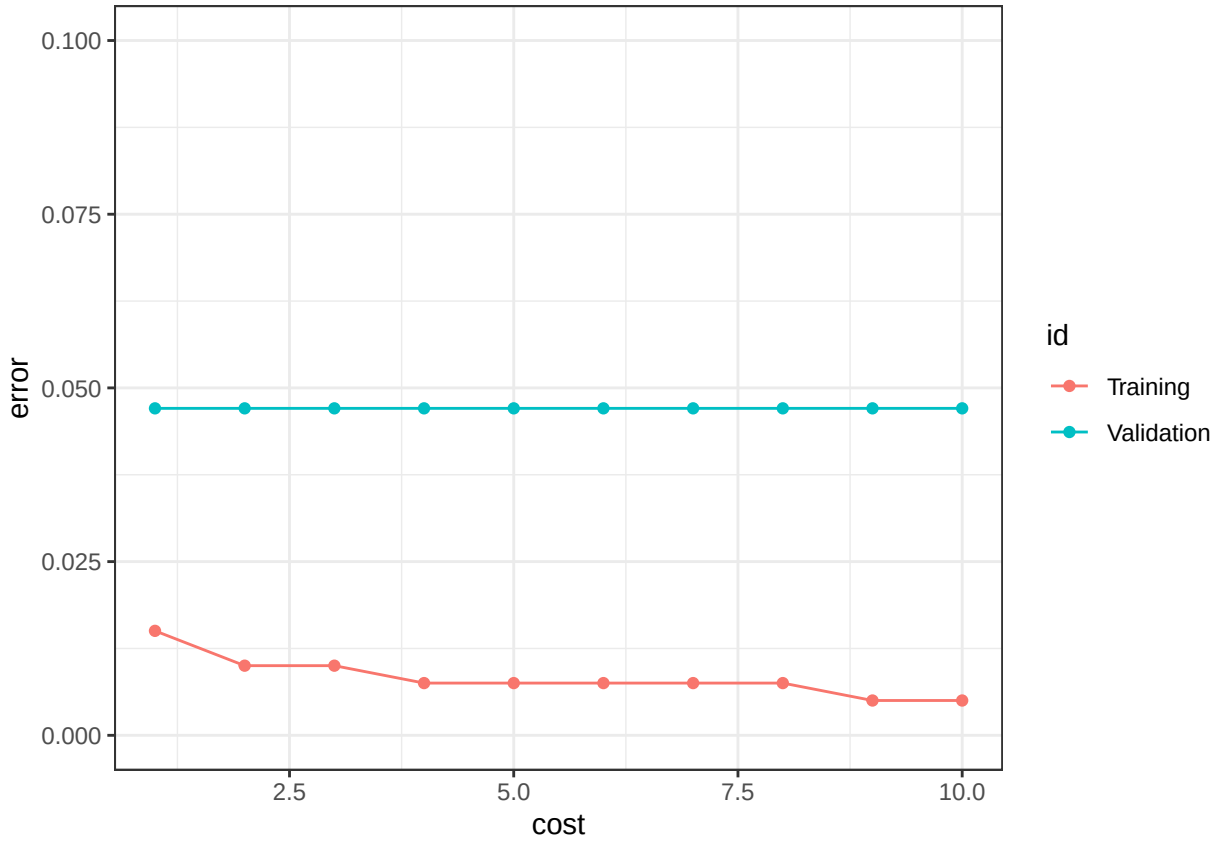
```
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
##  [1,] 0.03529412 0.03529412 0.03529412 0.02352941 0.02352941 0.02352941
##  [2,] 0.04705882 0.03529412 0.02352941 0.01176471 0.01176471 0.01176471
##  [3,] 0.04705882 0.02352941 0.02352941 0.01176471 0.01176471 0.01176471
##  [4,] 0.03529412 0.02352941 0.03529412 0.01176471 0.01176471 0.01176471
##  [5,] 0.03529412 0.02352941 0.01176471 0.01176471 0.01176471 0.01176471
##  [6,] 0.03529412 0.03529412 0.01176471 0.01176471 0.01176471 0.01176471
##  [7,] 0.03529412 0.03529412 0.01176471 0.01176471 0.01176471 0.02352941
##  [8,] 0.03529412 0.03529412 0.01176471 0.01176471 0.01176471 0.02352941
##  [9,] 0.03529412 0.02352941 0.01176471 0.01176471 0.01176471 0.02352941
## [10,] 0.03529412 0.01176471 0.01176471 0.01176471 0.01176471 0.02352941
##             [,7]       [,8]       [,9]      [,10]
##  [1,] 0.02352941 0.01176471 0.01176471 0.01176471
##  [2,] 0.02352941 0.02352941 0.01176471 0.01176471
##  [3,] 0.01176471 0.01176471 0.01176471 0.02352941
##  [4,] 0.01176471 0.02352941 0.02352941 0.02352941
##  [5,] 0.02352941 0.02352941 0.02352941 0.02352941
##  [6,] 0.02352941 0.02352941 0.02352941 0.02352941
##  [7,] 0.02352941 0.02352941 0.02352941 0.02352941
##  [8,] 0.02352941 0.02352941 0.02352941 0.02352941
##  [9,] 0.02352941 0.02352941 0.02352941 0.02352941
## [10,] 0.02352941 0.02352941 0.02352941 0.02352941
```

6.4 Spaghetti Plot

Per comprendere meglio la ricerca del modello ottimale possiamo aiutarci con una dimostrazione grafica, che ci mostri in ogni grafico come variino i risultati sia su training set che su validation set al variare dei parametri passati.

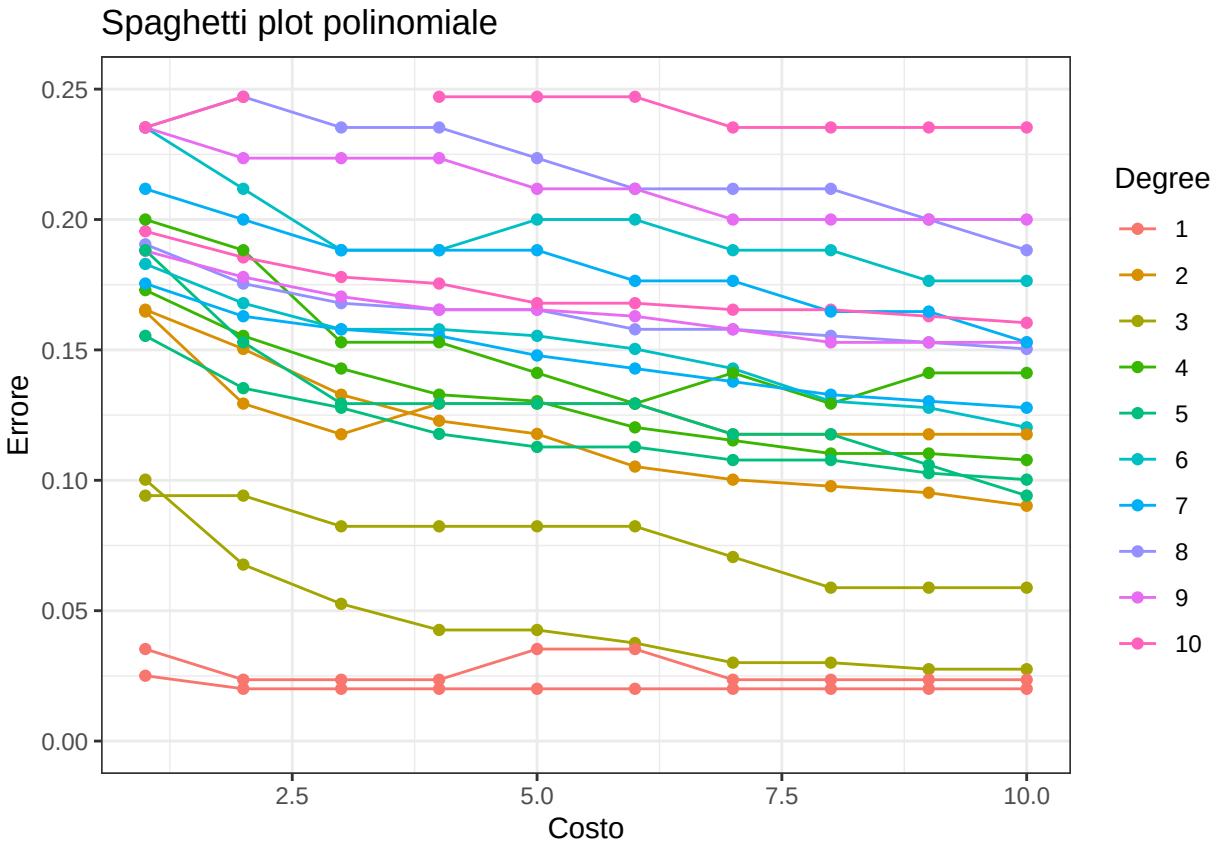
6.4.1 Lineare

In questo grafico vediamo come varii l'errore sia nel validation set che nel training set al variare del costo.

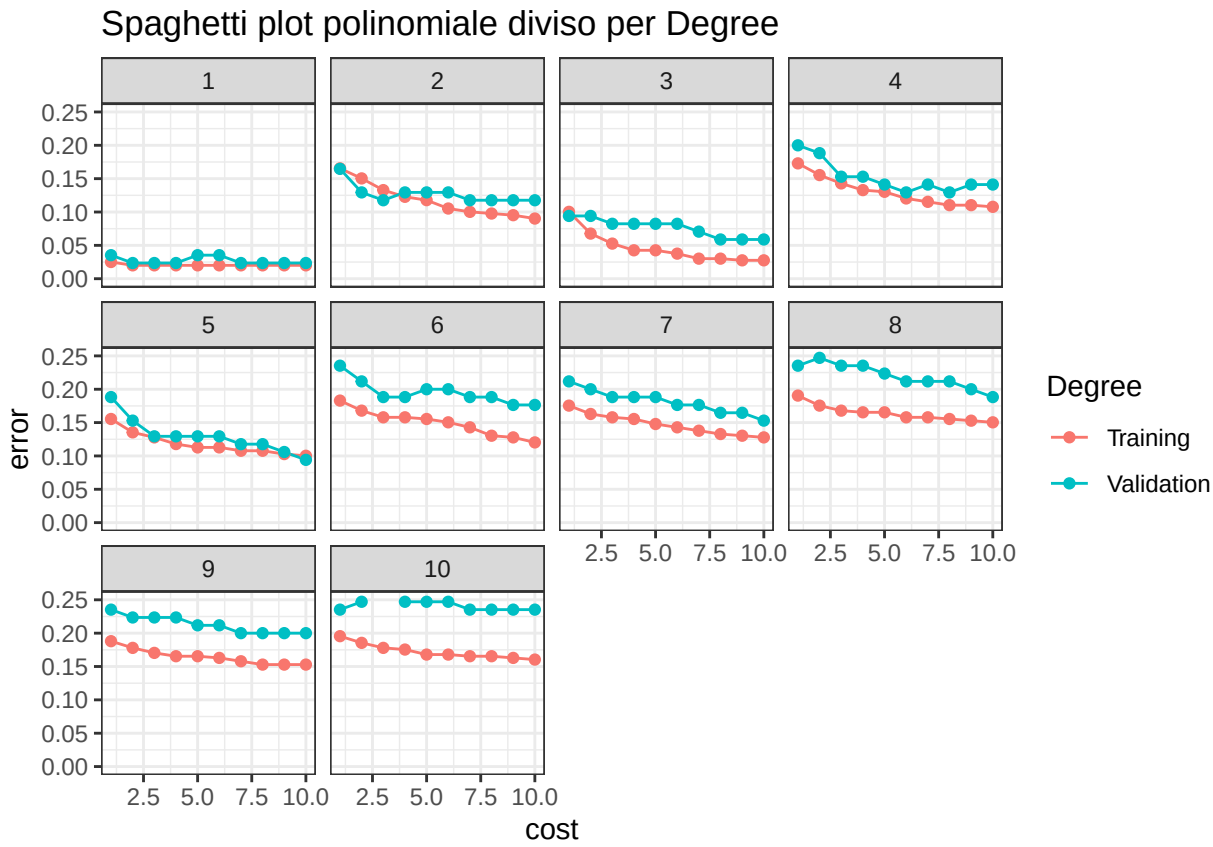


6.4.2 Polinomiale

Questo grafico ci mostra il variare dell'errore (asse delle ordinate) al variare del costo (asse delle ascisse) e del grado (colore della linea).

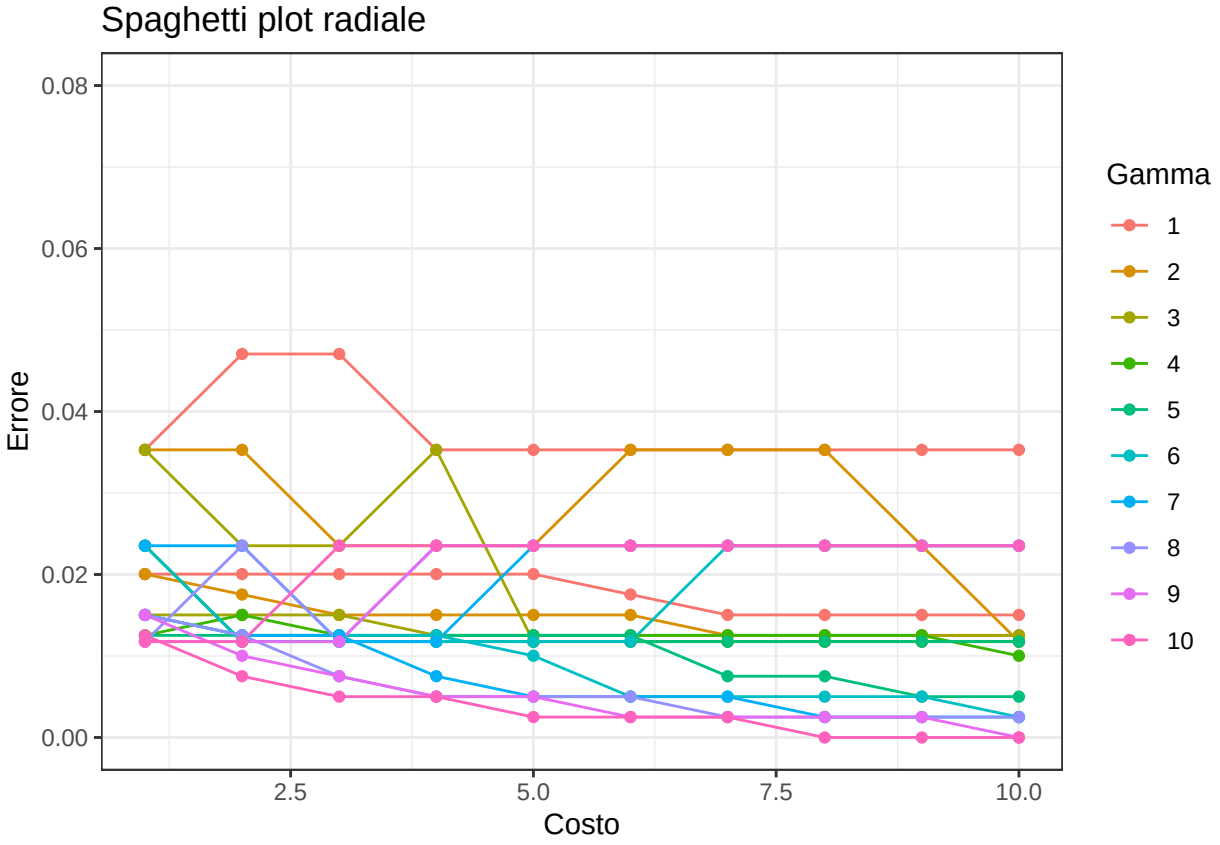


Questo grafico, chiamato in gergo Spaghetti plot non è chiarissimo essendoci tante linee (da qui il nome), abbiamo così deciso, per renderlo più chiaro di dividere il grafico in più grafici a seconda del degree, rappresentando in rosso i risultati del training set e in blu i risultati del validation set:

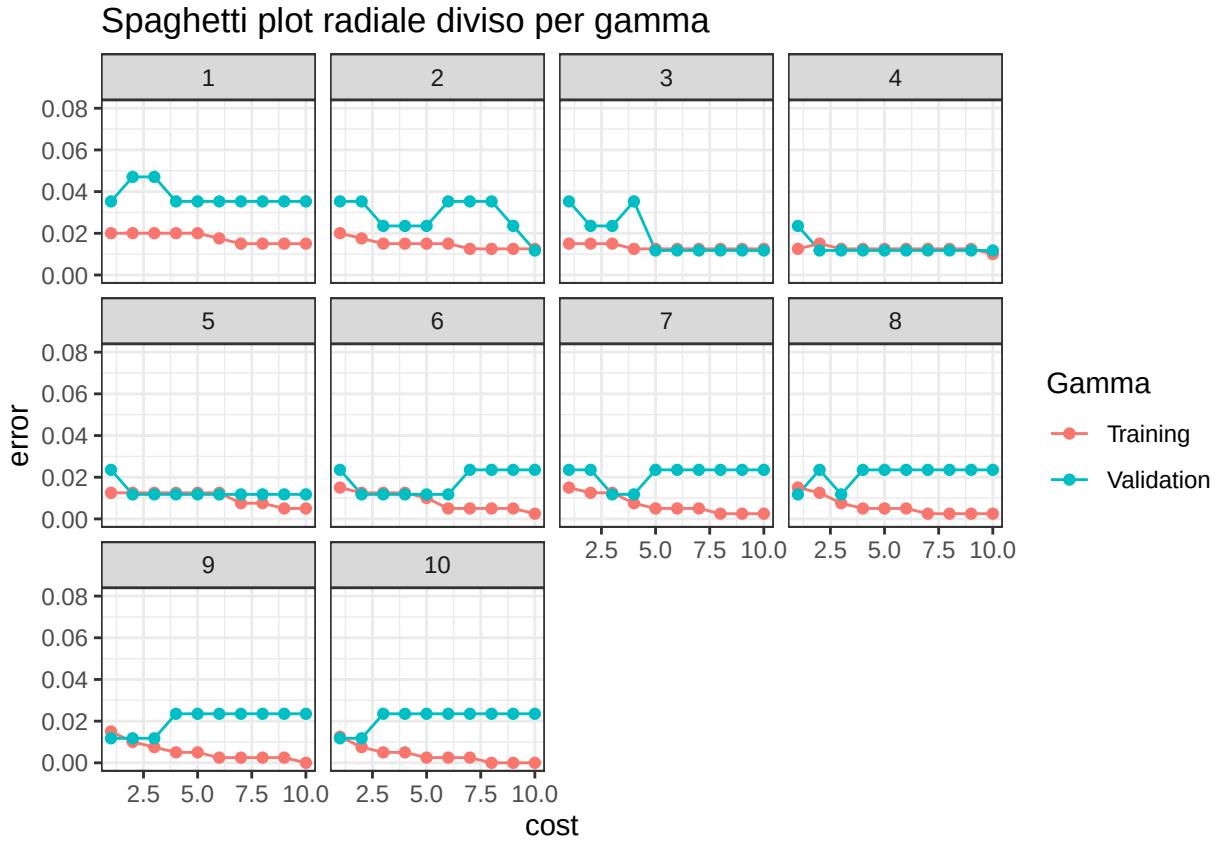

Spaghetti plot polinomiale diviso per Degree
1
2
3
4
5
6
7
8
9
10
error
cost
0.25
0.20
0.15
0.10
0.05
0.00
2.5
5.0
7.5
10.0
Degree
Training
Validation

6.4.3 Radiale

Questo grafico ci mostra il variare dell'errore (asse delle ordinate) al variare del costo (asse delle ascisse) e del gamma (colore della linea). Abbiamo usato altre unità di misura per rendere il grafico più leggibile, essendo l'entità dell'errore più piccolo rispetto all'errore del polinomiale.



Anche in questo caso abbiamo creato più grafici per rendere tutto più leggibile, rappresentando in rosso i risultati del training set e in blu i risultati del validation set:



7 Valutazione performance

Per trovare il modello più efficiente e quindi con l'errore minimo, andiamo a confrontare i risultati minimi ottenuti per ogni modello con gli altri minimi ottenuti e una volta trovato possiamo applicarlo all'ultimo set di dati che ci resta, ovvero il test set.

```
## [1] "lineare"
## [1] 0.04705882
## [1] "polinomiale"
## [1] 0.02352941
## [1] "radiale"
## [1] 0.01176471
```

```r
#Confronto tutti i valori minimi per trovare IL minimo in assoluto
if(minimo.Lin<minimo.Poly){
  if(minimo.Lin<minimo.Rad){
    errore_minimo<-minimo.Lin
    pos_errore_minimo<-which(MR.lin.total.Val==min(MR.lin.total.Val), arr.ind = TRUE)
    x<-pos_errore_minimo[1]

    #applichiamo il machine learning con i parametri migliori
    model<-svm(diagnosis ~ ., ds, kernel="linear", cost=x)
    y.pred<-predict(model.SVC, ds.test)
    Errore<-MR(y.pred, ds.test$diagnosis)
    Accuratezza<-Acc(y.pred, ds.test$diagnosis)
    modello<-"linear"
  } else {
    errore_minimo<-minimo.Rad
    pos_errore_minimo<-which(MR.rad.total.Val==min(MR.rad.total.Val), arr.ind = TRUE)
    y<-pos_errore_minimo[1,1]
    x<-pos_errore_minimo[1,2]


    #applichiamo il machine learning con i parametri migliori
    model.SVM<-svm(diagnosis~., ds, kernel="radial", cost=x, gamma=y/100)
    y.pred<-predict(model.SVM, ds.test)
    Errore<-MR(y.pred, ds.test$diagnosis)
    Accuratezza<-Acc(y.pred, ds.test$diagnosis)
    modello<-"radial"
  }
} else {
  if(minimo.Poly<minimo.Rad){
    errore_minimo<-minimo.Poly
    pos_errore_minimo<-which(MR.poly.total.Val==min(MR.poly.total.Val), arr.ind = TRUE)
    y<-pos_errore_minimo[1,1]
    x<-pos_errore_minimo[1,2]

    #applichiamo il machine learning con i parametri migliori
    model.SVM<-svm(diagnosis~., ds, kernel="polynomial", cost=x, degree=y)
    y.pred<-predict(model.SVM, ds.test)
    Errore<-MR(y.pred, ds.test$diagnosis)
    Accuratezza<-Acc(y.pred, ds.test$diagnosis)
    modello<-"polynomial"

  } else {
    errore_minimo<-minimo.Rad
    pos_errore_minimo<-which(MR.rad.total.Val==min(MR.rad.total.Val), arr.ind = TRUE)
    x<-pos_errore_minimo[1,1]
    y<-pos_errore_minimo[1,2]

    #applichiamo il machine learning con i parametri migliori
    model.SVM<-svm(diagnosis~., ds, kernel="radial", cost=x, gamma=y/100)
    y.pred<-predict(model.SVM, ds.test)
    Errore<-MR(y.pred, ds.test$diagnosis)
    Accuratezza<-Acc(y.pred, ds.test$diagnosis)
    modello<-"radial"
```

```
  }
}

errore_minimo #Stampo l'errore minimo
```

```
## [1] 0.01176471
```

```
x             #riga del minimo
```

```
## row
##  10
```

```
y             #colonna del minimo, eventuale perché potrebbe essere lineare
```

```
## col
##   2
```

```
Errore
```

```
## [1] 0.01176471
```

```
Accuratezza
```

```
## [1] 0.9882353
```

```
modello
```

```
## [1] "radial"
```

8 Interpretazione probabilistica

Invece di predirre una classe con SVM, vogliamo predirre una probabilità. Questo approccio vale solo per classi binarie. La probabilità che R predice si basa sulla distanza del punto dalla retta o dalla curva.

```
## [1] "probabilità di diagnosi"
```

```
##            9           11           20           31           33           43
## 9.903255e-01 9.528809e-01 2.245437e-02 9.999317e-01 9.999234e-01 9.706171e-01
##           54           62           64           71           77           80
## 9.936998e-01 1.990594e-03 3.182595e-04 9.999256e-01 3.181355e-03 1.842993e-03
##           83           95           99          100          101          128
## 9.823988e-01 9.948012e-01 1.073956e-03 2.355509e-01 9.670671e-01 9.968147e-01
##          132          140          141          155          168          170
## 9.992282e-01 1.367912e-04 3.360644e-04 4.005198e-02 9.679244e-01 2.970711e-03
##          173          182          183          195          196          218
## 9.833087e-01 9.936006e-01 9.958037e-01 6.673205e-01 1.750387e-03 7.048255e-05
##          220          221          228          230          232          259
## 9.998446e-01 2.764241e-03 2.015704e-02 9.698827e-01 8.595469e-03 9.832956e-01
##          261          267          268          273          284          296
## 9.999809e-01 4.350326e-04 1.185482e-03 9.992925e-01 9.758645e-01 3.548795e-04
##          299          302          304          310          313          314
## 7.552824e-02 4.460900e-04 1.388084e-04 1.151280e-03 2.189318e-04 2.353358e-04
##          335          344          347          357          359          367
## 7.403808e-04 9.989452e-01 1.460167e-03 4.516238e-03 2.228794e-04 9.997629e-01
##          392          398          408          416          417          424
## 6.960346e-04 4.625847e-04 3.486957e-04 1.341973e-03 7.329413e-03 2.145513e-02
##          426          432          436          437          440          443
## 5.016753e-04 1.859363e-04 9.894263e-01 5.628732e-03 1.674598e-03 1.187658e-03
##          447          456          463          474          476          493
```

```
## 9.999613e-01 7.959512e-02 1.466632e-01 7.268625e-03 7.399349e-03 9.991667e-01
##          494          495          502          517          523          532
## 1.091671e-04 2.051520e-03 9.610781e-01 9.999094e-01 4.905858e-04 2.938335e-02
##          534          537          540          549          564          565
## 9.991902e-01 5.429553e-01 2.005295e-03 2.009351e-04 9.991079e-01 9.997910e-01
##          566
## 9.995219e-01
```

```
## [1] "predizione del modello"
```

```
##  [1] 1 1 0 1 1 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 1 0 0 1 0 1 1 0
## [39] 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 1
## [77] 0 0 1 1 0 0 1 1 1
```

Matrice di confusione

```
##
## y.final  B  M
##       0 49  1
##       1  0 35
```

Per evitare falsi positivi e quindi dichiarare benigno ciò che in realtà è maligno abbassiamo il valore di *treshold*.
A questo punto il valore di riferimento per dichiarare una diagnosi maligna non sarà più con y.class > 0.5 ma mettiamo y.class > 0.2

```
##  [1] 1 1 0 1 1 1 1 0 0 1 0 0 1 1 0 1 1 1 1 0 0 0 1 0 1 1 1 1 0 0 1 0 0 1 0 1 1 0
## [39] 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 1
## [77] 0 0 1 1 0 0 1 1 1
```

Matrice di confusione

```
##
## y.final  B  M
##       0 49  0
##       1  0 36
```

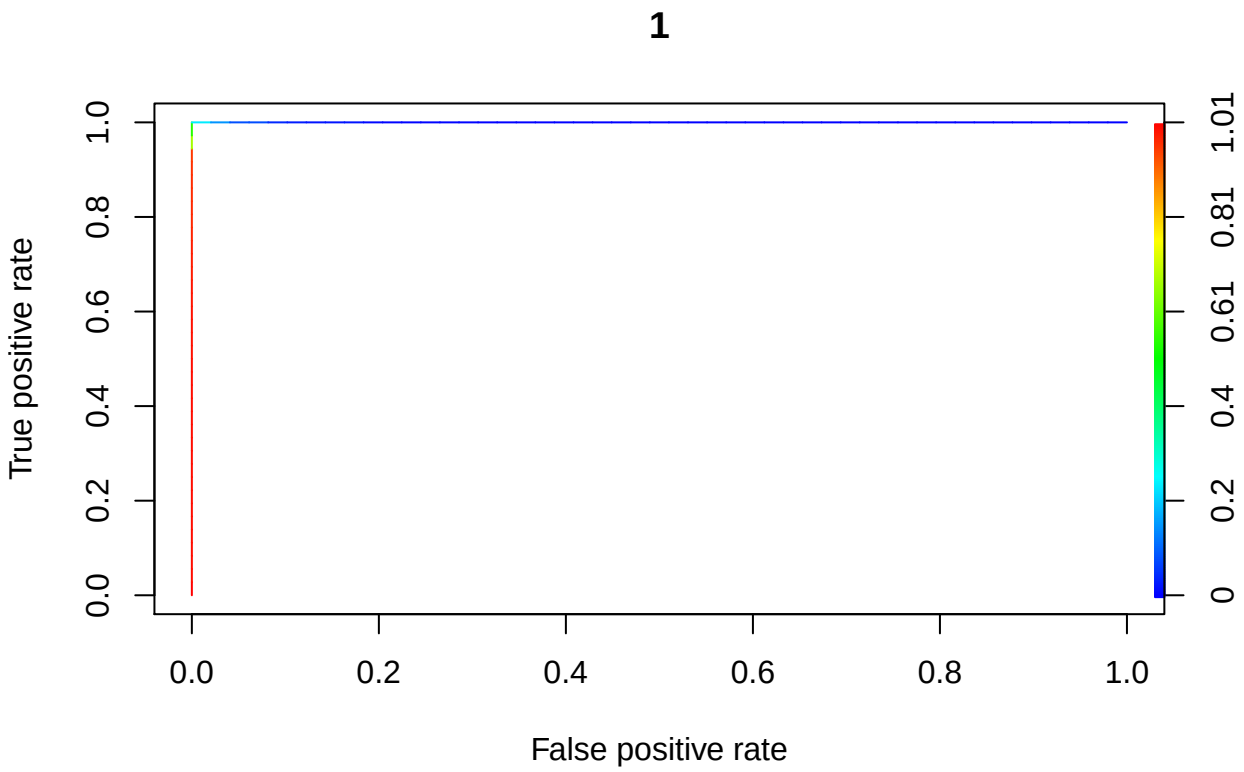
Una volta trovati i valori di treshold ottimali per il nostro studio si valuta nuovamente la performance e si stampano curva di ROC e AUC per concludere la valutazione

8.1 Curva di ROC

In questa curva il colore indica il treshold e mi dice per ogni valore del treshold il false positive rate e il true positive rate.
La curva partirà sempre dai valori di [0,0] e finirà sempre sui valori [1,1]. Una curva perfetta rispecchierebbe gli assi in quanto ogni parametro dovrebbe darci l'accuratezza massima possibile.

Ora valutiamo attraverso la curva di ROC la performance del modello.
La valutazione si effettua attraverso l'AUC che ci dice quanto è buono il modello in maniera assoluta, indipendentemente dal treshold.

8.1.1 AUC (area sotto la curva di ROC)



9 Studio statistico dei risultati

9.1 SRS(k)

Anche se il nostro modello ha dato ottimi risultati, la nostra analisi non può fermarsi qui, perché potrebbe essere comunque stato un caso dovuto a come siano stati selezionati i dati. Dobbiamo quindi generare dei nuovi training, validation e test set per poter confermare la validità del modello. Nel nostro caso abbiamo deciso di applicare il modello 50 volte.

```
for (c in 1:50){
  train.sample<-sample(Nstat, Nstat.train)
  dstat.train<-ds[train.sample, ]
  dstat.test<-ds[-train.sample, ]

  val.sample<-sample(Nstat.test+Nstat.val, Nstat.val)
  dstat.val<-ds.test[val.sample, ]
  dstat.test<-dstat.test[-val.sample, ]

  model.SVM<-svm(diagnosis~., dstat.train, kernel="radial", cost=x, gamma=y/100)
  y.pred<-predict(model.SVM, dstat.train)

  y.pred<-predict(model.SVM, dstat.test)
  MR.radStat<-MR(y.pred, dstat.test$diagnosis)
  Acc.radStat <- Acc(y.pred, dstat.test$diagnosis)
```

```
  MR.radStatVector[c] <-  MR.radStat
  Acc.radStatVector[c] <- Acc.radStat
}
```

```
## [1] "Errore"
```

```
##  [1] 0.03278689 0.03125000 0.03225806 0.04918033 0.03125000 0.05084746
##  [7] 0.01666667 0.01724138 0.06060606 0.00000000 0.06896552 0.09090909
## [13] 0.01785714 0.03225806 0.03278689 0.03278689 0.04615385 0.05084746
## [19] 0.05882353 0.03773585 0.10169492 0.05172414 0.05172414 0.03636364
## [25] 0.00000000 0.01724138 0.00000000 0.08474576 0.00000000 0.03125000
## [31] 0.01724138 0.03174603 0.03030303 0.01612903 0.03225806 0.01470588
## [37] 0.03125000 0.00000000 0.07272727 0.00000000 0.05263158 0.03174603
## [43] 0.07936508 0.01639344 0.01639344 0.05084746 0.04687500 0.08333333
## [49] 0.03076923 0.01724138
```

```
## [1] "Accuratezza"
```

```
##  [1] 0.9672131 0.9687500 0.9677419 0.9508197 0.9687500 0.9491525 0.9833333
##  [8] 0.9827586 0.9393939 1.0000000 0.9310345 0.9090909 0.9821429 0.9677419
## [15] 0.9672131 0.9672131 0.9538462 0.9491525 0.9411765 0.9622642 0.8983051
## [22] 0.9482759 0.9482759 0.9636364 1.0000000 0.9827586 1.0000000 0.9152542
## [29] 1.0000000 0.9687500 0.9827586 0.9682540 0.9696970 0.9838710 0.9677419
## [36] 0.9852941 0.9687500 1.0000000 0.9272727 1.0000000 0.9473684 0.9682540
## [43] 0.9206349 0.9836066 0.9836066 0.9491525 0.9531250 0.9166667 0.9692308
## [50] 0.9827586
```

Ora che abbiamo i risultati di 50 set diversi, possiamo confermare la validità e l'efficienza del nostro modello.

9.2 Statistica descrittiva

9.2.1 Calcolo del centro

9.2.1.1 Media

```
## [1] 0.03675823
```

```
## [1] 0.9632418
```

9.2.1.2 Mediana

```
## [1] 0.03225806
```

```
## [1] 0.9677419
```

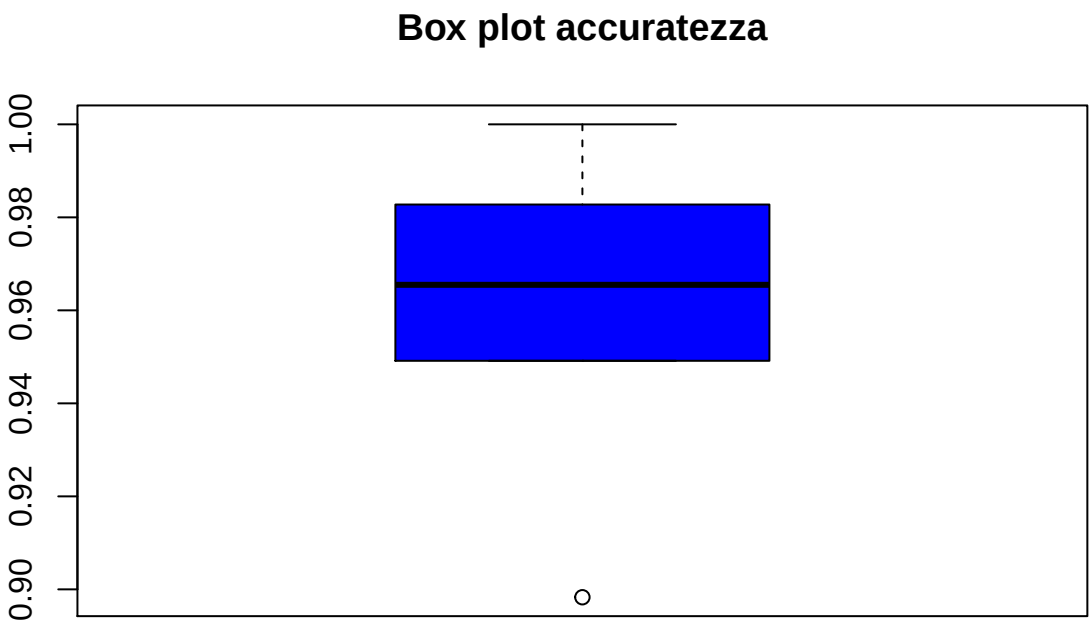
Abbiamo otttenuto che media e mediana sono diversi, possiamo quindi dire che ci troviamo difronte a una distribuzione asimmetrica. La media dell'errore è maggiore della mediana (asimmetria positiva) mentre la media dell'accuratezza è minore della mediana (asimmetria negativa). Sapendo che la mediana è il valore per cui nel set ci sono 50% dei dati maggiori di essa e il restante 50% è minore sappiamo essere posizionata sempre al centro, e se la media coincide con essa vuol dire che non ci sono nel nostro set outliers che vanno a influenzare il valore di quest'ultima.

9.2.1.3 Quantile semplice

```
##          0%         25%         50%         75%        100%
## 0.00000000 0.01724138 0.03225806 0.05084746 0.10169492
```

```
##         0%        25%        50%        75%       100%
## 0.8983051 0.9491525 0.9677419 0.9827586 1.0000000
```

Possiamo osservare come i quantili dell'errore e dell'accuratezza siano speculari, se sommiamo il primo dell'errore con l'ultimo dell'accuratezza otteniamo 1, ed è così anche per tutti gli altri. Questo fenomeno è evidente se pensiamo alla natura dell'accuratezza, viene calcolata sottraendo a 1 l'entità dell'errore.



9.2.2 Diffusione dei dati

9.2.2.1 Varianza campionaria

```
## [1] 0.0006317292
```

```
## [1] 0.0006317292
```

La varianza è la media di quanto si discostino i dati dalla media aritmetica ed è quindi logico che la varianza dell'errore e dell'accuratezza siano uguali perché l'accuratezza è calcolata sottraendo a 1 l'entità dell'errore, è quindi speculare all'errore, quindi anche se la media e i dati in se sono diversi, la loro variabilità sarà uguale e ce lo dimostra la varianza.

9.2.2.2 Daviazione standard

```
## [1] 0.02513422
```

```
## [1] 0.02513422
```

Lo scarto quadratico non è altro che la radice quadrata della varianza, quindi anche in questo caso sono logicamente uguali i due valori.

9.2.2.3 Range interquartile (IQR)

```
## [1] 0.03360608
```

```
## [1] 0.03360608
```

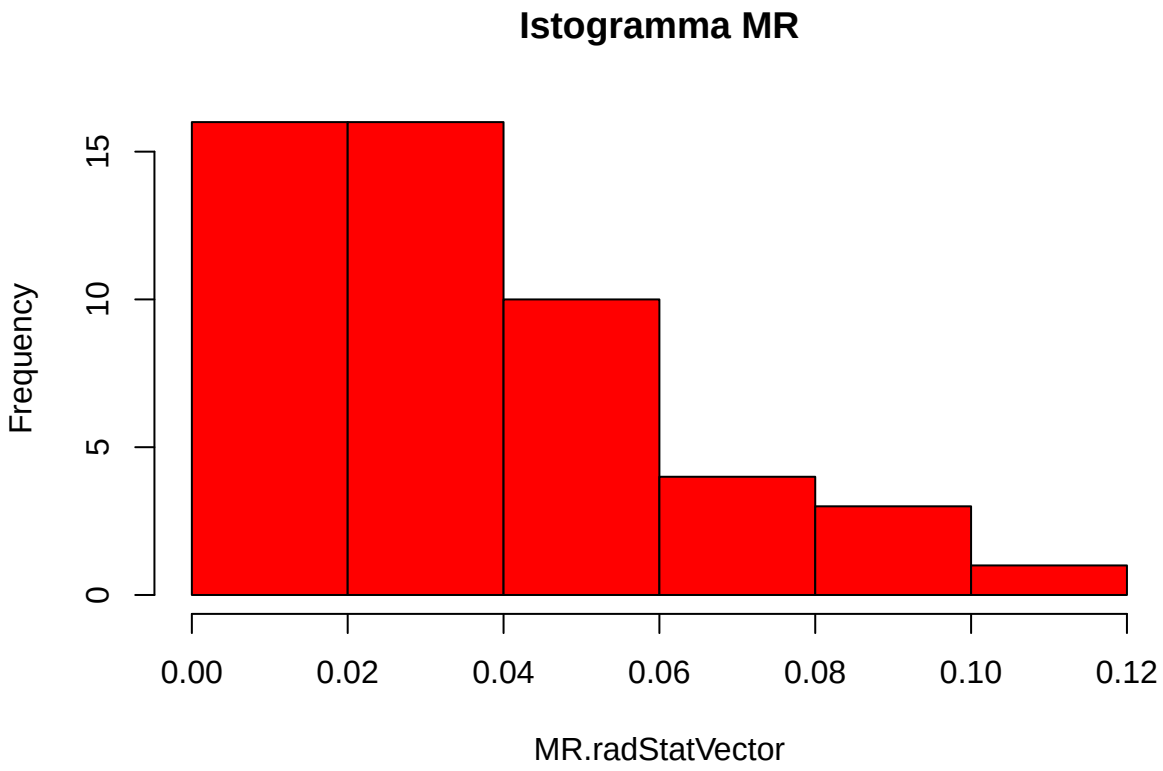
Questo dato viene spesso usato nei set in cui si sospetta ci siano outliers, e considera i dati centrali del set essendo la differenza tra il primo e il terzo quartile.

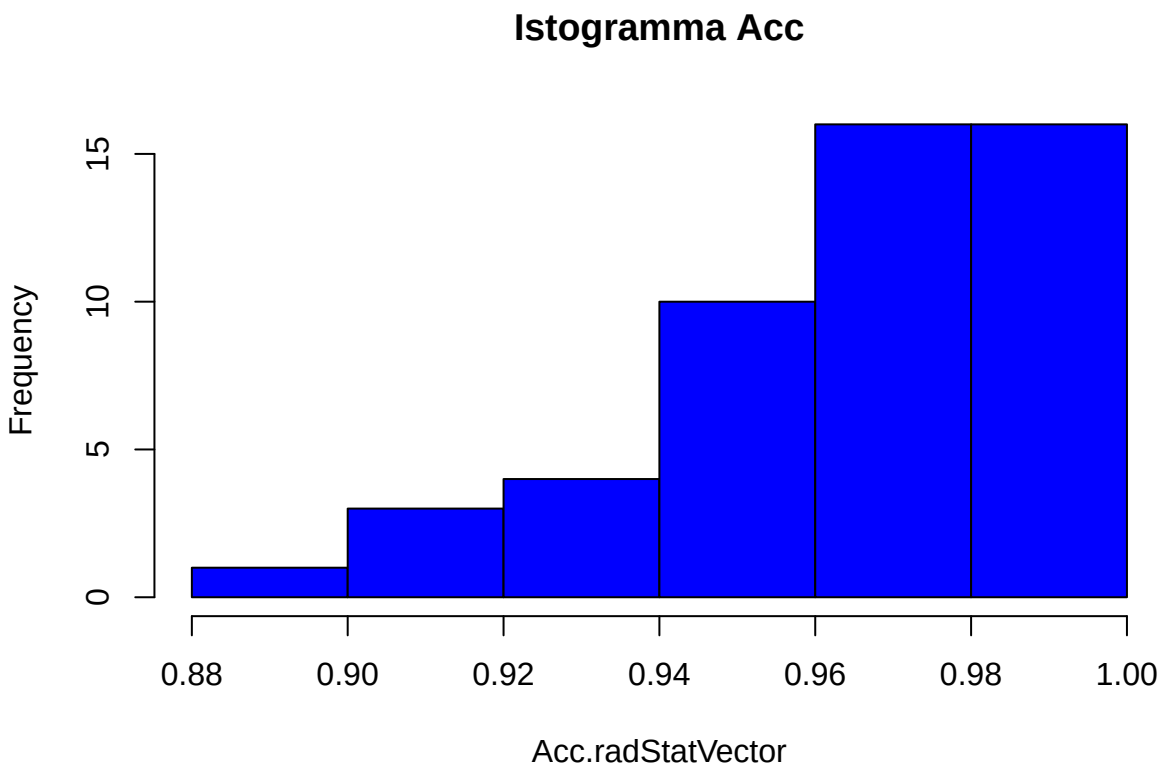
9.2.2.4 Simmetria

```
## [1] 0.5891957
```

```
## [1] -0.5891957
```

La simmetria conferma la nostra analisi iniziale su media e mediana, l'accuratezza ha una asimmetria negativa mentre l'errore una asimmetria positiva.





9.2.2.5 Curtosi

```
## [1] 2.802157
```

```
## [1] 2.802157
```

Entrambe le distribuzioni sono iponormali, ovvero rispetto alla distribuzione normale si ha una minore frequenza per valori centrali ed estremi e maggiore per valori intermedi.

9.2.2.6 Deviazione assoluta dalla media (MAD)

```
## [1] 0.02352089
```

```
## [1] 0.02352089
```

Anche questo dato è un rilevatore delle distribuzioni più forte rispetto alla varianza e alla deviazione strandard in quanto non si fa influenzare dagli outliers.

9.3 Statistica inferenziale

Supponiamo che la nostra SRS(n) formato da valori di MR appartenga ad una distribuzione normale con media e deviazione standard note. Ne segue che la variabile aleatoria data dalla media campionaria ha anche essa una distribuzione normale con media=mu e deviazione standard=sigma/sqrt(n)

Verifichiamo che, preso un campione casuale da una distribuzione che abbiamo supposto fosse normale con media=mu e sd=sigma, la variabile aleatoria media campionaria avrà anch'essa una distribuzione normale con media=mu e sd=sigma/sqrt(n)

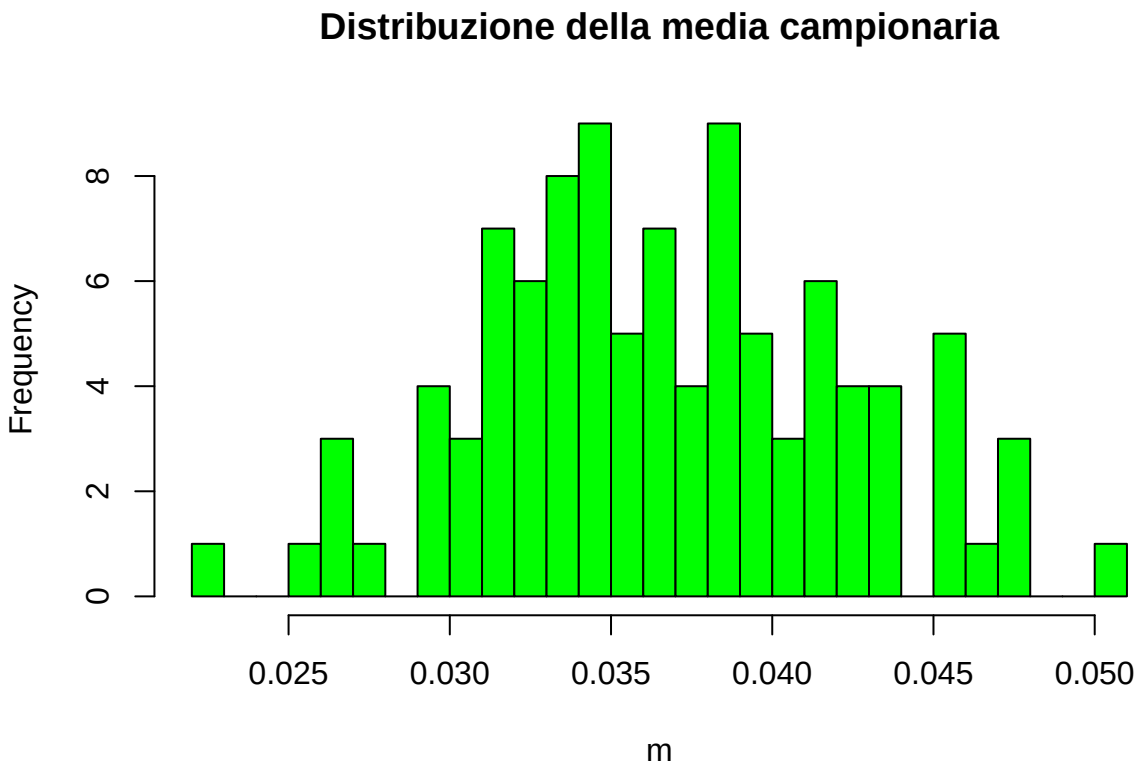
9.3.1 Distribuzione della media campionaria

```
mu <- mean(MR.radStatVector)
sigma <- sd(MR.radStatVector)
k <- 100 #numero di SRS che prendo
n <- 20 #dimensione del campione
```

Aumentando k o n i valori calcolati si avvicineranno sempre di più ai valori esatti.

```
 # In m io avrò k realizzazioni della variabile aletoria media campionaria
m <- replicate(k, mean(rnorm(n, mu, sigma)))
# La media campionaria è uno stimatore non distorto della media mu
mu_c <- mean(m)
```

```
hist(m, main="Distribuzione della media campionaria",breaks=20, col="green")
```



Mostriamo la media calcolata dalle k volte in cui realizzo la media campionaria e la media esatta, infine calcoliamo l'errore.

```
## [1] 0.03671015
```

```
## [1] 0.03675823
```

```
## [1] 4.808555e-05
```

Mostriamo adesso la deviazione standard, che dovrebbe essere uguale o quasi a sigma/sqrt(n), poi calcoliamo l'errore

```
## [1] 0.005526556
```

```
## [1] 0.005620183
```

```
## [1] 9.362741e-05
```

9.3.2 Distribuzione della variabile aleatoria Z

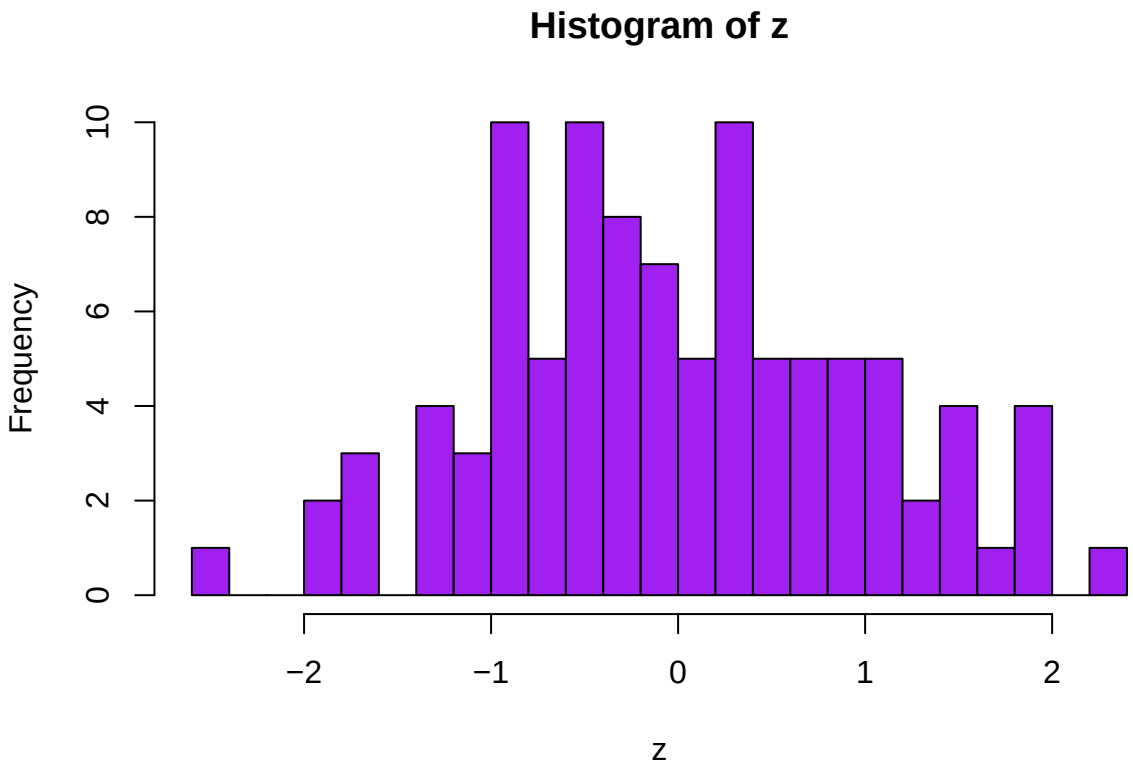
Ora dimostriamo che la variabile aleatoria Z definita come:

```
z <- (m-mu)/(sigma/sqrt(n))
```

appartiene ad una distribuzione normale standard (mean=0, sd=1)
Mostriamo la media e la deviazione standard di Z, la media si avvicinerà a 0 e la deviazione a 1, dimostrando quello detto in precedenza

```
## [1] -0.00855587
```

```
## [1] 0.9833409
```



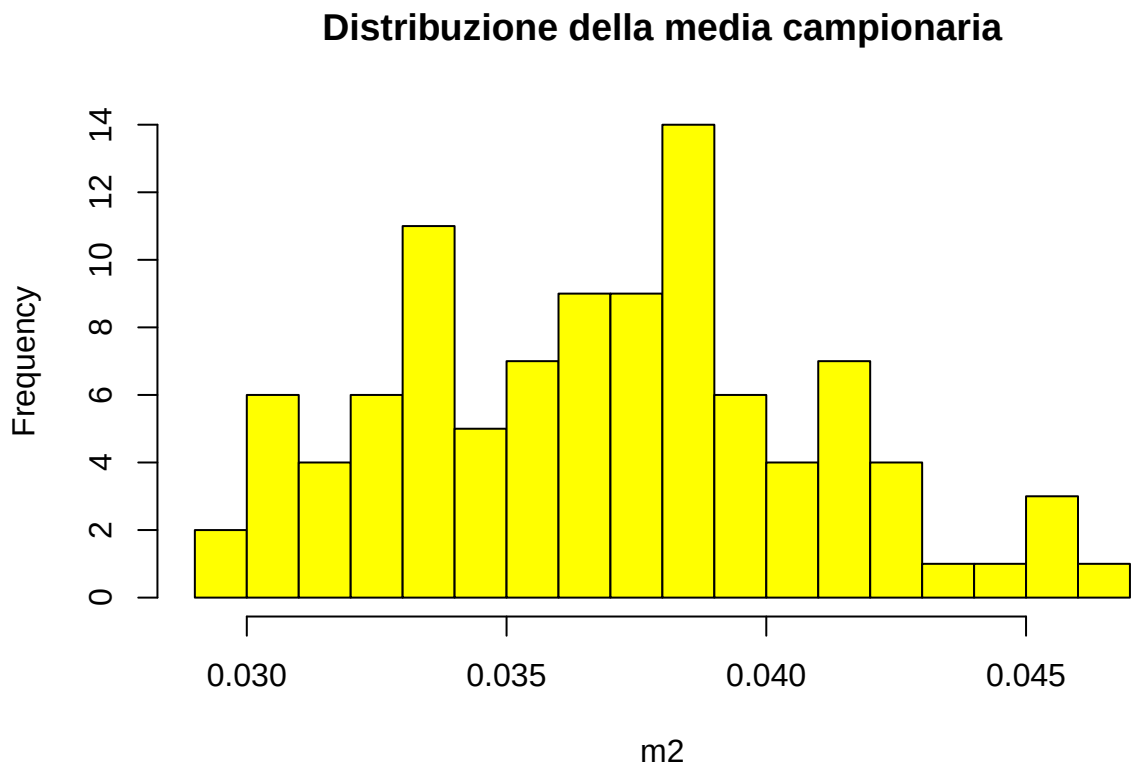
Ora supponiamo che i nostri campioni appartengano ad una distribuzione diversa da quella normale, di cui però sono note media e deviazione standard.

Per il teorema del limite centrale sappiamo che data una SRS(n) con media e deviazione standard note, la variabile Z enunciata precedentemente ha una distribuzione campionaria che ha come limite per n che tende a infinito la distribuzione normale standard mentre la variabile aleatoria media campionaria ha come limite per n che tende ad infinito una distribuzione normale con media=mu e sd=sigma/sqrt(n)
Questo vale quando il campione è sufficientemente grande.

Se prendo dei campioni da una qualsiasi distribuzione, supponendo di avere un n sufficientemente grande, mi posso sempre ricondurre allo studio di campioni con una distribuzione normale per quanto riguarda informazioni relative alla statistica media campionaria, varianza e Z.

```
mu2 <- mean(MR.radStatVector)
sigma2 <- sd(MR.radStatVector)
k <- 100 #numero di SRS che prendo
n2 <- 35 #dimensione del campione

m2 <- replicate(k, mean(rnorm(n2, mu2, sigma2)))
```




```
mu_c2 <- mean(m2)
mu_c2
```

```
## [1] 0.03685693
```

```
mu2
```

```
## [1] 0.03675823
```

```
errM2 <- abs(mu2-mu_c2)
errM2
```

```
## [1] 9.869214e-05
```

```
sigma_c2 <- sd(m2)
sigma_c2
```

```
## [1] 0.00398063
```

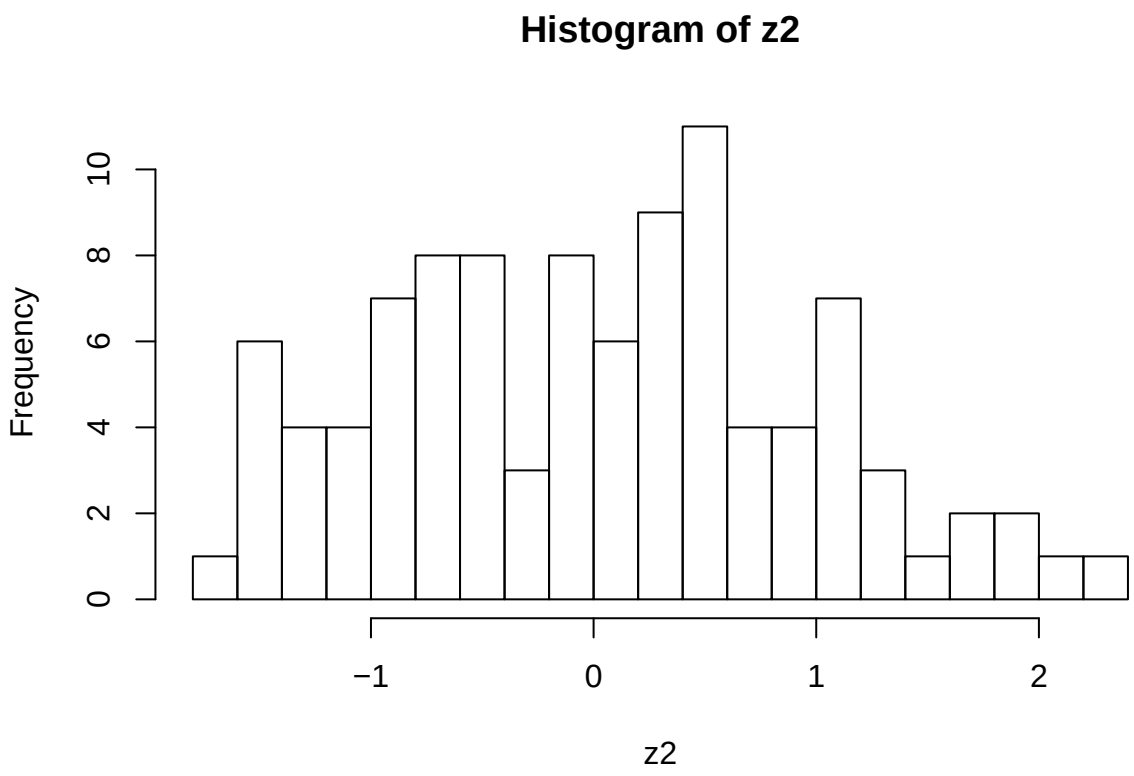
```
h2 <- sigma2/sqrt(n2)
h2
```

```
## [1] 0.004248459
```

```
errSD2 <- abs(h2-sigma_c2)
errSD2
```

```
## [1] 0.0002678291
```

```
z2 <- (m2-mu2)/(sigma2/sqrt(n2))
hist(z2, breaks=20)
```




```
zM2 <- mean(z2)
zM2
```

```
## [1] 0.0232301
```

```r
zSD2 <- sd(z2)
zSD2
```

```
## [1] 0.9369585
```

Quanto è buona la stima della media campionaria?
L'affidabilità della stima dipende dal livello di probabilità che scegliamo, nel nostro caso voglimao calcolare un intervallo di confidenza con probabilità pari al 95%, ovvero con un livello di confidenza alpha=0.05

L'intervallo è una variabile casuale poiché i 2 estremi dipendono dalla variabile aleatoria media campionaria. Possiamo dire che la probabilità che il valore esatto mu sia all'interno dell'intervallo è del 100(1-alpha)%

```r
mu <- mean(MR.radStatVector)
sigma <- sd(MR.radStatVector)
n3 <- 50
alpha <- 0.05

x3 <- rnorm(n3, mu, sigma)
mCamp <- mean(x3) #media campionaria
S <- sd(x3) #deviazione standard campionaria

z_alpha <- qnorm(1-alpha, mean=0, sd=1)

# Calcolo i 2 estremi dell'intervallo
infM <- mCamp - z_alpha*S/sqrt(n3)
supM <- mCamp + z_alpha*S/sqrt(n3)

infM
```

```
## [1] 0.03494187
```

```r
supM
```

```
## [1] 0.04633652
```

Stampiamo la media esatta e vediamo se sta nell'intervallo di confidenza

```r
mu
```

```
## [1] 0.03675823
```

Proviamo a vedere su 100 prove se effettivamente rispetta la probabilità del 95%

```r
k <- 0
V.infM <- 1:100
V.supM <- 1:100

for(i in 1:100){
  y=rnorm(n3, mu, sigma)
  mCamp1 <- mean(y)
  S1<-sd(y)
  V.infM[i] <- mCamp1 - z_alpha*S1/sqrt(n3)
  V.supM[i] <- mCamp1 + z_alpha*S1/sqrt(n3)
  if (V.infM[i] <=  mu & mu <= V.supM[i]){
    k <- k+1
  }
}
```

```
# Ora stampo il numero di volte in cui la media esatta si trova all'interno
# dell'intervallo di confidenza calcolato sulla media campionaria
k
```

```
## [1] 94
```

10 Features Selection

Questa parte di analisi consiste nell'andare a studiare quali dati possano essere ignorati lasciando la performance del modello invariata. Nello specifico consiste nel provare il modello senza alcuni dati selezionati, con il fine di identificare i features necessari per ottenere la stessa performance ottenuta con tutti i dati, i dati identificati saranno quelli che influenzano maggiormente la diagnosi della cellula.
Modello che usa tutti i features.

```
## [1] 0.05882353
```

```
## [1] 0.9411765
```

La prima prova la facciamo togliendo i dati relativi al raggio, secondo la matrice di correlazione la diagnosi era fortemente influenzata dal raggio, quindi ci aspettiamo che la performance diminuisca.

```
## [1] 0.09411765
```

```
## [1] 0.9058824
```

La seconda prova la facciamo sul perimentro, e anche in questo caso ci aspettiamo che la performance diminuisca.

```
## [1] 0.09411765
```

```
## [1] 0.9058824
```

La terza prova la facciamo sulla texture. La diagnosi abbiamo visto dal primo grafico essere influenzata dalla texture, ma non tanto quanto dal raggio o dal perimetro, quindi ci aspettiamo che la performance peggiori, ma non tanto quanto i features precedenti.

```
## [1] 0.07058824
```

```
## [1] 0.9294118
```

Sta volta togliamo i dati relativi all'area, ci aspettiamo lo stesso risultato di raggio e perimetro.

```
## [1] 0.09411765
```

```
## [1] 0.9058824
```

Continuiamo la nostra analisi togliendo i valori di smoothness, questo dato non influenzava più di tanto la diagnosi esattamente come texture, quindi potremmo aspettarci che l'accuratezza diminuisca.

```
## [1] 0.07058824
```

```
## [1] 0.9294118
```

Il dato compactness influenzava, secondo la matrice di correlazione, la diagnosi più o meno come l'area, vediamo come varia la performance togliendo questo dato.

```
## [1] 0.05882353
```

```
## [1] 0.9411765
```

Analizziamo senza i dati della concavità, anche questo era un dato che influenzava la diagnosi.

```
## [1] 0.07058824
```

```
## [1] 0.9294118
```

Analizziamo senza i dati sui punti concavi, ci aspettiamo che l'accuratezza diminuisca come per il raggio.

```
## [1] 0.02352941
```

```
## [1] 0.9764706
```

Togliamo i dati relativi alla simmetria, ricordandoci che in più grafici avevamo costato non influenzasse la diagnosi più di tanto.

```
## [1] 0.05882353
```

```
## [1] 0.9411765
```

Togliamo i dati relativi alla fractal dimension, ricordando che questo dato non influenzasse in alcun modo la diagnosi, è il primo dato del dataset che non influenzi per niente la diagnosi.

```
## [1] 0.05882353
```

```
## [1] 0.9411765
```

Iniziamo un'analisi inversa, invece di escludere solamente dei dati, sta volta prendiamo in cosiderazione solo dei dati. Prendiamo solamenti i valori peggiori.

```
## [1] 0.1411765
```

```
## [1] 0.8588235
```

11 Codice completo

Caricamento del dataset ds<-read.csv(“data.csv”)

Pulizia del dataset
ds$X<-NULL
ds<-na.omit(ds)
levels(ds$diagnosis)

Salviamo il backup
dsBU<-ds

Grafico 1: Matrice di correlazione
mydata<-dsBU
mydata$id<-NULL
numero<-as.numeric(mydata$diagnosis)
mydata$diagnosis<-numero
cor.matrix<- cor(mydata)
corrplot(cor.matrix,method= c(“circle”),tl.cex=0.6,number.font=6,tl.col=“black”)

Grafico 2: Pazienti per range di ragggio
raggruppamentoradius<-c(“6.98-10”,“10.01-13.03”,“13.04-16.06”,“16.07-19.09”,“19.1-22.12”,“22.13-25.15”,“25.16-29”)
rrad<-ds$radius_mean
rrad<-cut(ds$radius_mean,breaks = c(6.98,10,13.03,16.06,19.09,22.12,25.15,29),labels = raggruppamentoradius)
ds$appoggio<-rrad

ggplot(data=NULL, aes(x=ds$appoggio, y=1)) +
geom_bar(stat = “identity”,color=“pink”,fill=“pink”)+
xlab(“range”)+
ylab(“numero pazienti”)+
labs(
title=“distribuzione delle dimensioni del raggio”)

Grafico 3: Barplot benigni maligni per range di raggio
conta<-c((table(ds$diagnosis[ds$appoggio==“6.98-10”])),(table(ds$diagnosis[ds$appoggio==“10.01-13.03”])),(table(ds$diagnosis[ds$appoggio==“13.04-16.06”])),(table(ds$diagnosis[ds$appoggio==“16.07-19.09”])),(table(ds$diagnosis[ds$appoggio==“19.1-22.12”])),(table(ds$diagnosis[ds$appoggio==“22.13-25.15”])),(table(ds$diagnosis[ds$appoggio==“25.16-29”])))
data2 <- matrix(conta, nrow=2)
colnames(data2)<- c(raggruppamentoradius)
rownames(data2)<- c(“benigno”,“maligno”)
media<-mean(ds$radius_mean)
mediana<-median(ds$radius_mean)

barplot(data2,
border=“white”,
axis.lty = 1,
col=colors()[c(49,554)] ,
font.axis=6,

cex.axis = 0.85,
cex.names = 0.85,
las=2,
beside=T,
main = “maligni e benigni per range di raggio”,
ylab=“numero di pazienti”,
font.lab=1)
legenda2<-legend(15,140,cex=0.7,fill=colors()[c(49,554)] ,legend=rownames(data2),title=“legenda”)

Grafico 4: Correlazione raggio e punti concavi
basic <- ggplot(ds, aes(ds$radius_mean, ds$concave.points_mean,colour=ds$diagnosis,
shape = factor(ds$diagnosis))) +

labs(title=“Analisi punti concavi e raggio”)+
xlab(“raggio”)+
ylab(“punti concavi”)+
geom_point(size=2)+
scale_color_brewer(palette=“Set2”)
basic+
labs(
colour=“legenda”,
shape=“legenda”)

Grafico 5: Boxplot simmetrie
box2<-boxplot(ds$symmetry_mean[ds$diagnosis==“M”],ds$symmetry_mean[ds$diagnosis==“B”],
names=c(“maligno”,“benigno”), main =“box plot delle simmetrie”,ylab=“simmetria”,
las=2,col=c(“red”,“green”))

Grafico 6: Correlazione compattezza e simmetria
basic2 <- ggplot(ds, aes(ds$compactness_mean, ds$symmetry_mean,colour=ds$diagnosis,
shape = factor(d$diagnosis))) +
labs(title=“Analisi compattezza e simmetria”)+
xlab(“compattezza”)+
ylab(“simmetria”)+
geom_point(size=2)+
scale_color_brewer(palette=“Set2”)
basic2+
labs(colour=“legenda”,
shape=“legenda”)

Grafico 7: Box plot Fractal dimension
box1<-boxplot(ds$fractal_dimension_mean[ds$diagnosis==“M”],ds$fractal_dimension_mean[ds$diagnosis==“B”],
names=c(“maligno”,“benigno”),main =“box plot fractal dimension”,ylab=“fractal_dimension”,
las=2,col=c(“red”,“green”))

Machine learning

Splitting
N<-nrow(ds)

```
N.train<- 399
N.val<-85
N.test<-85


train.sample<-sample(N, N.train)
ds.train<-ds[train.sample, ]
ds.test<-ds[-train.sample, ]


val.sample<-sample(N.test+N.val, N.val)
ds.val<-ds.test[val.sample, ]
ds.test<-ds.test[-val.sample, ]
```

Funzioni che useremo per valutare il modello

Errore

```
MR<- function(y.pred, y.true){
res<-mean(y.pred != y.true)
return(res)
}
```

Accuratezza

```
Acc<-function(y.pred, y.true){
res<-1-mean(y.pred!=y.true)
return(res)
}
```

Addestramento del modello

Lineare

```
MR.lin.total.Train<-1:10
for(c in 1:10){
model.SVC<-svm(diagnosis~ ., ds, kernel="linear", cost=c)
y.pred<-predict(model.SVC, ds)
MR.lin<-MR(y.pred, ds$diagnosis)
MR.lin.total.Train[c]<-MR.lin
}
MR.lin.total.Train
```

Polinomiale

```
MR.poly.total.Train<-matrix(ncol=10, nrow=10)
Error.poly.train<-c(0)
for(c in 1:10){
for(d in 1:10){
model.SVM<-svm(diagnosis~., ds, kernel="polynomial", cost=c, degree=d)
y.pred<-predict(model.SVM, ds)
MR.poly<-MR(y.pred, ds$diagnosis)
MR.poly.total.Train[c,d]<-MR.poly
Error.poly.train<-append(Error.poly.train, MR.poly)
```

```
}
}
Error.poly.train<-Error.poly.train[-1]
MR.poly.total.Train
```

Radiale

```
MR.rad.total.Train<-matrix(ncol=10, nrow=10)
Error.rad.train<-c(0)
for(c in 1:10){
for(g in 1:10){
model.SVM<-svm(diagnosis ~ ., ds, kernel="radial", cost=c, gamma=g/100)
y.pred<-predict(model.SVM, ds)
MR.rad<-MR(y.pred, ds$diagnosis)
MR.rad.total.Train[c,g]<-MR.rad
Error.rad.train<-append(Error.rad.train, MR.rad)
}
}
Error.rad.train<-Error.rad.train[-1]
MR.rad.total.Train
```

Hyperparameter Tuning

Lineare

```
MR.lin.total.Val<-1:10
for(c in 1:10){
model.SVC<-svm(diagnosis ~ ., ds, kernel="linear", cost=c)
y.pred<-predict(model.SVC, ds.val)
MR.lin<-MR(y.pred, ds.val$diagnosis)
MR.lin.total.Val[c]<-MR.lin
}
MR.lin.total.Val
```

Polinomiale

```
MR.poly.total.Val<-matrix(ncol=10, nrow=10)
Error.poly.val<-c(0)
for(c in 1:10){
for(d in 1:10){
model.SVM<-svm(diagnosis~., ds, kernel="polynomial", cost=c, degree=d)
y.pred<-predict(model.SVM, ds.val)
MR.poly<-MR(y.pred, ds.val$diagnosis)
MR.poly.total.Val[c,d]<-MR.poly
Error.poly.val<-append(Error.poly.val, MR.poly)
}
}
Error.poly.val<-Error.poly.val[-1]
MR.poly.total.Val
```

Radiale

```
MR.rad.total.Val<-matrix(ncol=10, nrow=10)
Error.rad.val<-c(0)
for(c in 1:10){
```

```
for(g in 1:10){
model.SVM<-svm(diagnosis~., ds, kernel="radial", cost=c, gamma=g/100)
y.pred<-predict(model.SVM, ds.val)
MR.rad<-MR(y.pred, ds.val$diagnosis)
MR.rad.total.Val[c,g]<-MR.rad
Error.rad.val<-append(Error.rad.val, MR.rad)
}
}
Error.rad.val<-Error.rad.val[-1]
MR.rad.total.Val
```

Spaghetti Plot

Lineare

Creo i dataframe per i plot

```
df_Spaghetti_Linear_Train<-data.frame(
cost=c(1:10),
error=MR.lin.total.Train,
id="Training")

df_Spaghetti_Linear_Val<-data.frame(
cost=c(1:10),
error=MR.lin.total.Val,
id="Validation")

df_Spaghetti_Linear<-rbind(df_Spaghetti_Linear_Train, df_Spaghetti_Linear_Val)
```

Plot

```
ggplot(df_Spaghetti_Linear, aes(x=cost, y=error, color=id))+
geom_line()+geom_point()+theme_bw()+scale_y_continuous(limits = c(0,0.25))
```

Polinomiale

Creo i dataframe per i plot

```
df_Spaghetti_Polinomial_Train<-data.frame(cost=ceiling(c(1:100)/10),
degree=c(1:10),
error=Error.poly.train,
id="Training")

df_Spaghetti_Polinomial_Val<-data.frame(cost=ceiling(c(1:100)/10),
degree=c(1:10),
error=Error.poly.val,
id="Validation")

df_Spaghetti_Polinomial<-rbind(df_Spaghetti_Polinomial_Train, df_Spaghetti_Polinomial_Val)
```

Plot

```
ggplot(df_Spaghetti_Polinomial, aes(x=cost, y=error, color=factor(degree), fill=interaction(id, degree)),
xlim=degree) +
geom_line()+
geom_point()+
labs(title = “Spaghetti plot polinomiale”, x = “Costo”, y = “Errore”, color = “Degree”)+
guides(fill=“none”)+
theme_bw()+
scale_y_continuous(limits = c(0,0.25))
```

Plot diviso

```
ggplot(df_Spaghetti_Polinomial, aes(x=cost, y=error, color=factor(id), fill=interaction(id, degree))) +
geom_line() +
geom_point() +
labs(title = “Spaghetti plot polinomiale diviso per Degree”, color = “Degree”)+
theme_bw() +
guides(fill=“none”)+
scale_y_continuous(limits = c(0,0.25)) +
facet_wrap(~degree)
```

Radiale

Creo i dataframe per i plot

```
df_Spaghetti_Radial_Train<-data.frame(cost=ceiling(c(1:100)/10),
gamma=c(1:10),
error=Error.rad.train,
id=“Training”)

df_Spaghetti_Radial_Val<-data.frame(cost=ceiling(c(1:100)/10),
gamma=c(1:10),
error=Error.rad.val,
id=“Validation”)
```

Plot

```
df_Spaghetti_Radial<-rbind(df_Spaghetti_Radial_Train, df_Spaghetti_Radial_Val)
ggplot(df_Spaghetti_Radial, aes(x=cost, y=error, color=factor(gamma), fill=interaction(id, gamma))) +
geom_line() +
geom_point() +
labs(title = “Spaghetti plot radiale”, x = “Costo”, y = “Errore”, color = “Gamma”)+
guides(fill=“none”)+
theme_bw() +
scale_y_continuous(limits = c(0,0.05))
```

Plot diviso

```
ggplot(df_Spaghetti_Radial, aes(x=cost, y=error, color=factor(id), fill=interaction(id, gamma))) +
geom_line() +
geom_point() +
labs(title = “Spaghetti plot radiale diviso per gamma”, color = “Gamma”)+
theme_bw() +
guides(fill=“none”)+
scale_y_continuous(limits = c(0,0.05)) +
facet_wrap(~gamma)
```

Valutazione performance

Ricerca del minimo

Lineare
modello<-'lineare'
modello
minimo.Lin<-min(MR.lin.total.Val)
minimo.Lin #Trova il valore minimo nel array degli errori del lineare

Polinomiale
modello<-'polinomiale'
modello
minimo.Poly<-min(MR.poly.total.Val) #Trova il valore minimo nella matrice degli errori del polinomiale
minimo.Poly

Radiale
modello<-'radiale'
modello
minimo.Rad<-min(MR.rad.total.Val) #Trova il valore minimo nella matrice degli errori del radiale
minimo.Rad

Confronto tutti i valori minimi per trovare IL minimo in assoluto
if(minimo.Lin<minimo.Poly){
if(minimo.Lin<minimo.Rad){
errore_minimo<-minimo.Lin
pos_errore_minimo<-which(MR.lin.total.Val==min(MR.lin.total.Val), arr.ind = TRUE)
x<-pos_errore_minimo[1]

applichiamo il machine learning con i parametri migliori
model<-svm(diagnosis ~ ., ds, kernel="linear", cost=x)
y.pred<-predict(model.SVC, ds.test)
Errore<-MR(y.pred, ds.test$diagnosis
) Accuratezza<-Acc(y.pred, ds.test$diagnosis)
modello<-"linear"
} else {
errore_minimo<-minimo.Rad
pos_errore_minimo<-which(MR.rad.total.Val==min(MR.rad.total.Val), arr.ind = TRUE)
y<-pos_errore_minimo[1,1]
x<-pos_errore_minimo[1,2]

applichiamo il machine learning con i parametri migliori
model.SVM<-svm(diagnosis~., ds, kernel="radial", cost=x, gamma=y/100)
y.pred<-predict(model.SVM, ds.test)
Errore<-MR(y.pred, ds.test$diagnosis)
Accuratezza<-Acc(y.pred, ds.test$diagnosis)
modello<-"radial"
}

```
} else {
if(minimo.Poly<minimo.Rad){
errore_minimo<-minimo.Poly
pos_errore_minimo<-which(MR.poly.total.Val==min(MR.poly.total.Val), arr.ind = TRUE)
y<-pos_errore_minimo[1,1]
x<-pos_errore_minimo[1,2]


applichiamo il machine learning con i parametri migliori
model.SVM<-svm(diagnosis~., ds, kernel="polynomial", cost=x, degree=y)
y.pred<-predict(model.SVM, ds.test)
Errore<-MR(y.pred, ds.test$diagnosis)
Accuratezza<-Acc(y.pred, ds.test$diagnosis)
modello<-"polynomial"


} else {
errore_minimo<-minimo.Rad
pos_errore_minimo<-which(MR.rad.total.Val==min(MR.rad.total.Val), arr.ind = TRUE)
x<-pos_errore_minimo[1,1]
y<-pos_errore_minimo[1,2]


applichiamo il machine learning con i parametri migliori
model.SVM<-svm(diagnosis~., ds, kernel="radial", cost=x, gamma=y/100)
y.pred<-predict(model.SVM, ds.test)
Errore<-MR(y.pred, ds.test$diagnosis)
Accuratezza<-Acc(y.pred, ds.test$diagnosis)
modello<-"radial"
}
}


errore_minimo #Stampo l'errore minimo
x #riga del minimo
y #colonna del minimo, eventuale perch? potrebbe essere lineare
Errore
Accuratezza
modello


Interpretazione probabilistica
SVM.probs <- svm(diagnosis ~ ., data=ds, kernel="radial", cost=x, gamma=y/100, probability=TRUE)
y.pred <- predict(SVM.probs, ds.test, probability=TRUE)
"probabilità di diagnosi"
y.pred <- attr(y.pred, "probabilities")
y.vec <- y.pred[, 2]
y.vec


"predizione del modello"
y.final <- rep(0, 85)
y.final[y.vec > 0.5] <-1
y.final


table(y.final, ds.test$diagnosis)
```

```
y.final <- rep(0, 85)
y.final[y.vec > 0.2] <- 1
y.final


table(y.final, ds.test$diagnosis)
```

Curva di ROC

```
pred <- prediction(y.vec, ds.test$diagnosis)
perf <- performance(pred, "tpr", "fpr")
```

AUC

```
auc <- performance(pred, "auc")
auc <- auc@y.values[[1]]


plot(perf, main=auc, colorize=TRUE)
```

Studio statistico dei risultati

SRS(k)

```
dstat<-dsBU #Carico il dataset
Nstat<-nrow(dstat)


Nstat.train<- 399
Nstat.val<-85
Nstat.test<-85


MR.radStatVector <- 1:50 #vettore che conterrà i 50 valori di MR (kernel radiale)
Acc.radStatVector <- 1:50


for (c in 1:50){
train.sample<-sample(Nstat, Nstat.train)
dstat.train<-ds[train.sample, ]
dstat.test<-ds[-train.sample, ]


val.sample<-sample(Nstat.test+Nstat.val, Nstat.val)
dstat.val<-ds.test [val.sample, ]
dstat.test<-dstat.test[-val.sample, ]


model.SVM<-svm(diagnosis~., dstat.train, kernel="radial", cost=x, gamma=y/100)
y.pred<-predict(model.SVM, dstat.train)


y.pred<-predict(model.SVM, dstat.test)
MR.radStat<-MR(y.pred, dstat.test$diagnosis)
Acc.radStat <- Acc(y.pred, dstat.test$diagnosis)
```

```
MR.radStatVector[c] <- MR.radStat
Acc.radStatVector[c] <- Acc.radStat
}
```

Vettori con 50 valori di MR e Acc ottenuti valutando modelli con kernel radiale

```
'Errore'
MR.radStatVector
'Accuratezza'
Acc.radStatVector
```

Statistica descrittiva

Calcolo del centro

Media

```
MR.radMean <- mean(MR.radStatVector)
MR.radMean

Acc.radMean <- mean(Acc.radStatVector)
Acc.radMean
```

Mediana

```
MR.radMedian <- median(MR.radStatVector)
MR.radMedian

Acc.radMedian <- median(Acc.radStatVector)
Acc.radMedian
```

Quantile semplice

```
MR.radQuant <- quantile(MR.radStatVector)
MR.radQuant

Acc.radQuant <- quantile(Acc.radStatVector)
Acc.radQuant
```

Plot quantili

```
sum.MR <- summary(MR.radStatVector)
sum.Acc <- summary(Acc.radStatVector)

boxplot(sum.MR, col="red",main="Box plot errore")
boxplot(sum.Acc, col="blue",main="Box plot accuratezza")
```

Diffusione dei dati

Varianza campionaria

```
MR.varCamp <- var(MR.radStatVector)
```

```
MR.varCamp

Acc.varCamp <- var(Acc.radStatVector)
Acc.varCamp

Daviazione standard
MR.sd <- sd(MR.radStatVector)
MR.sd

Acc.sd <- sd(Acc.radStatVector)
Acc.sd

Range interquartile (IQR)
MR.iqr <- IQR(MR.radStatVector)
MR.iqr

Acc.iqr <- IQR(Acc.radStatVector)
Acc.iqr

Simmetria
sym <- function(y){
m1 <- mean((y-mean(y))^3)
s <- m1/(sd(y)^3)
}

sym.MR <- sym(MR.radStatVector)
sym.MR

sym.Acc <- sym(Acc.radStatVector)
sym.Acc

Curtosi
cur <- function(y){
m2 <- mean((y-mean(y))^4)
c <- m2/(sd(y)^4)
}

cur.MR <- cur(MR.radStatVector)
cur.MR
cur.Acc <- cur(Acc.radStatVector)
cur.Acc

Deviazione assoluta dalla media (MAD)
MR.mad <- mad(MR.radStatVector)
MR.mad

Acc.mad <- mad(Acc.radStatVector)
Acc.mad
```

Statistica inferenziale

Distribuzione della media campionaria
mu <- mean(MR.radStatVector)
sigma <- sd(MR.radStatVector)
k <- 100 #numero di SRS che prendo
n <- 20 #dimensione del campione
hist(m, main="Distribuzione della media campionaria",breaks=20, col="green")
plot

Mostro la media calcolata dalle k volte in cui realizzo la media campionaria e la media esatta, poi calcolo l'errore
mu_c
mu
errM <- abs(mu-mu_c)
errM

Mostro la deviazione standard calcolata, che deve essere uguale o avvicinarsi a sigma/sqrt(n), poi calcolo l'errore.
sigma_c <- sd(m)
sigma_c
h <- sigma/sqrt(n)
h
errSD <- abs(h-sigma_c)
errSD

Distribuzione della variabile aleatoria Z
z <- (m-mu)/(sigma/sqrt(n))

Mostro media e deviazione standard di Z, la media si avviciner? a 0 e la deviazione standard a 1
zM <- mean(z)
zM
zSD <- sd(z)
zSD
hist(z, breaks=20, col="purple")

mu2 <- mean(MR.radStatVector)
sigma2 <- sd(MR.radStatVector)
k <- 100 #numero di SRS che prendo
n2 <- 35 #dimensione del campione

m2 <- replicate(k, mean(rnorm(n2, mu2, sigma2)))

hist(m2, main="Distribuzione della media campionaria",breaks=20, col="yellow")
plot

mu_c2 <- mean(m2)
mu_c2

```
mu2
errM2 <- abs(mu2-mu_c2)
errM2


sigma_c2 <- sd(m2)
sigma_c2
h2 <- sigma2/sqrt(n2)
h2
errSD2 <- abs(h2-sigma_c2)
errSD2


z2 <- (m2-mu2)/(sigma2/sqrt(n2))
hist(z2, breaks=20)


zM2 <- mean(z2)
zM2
zSD2 <- sd(z2)
zSD2


mu <- mean(MR.radStatVector)
sigma <- sd(MR.radStatVector)
n3 <- 50
alpha <- 0.05


x3 <- rnorm(n3, mu, sigma)
mCamp <- mean(x3) #media campionaria
S <- sd(x3) #deviazione standard campionaria


z_alpha <- qnorm(1-alpha, mean=0, sd=1)


Calcolo i 2 estremi dell'intervallo
infM <- mCamp - z_alpha*S/sqrt(n3)
supM <- mCamp + z_alpha*S/sqrt(n3)


infM
supM


Stampiamo la media esatta e vediamo se sta nell'intervallo di confidenza
mu


Proviamo a vedere su 100 prove se effettivamente rispetta la probabilità del 95%
k <- 0
V.infM <- 1:100
V.supM <- 1:100


for(i in 1:100){
y=rnorm(n3, mu, sigma)
mCamp1 <- mean(y)
```

```
S1<-sd(y)
V.infM[i] <- mCamp1 - z_alpha*S1/sqrt(n3)
V.supM[i] <- mCamp1 + z_alpha*S1/sqrt(n3)
if (V.infM[i] <= mu & mu <= V.supM[i]){
k <- k+1
}
}
```

Ora stampo il numero di volte in cui la media esatta si trova all'interno dell'intervallo di confidenza calcolato sulla media campionaria

```
k
```

Features Selection

Modello che usa tutti i features

```
model.SVM <- svm(diagnosis ~ ., ds, kernel="radial", cost=x, gamma=y/100)
y.pred <- predict(model.SVM, ds.test)

MR(y.pred, ds.test$diagnosis)
Acc(y.pred, ds.test$diagnosis)
```

togliano dati relativi a: Raggio

```
model.SVM <- svm(diagnosis ~ .  -radius_mean -radius_se -radius_worst, ds, kernel="radial", cost=x, gamma=y/100)
y.pred <- predict(model.SVM, ds.test)
MR(y.pred, ds.test$diagnosis)
Acc(y.pred, ds.test$diagnosis)
```

togliano dati relativi a: Perimetro

```
model.SVM <- svm(diagnosis ~ . -perimeter_mean -perimeter_se -perimeter_worst, ds, kernel="radial", cost=x, gamma=y/100)
y.pred <- predict(model.SVM, ds.test)
MR(y.pred, ds.test$diagnosis)
Acc(y.pred, ds.test$diagnosis)
```

togliano dati relativi a: Texture

```
model.SVM <- svm(diagnosis ~ . -texture_mean -texture_se -texture_worst, ds, kernel="radial", cost=x, gamma=y/100)
y.pred <- predict(model.SVM, ds.test)
MR(y.pred, ds.test$diagnosis)
Acc(y.pred, ds.test$diagnosis)
```

togliano dati relativi a: Smoothness

```
model.SVM <- svm(diagnosis ~ . -smoothness_mean -smoothness_se -smoothness_worst, ds, kernel="radial", cost=x, gamma=y/100)
y.pred <- predict(model.SVM, ds.test)
MR(y.pred, ds.test$diagnosis)
Acc(y.pred, ds.test$diagnosis)
```

togliano dati relativi a: Compactness
model.SVM <- svm(diagnosis ~ . -compactness_mean -compactness_se -compactness_worst, ds, kernel="radial", cost=x, degree=y/100)
y.pred <- predict(model.SVM, ds.test)
MR(y.pred, ds.test$diagnosis)
Acc(y.pred, ds.test$diagnosis)

togliano dati relativi a: Concavity
model.SVM <- svm(diagnosis ~ . -concavity_mean -concavity_se -concavity_worst, ds, kernel="radial", cost=x, gamma=y/100)
y.pred <- predict(model.SVM, ds.test)
MR(y.pred, ds.test$diagnosis)
Acc(y.pred, ds.test$diagnosis)

togliano dati relativi a: Concave
model.SVM <- svm(diagnosis ~ . -concave.points_mean -concave.points_se -concave.points_worst, ds, kernel="radial", cost=x, degree=y/100)
y.pred <- predict(model.SVM, ds.test)
MR(y.pred, ds.test$diagnosis)
Acc(y.pred, ds.test$diagnosis)

togliano dati relativi a: Simmetry
model.SVM <- svm(diagnosis ~ . -symmetry_mean -symmetry_se -symmetry_worst, ds, kernel="radial", cost=x, gamma=y/100)
y.pred <- predict(model.SVM, ds.test)
MR(y.pred, ds.test$diagnosis)
Acc(y.pred, ds.test$diagnosis)

togliano dati relativi a: fractal
model.SVM <- svm(diagnosis ~ . -fractal_dimension_mean -fractal_dimension_se -fractal_dimension_worst, ds, kernel="radial", cost=x, gamma=y/100)
y.pred <- predict(model.SVM, ds.test)
MR(y.pred, ds.test$diagnosis)
Acc(y.pred, ds.test$diagnosis)

Teniamo solamente i peggiori
model.SVM <- svm(diagnosis ~ radius_worst + perimeter_worst + area_worst +
texture_worst + smoothness_worst + compactness_worst +
concavity_worst + concave.points_worst + concavity_worst +
symmetry_worst + fractal_dimension_worst, ds, kernel="radial",
cost=x, gamma=y/100)

y.pred <- predict(model.SVM, ds.test)
MR(y.pred, ds.test$diagnosis)
Acc(y.pred, ds.test$diagnosis)