



**Università degli Studi di Bari**

---

DIPARTIMENTO DI INFORMATICA  
Corso di Laurea Magistrale in Informatica

PROGETTO DI INTELLIGENZA ARTIFICIALE

# CONFRONTO TRA DUE ALGORITMI DI APPRENDIMENTO

Esaminando:

**Giuseppe Rizzi**

Matricola 591275

Docenti:

**Prof. Floriana Esposito**

**Prof. Nicola Di Mauro**

---

# Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
<b>2</b>	<b>Descrizione dei dati</b>	<b>3</b>
2.1	German Credit dataset . . . . .	3
2.2	Image Segmentation dataset . . . . .	4
2.3	Vehicle Silhouettes dataset . . . . .	5
2.4	Wisconsin Breast Cancer dataset . . . . .	6
<b>3</b>	<b>REPTree</b>	<b>8</b>
3.1	Information Gain . . . . .	8
3.2	Reduced Error Pruning . . . . .	9
<b>4</b>	<b>RIPPER</b>	<b>11</b>
4.1	Incremental Reduced Error Pruning . . . . .	12
4.2	Miglioramenti ad IREP . . . . .	14
<b>5</b>	<b>Esecuzione</b>	<b>17</b>
5.1	Risultati su German Credit . . . . .	17
5.2	Risultati su Image Segmentation . . . . .	22
5.3	Risultati su Vehicle Silhouettes . . . . .	26
5.4	Risultati su Wisconsin Breast Cancer . . . . .	30

# Capitolo 1: Introduzione

Il seguente lavoro si propone di confrontare due algoritmi di apprendimento supervisionati, *REPTree* e *RIPPER*: il primo sfrutta la metodologia di costruzione di alberi di decisione, il secondo quello di costruzione di regole.

Verranno testati su quattro dataset messi a disposizione dall'*UCI Machine Learning Repository*<sup>1</sup>, procedendo con la presentazione dei risultati e dei modelli di predizione ottenuti.

Il software utilizzato è *Weka*<sup>2</sup>, una suite di algoritmi di *machine learning*, fortemente utilizzato sia in ambito accademico che industriale.

---

<sup>1</sup><https://archive.ics.uci.edu/ml/datasets.html>

<sup>2</sup><http://www.cs.waikato.ac.nz/ml/weka/>

# Capitolo 2: Descrizione dei dati

Di seguito vengono descritti i 4 dataset utilizzati nella sperimentazione

## 2.1 German Credit dataset

Il dataset contiene informazioni in ambito finanziario su clienti ritenuti a rischio o meno.

- Numero di istanze: 1000
- Numero di attributi: 21
- Attributo target: **class**
- Valori target: {good, bad}

Attributo	Tipo
checking_status	nominal
duration	numeric
credit_history	nominal
purpose	nominal
credit_amount	numeric
savings_status	nominal
employment	nominal
installment_commitment	numeric
personal_status	nominal
other_parties	nominal
residence_since	numeric
property_magnitude	nominal
age	numeric
other_payment_plans	nominal
housing	nominal
existing_credits	numeric
job	nominal
num_dependents	numeric
own_telephone	nominal
foreign_worker	nominal
<b>class</b>	nominal

## 2.2 Image Segmentation dataset

Il dataset contiene informazioni di sette immagini all'aperto che sono state segmentate a mano. Ogni istanza rappresenta una regione 3x3.

- Numero di istanze: 2310
- Numero di attributi: 20
- Attributo target: **class**
- Valori target: {brickface, sky, foliage, cement, window, path, grass}

Attributo	Tipo
region-centroid-col	numeric
region-centroid-row	numeric
region-pixel-count	numeric
short-line-density-5	numeric
short-line-density-2	numeric
vedge-mean	numeric
vegde-sd	numeric
hedge-mean	numeric
hedge-sd	numeric
intensity-mean	numeric
rawred-mean	numeric
rawblue-mean	numeric
rawgreen-mean	numeric
exred-mean	numeric
exblue-mean	numeric
exgreen-mean	numeric
value-mean	numeric
saturation-mean	numeric
hue-mean	numeric
<b>class</b>	nominal

## 2.3 Vehicle Silhouettes dataset

Il dataset contiene informazioni per discriminare le silhouette di diversi veicoli tra automobili, van e bus.

- Numero di istanze: 846
- Numero di attributi: 19
- Attributo target: **Class**
- Valori target: {opel, saab, bus, ban}

Attributo	Tipo.
COMPACTNESS	numeric
CIRCULARITY	numeric
DISTANCE CIRCULARITY	numeric
RADIUS RATIO	numeric
PR.AXIS ASPECT RATIO	numeric
MAX.LENGTH ASPECT RATIO	numeric
SCATTER RATIO	numeric
ELONGATEDNESS	numeric
PR.AXIS RECTANGULARITY	numeric
MAX.LENGTH RECTANGULARITY	numeric
SCALED VARIANCE_MAJOR	numeric
SCALED VARIANCE_MINOR	numeric
SCALED RADIUS OF GYRATION	numeric
SKEWNESS ABOUT_MAJOR	numeric
SKEWNESS ABOUT_MINOR	numeric
KURTOSIS ABOUT_MAJOR	numeric
KURTOSIS ABOUT_MINOR	numeric
HOLLOWS RATIO	numeric
<b>Class</b>	nominal

## 2.4 Wisconsin Breast Cancer dataset

Il dataset contiene informazioni riguardo a vari casi di tumore al seno, che permettono di stabilire se esso è benigno o maligno.

- Numero di istanze: 699
- Numero di attributi: 10
- Attributo target: **Class**
- Valori target: {benign, malignant}

Attributo	Tipo
Clump_Thickness	numeric
Cell_Size_Uniformity	numeric
Cell_Shape_Uniformity	numeric
Marginal_Adhesion	numeric
Single_Epi_Cell_Size	numeric
Bare_Nuclei	numeric
Bland_Chromatin	numeric
Normal_Nucleoli	numeric
Mitoses	numeric
<b>Class</b>	nominal



# Capitolo 3: REPTree

Come primo algoritmo si è scelto di usare **REPTree**, che costruisce alberi di decisione usando l'*information gain* per i valori nominali e la varianza per i valori numerici[16].

Visto che sono stati presi in considerazione dataset con attributi di classe nominali per un task di classificazione e non di regressione, verrà discusso il criterio dell'*information gain*, analogo all'algoritmo *C4.5*[9].

## 3.1 Information Gain

Per selezionare l'attributo che meglio classifica i dati  $D$  ed in particolare, su quale dei suoi valori occorre fare uno *split*, può convenire usare l'*entropia*[13], ossia l'incertezza contenuta nei dati, che è calcolata come:

$$E(D) = - \sum_i p_i \log_2 p_i$$

cioè la media dei logaritmi delle probabilità di ciascun oggetto  $i$  pesato per la probabilità stessa. Nel contesto di classificazione, gli oggetti sono gli esempi nel dataset di cui viene calcolata la probabilità che essi appartengano o meno ad una delle classi presenti nell'attributo target. Più è probabile che un esempio appartenga ad una certa classe, più la sua influenza nel calcolo della media sarà mitigata dal logaritmo della sua stessa probabilità.

È possibile calcolare l'entropia anche per sottoinsiemi del dataset, in particolare quegli esempi  $D_v$  che presentano lo stesso valore  $v$  di un certo attributo  $a$ , poi sommare tutte le entropie relative a tutti i valori  $V$  dell'attributo per ottenere l'entropia dei dati dopo aver preso in considerazione l'attributo  $a$ . Ogni entropia viene pesata per il numero di esempi che presentano quel valore diviso per il totale di esempi esistenti nel dataset:

$$E(D|a) = \sum_{v \in V} \frac{|D_v|}{|D|} \cdot E(D_v)$$

Da queste formule si ricava l'information gain, cioè la riduzione di incertezza totale prendendo in considerazione un attributo  $a$ :

$$IG = E(D) - E(D|a)$$

Come radice verrà utilizzato l'attributo che massimizza l'information gain, come archi i valori dell'attributo e si ripete la procedura per i nodi figli fino a generare le foglie.

## 3.2 Reduced Error Pruning

Per evitare l'*overfitting*, ossia un sovra-adattamento del modello ai dati di training che compromette la bontà delle sue predizioni su nuovi esempi, può essere ragionevole semplificare il modello, rischiando di commettere qualche errore ma garantendoci una migliore copertura per dati non visti.

Questa semplificazione viene chiamata *pruning* (potatura), in cui, una volta costruito il modello utilizzando i dati del *growing set*, esso viene testato su una parte dei dati, accantonati e non adoperati per la predizione, che fanno parte del *pruning set*.

Una tecnica di potatura è REP (**R**educed **E**rror **P**runing)[8] che utilizza un pruning set per stimare l'accuratezza dei nodi intermedi e confrontarla con quella dei suoi sottoalberi.

Viene calcolato il guadagno dall'eventuale potatura sottraendo il numero di errori (esempi classificati scorrettamente) al sottoalbero  $T$  al numero di errori al nodo radice  $v$  del sottoalbero:

$$Gain_{REP} = \varepsilon_T - \varepsilon_v$$

L'albero è potato se il guadagno è positivo quando vengono commessi più errori nell'intero sottoalbero, e non al nodo. C'è un'altra condizione da rispettare per procedere alla potatura: può avvenire solo se il sottoalbero  $T$  non ha un sottoalbero che ha un tasso d'errore minore di  $T$  stesso (*bottom-up restriction*).

L'algoritmo di REP è il seguente:

- Si parte dall'albero completo e lo si visita in post-ordine.
- Per ogni nodo intermedio  $v$ 
  - Calcolo l'accuratezza sul pruning set dell'albero completo.
  - Calcolo l'accuratezza sul pruning set rispetto a  $v$  e al suo sottoalbero  $T$ .

- Se l'accuratezza aumenta, pota. In caso di uguaglianza pota per semplificare (rasoio di Occam).

Inoltre è dimostrato che, tra tutti i possibili sottoalberi potati che è possibile generare, REP trova il sottoalbero più piccolo e più accurato rispetto al pruning set[4].

## Capitolo 4: RIPPER

Come secondo algoritmo si è scelto di usare **RIPPER**[3], in particolare nella versione implementata da Weka, **JRip**[17].

RIPPER (**R**epeated **I**ncremental **P**runing to **P**roduce **E**rror **R**eduction) è un algoritmo di induzione di regole proposto da William W. Cohen nel 1995. Esso si è dimostrato competitivo con C4.5Rules rispetto ai tassi di errore, scala in maniera lineare con il numero di esempi di training e può elaborare in maniera efficiente dataset rumorosi che contengono centinaia di migliaia di esempi. RIPPER si basa su IREP (**I**ncremental **R**educed **E**rror **P**runing))[5], di cui si discuterà nei prossimi paragrafi.

Molte delle tecniche usate nei moderni sistemi di apprendimento di regole sono state adattate dall'apprendimento degli alberi di decisione. La maggior parte dei sistemi di apprendimento di alberi di decisione usa una strategia di apprendimento *overfit-and-simplify* (sovradata-e-semplifica) per gestire dati rumorosi: viene generata un'ipotesi prima facendo crescere un albero complesso che "overfitta" i dati, e poi si semplifica o pota tale albero (un'operazione di *pruning*). Una tecnica di pruning efficace è *reduced error pruning* (*REP*), discussa in 3.2. Essa può essere facilmente adattata ai sistemi di apprendimento di regole[6][1].

In REP per le regole, il training set viene diviso in *growing set* e *pruning set*. All'inizio, viene creato un *rule set* di partenza che overfitta il growing set, usando qualche metodo euristico. Questo rule set spropositato viene poi semplificato ripetutamente applicando qualche operatore di pruning. Ad ogni fase di semplificazione, l'operatore di pruning scelto è quello che produce la più grande riduzione di errore sul pruning set. La semplificazione finisce quando il tasso di errore non si riduce ulteriormente applicando gli operatori di pruning.

REP per le regole di solito migliora davvero la performance di generalizzazione sui dati rumorosi[6][1][14][5]; tuttavia è computazionalmente costoso per grandi dataset[2].

In risposta all'inefficienza di REP, Fürnkranz e Widmer [1994] proposero un algoritmo di apprendimento chiamato *incremental reduced error pruning*

(IREP)[5].

## 4.1 Incremental Reduced Error Pruning

L'idea di usare un pruning set separato per la potatura è REP. La variante che pota una regola subito dopo averla "fatta crescere" si chiama *incremental reduced error pruning* (IREP)[15]. Quest'ultima integra saldamente REP con un algoritmo di apprendimento di regole *separate-and-conquer*. L'algoritmo 1 ne presenta una versione a due classi. Come ogni algoritmo *separate-and-conquer* standard, IREP costruisce un ruleset in maniera *greedy*, una regola alla volta. Dopo averne trovata una, tutti gli esempi coperti da quella regola (sia positivi che negativi) sono cancellati. Questo processo si ripete finché non ci sono più esempi positivi, o finché la regola trovata da IREP non presenta un grande tasso di errore, cosa inaccettabile.

Per costruire una regola, IREP usa la seguente strategia. Prima, gli esempi non coperti sono partizionati a caso in due sottoinsiemi, un growing e un pruning set. Nell'implementazione di Cohen, il growing set contiene 2/3 degli esempi.

Poi, una regola viene "fatta crescere". L'implementazione di Cohen di *GrowRule* è una versione proposizionale di FOIL (First Order Inductive Learner), dove i letterali non si servono di predicati ma di uguaglianze (per valori discreti) e confronti numerici (per valori continui)[12]. Esso inizia con una congiunzione vuota di condizioni (la regola vuota) e considera di aggiungere a questa qualsiasi condizione nella forma  $A_d = v$ ,  $A_c \leq \theta$  oppure  $A_c \geq \theta$  dove  $A_d$  è un attributo discreto e  $v$  è un valore che può assumere, mentre  $A_c$  è un attributo continuo e  $\theta$  è un valore soglia. *GrowRule* aggiunge ripetutamente la condizione che massimizza un'euristica di *information gain*, nello specifico quella di FOIL, finché la regola non copre più esempi negativi nel growing set.

Siano  $R_0$  e  $R_1$  due regole, la seconda ottenuta dall'aggiunta di una condizione nel corpo della prima. L'information gain viene così calcolato:

$$Gain_{IREP}(R_0, R_1) = t \cdot \left( \log \frac{p_1}{p_1 + n_1} - \log \frac{p_0}{p_0 + n_0} \right)$$

dove  $t$  riguarda gli esempi positivi coperti da  $R_0$  che soddisfano anche  $R_1$  dopo aver aggiunto una condizione,  $p_0$  (rispettivamente  $p_1$ ) sono gli esempi positivi coperti da  $R_0$  (rispettivamente  $R_1$ ) e  $n_0$  (rispettivamente  $n_1$ ) sono gli esempi negativi coperti da  $R_0$  (rispettivamente  $R_1$ ).

L'idea alla base è che l'informazione totale che si guadagna è dato dal numero di tuple che soddisfano la nuova condizione moltiplicato l'informazione guadagnata in merito a ciascuna[7].

Dopo aver espanso una regola, essa viene immediatamente potata. Per prunarla, l'implementazione di Cohen cancella qualsiasi sequenza finale di condizioni dalla regola e sceglie l'eliminazione che massimizza la funzione

$$v(Rule, PrunePos, PruneNeg) \equiv \frac{p + (N - n)}{P + N} \quad (4.1)$$

dove  $P$  (rispettivamente  $N$ ) è il numero totale di esempi in  $PrunePos$  ( $PruneNeg$ ) e  $p$  ( $n$ ) è il numero di esempi in  $PrunePos$  ( $PruneNeg$ ) coperti da  $Rule$ . Questo processo è ripetuto finché nessun'altra cancellazione migliora il valore di  $v$ .

L'algoritmo IREP descritto sopra è per i problemi di apprendimento a due classi. L'implementazione di Cohen gestisce classi multiple, come spiegato di seguito:

1. Le classi vengono ordinate secondo la prevalenza, cioè l'ordine è  $C_1, \dots, C_k$  dove  $C_1$  è la classe di minoranza e  $C_k$  è la classe di maggioranza.
2. Viene trovata una regola che separi  $C_1$  dal resto delle classi; questo viene fatto con una singola chiamata ad IREP dove  $PosData$  contiene gli esempi di classe  $C_1$  e  $NegData$  contiene gli esempi di classi  $C_2, C_3, \dots, C_k$ .
3. Tutte le istanze coperte dal ruleset appena addestrato sono rimosse dal dataset e IREP si appresta a separare  $C_2$  dalle classi  $C_3, \dots, C_k$ .
4. Si ripete finché rimane la sola classe  $C_k$ . Quest'ultima verrà usata come classe di default.

L'implementazione di Cohen differisce da quella di Fürnkranz e Widmer sotto molti aspetti. Quando le regole vengono potate, la nuova implementazione permette di cancellare qualsiasi sequenza finale di condizioni, mentre l'implementazione di Fürnkranz e Widmer permette solo la cancellazione di una singola condizione finale. L'algoritmo rivisitato permette anche di fermare l'aggiunta di regole al ruleset quando la regola appresa ha un tasso di errore superiore al 50%, mentre quello di Fürnkranz e Widmer la ferma quando l'accuratezza della regola è minore dell'accuratezza della regola vuota.

---

**Algoritmo 1** IREP( $Pos, Neg$ )

---

```
1:  $Ruleset \leftarrow \emptyset$ 
2: while  $Pos \neq \emptyset$  do
3:   dividi  $(Pos, Neg)$  in  $(GrowPos, GrowNeg)$  e  $(PrunePos, PruneNeg)$ 
4:    $Rule \leftarrow GrowRule(GrowPos, GrowNeg)$ 
5:    $Rule \leftarrow PruneRule(Rule, PrunePos, PruneNeg)$ 
6:   if il tasso di errore su  $(PrunePos, PruneNeg) > 50\%$  then
7:     return  $Ruleset$ 
8:   else
9:     aggiungi  $Rule$  a  $Ruleset$ 
10:    rimuovi gli esempi coperti da  $Rule$  da  $(Pos, Neg)$ 
11: return  $Ruleset$ 
```

---

## 4.2 Miglioramenti ad IREP

Sono state implementate tre modifiche ad IREP: una metrica alternativa per determinare il valore delle regole nella fase di potatura; una nuova euristica per decidere quando fermare l'aggiunta di regole al ruleset; un successivo passaggio di "ottimizzazione" del ruleset per tentare di avvicinarsi di più al REP convenzionale (cioè, non incrementale).

### Metrica per il valore delle regole

Il fallimento occasionale di IREP a convergere al crescere del numero degli esempi può essere facilmente fatto risalire alla metrica usata per guidare la potatura (ossia la (4.1)). Le scelte intraprese nella definizione di tale metrica non sono intuitive; per esempio (assumendo che  $P$  e  $N$  siano fissati) la metrica preferisce una regola  $R_1$  che copre  $p_1 = 2000$  esempi positivi e  $n_1 = 1000$  esempi negativi rispetto ad una regola  $R_2$  che copre  $p_2 = 1000$  esempi positivi e  $n_2 = 1$  esempio negativo; si noti comunque che  $R_2$  è altamente predittiva, al contrario di  $R_1$ . Quindi si è deciso di sostituire la metrica di IREP con

$$v^*(Rule, PrunePos, PruneNeg) \equiv \frac{p - n}{p + n}$$

che sembra avere un comportamento più intuitivo e soddisfacente.

### Condizione di stop

L'implementazione di IREP di Cohen si ferma in maniera greedy aggiungendo regole al ruleset quando l'ultima regola costruita ha un tasso d'errore

maggiore del 50% sui dati di pruning. Questa euristica, spesso, si ferma troppo presto con campioni di dimensioni moderate; questo è vero soprattutto quando si apprende un concetto con regole a bassa copertura (pochi esempi coperti).

La soluzione a questo problema è la seguente. Dopo l'aggiunta di ogni regola, viene calcolata la *description-length* totale del ruleset e degli esempi. La nuova versione di IREP ferma l'aggiunta di regole quando questa *description-length* è maggiore di  $d$  bit rispetto alla più piccola *description-length* ottenuta sinora, o quando non ci sono più esempi positivi. Nell'implementazione si è usato  $d = 64$ . Il ruleset viene poi semplificato esaminando ogni regola a turno (cominciando dall'ultima) e cancellando regole così da ridurre la *description-length* totale.

Il principio *MDL* (Minimum Description Length) può essere meglio espresso immaginando un modello di comunicazione in cui un mittente trasmette ad un ricevente una descrizione che consiste in una teoria  $T$  e i dati  $D$  da cui essa è derivata[11].

Il metodo usato per la codifica è lo stesso usato in *C4.5rules*[10]. Esso parte da un *bias* in cui il numero di falsi positivi e falsi negativi sia lo stesso e si procede come segue: i messaggi da inviare si presentano con probabilità  $p_j$ , e servono  $-\log(p_j)$  bit (in base 2) per costruirli: più un messaggio è frequente, meno bit saranno necessari per rappresentarlo. Si inviano i dati codificati, poi anziché inviare i messaggi di errore per tutti i dati, il mittente prima trasmette gli errori  $e$  nei  $C$  casi coperti dalla teoria e poi negli  $U$  casi non coperti. Sotto l'assunzione che i falsi positivi  $fp$  e i falsi negativi  $fn$  siano bilanciati, la probabilità di errore nei casi coperti è  $e/2C$  e questa probabilità è usata per codificare i messaggi di errore per i casi coperti. Una volta che i falsi positivi sono stati identificati, il destinatario può calcolare il vero numero dei falsi negativi come  $e - fp$ , quindi la probabilità di errore per i casi non coperti è  $fn/U$ . Il costo totale quindi diventa:

$$\begin{aligned}
& \log(|D| + 1) \\
& + fp \times (-\log(\frac{e}{2C})) \\
& + (C - fp) \times (-\log(1 - \frac{e}{2C})) \\
& + fn \times (-\log(\frac{fn}{U})) \\
& + (U - fn) \times (-\log(1 - \frac{fn}{U}))
\end{aligned}$$



## Ottimizzazione delle regole

L'approccio ripetuto *grow-and-simplify* usato in IREP può produrre risultati abbastanza differenti dal REP convenzionale (non incrementale). Un modo per migliorarlo è elaborare a posteriori le regole prodotte da IREP così da avvicinarsi di più all'effetto del REP convenzionale. Per esempio, si potrebbe ri-potare ogni regola al fine di minimizzare l'errore del ruleset completo.

Il metodo sviluppato per ottimizzare un ruleset  $R_1, R_2, \dots, R_k$  consiste nel costruire due regole alternative per ogni  $R_i$ . La *sostituta* di  $R_i$  viene generata espandendo e poi potando  $R_i$ . La *revisione* di  $R_i$  viene generata in maniera analoga, tranne per il fatto che la revisione è espansa in modo greedy aggiungendo condizioni a  $R_i$ , piuttosto che alla regola vuota. Infine si sceglie tra le tre regole quale includere nella teoria. Questa decisione viene presa in base all'euristica MDL. L'implementazione di questo metodo in IREP avviene in questo modo:

1. Viene usato IREP per ottenere un ruleset iniziale.
2. Esso viene ottimizzato, come descritto sopra.
3. Vengono aggiunte le regole in modo tale da coprire gli esempi positivi rimanenti.

L'ottimizzazione può essere ripetuta più volte elaborando il ruleset ottenuto dalla passata precedente dell'algoritmo.

IREP, con l'aggiunta del passo di post-ottimizzazione, forma un nuovo algoritmo che è stato chiamato **RIPPER** (**R**epeated **I**ncremental **P**runing to **P**roduce **E**rror **R**eduction).

L'implementazione in Weka di RIPPER si chiama JRip.

# Capitolo 5: Esecuzione

Qui vengono confrontati i due algoritmi REPTree e RIPPER/JRip. Entrambi hanno sfruttato una *10-fold cross validation*.

## 5.1 Risultati su German Credit

Esecuzione REPTree	
=== Run information ===	
Scheme:weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1	
Relation:	german_credit
Instances:	1000
Attributes:	21
	checking_status
	duration
	credit_history
	purpose
	credit_amount
	savings_status
	employment
	installment_commitment
	personal_status
	other_parties
	residence_since
	property_magnitude
	age
	other_payment_plans
	housing
	existing_credits
	job
	num_dependents
	own_telephone
	foreign_worker
	class
Test mode:10-fold cross-validation	

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	718	71.8	%
Incorrectly Classified Instances	282	28.2	%
Kappa statistic	0.2702		
Mean absolute error	0.3417		
Root mean squared error	0.4424		
Relative absolute error	81.3157	%	
Root relative squared error	96.532	%	
Total Number of Instances	1000		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.859	0.61	0.767	0.859	0.81	0.72	good
	0.39	0.141	0.542	0.39	0.453	0.72	bad
Weighted Avg.	0.718	0.469	0.699	0.718	0.703	0.72	

=== Confusion Matrix ===

a	b	<-- classified as
601	99	a = good
183	117	b = bad



Figura 5.1: Modello di REPTree

## Esecuzione JRip

=== Run information ===

Scheme:weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1

Relation:        german\_credit

Instances:       1000

Attributes:      21

checking\_status  
duration  
credit\_history  
purpose  
credit\_amount  
savings\_status  
employment  
installment\_commitment  
personal\_status  
other\_parties  
residence\_since  
property\_magnitude  
age  
other\_payment\_plans  
housing  
existing\_credits  
job  
num\_dependents  
own\_telephone  
foreign\_worker  
class

Test mode:10-fold cross-validation

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	717	71.7	%
Incorrectly Classified Instances	283	28.3	%
Kappa statistic	0.2513		
Mean absolute error	0.3781		
Root mean squared error	0.4472		
Relative absolute error	89.9974	%	
Root relative squared error	97.5906	%	
Total Number of Instances	1000		

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.873	0.647	0.759	0.873	0.812	0.593	good
0.353	0.127	0.544	0.353	0.428	0.593	bad

```
Weighted Avg.    0.717    0.491    0.694    0.717    0.697    0.593
```

```
=== Confusion Matrix ===
```

```
   a   b  <-- classified as
611  89 |   a = good
194 106 |   b = bad
```

Regole:

1. (checking\_status = "<0") **AND** (job = skilled)  $\Rightarrow$  class=bad (172.0/76.0)
2. (checking\_status = "0<=X<20") **AND** (duration >= 24) **AND** (savings\_status = "<100")  $\Rightarrow$  class=bad (61.0/19.0)
3. [Empty Rule]  $\Rightarrow$  class=good (767.0/162.0)

## 5.2 Risultati su Image Segmentation

### Esecuzione REPTree

=== Run information ===

```

Scheme:      weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0
Relation:    segment
Instances:   2310
Attributes:  20
              region-centroid-col
              region-centroid-row
              region-pixel-count
              short-line-density-5
              short-line-density-2
              vedge-mean
              vegde-sd
              hedge-mean
              hedge-sd
              intensity-mean
              rawred-mean
              rawblue-mean
              rawgreen-mean
              exred-mean
              exblue-mean
              exgreen-mean
              value-mean
              saturation-mean
              hue-mean
              class
Test mode:   10-fold cross-validation

```

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	2196	95.0649 %
Incorrectly Classified Instances	114	4.9351 %
Kappa statistic	0.9424	
Mean absolute error	0.0186	
Root mean squared error	0.1108	
Relative absolute error	7.5874 %	
Root relative squared error	31.665 %	
Total Number of Instances	2310	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,973	0,007	0,961	0,973	0,967	0,961	0,996	0,978	brickface
	1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	sky
	0,921	0,013	0,924	0,921	0,923	0,910	0,979	0,922	foliage
	0,909	0,014	0,917	0,909	0,913	0,899	0,983	0,952	cement
	0,894	0,023	0,865	0,894	0,879	0,859	0,970	0,870	window
	0,979	0,002	0,991	0,979	0,985	0,982	0,998	0,992	path
	0,979	0,000	1,000	0,979	0,989	0,988	0,994	0,989	grass
Weighted Avg.	0,951	0,008	0,951	0,951	0,951	0,943	0,988	0,957	

=== Confusion Matrix ===

a b c d e f g <-- classified as

```

321  0  0  3  6  0  0 | a = brickface
0 330  0  0  0  0  0 | b = sky
2  0 304  4 20  0  0 | c = foliage
5  0  4 300 18  3  0 | d = cement
5  0 18 12 295  0  0 | e = window
0  0  0  7  0 323  0 | f = path
1  0  3  1  2  0 323 | g = grass

```

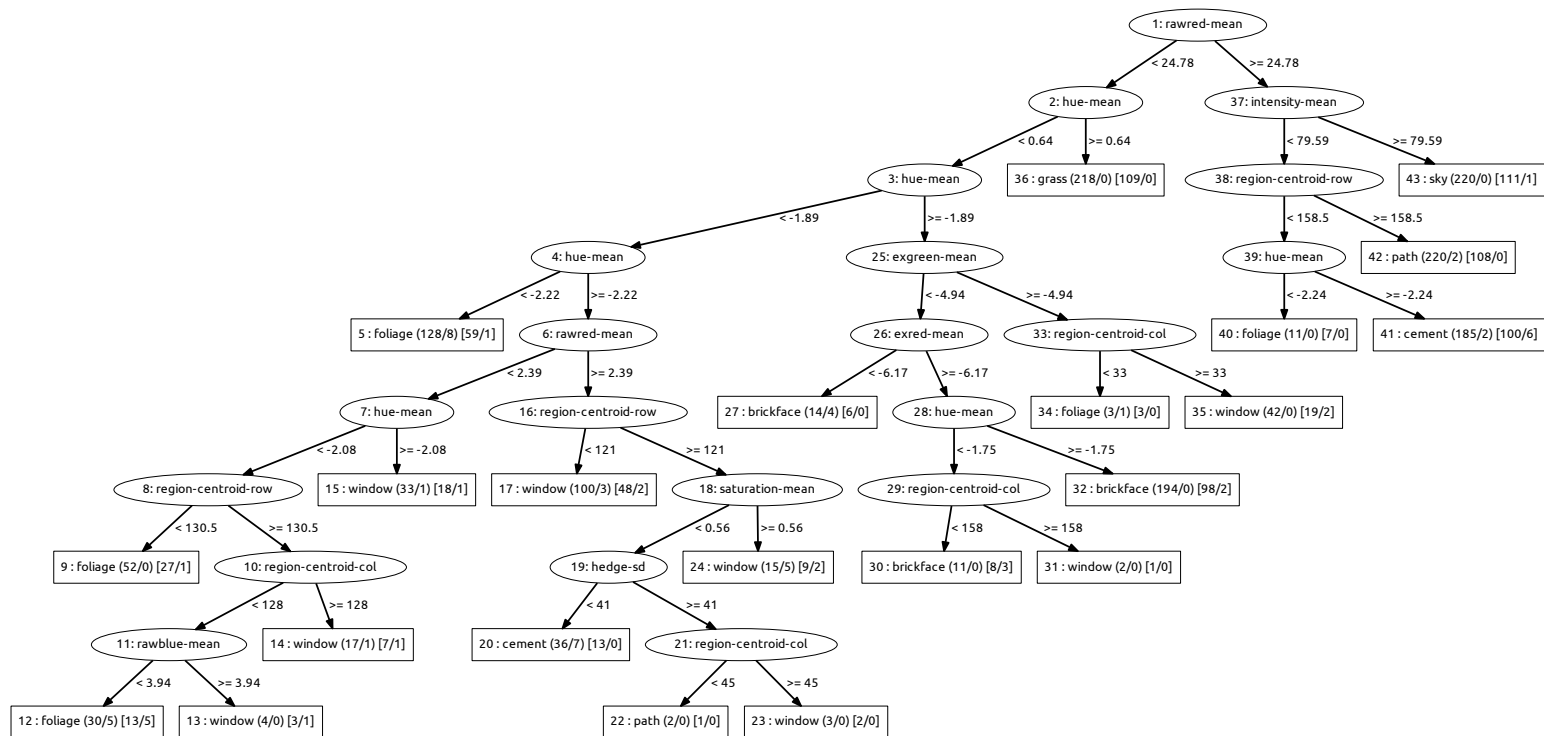


Figura 5.2: Modello di REPTree

### Esecuzione JRip

=== Run information ===

```

Scheme:      weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1
Relation:    segment
Instances:   2310
Attributes:  20
              region-centroid-col
              region-centroid-row
              region-pixel-count
              short-line-density-5
              short-line-density-2

```



```

    vedge-mean
    vegde-sd
    hedge-mean
    hedge-sd
    intensity-mean
    rawred-mean
    rawblue-mean
    rawgreen-mean
    exred-mean
    exblue-mean
    exgreen-mean
    value-mean
    saturation-mean
    hue-mean
    class
Test mode:    10-fold cross-validation

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      2204           95.4113 %
Incorrectly Classified Instances    106           4.5887 %
Kappa statistic                    0.9465
Mean absolute error                 0.0172
Root mean squared error             0.1115
Relative absolute error              7.0261 %
Root relative squared error         31.8519 %
Total Number of Instances          2310

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
      0,982   0,005   0,970     0,982   0,976     0,972   0,991   0,958   brickface
      0,997   0,000   1,000     0,997   0,998     0,998   0,998   0,997   sky
      0,927   0,018   0,897     0,927   0,912     0,897   0,972   0,899   foliage
      0,930   0,012   0,930     0,930   0,930     0,919   0,977   0,932   cement
      0,864   0,014   0,913     0,864   0,888     0,870   0,965   0,874   window
      0,988   0,005   0,970     0,988   0,979     0,975   0,996   0,986   path
      0,991   0,001   0,997     0,991   0,994     0,993   0,999   0,995   grass
Weighted Avg.   0,954   0,008   0,954     0,954   0,954     0,946   0,986   0,949

=== Confusion Matrix ===

  a  b  c  d  e  f  g  <-- classified as
324  0  1  3  2  0  0 |  a = brickface
  0 329  0  1  0  0  0 |  b = sky
  2  0 306  6 14  2  0 |  c = foliage
  3  0  6 307 10  3  1 |  d = cement
  5  0 24 11 285  5  0 |  e = window
  0  0  3  1  0 326  0 |  f = path
  0  0  1  1  1  0 327 |  g = grass

```

## Regole:

1. (intensity-mean  $\geq$  26.1111) **AND** (hue-mean  $\geq$  -2.17447) **AND** (region-centroid-row  $\leq$  159) **AND** (intensity-mean  $\leq$  72.8889) **AND** (rawgreen-mean  $\geq$  22.3333)  $\Rightarrow$  class=cement (281.0/0.0)
2. (vedge-mean  $\geq$  1.72222) **AND** (region-centroid-row  $\leq$  160) **AND** (region-centroid-row  $\geq$  146) **AND** (hedge-sd  $\leq$  1.86667) **AND** (saturation-mean  $\leq$  0.541667)  $\Rightarrow$  class=cement (20.0/1.0)

3. (region-centroid-row  $\geq 123$ ) **AND** (hue-mean  $\leq -2.10408$ ) **AND** (hue-mean  $\geq -2.17535$ ) **AND** (rawred-mean  $\geq 8$ ) **AND** (region-centroid-row  $\leq 156$ )  $\Rightarrow$  class=cement (19.0/1.0)
4. (intensity-mean  $\geq 86.2963$ )  $\Rightarrow$  class=sky (330.0/0.0)
5. (hue-mean  $\geq 1.28706$ )  $\Rightarrow$  class=grass (327.0/0.0)
6. (hedge-mean  $\leq 0.777777$ ) **AND** (region-centroid-col  $\geq 128$ ) **AND** (saturation-mean  $\leq 0.533928$ ) **AND** (exred-mean  $\leq 0.111111$ )  $\Rightarrow$  class=window (91.0/0.0)
7. (rawred-mean  $\leq 18.2222$ ) **AND** (region-centroid-col  $\geq 152$ ) **AND** (rawblue-mean  $\geq 9.55556$ ) **AND** (hue-mean  $\geq -2.20829$ )  $\Rightarrow$  class=window (82.0/0.0)
8. (intensity-mean  $\leq 3.7037$ ) **AND** (hue-mean  $\geq -2.08783$ ) **AND** (region-centroid-col  $\geq 34$ )  $\Rightarrow$  class=window (62.0/1.0)
9. (hue-mean  $\leq -2.0793$ ) **AND** (hue-mean  $\geq -2.21646$ ) **AND** (rawred-mean  $\geq 0.666667$ ) **AND** (rawred-mean  $\leq 25.6667$ ) **AND** (exgreen-mean  $\leq -6.22222$ ) **AND** (exblue-mean  $\leq 33.6667$ )  $\Rightarrow$  class=window (51.0/2.0)
10. (vedge-mean  $\leq 0.277778$ ) **AND** (region-centroid-row  $\geq 131$ ) **AND** (region-centroid-col  $\geq 125$ )  $\Rightarrow$  class=window (8.0/1.0)
11. (exgreen-mean  $\geq -6.11111$ ) **AND** (region-centroid-row  $\geq 133$ ) **AND** (hue-mean  $\geq -2.1753$ ) **AND** (exgreen-mean  $\leq -3.11111$ ) **AND** (region-centroid-col  $\geq 38$ )  $\Rightarrow$  class=window (18.0/3.0)
12. (intensity-mean  $\leq 2.96296$ ) **AND** (region-centroid-row  $\geq 133$ ) **AND** (rawred-mean  $\geq 0.888889$ )  $\Rightarrow$  class=window (5.0/0.0)
13. (exgreen-mean  $\geq -6.33333$ ) **AND** (region-centroid-row  $\leq 133$ )  $\Rightarrow$  class=foliage (233.0/5.0)
14. (hue-mean  $\leq -2.0944$ ) **AND** (region-centroid-row  $\leq 145$ )  $\Rightarrow$  class=foliage (98.0/11.0)
15. (rawred-mean  $\leq 18.4444$ ) **AND** (exred-mean  $\leq -6$ )  $\Rightarrow$  class=foliage (13.0/4.0)
16. (region-centroid-row  $\leq 149$ )  $\Rightarrow$  class=brickface (334.0/7.0)
17. [Empty Rule]  $\Rightarrow$  class=path (338.0/10.0)

## 5.3 Risultati su Vehicle Silhouettes

```

Esecuzione REPTree

=== Run information ===

Scheme:      weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0
Relation:    vehicle
Instances:   846
Attributes:  19
             COMPACTNESS
             CIRCULARITY
             DISTANCE CIRCULARITY
             RADIUS RATIO
             PR.AXIS ASPECT RATIO
             MAX.LENGTH ASPECT RATIO
             SCATTER RATIO
             ELONGATEDNESS
             PR.AXIS RECTANGULARITY
             MAX.LENGTH RECTANGULARITY
             SCALED VARIANCE_MAJOR
             SCALED VARIANCE_MINOR
             SCALED RADIUS OF GYRATION
             SKEWNESS ABOUT_MAJOR
             SKEWNESS ABOUT_MINOR
             KURTOSIS ABOUT_MAJOR
             KURTOSIS ABOUT_MINOR
             HOLLOWS RATIO
             Class
Test mode:   10-fold cross-validation

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      612           72.3404 %
Incorrectly Classified Instances    234           27.6596 %
Kappa statistic                    0.6313
Mean absolute error                 0.1617
Root mean squared error             0.3109
Relative absolute error             43.1254 %
Root relative squared error         71.8227 %
Total Number of Instances          846

=== Detailed Accuracy By Class ===

            TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
            0,575    0,137    0,584     0,575    0,580      0,440    0,842     0,586     opel
            0,475    0,129    0,560     0,475    0,514      0,366    0,795     0,561     saab
            0,945    0,040    0,892     0,945    0,918      0,889    0,968     0,895     bus
            0,910    0,063    0,815     0,910    0,860      0,816    0,972     0,879     van
Weighted Avg.    0,723    0,093    0,711     0,723    0,716      0,625    0,893     0,728

=== Confusion Matrix ===

  a   b   c   d  <-- classified as
122  68   8  14 |  a = opel
 81 103  13  20 |  b = saab
  1   4 206   7 |  c = bus
  5   9   4 181 |  d = van

```

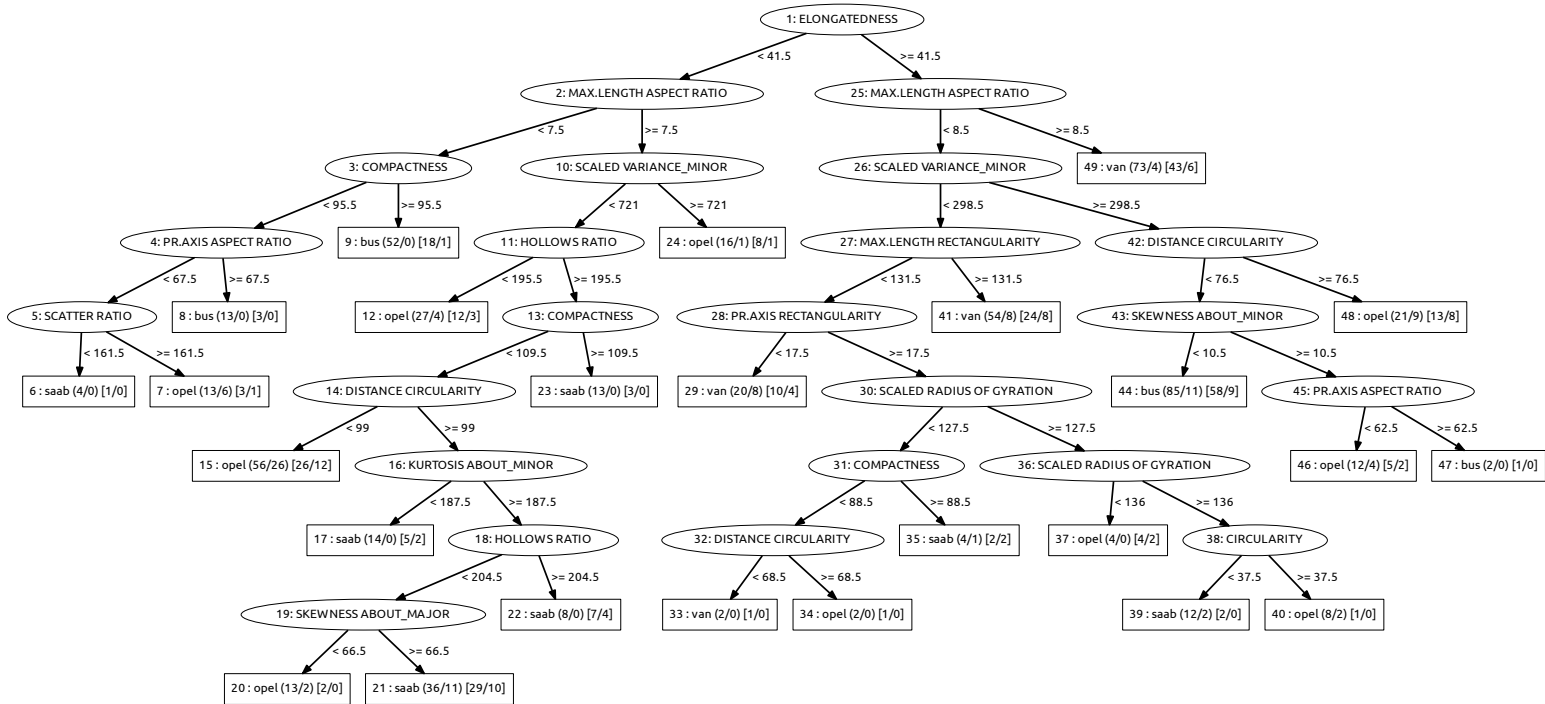


Figura 5.3: Modello di REPTree

#### Esecuzione JRip

=== Run information ===

```

Scheme:      weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1
Relation:    vehicle
Instances:   846
Attributes:  19
COMPACTNESS
CIRCULARITY
DISTANCE CIRCULARITY
RADIUS RATIO
PR.AXIS ASPECT RATIO
MAX.LENGTH ASPECT RATIO
SCATTER RATIO
ELONGATEDNESS
PR.AXIS RECTANGULARITY
MAX.LENGTH RECTANGULARITY
SCALED VARIANCE_MAJOR
SCALED VARIANCE_MINOR
SCALED RADIUS OF GYRATION
SKEWNESS ABOUT_MAJOR

```

```

        SKEWNESS ABOUT_MINOR
        KURTOSIS ABOUT_MAJOR
        KURTOSIS ABOUT_MINOR
        HOLLOWS RATIO
        Class
Test mode: 10-fold cross-validation

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      584          69.0307 %
Incorrectly Classified Instances    262          30.9693 %
Kappa statistic                    0.5868
Mean absolute error                 0.1914
Root mean squared error             0.3323
Relative absolute error             51.0598 %
Root relative squared error         76.7524 %
Total Number of Instances          846

=== Detailed Accuracy By Class ===

        TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
        0,481    0,126    0,560     0,481    0,518      0,374    0,785     0,545     opel
        0,484    0,146    0,533     0,484    0,507      0,349    0,793     0,501     saab
        0,940    0,099    0,768     0,940    0,845      0,792    0,940     0,812     bus
        0,864    0,043    0,860     0,864    0,862      0,820    0,927     0,861     van
Weighted Avg.   0,690    0,105    0,677     0,690    0,680      0,580    0,860     0,677

=== Confusion Matrix ===

  a   b   c   d  <-- classified as
102  71  27  12 |  a = opel
 74 105  27  11 |  b = saab
  2   6 205   5 |  c = bus
  4  15   8 172 |  d = van

```

## Regole:

1. (ELONGATEDNESS  $\geq 43$ ) **AND** (MAX.LENGTH ASPECT RATIO  $\geq 9$ ) **AND** (DISTANCE CIRCULARITY  $\geq 73$ )  $\Rightarrow$  Class=van (86.0/0.0)
2. (SCALED VARIANCE.MINOR  $\leq 309$ ) **AND** (MAX.LENGTH RECTANGULARITY  $\geq 132$ ) **AND** (DISTANCE CIRCULARITY  $\leq 64$ ) **AND** (SCALED RADIUS OF GYRATION  $\leq 157$ )  $\Rightarrow$  Class=van (23.0/0.0)
3. (PR.AXIS RECTANGULARITY  $\leq 18$ ) **AND** (MAX.LENGTH RECTANGULARITY  $\geq 128$ ) **AND** (SCALED RADIUS OF GYRATION  $\leq 140$ )  $\Rightarrow$  Class=van (42.0/6.0)
4. (SCALED VARIANCE.MINOR  $\leq 309$ ) **AND** (MAX.LENGTH RECTANGULARITY  $\geq 142$ )  $\Rightarrow$  Class=van (33.0/5.0)
5. (ELONGATEDNESS  $\geq 53$ ) **AND** (SCALED RADIUS OF GYRATION  $\geq 137$ )  $\Rightarrow$  Class=van (15.0/5.0)
6. (SCALED VARIANCE.MAJOR  $\leq 177$ ) **AND** (MAX.LENGTH ASPECT RATIO  $\geq 10$ )  $\Rightarrow$  Class=van (8.0/1.0)
7. (MAX.LENGTH ASPECT RATIO  $\geq 8$ ) **AND** (MAX.LENGTH RECTANGULARITY  $\geq 173$ )  $\Rightarrow$  Class=opel (45.0/8.0)

8. (MAX.LENGTH ASPECT RATIO  $\geq 8$ ) **AND** (COMPACTNESS  $\leq 103$ ) **AND** (ELONGATEDNESS  $\leq 37$ ) **AND** (HOLLOWS RATIO  $\leq 195$ )  $\Rightarrow$  Class=opel (14.0/0.0)
9. (MAX.LENGTH ASPECT RATIO  $\geq 8$ ) **AND** (HOLLOWS RATIO  $\leq 198$ ) **AND** (KURTOSIS ABOUT\_MINOR  $\geq 189$ )  $\Rightarrow$  Class=opel (42.0/17.0)
10. (SKEWNESS ABOUT\_MAJOR  $\leq 67$ ) **AND** (HOLLOWS RATIO  $\leq 203$ )  $\Rightarrow$  Class=opel (66.0/30.0)
11. (SCALED RADIUS OF GYRATION  $\leq 142$ ) **AND** (HOLLOWS RATIO  $\leq 194$ ) **AND** (DISTANCE CIRCULARITY  $\geq 57$ )  $\Rightarrow$  Class=opel (17.0/2.0)
12. (MAX.LENGTH ASPECT RATIO  $\geq 9$ ) **AND** (DISTANCE CIRCULARITY  $\geq 100$ ) **AND** (SCALED VARIANCE\_MAJOR  $\leq 231$ )  $\Rightarrow$  Class=saab (71.0/9.0)
13. (MAX.LENGTH ASPECT RATIO  $\geq 9$ ) **AND** (PR.AXIS ASPECT RATIO  $\leq 61$ )  $\Rightarrow$  Class=saab (23.0/7.0)
14. (SCALED VARIANCE\_MAJOR  $\leq 165$ ) **AND** (DISTANCE CIRCULARITY  $\leq 66$ )  $\Rightarrow$  Class=saab (36.0/11.0)
15. (SKEWNESS ABOUT\_MAJOR  $\leq 72$ ) **AND** (PR.AXIS ASPECT RATIO  $\leq 65$ ) **AND** (DISTANCE CIRCULARITY  $\geq 81$ ) **AND** (SKEWNESS ABOUT\_MAJOR  $\geq 66$ )  $\Rightarrow$  Class=saab (27.0/7.0)
16. (CIRCULARITY  $\leq 40$ ) **AND** (RADIUS RATIO  $\leq 144$ )  $\Rightarrow$  Class=saab (16.0/6.0)
17. [Empty Rule]  $\Rightarrow$  Class=bus (282.0/69.0)

## 5.4 Risultati su Wisconsin Breast Cancer

```

Esecuzione REPTree

=== Run information ===

Scheme:weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1
Relation:      wisconsin-breast-cancer
Instances:     699
Attributes:    10
               Clump_Thickness
               Cell_Size_Uniformity
               Cell_Shape_Uniformity
               Marginal_Adhesion
               Single_Epi_Cell_Size
               Bare_Nuclei
               Bland_Chromatin
               Normal_Nucleoli
               Mitoses
               Class

Test mode:10-fold cross-validation

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      656           93.8484 %
Incorrectly Classified Instances    43           6.1516 %
Kappa statistic                    0.8653
Mean absolute error                 0.083
Root mean squared error             0.2234
Relative absolute error             18.3662 %
Root relative squared error         47.0064 %
Total Number of Instances          699

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area   Class
               0.941     0.066     0.964       0.941     0.952       0.964     benign
               0.934     0.059     0.893       0.934     0.913       0.964     malignant
Weighted Avg.   0.938     0.064     0.94        0.938     0.939       0.964

=== Confusion Matrix ===

  a   b   <-- classified as
431  27 |   a = benign
 16 225 |   b = malignant

```

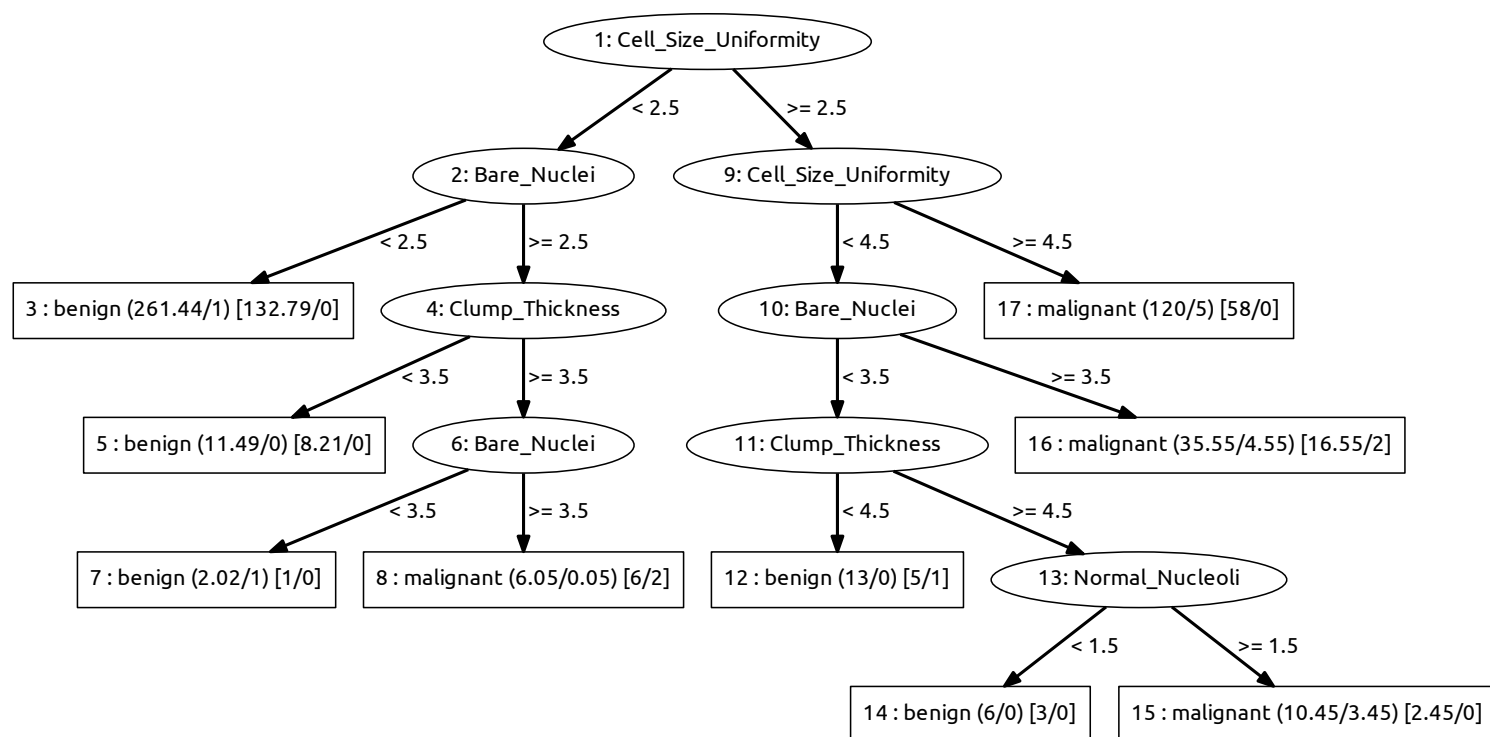


Figura 5.4: Modello di REPTree

Esecuzione JRip	
=== Run information ===	
Scheme:	weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1
Relation:	wisconsin-breast-cancer
Instances:	699
Attributes:	10
	Clump_Thickness
	Cell_Size_Uniformity
	Cell_Shape_Uniformity
	Marginal_Adhesion
	Single_Epi_Cell_Size
	Bare_Nuclei
	Bland_Chromatin
	Normal_Nucleoli
	Mitoses
	Class



Test mode:10-fold cross-validation

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	667	95.422 %
Incorrectly Classified Instances	32	4.578 %
Kappa statistic	0.8999	
Mean absolute error	0.0618	
Root mean squared error	0.2022	
Relative absolute error	13.676 %	
Root relative squared error	42.5462 %	
Total Number of Instances	699	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.952	0.041	0.978	0.952	0.965	0.973	benign
	0.959	0.048	0.913	0.959	0.935	0.973	malignant
Weighted Avg.	0.954	0.044	0.955	0.954	0.954	0.973	

=== Confusion Matrix ===

a	b	<-- classified as
436	22	a = benign
10	231	b = malignant

Regole:

1. (Cell\_Size\_Uniformity  $\geq 3$ ) **AND** (Cell\_Size\_Uniformity  $\geq 5$ )  $\Rightarrow$  Class=malignant (178.0/5.0)
2. (Bare\_Nuclei  $\geq 4$ ) **AND** (Bare\_Nuclei  $\geq 7$ )  $\Rightarrow$  Class=malignant (48.0/4.0)
3. (Normal\_Nucleoli  $\geq 3$ ) **AND** (Clump\_Thickness  $\geq 6$ )  $\Rightarrow$  Class=malignant (13.0/1.0)
4. (Bare\_Nuclei  $\geq 3$ ) **AND** (Clump\_Thickness  $\geq 5$ )  $\Rightarrow$  Class=malignant (11.0/3.0)
5. (Marginal\_Adhesion  $\geq 8$ )  $\Rightarrow$  Class=malignant (2.0/0.0)
6. [Empty Rule]  $\Rightarrow$  Class=benign (447.0/2.0)

# Bibliografia

- [1] C. Brunk and M. J. Pazzani. An investigation of noise-tolerant relational concept learning algorithms, 1991.
- [2] W. W. Cohen. Efficient pruning methods for separate-and-conquer rule learning systems. In *In Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 988–994. Morgan Kaufmann, 1993.
- [3] W. W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.
- [4] F. Esposito, D. Malerba, and G. Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(5):476–491, May 1997. ISSN 0162-8828.
- [5] J. Fürnkranz and G. Widmer. Incremental reduced error pruning, 1994.
- [6] G. Pagallo, D. Haussler, and P. Rosenbloom. © 1990 kluwer academic publishers. manufactured in the netherlands. boolean feature discovery in empirical learning, 1990.
- [7] J. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3): 239–266, 1990. ISSN 1573-0565.
- [8] J. R. Quinlan. Simplifying decision trees. *Int. J. Man-Mach. Stud.*, 27(3):221–234, Sept. 1987. ISSN 0020-7373.
- [9] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.
- [10] J. R. Quinlan. MDL and categorical theories (continued). In *In Machine Learning: Proceedings of the Twelfth International Conference, Lake Tahoe*, pages 464–470. Morgan Kaufmann, 1995.
- [11] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Inf. Comput.*, 80(3):227–248, Mar. 1989. ISSN 0890-5401.
- [12] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003. ISBN 0137903952.

- [13] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.
- [14] S. M. Weiss and N. Indurkha. Reduced complexity rule induction. In *In Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 678–684. Morgan Kaufmann, 1991.
- [15] I. H. Witten, E. Frank, and M. A. Hall. Decision trees. In *Data Mining: Practical Machine Learning Tools and Techniques*, chapter 6.2, page 206. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011. ISBN 0123748569, 9780123748560.
- [16] I. H. Witten, E. Frank, and M. A. Hall. Decision trees. In *Data Mining: Practical Machine Learning Tools and Techniques*, chapter 11.4, page 456. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011. ISBN 0123748569, 9780123748560.
- [17] I. H. Witten, E. Frank, and M. A. Hall. Decision trees. In *Data Mining: Practical Machine Learning Tools and Techniques*, chapter 6.11, page 303. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011. ISBN 0123748569, 9780123748560.