UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

LAUREA MAGISTRALE
IN INGEGNERIA INFORMATICA

TESI DI LAUREA

RICONOSCIMENTO DELLA SPECIE ARBOREA DA IMMAGINI TOMOGRAFICHE

Nicola Bertolini

RELATORE: Emanuele Menegatti

CORRELATORE: Enrico Vicario

Anno Accademico 2013–2014

Prefazione

Negli ultimi quarant'anni si sono succeduti i tentativi di utilizzare gli strumenti per la diagnosi medica, come il tomografo, per la rilevazione dei difetti interni degli alberi in modo non distruttivo. Microtec, per prima al mondo, è riuscita a produrre uno scanner di nuova concezione che adempisse a questo scopo con efficienza ed affidabilità. Dal momento della messa in funzione del primo CT.LOG le possibilità di sviluppo si sono moltiplicate e sono ancora molte le strade da esplorare. La tomografia in quest'ambito serve a massimizzare il valore, e quindi il profitto, per ogni tronco. Il valore e la lavorazione del legno variano molto in base alla specie: non sempre però questa è certificata, soprattutto nella produzione in serie. È per questo che è stato chiesto da un cliente di creare un sistema automatico di catalogazione per distinguere fra 4 diverse specie che spesso vengono confuse fra di loro: Abeti Bianchi, Abeti Rossi, Pini e Abeti di Douglas. Lo scopo primario di questo lavoro è lo sviluppo di un classificatore che riconosca le specie dei tronchi da immagini tomografiche in modo efficiente e con confidenza sufficiente a garantire un incremento del profitto sul tagliato, partendo da un dataset di 2700 scansioni di tronchi. Per questo motivo verranno indagati e sviluppati 3 diversi sistemi di classificazione automatica, attingendo al mondo del Machine Learning e della Computer Vision. I risultati presentati mostrano come è possibile distinguere fra queste specie affidandosi a poche caratteristiche non sempre facili da valutare se non tramite la tomografia computerizzata e una conoscenza biologica di base.

Indice

Capitolo 1

Introduzione

Dal 1980 microtec si occupa di innovare il mondo dell'industria del legno, introducendo ogni giorno nuove tecnologie per automatizzare il processo produttivo e massimizzare il valore del prodotto finito. Tutto questo è permesso dai grandi investimenti che l'azienda compie nel campo della ricerca e sviluppo e della forte coesione che promuove con i partner commerciali. In questo ambito di sviluppo positivo e propositivo, sono stati messi a punto sistemi completi che hanno rivoluzionato la catena di lavorazione del legname, dalla rilevazione dei difetti del legno all'ottimizzazione del valore del tagliato, dal semplice taglio al riconoscimento della specie arborea. In particolare da metà anni novanta è stata introdotta la tecnologia a raggi X per la scansione profonda del legno in modo non distruttivo obbiettivo ormai decennale di quest'industria. Dallo studio di questa tecnologia sono fioriti negli anni i progetti che hanno portato alla produzione di diversi prodotti, dal Goldenye(1995) per l'elaborazione di tavole al Tomolog(2001) per il riconoscimento delle proprietà interne dei tronchi da sistemi X–ray multi vista, fino allo sviluppo del CT.LOG lo scanner tomografico definitivo per tronchi che realizza le ambizioni di molti studiosi del settore: portare la scansione tomografica ad alta risoluzione, alta velocità ed alta efficienza all'interno delle segherie. CT.LOG ricostruisce il tronco con un modello tridimensionale denso tramite un tomografo di nuova concezione, e potenti algoritmi per l'interpolazione dei dati. Dai dati grezzi poi cerca di individuare tutte le proprietà utili a definire il valore del tronco e come lavorarlo in seguito. Partendo dall'individuazione delle parti basilari del tronco come pith, hearthwood, e sapwood, passando per le caratteristiche visivamente più facili da individuare come nodi, anelli di accrescimento e marciume, fino all'individuazione di difetti rari e non banali, come sacche di resina, compressionwood, crepi, attacchi di funghi o insetti. Per poter aumentare ancora di più l'automatizzazione di questo processo ed insieme aumentare il valore della produzione, è stato richiesto da un cliente, Siat-Braun forse la più grande segheria francese per metri cubi prodotti, di cercare di definire la specie arborea dei tronchi scansionati, in quanto definirla in impianto risulta un lavoro molto dispendioso in termini di tempo, in particolare se si elaborano decine di tronchi al minuto, e la specie definita in foresta non sempre è affidabile.

Per ottenere un'informazione certa sulla specie si è dovuto studiare più ambiti, il primo è la tomografia e in particolare il tomografo CT.LOG di Microtec e il tipo di informazioni che una scansione mette a disposizione. Verrà presentata una pa-

noramica sulla macchina e sul software che interpola i dati e ricostruisce il modello tridimensionale, verranno inoltre brevemente analizzati anche gli algoritmi di computer vision che estraggono le informazioni basilari sui tronchi e quelli che scavano più in profondità per trovare quelle differenze fondamentali fra una specie e l'altra. La classificazione verrà affidata in prima analisi a algoritmi di machine learning stabili e conosciuti che faranno affiorare informazioni non palesi ed indispensabili al riconoscimento della specie. Infatti per alcune specie ci sono caratteristiche palesi che si distinguono ad occhio nudo sulle scansioni, queste caratteristiche verranno estratte con procedimenti di computer vision e ci si affiderà al data mining solo per metterle in correlazione con le altre, mentre altri tipi di informazioni (feature) verranno estratte e calcolate nella speranza che offrano una buona correlazione con le specie. Questo lavoro di tipo statistico è quasi esclusivamente affidato al data mining. Una volta definite tutte le informazioni necessarie verranno presentati diversi classificatori di natura diversa: sia sviluppati automaticamente sia studiati appositamente per il caso, per mettere maggiormente in risalto determinate differenze fra una specie e l'altra.

Lo sviluppo di questo progetto parte dalle immagini tomografiche degli alberi, nel capitolo 2 verrà spiegato come vengono ottenute, quali sono i vantaggi e i limiti rispetto altri tipi di scansioni. Nel capitolo 3 verranno presentate le diverse specie di alberi e verrà spiegato quali sono le conoscenze biologiche a priori che permettono la distinzione di una specie dall'altra. Inoltre verrà presentato l'ampio dataset a disposizione. Il capitolo 4 presenterà le diverse strategie di data mining adottate, le conoscenze teoriche preliminari e come esse hanno portato alla definizione del classificatore finale. Il cuore dell'elaborato risiede nel capitolo 5 dove viene spiegato come sono elaborati i dati grezzi fino ad ottenere le informazioni utili alla classificazione, in particolare verrà esposto come sono state calcolate le feature prese in considerazione e quali alla fine si sono rivelate utili. I risultati e le conclusioni verranno presentati nel capitolo 6

Capitolo 2

Tomografia computerizzata

Nel 1972 è stato introdotto, nell'ambiente della diagnosi medica, un nuovo strumento che permetteva di analizzare approfonditamente e con alta risoluzione il corpo umano: il tomografo. Subito ha attratto l'attenzione dell'industria del legno, in particolare di chi puntava ad un'automazione del processo produttivo. In virtù della composizione del legno, principalmente acqua e carbonio, i risultati dei primi test su tronchi, compiuti pochi anni dopo, non furono dissimili a quelli in ambito medicale. La particolare attenzione era dovuta alla necessità di trovare metodi non distruttivi per trovare i difetti interni ai tronchi per poter ottimizzare il valore del segato. Di qui la necessità di estrarre caratteristiche dei tronchi quali forma, dimensione e aderenza dei nodi, stato di essiccazione, volume dell'hearthwood, posizione del pith, crepi, e quindi di implementare algoritmi che trovassero queste peculiarità a partire dalle ricostruzioni tridimensionali.

2.1 CT.LOG

La principale difficoltà stava nel produrre uno scanner industriale che desse un buon trade–off fra velocità di elaborazione/scansione e qualità, infatti, di fronte ad un'ottima risoluzione, gli scanner medicali pongono anche un forte limite su velocità di scansione e tempi di utilizzo. Altri scanner a raggi X per tronchi con un numero di viste che varia da 1 a 4, invece, davano alta velocità ma scarsa risoluzione e difficoltà nella ricostruzione tridimensionale.

Microtec voleva colmare queste lacune cercando di produrre uno scanner Ct che:

1. garantisse una velocità di scansione dai 60 ai 240 metri al minuto ($1 \div 4 m/s$);
2. avesse la capacità di lavorare continuativamente senza dover fermare la linea di produzione per raffreddare le macchine;
3. assicurasse un rapporto segnale rumore accettabile per tronchi fino a $70cm$ di diametro ad una data velocità;
4. fosse strutturalmente robusto per lavorare con pesanti tronchi;
5. fornisse una risoluzione di circa $1mm$ in direzione trasversale e $1cm$ in direzione assiale;

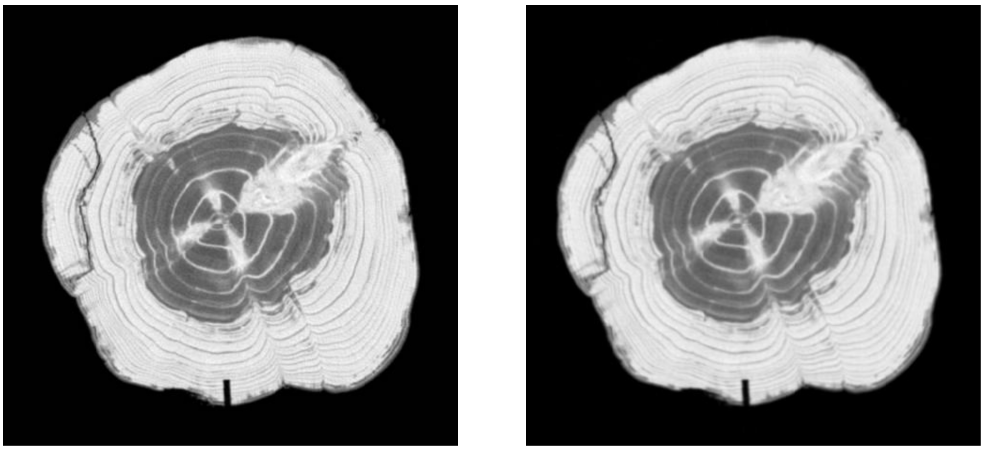
6. permettesse la ricostruzione del modello tridimensionale e la sua analisi in tempo reale.

Per comprendere le difficoltà superate da Microtec nel progettare il proprio prodotto, si pensi che uno scanner medicale ad alta risoluzione lavora a $0.5m/s$ ma per molto tempo resta inutilizzato per far raffreddare il tubo a raggi X, mentre gli scanner per bagagli degli aeroporti lavorano in continuazione ma a $0.3m/s$
. Il problema principale legato a questa lentezza di elaborazione sta nella configurazione standard degli scanner tomografici detta Fan Beam, che prevede una sorgente di raggi X ed un array di sensori lineare rotanti attorno all'oggetto da scansionare. Per poter ricostruire una sezione dell'oggetto, bisogna allora acquisire molteplici proiezioni della stessa sezione, in particolare, per una risoluzione di un'immagine a centimetro è necessaria più di mezza rivoluzione della sorgente e dei sensori attorno al oggetto della scansione. Il limite risiede quindi nella velocità di rotazione che difficilmente è superiore a 4 rivoluzioni al secondo, il che limita a $8cm/s$ l'avanzamento della scansione.

Moltiplicando le linee di sorgenti e sensori si può incrementare la velocità, ma questi devono essere posizionati distanti fra loro per non influenzarsi. Un'altra possibilità è data dalla geometria detta Cone Beam che utilizza una sola sorgente ed un array 2D di sensori. Fino al 2001, però, esistevano solo metodi teorici, eccessivamente dispendiosi in termini di tempo, o approssimati, che non permettevano sufficiente affidabilità. Dal 2001, Alexander Katsevich ha pubblicato una serie di articoli, [26, 27, 28, 29] nei queli propone e migliora una soluzione esatta ed efficiente al problema della ricostruzione per la cone beam tomography. La prima realizzazione di uno scanner che utilizza la tecnologia cone beam implementando l'idea di Katsevich, è proprio CT.LOG, che utilizza una matrice di 752×1380 sensori. Grazie alla velocità di ricostruzione garantita dall'algoritmo di Katsevich e con il vincolo di 120 rivoluzioni al minuto, si riesce a incrementare la velocità di scansione fino a $120m/min$.

I dati prodotti dal tomografo di Microtec sono matrici dense di valori a 8 bit senza segno, in pratica in scala di grigi. Le dimensioni della matrice variano nella direzione assiale in relazione alla lunghezza del tronco analizzato. Avendo risoluzione lungo l'asse z di 1 voxel per 11 millimetri, quindi di circa 91 voxel al metro, e dimensioni fisse di 700 voxel per le altre due direzioni, con risoluzione di 1 voxel a millimetro, si intuisce subito che un tronco,di lunghezza tipicamente sopra i 4 metri, corrisponderà ad una grande mole di dati analizzare. Questo porta alla necessità di trovare modi molto efficienti di portare a termine anche i task più facili. In particolare, risulta facile fare molte elaborazioni partendo dalle sezioni trasversali, riconoscere in queste determinate particolarità e poi seguirle di slice in slice. Compiuto questo passo sarà più facile analizzare i difetti del tronco grazie a viste dedicate, come immagini polari, intorno al pith, di sezioni trasversali, o tramite le superfici cilindriche concentriche.

La velocità di elaborazione assicurata dall'approccio Cone Beam non introduce significativi errori, artefatti e rumore, come si può osservare dal confronto delle immagini 2.1(a) e 2.1(b).



(a) Fan Beam (b) Cone Beam

Figura 2.1: Ricostruzione della stessa sezione trasversale con Fan Beam CT, e simulando un Cone Beam CT a $2m/s$

2.2 Scala di Hounsfiled

Gli scanner tomografici, in sostanza misurano la densità di un oggetto voxel per voxel, Per fare ciò si servono delle radiazioni a raggi X e con dei sensori, posti diametralmente opposti alla sorgente rispetto all'oggetto, misurano l'attenuazione dell'energia del fascio di raggi che li colpisce: di fatto viene misurata la radiodensità dei raggi X rispetto al materiale attraversato. Attraverso gli algoritmi di ricostruzione (backward–projection), partendo da una serie di immagini monodimensionali (o bidimensionali nel caso della cone beam tomography), si riesce a trovare la densità di ogni singolo elemento tridimensionale, con risoluzione a piacere a discapito della velocità. Comunemente, i dati in output ai tomografi sono espressi con la scala di Hounsfield. Questa scala punta ad aumentare la precisione nella zona d'interesse classica per applicazioni medicali. In questo caso l'elemento con densità inferiore attraversato dai raggi x sarà sicuramente l'aria, mentre quello più denso sono le ossa umane. La scala di Hounsfield fa combaciare la densità dell'aria con il valore $-1000HU$ (hounsfield unit), quella dell'acqua, principale elemento del corpo umano , con $0HU$, e quindi le ossa si ritrovano ad occupare il range $700 - 1000HU$. Per le proprie applicazioni, Microtec ha deciso di tenere come range operativo quello che va dalla densità dell'aria fino a $400HU$, la densità massima riscontrabile in un albero. Come già specificato, le immagini sono in scala di grigio a 8 bit, quindi possono assumere valori da 0 a 255. Quindi lo 0 corrisponderà alla densità dell'aria $-1000HU$, mentre a 255 corrisponderà il massimo valore.

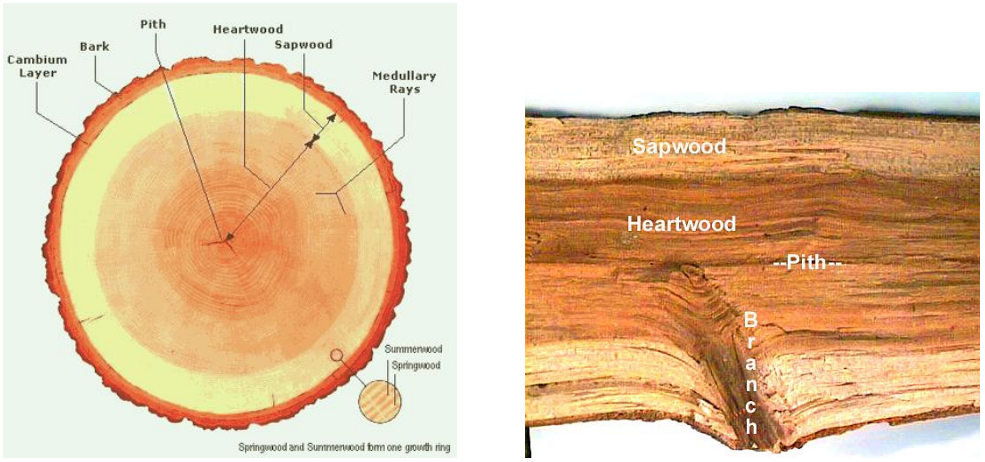
In definitva, le matrici prodotte da CT.LOG, per quanto compresse con l'algoritmo lossless LZW, occupano su disco circa $80KB$ per slice

Capitolo 3

Biologia

Gli alberi che verranno analizzati, come spiegato in 3.1, sono tutte conifere cioè alberi dal legno abbastanza morbido e lavorabile. Nonostante siano di specie diverse, presentano caratteristiche comuni a tutti gli alberi. In primo luogo, sezionando il tronco con un piano parallelo al terreno, che chiameremo sezione trasversale, sono evidenti una serie di elementi di primaria importanza. Come si vede in figura3.1(a), partendo dal centro e procedendo verso l'esterno del tronco, troviamo il midollo(in inglese pith), che percorre tutta la pianta, alle volte questa parte marcisce o scompare lasciando cavità. Dal pith si staccano i nodi che arrivati all'esterno diventano rami. Attorno al pith si trova il durame (En.: Heartwood) parte di legno "morto" che ha la sola funzione di sostegno della pianta, è questa parte che è viene impiegata per ricavare le tavole di legno. La corona esterna al durame prende il nome di alburno (En.: Sapwood), parte di legno ancora viva, attraverso la quale l'acqua viene condotta per tutta la pianta, questa parte è troppo morbida per essere usata come legno per le tavole e presenta densità molto più alta che nel durame, per la presenza di acqua. L'utlima fascia, prima della corteccia è detta legno di cambio o di crescita, è la parte del tronco che trasporta la linfa vitale. Spesso non distinguibile a occhio nudo, negli alberi che formano anelli annuali é la parte che viene rigenerata di anno in anno e, in base alla stagione, cresce più o meno densa: in primavera cresce rapidamente ed è poco densa per poter trasportare maggior linfa alle foglie, in autunno cresce meno ma è molto più densa per preparasi al freddo invernale, questa differenza di densità crea gli anelli L'ultimo strato è la corteccia (En.: bark), strato protettivo, in alcune specie la resina si trova solo qui. La parte più esterna è morta e si stacca con facilità.

I nodi si presentano come blocchi di legno più duro e denso, partono dal pith e si allontanano verso la corteccia, generalmente con una pendenza verso l'alto, per alcune specie è caratteristica la pendenza accentuata o la pendenza verso il basso. Di fatto il nodo è il ramo che una volta nato viene inglobato nel tronco. In tutti gli alberi che analizziamo i rami principali si trovano generalmente in corone di $3 \div 6$ elementi che hanno origine tutti alla stessa altezza dell'albero, queste corone sono dette verticilli. Solo in alcune specie sono invece presenti rami sporadici meno robusti ad altezze che possono essere considerate casuali.



(a) Trasversale (b) Radiale

Figura 3.1: Sezioni Trasversale e radiale di tronco con evidenziate le caratteristiche fondamentali

3.1 Gli Alberi

Le conifere [5] sono fra gli alberi più diffusi al mondo, le specie ancora viventi rappresentano infatti la principale classe fra gli alberi che popolano le grandi foreste boreali dell'emisfero settentrionale. L'unico ordine di specie ancora esistenti, le Pinales, raggruppa 6 famiglie, per più di 600 specie. La famiglia più importante, e d'interesse specifico in questo lavoro, è quella delle pinaceae che raggruppano 11 generi fra cui larici, pini, abeti, pecci e cedri, per un totale di 232 specie viventi riconosciute. In particolare, la nostra attenzione si focalizzerà su solo alcune di queste: saranno le specie sulle quali verrà addestrato e testato il calssificatore, le specie interessanti per il cliente.

3.1.1 Picea Abies

Conosciuto in italiano come Peccio comune o Abete rosso o Abete di Norvegia [30], fa parte del genere delle Piceae, in questo lavoro ci riferiremo a lui con il nome inglese Spruce (Norway Spruce). Confuso molto spesso con un abete, esso è relativamente distante alle Abies (Abeti) sull'albero filogenico. Specie endemica di centro, nord ed est Europa, si spinge fino alla grecia e la Massiccio Centrale in Francia, in climi più freddi si trova sul livello del mare, scendendo sotto il 47 parallelo, solo fra 500 e $2200m$, è stata naturalizzata in tutta Europa e nella regione centro settentrionale degli stati uniti, e le regioni adiacenti del Canada. Si presenta come la più classica delle conifere, alto fino a 50 metri, e largo ad un metro dalla base fino ad $1,5m$ con rami corti e robusti presenti lungo tutto il fusto. Oltre ai caratteristici verticilli di rami, lungo il resto del tronco, presenta rami secondari più piccoli. I rami superiori presentano un'inclinazione verso l'alto, quelli bassi, più pesanti, tendono, con il tempo, verso il terreno. Le foglie, elemento fortemente distintivo dal simile abete bianco (sez. 3.1.2) sono aghi lunghi fino a $2.5cm$ con sezione quadrata, con colore uniforme dal verde chiaro al verde scuro, cadono periodicamente sostituite

subito da nuove, sono presenti su tutta la lunghezza dei rami minori. La corteccia è da prima rossastra, da cui il nome Abete rosso, quando invecchia tende dal grigio marrone e si stacca in scaglie. I coni (pigne), di colore rosso–marrone e forma particolarmente allungata, nascono sulle punte dei rami più robusti e sono cadenti. Il legno, abbastanza pregiato e versatile, viene impiegato nell'edilizia come elemento estetico o strutturale, abbastanza tenero, ha un tempo d'essiccazione maggiore rispetto al Fir. Si adatta particolarmente alla coltivazione intensiva perchè crea boschi puri, cresce relativamente in fretta, e non rovina l'ecosistema.

La sezione del tronco non presenta anelli particolarmente marcati, anche se restano evidenti come in tutte le conifere; alle volte sono presenti però sacche di resina che compromettono alcune proprietà meccaniche.

3.1.2 Abies Alba

L'abete bianco [31] è un degli alberi più diffusi sull'arco alpino italiano, è presente su tutti i rilievi europei spingendosi sui balcani fino in grecia, su tutti i pirenei, sugli appennini calabri e fino in ucraina, comunque non sotto i $500m$ e mai sopra i 2100. Non crea foreste pure ma si trova spesso affianco ai faggi a quote basse o agli abeti rossi e ai larici a quote più elevate. Noto in inglese come Fir, il suo nome scientifico, Abies Alba, si rifà all'appartenenza al genere degli Abies (abeti) e al colore di un lato delle foglie che varia dal bianco al ceruleo chiaro. Tali foglie, a sezione piatta, sono lunghe fino a $3cm$ e si trovano in coppie diametralmente opposte sul rametto e non cadono. Esse sono l'elemento fortemente distintivo rispetto all'abete rosso per forma, disposizione, colore, dimensione, e durata: la loro non caducità fa preferire questa specie come "albero di natale" casalingo.

Gli esemplari adulti sono fra i più imponenti alberi europei, visto che raggiungono $55m$ di altezza con un diametro che sfiora i 3 metri. La parte bassa del tronco presenta verticilli e altri rami sporadici. Con l'età l'aspetto a cono della chioma si trasforma con i rami più bassi che tendono verso l'alto appiattendo l'estremità.

La corteccia è grigiastra per tutta la vita dell'albero, si stacca a scaglie ed è molto resinosa, infatti, a differenza di altri alberi simili, la resina è presente solo nella parte più esterna del tronco e non nell'alburno e nel durame: gli abeti bianchi sono totalmente privi di sacche di resina.

I coni che cambiano colore dal verde al rosso–marrone con la maturazione, svettano verso l'alto dai rami principali.

L'impiego del legno è similare a quello dell'abete rosso e in più particolarmente rinomato per la costruzione di strumenti musicali. Presenta proprietà meccaniche molto simili allo Spruce e per questo spesso ne viene fatta un'unica classificazione. Differente invece è il tempo d'essicazione, molto più breve per il fir, particolare che rende necessario distinguere bene fra queste due specie.

3.1.3 Pseudotsuga Menziesii

Noto come Douglas Fir, o in Italia come douglasia, deve il suo nome allo scopritore di questa specie [32]. Anche se spesso associato agli abeti (Fir) per l'aspetto,

la sua filogenia lo pone nel genere delle Pseudotsuga, ed infatti le differenze sono parecchie e, per alcuni aspetti, molto evidenti.

È originario delle regioni occidentali del Canada e dei vicini stati degli USA, si spinge in alcune sue sottospecie sulle montagne rocciose fino in messico. È stato trapiantato anche in Europa per la coltivazione a scopo di produrre legno da costruzione di qualità e bell'aspetto per le sue spiccate proprietà meccaniche e la rapida crescita. Viene impiegato non solo per l'architettura domestica, ma anche per opere più complesse come ponti, o nella realizzazione di imbarcazioni. Il suo legno ha un intenso colore rossastro che non schiarisce con l'essicazione. Predilige quote relativamente basse.

Gli esemplari selvatici possono arrivare anche a 100 metri d'altezza con un diametro alla base fino a $4.5m$, rendendo questa specie la più imponente fra le Pinacee, seconda in altezza solo alle sequoie californiane fra tutti gli esseri viventi. Ha corteccia e foglie molto simili all'abete bianco solo di tonalità più scure e disposizione di rami e chioma, che diventa piatta con l'avanzare degli anni, del tutto omologa. L'elemento distintivo più forte, a parte le dimensioni, sono i coni che non svettano verso l'alto ma soprattutto appaiono totalmente diversi: di colore marrone chiaro sono molto più larghi, con molto meno scaglie e presentano spine tricuspidi sopra ogni scaglia. La rapida crescita contribuisce allo sviluppo di anelli di crescita particolarmente ampi

3.1.4 Pinus Nigra

Il pino nero [33], come si evince dal nome "americano" European black pine, è una specie endemica dei rilievi europei, con particolare concentrazione in Austria e tutta la regione sud–orientale del vecchio continente. Le varie sottospecie prediligono altitudini e climi diversi, ma si trovano con continuità dall'Italia al mar Nero, dalla Grecia all'Austria apparendo anche nella parte occidentale della Turchia e sull'isola di Cipro. È stato naturalizzato anche in Canada, nei territori dal clima più mite e in tutta la costa atlantica degli Stati Uniti

È un membro della famiglia dei pini, e come tutti questi presenta rami robusti in verticilli e nessun ramo singolo. Per tutta la lunghezza del fusto, che può arrivare a $45m$ di altezza e 1,5 metri di larghezza, è ricoperto da uno speso strato di corteccia grigio–marrone con sfumature rosate. La chioma ha una sagoma ovale nei primi anni di vita per poi tendere ad appiattirsi nella parte superiore e assomigliare molto al pino marittimo. Il cono dalla forma allungata e di tonalità fra il giallo e il marrone, presenta scaglie molto più grosse delle altre specie analizzate. Le foglie crescono a coppie alle volte attorcigliate fra loro , raggiungono i $18cm$ di lunghezza e si presentano di un verde particolarmente scuro, da questo l'appellativo Nero.

È coltivato in Europa e negli USA nelle zone con clima troppo rigido per la coltivazione di altre specie e per la rapidità di crescita. Il suo legno, molto resinoso, viene utilizzato soprattutto per elementi decorativi esterni per la facilità di lavorazione e per la buona resistenza agli agenti atmosferici. È considerato l'albero che meglio sopporta l'ambiente inquinato urbano

3.1.5 Pinus Sylvestris

Leggermente più basso e fino, questo pino [34] ha struttura e crescita molto simili a quelle del cugino della specie Nigra. I coni sono leggermente più piccoli di colore grigio, e la corteccia nella parte bassa è simile al pino nero ma in alto è più fina e con sfumature arancioni. Le foglie cambiano lunghezza con gli anni, da un massimo di $9cm$ in gioventù, a $5cm$ in età avanzata.Si presentano sempre in coppie di 2 attorcigliate fra loro con colorazione glauca.

Presente in praticamente tutta l'Europa centrale, l'arco alpino, la penisola scandinava, la russia, i rilievi maggiori di francia, spagna, penisola balcanica e anatolia e in scozia, dal quale prende il nome di Scots Pine, nelle diverse sottospecie. È stato naturalizzato in molte regione del nord america.

Viene impiegato per gli stessi usi del pino nero e per questo tratteremo i pini come unica specie(Pine)

3.2 Dataset

Per lo sviluppo del classificatore ci è stato messo a disposizione da Siat Braun[1] un dataset di 2893 scansioni di tronchi catalogati accuratamente secondo la loro specie dai ricercatori di FVA[2] per un totale di $254GB$ di dati. Oltre a questi erano presenti diverse specie che al momento non sono considerate d'interesse. Fra Larici, Pecci di Sitka ed Abeti di Vancouver 376 campioni non sono utili per lo scopo. Un eventuale ampliamento del dataset con l'aggiunta di altri campioni per queste specie permetterebbe l'estensione del numero delle classi. A disposizione ci sono, quindi 1180 Spruce, 739 Fir, 336 Pine, 638 Douglas. Come vedremo in seguito alcuni tronchi, al momento della scansione erano eccessivamente secchi e quindi non in condizioni ottimali di lavorazione, infatti per essere lavorato è necessario che il tronco non si sia essiccato in foresta. I tronchi secchi spesso presentano ampie fenditure esterne(cretti da sole), attacchi di funghi e larve. Per il nostro interesse, abbiamo notato che i tronchi secchi presentano variazione nei valori di alcune feature, che incidevano pesantemente nei risultati del classificatore, nella sezione 5.1.1 verrà approfondito il problema. Per questi motivi abbiamo scelto di lavorare solo con i tronchi che consideriamo buoni ed elaborare il classificatore sulla base di 1099 Spruce, 738 Fir, 333 Pine, e 530 Douglas.

Oltre all'evidente sbilanciamento del numero di record, il nostro dataset ha una debolezza relative al fatto che i tronchi scansionati sono stati tagliati tutti nella stessa zona e quindi il classificatore risultante, usando questi tronchi come training set, può non risultare valido per segherie in altre zone, soprattutto se con climi differenti.

[1] Cliente di Microtec. Segheria di Urmatt (Alsazia, Francia) dov'è installato un CT.LOG

[2] Forstliche Versuchsund Forschungsanstalt Baden-Württemberg (Freiburg, Germania), collabora con Microtec fornendo supporto per le conoscenze biologiche e di scienze forestali

Capitolo 4

Classificatori

In questo capitolo verranno presentati brevemente gli aspetti fondamentali degli strumenti utilizzati allo scopo di sviluppare il classificatore finale. Per arrivare all'implementazione stabile e funzionante alla quale si riferiscono i risultati nella sezione 6, sono state fatte scelte di studio e progettazione che permettessero un ampliamento delle conoscenze sulle features (capitolo 5) e come esse siano influenzate dalle classi. Lo strumento maggiormente utilizzato a questo scopo è stato Weka (sez. 4.1) e le sue implementazioni di alcuni algoritmi (sezioni 4.2, 4.3, 4.4) per lo sviluppo di diversi classificatori. Nel paragrafo 4.5 verrà spiegato come sono state valutate le prestazioni dei classificatori in base anche alle esigenze dei clienti.

4.1 WEKA

Weka è una piattaforma opensource per il datamining e il machine learning sviluppata dall'università di Waikato [3, 4], e nota in tutto il mondo per la sua versatilità, completezza, e facilità di utilizzo.

Listing 4.1: Esempio di file arff

```
% 1. Title Iris Plants Database
%
% 2. Sources
%      (a) Creator R.A. Fisher
%      (b) Donor Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date July, 1988
%
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

Scritto in java, offre, oltre all'interfaccia grafica ricca e completa, un'ampia collezione di funzionalità per lo studio di grandi dataset, in particolare permette la costruzione di classificatori, con moltissimi modelli a disposizione, la clasterizzazione

dei dati secondo diversi algoritmi, e l'Association Analysis con alcuni approcci diversi fra loro. Accetta in input diversi tipi di file che poi converte nel formato standard ARFF, formato particolarmente adatto al preprocessing per la classificazione

Il formato vuole un header composto dal nome della relazione, il formato dei dati, specificandone il nome e la natura(numero intero, reale, nominale, binario o stringa) e poi tutti i dati, un record per riga, con gli attributi separati da una virgola. File di tipo CSV vengono accettati e rielaborati per creare l'header. In seguito, alle volte, è necessario rieditare tale header del file arff ottenuto dal file csv per rendere l'elaborazione più semplice.

In questo lavoro, Weka, è stato utilizzato per lo sviluppo del sistema di classificazione, in particolare utilizzando le funzioninalità di preprocessing in fase di sperimentazione, per eliminare i record di una o l'altra classe e per ridurre il set di feature alle volte molte delle quali inutili.

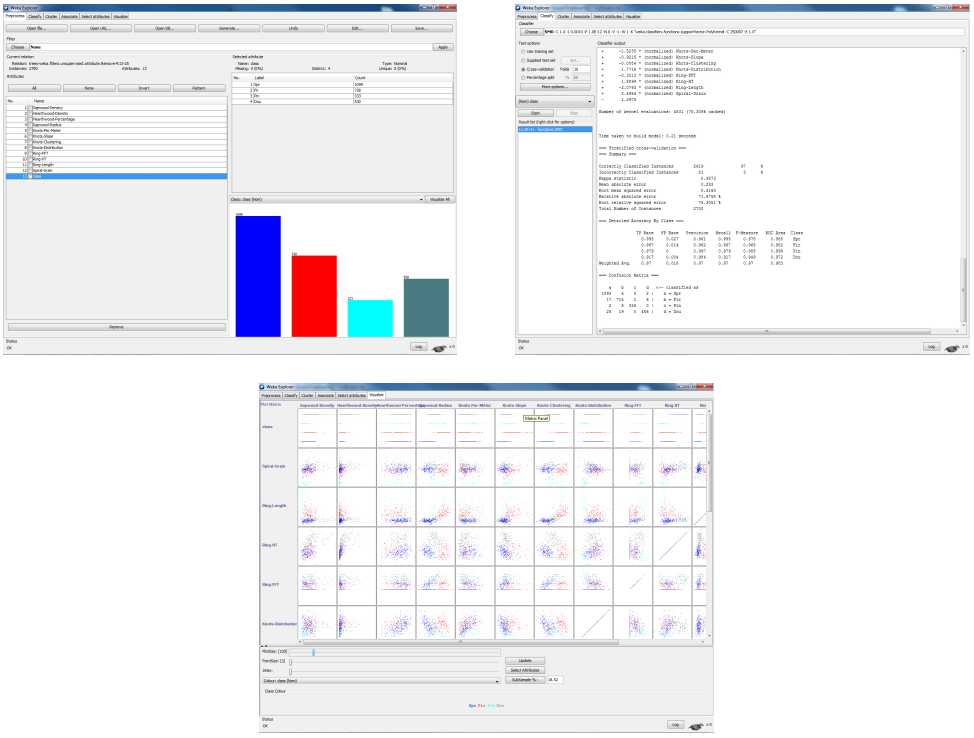


Figura 4.1: Le principali tab di Weka utilizzate

Come classificatori abbiamo utilizzato le implementazioni native di Weka degli algoritmi C4.5, SVM e ANN, rispettivamente J48, SMO e MultilayerPerceptron, ma sono state testate altre opzioni, come i classificatori bayesiani e molti fra gli altri classificatori basati su funzioni e su alberi decisionali. L'output di tutti i tool provati risulta ampiamente comprensibile e completo, dando informazioni aggiuntive anche sulle misure di validazione.

Essendo il nostro dataset affetto da un serie di debolezze (cfr sez. 3.2) abbiamo anche fatto largo uso di tutti i possibili schemi di validazione messi a disposizione da Weka, in particolar modo la cross validation, utilizzando un numero di fold standard, ma anche provando l'approccio leave-one-out abbastanza dispendioso in termini di tempo. Questo argomento verrà discusso approfonditamente nel paragrafo 4.5.

Estremamente utili e facili da usare sono i tool di visualizzazione dei record su grafici 2d per confrontare la distribuzione di questi nelle varie classi rispetto a due feature a scelta. La consultazione dei grafici risulta ancora più facile tramite strumenti di zoom e di selezione dei record per visualizzare in particolare i valori di ogni singola istanza.

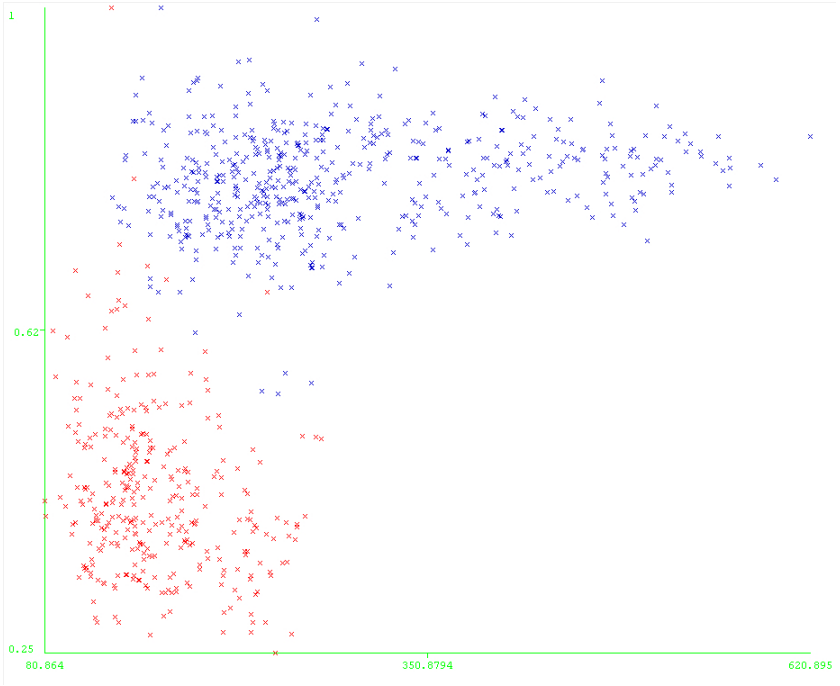


Figura 4.2: Tool di visualizzazione, nel grafico è evidente come le due feature separino quasi perfettamente le due classi

Le uniche limitazioni per Weka sono date dalla java virtual machine, non particolarmente veloce ed adatta a lavorare in multithreading con limiti di memoria troppo stringenti per i computer d'oggi e le grandi quantità di dati dei dataset

4.2 C4.5

Nella necessità di acquisire maggiori informazioni riguardo le possibili feature caratterizzanti di ogni specie, è stata fatta la scelta di iniziare a lavorare con alberi decisionali, che fra i classificatori si distinguono per facilità di interpretazione e limitata parametrizzazione che consente comunque robustezza al rumore e limitazione dell'overfitting. Nonostante il problema da risolvere può essere considerato NP-completo, molti approcci utilizzano euristiche: l'algoritmo che presenteremo utilizza un approccio greedy, top–down, con una strategia di partizionamento ricorsivo. In questo modo il tempo di costruzione dell'albero non dipenderà più dalla taglia del dataset. Il test di un record è $O(w)$, con w profondità massima dell'albero. Gli alberi decisionali permettono una classificazione multi-classe.

Sono classificatori che risolvono il problema della classificazione ponendo una serie di domande sui valori delle feature del record da classificare. Per ogni domanda, posta in un nodo interno dell'albero, le possibili risposte portano, attraverso gli archi

uscenti, in altri nodi interni o alle foglie.
In questo modo le domande poste una dopo l'altra partizionano lo spazio delle feature e dividono il dataset in sottoinsiemi di record che condividono le proprietà specificate dalle domande e dalle risposte. Quindi un albero decisionale è composto da una radice, che corrisponde alla prima domanda posta per ogni record, senza archi entranti e con più archi uscenti, i nodi interni che pongono le domande successive, con un solo arco entrante e più archi uscenti (il vincolo del singolo arco entrante garantisce la forma di grafo ad albero e non grafi più complessi), e foglie, le quali corrispondono ad una possibile classificazione. Ogni record analizzato, attraverso il percorso dettato dalle risposte alle domande incontrate nel cammino sui nodi, porta ad una ed una sola foglia e quindi ad una classificazione univoca.

Come accennato all'inizio del paragrafo, la prima scelta di classificatore, è caduta su di un classificatore ad albero soprattutto per la facilità d'interpretazione che ha consentito un'analisi approfondita su quali feature fossero più o meno determinanti per ogni specie di alberi presa in esame. Questo step è stato eseguito controllando come veniva diviso il training set fra le diverse classi da ogni nodo, e valutando di volta in volta, se la feature fosse una buona feature, in quanto riusciva a dividere i record secondo le classi e secondo le nostre conoscenze biologiche descritte nel capitolo 3. L'algoritmo scelto, che verrà descritto in seguito, alle volte sceglieva dei test anche troppo particolareggiati, che non si affiancavano bene con l'errore statistico insito nelle feature prelevate da noi. Inoltre, alle volte, scegliendo determinati insiemi di feature, arrivavamo ad avere una classificazione molto fedele ma evidentemente overfitted per il nostro dataset, anche alla luce delle considerazioni sulla varietà di alberi a nostra disposizione espressa nel paragrafo 3.2.

Con questo approccio, e appoggiandoci alle funzionalità di Weka, siamo riusciti ad isolare una serie di feature, non più di 4 per specie e molte ripetute, i cui valori riuscivano a distinguere bene una specie da tutte le altre. In parte queste feature erano quelle attese, caso eclatante quello per i Pine che si dividono quasi perfettamente grazie alla sola misura di distribuzione dei nodi in verticilli, e per i quali, aggiungere altre feature non portava a miglioramenti sostanziali alla classificazione.

Weka, l'ambiente nel quale abbiamo calcolato i classificatori, mette a disposizione molti algoritmi che danno in output un albero decisionale, la nostra scelta è caduta su J48, implementazione in Java dell'algoritmo C4.5 [8, 9] per familiarità, conoscenze pregresse e perché è spesso presentato come uno dei migliori algoritmi per assolvere a questo scopo.

In sostanza, C4.5, è un'implementazione efficiente dell'algoritmo di hunt [10]. Come descritto sopra, l'algoritmo di hunt si occupa di dividere il training set, ad ogni passo, in training set più piccoli dividendo l'input in base ad un test su di una singola feature fino ad avere record di una singola classe nelle foglie. I challenge a cui deve rispondere l'algoritmo sono la scelta della feature, la scelta del test da fare e le possibili risposte, le condizioni per fermare la ricorsione e l'etichettatura delle foglie con la classe. Lo pseudo codice del procedimento è descritto dall'algoritmo 1

Le funzioni stoppingCondition, Classify e findBestSplit esprimono rispettivamente la sfida per fermare la ricorsione, quella di etichettare le foglie e quella di trovare la miglior feature, fra tutte le feature F e le migliori possibili risposte che dividano al meglio il training set E.

Algoritmo 1 Algoritmo di Hunt

```
1: Hunt(E,F)
2: if stoppingCondition(E,F) = true then
3:     leaf = creatNode()
4:     leaf.label=classify(E) return leaf
5: else
6:     root = creatNode()
7:     root.test–cond=findBestSplit(E,F)
8:     sia V = {v|v una possibile risposta a root.test–cond}
9:     for all v ∈ V do
10:        E_v = {e|root.test–cond(e)= v ∧ e ∈ E}
11:        child = Hunt(E_v,F) root→child con etichetta vù
12:    end for
13: end if
14: return root
```

StoppingCondition(E,F) generalmente usa una soglia di record minimi nelle foglie, se nel training set ci sono un numero minore o uguale alla soglia allora la crescita dell'albero viene troncata. Altra condizione è la presenza di record di una singola classe. Alle volte viene usato anche l'approccio di limitare la profondità dell'albero.
Per evitare overfitting si può compire un pre–pruning utilizzando una soglia sul guadagno introdotto dal miglor split, se tale guadagno è inferiore ad una data soglia, la cresita dell'albero viene interrotta.

Classify(E) impone l'etichetta alla foglia controllando quale è la classe maggioritaria in E.

findBestSplit(E,F) fa uso di diverse possibili misure di impurità e in base anche a questa scelta di misura può dare output diversi, in particolare può usare misure come l'Entropia, il Gini Index, o la statistica Chi quadro.

L'output di J48 non prevede split multipli, ma solo split binari a test a cui si possa rispondere vero o falso: per attributi numerici, continui o discreti, viene fatto un test del tipo maggiore–minore, per attributi nominali viene fatto un test del tipo uguale o appartente ad un sottoinsieme dei valori, per attributi binari è triviale.

Per limitare l'overfitting, l'approccio post–pruning da risultati migliori rispetto al pre–pruning perché lavora sull'albero completamente sviluppato ed evita una terminazione prematura della crescita. Una volta costruito l'albero ottimale, ne viene fatto un pruning dal basso verso l'alto dei nodi, unendo le foglie derivanti dal test corrispondente a tale nodo. L'idea è comune a quella del pre–pruning. Questo approccio post–pruning è conosciuto come Subtree Raising.

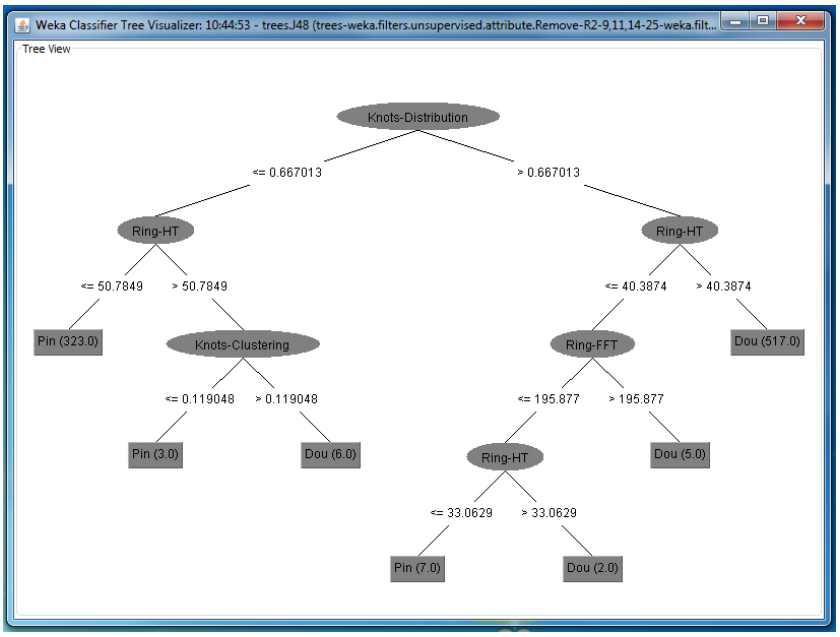


Figura 4.3: Albero di decisione: Output di J48

4.3 Support Vector Machine (SVM)

Le SVM sono fra i classificatori più utilizzati al mondo per la loro flessibilità e velocità di verifica. Sono considerate lo stato dell'arte per risolvere un problema di classificazione. Particolarmente adatte a lavorare con dati con un numero di dimensioni elevate, vincono le debolezze degli alberi di decisione potendo dividere lo spazio non solo per una dimensione alla volta. La risoluzione del modello, come vedremo in seguito, si basa su problemi largamente affrontati in letteratura per i quali esistono algoritmi efficienti per trovare l'ottimo globale, mentre per alberi di decisione e reti neurali vengono adottate soluzioni euristiche di tipo greedy che portano solo ad un ottimo locale.

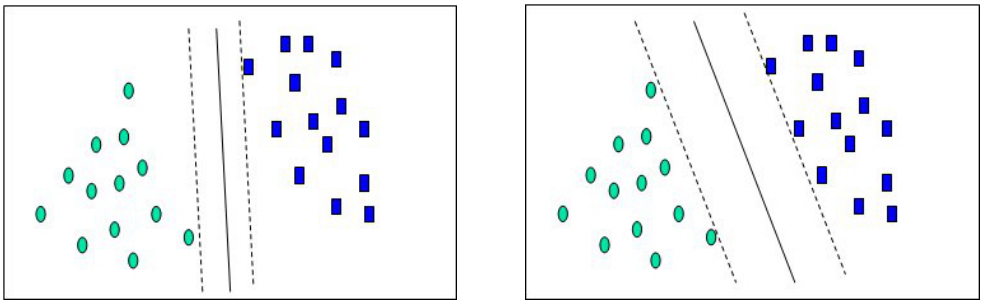
D'altro canto le SVM richiedono una forte parametrizzazione e, a meno del caso base linearmente separabile, sono di difficile interpretazione. Operano bene solo con attributi numerici o binari, quindi eventuali attributi nominali vengono tradotti in molti attributi binari, uno per ogni possibile valore dell'attributo nominale, causando un aumento della dimensione e una possibile inconsistenza di stati. Nel nostro caso non ci saranno attributi nominali, ma solo attributi numerici che possono essere considerati reali

Molte implementazioni delle SVM permettono solo classi binarie e, per le implementazioni che lo permettono, la risoluzione del problema per classi non binarie ha una complessità più alta.

Nonostante tutto ciò abbiamo scelto di lavorare con questo modello perché ci garantiva ottime prestazioni in fase di verifica e robustezza al rumore, oltre al fatto che induce implicitamente ad una misura di distanza fra il record in esame e le varie classi.

L'idea di base dietro a le SVM è la possibilità di dividere i record di una classe da quelli dell'altra attraverso un iperpiano su n dimensioni con i record definiti su

$m \leq n$ dimensioni e di farlo con l'iperpiano che lasci maggior margine. Per meglio capire questa idea, si può ragionare con solo 2 dimensioni.



(a) Margine stretto (b) Margine massimo

Figura 4.4: SVM valida con margine stretto e con margine largo

Come si vede dall'immagine 4.4, per un dato insieme di record possono esistere più iperpiani che dividono perfettamente le due classi. Il problema di ottimizzazione legato alle SVM vuole scegliere quello che massimizza la distanza fra l'iperpiano stesso e il record più vicino all'iperpiano per ogni classe. Cioè, facendo allontanare due piani paralleli all'iperpiano nei due versi e facendoli fermare quando uno dei due incontra un record di una classe, si vuole che l'iperpiano scelto sia quello che può far allontanare il più possibile tali piani paralleli. Il record che incontra l'iperpiano si chiama Support Vector. La distanza fra i due piani paralleli viene definita margine.

Non sempre però le classi si possono dividere così nettamente anche se su più dimensioni, e non sempre i piani che incontrano i Support Vector sono ad una distanza soddisfacente, per questo motivo si possono adottare alcuni trucchi per riuscire a descrivere al meglio il comportamento delle classi sulle m dimensioni dei record.

In generale si può permettere che un numero limitato di record sia all'interno del margine, anche dalla parte opposta dell'iperpiano rispetto alla sua classe (*soft margin approach*), e soprattutto si può studiare una trasformazione dello spazio in modo da poter trovare un piano lineare su questo nuovo spazio che nello spazio originale non esisterebbe. Questo secondo espediente si chiama *Kernel Trick*.

Consideriamo un problema di classificazione binaria con N record nel training set $T = \{t_i = (\boldsymbol{x_i}, y_i) i = 1, ..., N\}$ con $\boldsymbol{x_i} \in \Re^n$ l' i–esimo record e $y_i \in \{-1, 1\}$ l'etichetta della classe di tale record. Il decision bondary avrà quindi equazione

$$\boldsymbol{w}^T \boldsymbol{x} + b = 0$$

dove $\boldsymbol{w} \in \Re^n$ e $b \in \Re$ sono parametri del modello. Valutando l'equazione per un punto $\hat{\boldsymbol{x}}$ non appartenente al al piano si avrà $\boldsymbol{w} \cdot \hat{\boldsymbol{x}} + b = k$ con $k < 0$ se il punto si trova sotto al piano e $k > 0$ se si trova sopra. Si intuisce subito quindi che la classe predetta per tale punto sarà

$$\hat{y} = sign(\boldsymbol{w} \cdot \hat{\boldsymbol{x}} + b)$$

Il problema da risolvere consiste nel trovare $\boldsymbol{w}$ e b tale che i piani paralleli al decision boundary che determinano il margine abbiano equazione

$$p_1 : \boldsymbol{w} \cdot \boldsymbol{x} + b = 1$$
$$p_2 : \boldsymbol{w} \cdot \boldsymbol{x} + b = -1$$

e che il margine, cioè la distanza fra tali iperpiani, sia massimo. L'ampiezza del margine è data da

$$d = \frac{2}{||\boldsymbol{w}||} \tag{4.1}$$

Il margine più ampio possibile obbliga che almeno un record per classe stia su p_1 e p_2. Volendo massimizzare (4.1), e volendo che non ci siano punti all'interno del margine si ottiene il problema di ottimizzazione

$$\min_{\boldsymbol{w}} \frac{||\boldsymbol{w}||^2}{2} \tag{4.2}$$
$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1 \quad i = 1, ..., N \tag{4.3}$$

che essendo un noto problema di ottimizzazione convessa è facilmente risolvibile con il metodo dei moltiplicatori lagrangiani. Risolvendo tale problema, attraverso i valori dei moltiplicatori si evince che l'intera SVM dipende unicamente dai Support Vector, cioè quei record che si trovano sui piani delimitanti il margine (i soli per i quali i moltiplicatori di lagrange siano uguali a 1 e diversi da 0).

Se il margine non è sufficientemente ampio, magari a causa di alcuni record spuri si può rilassare il vincolo riguardante la presenza di record all'interno del margine stesso con l'aggiunta di N variabili di *Slack* ξ. Il problema (4.2) diventa quindi

$$\min_{\boldsymbol{w}} \frac{||\boldsymbol{w}||^2}{2} + C(\sum_{i=1}^{N} \xi_i)^k \tag{4.4}$$
$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1 - \xi_i \text{ if } y_i = 1 \tag{4.5}$$
$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \leq -1 + \xi_i \text{ if } y_i = -1 \tag{4.6}$$
$$\xi_i \geq 0 \ \forall i \tag{4.7}$$

dove C e k servono a penalizzare la classificazione errata, quindi a limitare il rilassamento del problema e le ξ_i sono diverse da zero solo per i record che si trovano all'interno del margine. In particolare $\xi_i \geq 1$ se e solo se t_i verrà classificato sbagliato.

Se anche questo passaggio non da risultati soddisfacenti si può ricorrere ad un'ulteriore generalizzazione delle SVM. L'idea di base consiste nel trasformare lo spazio delle feature tramite la mappa non lineare $\phi : \Re^n \to \Re^m$ con $m > n$ nel quale sia possibile trovare un iperpiano che separa le classi. Tale mappa $\phi(\boldsymbol{x})$ è implicitamente definita dalla *Kernel Function* $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \phi(\boldsymbol{x_i}) \cdot \phi(\boldsymbol{x_j})$ il prodotto interno fra due punti nello spazio trasformato.

Fra le *Kernel Function* più utilizzate ci sono le funzioni polinomiali

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \left(\boldsymbol{x_i}^T \boldsymbol{x_j} + 1\right)^d$$

e le *Radial Basis Function*

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = exp\left(-\frac{||\boldsymbol{x_i} - \boldsymbol{x_j}||^2}{\sigma^2}\right)$$

Se si sceglie $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{x_i}^T \boldsymbol{x_j}$ la SVM non lineare si riduce alla sua versione lineare.

In questo caso la classe viene decisa dalla funzione

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x}) + b = \sum_{i=1}^{M} \alpha_i y_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b \tag{4.8}$$

dove gli $\boldsymbol{x_i}$ sono i support vector di classe y_i e sono M.

4.3.1 SVM Multi classe

Consideriamo il problema di assegnare una sola classe ad un record scegliendo fra C possibili classi, avendo a disposizione un dataset $T = \{t_i = (\boldsymbol{x_i}, y_i) : i = 1, ..., N\}$ con $\boldsymbol{x_i} \in \Re^n$ e $y_i \in \{1, ..., C\}$. Le SVM, come spiegato all'inizio del paragrafo4.3, a differenza degli alberi di decisione, funzionano solo con classi binarie. Esistono alcuni lavori [13, 14, 15, 16, 17, 18] che propongono generalizzazioni delle svm per gestire il caso multi classe, ma come sostenuto da Rifkin et al. in [12] e da Liu et al. in [20] le SVM, se progettate con accuratezza forniscono automaticamente un approccio multi classe basato sulla misura derivata dal valore della decision function. Tale approccio prende il nome di One–vs–All perché si propone di ottenere uno score relativo ad ogni classe calcolando un singola SVM per determinare o meno l'appartenenza di un record ad ogni classe. Avendo, come nell'esempio, C classi diverse, bisogna costruire C diversi classificatori con l'intero training set, cambiando le etichette delle classi di volta in volta. L'i–esimo classificatore farà corrispondere alla classe positiva la classe i, e alla classe negativa tutte le altre

$$\hat{y_j} = \begin{cases} 1\, if & y_j = i \\ -1\, if & y_j \neq i \end{cases} \tag{4.9}$$

Per il record $\boldsymbol{x}$ verranno quindi calcolate tutte le decision function $f_i(\boldsymbol{x}) = \boldsymbol{w_i}^T \phi(\boldsymbol{x}) + b_i$ e verrà assegnata la classe $y = i^*$ dove f_{i^*} è il valore massimo, quindi

$$i^* = \underset{i=1,...,C}{\arg\max}\, f_i(\boldsymbol{x}) \tag{4.10}$$

4.4 Artificial Neural Network

Le reti neurali si rifanno al funzionamento del cervello umano, nel quale il processo di apprendimento si articola attraverso la comunicazione ripetuta fra neuroni. Il nostro cervello è composto dalle cellule neurali dette neuroni collegate fra loro tramite gli assoni(fasci di fibre), i dendriti(propaggini dei neuroni) e le sinapsi (punti di contatto fra dendriti e assoni) , il cervello impara cambiando lo stato delle sinapsi. Analogamente alla struttura cerebrale umana, le ANN sono composte da

nodi interconnessi tra loro, ognuno dei quali ha multipli archi entranti, che portano diverse informazioni, e multipli archi uscenti che inoltrano la stessa informazione / decisione ad altri nodi. La rete neurale più semplice composta da un solo neurone è detta perceptron e consiste in una funzione $f_p(x, w)$ che ricevuto in input il vettore $x = (1, x_1, x_2, ..., x_N)$ compie una combinazione lineare delle sue componenti tramite i pesi w

$$\hat{y} = sign\left(f_p(x, w)\right) = sign\left(\sum_{i=0}^{N} w_i x_i\right) \tag{4.11}$$

Il processo di apprendimento del singolo nodo è dato da:

Algoritmo 2 Algoritmo d'apprendimento di un perceptron

1: Sia D = $\{(x_i, y_i)|i = 1, ..., M\}$ il Training set
2: $w_j^0 = random\ \forall j = 0, 1, ..., N$
3: **repeat**
4: **for all** $(x_i, y_i) \in D$ **do**
5: $\hat{y}_i^k = sign(f_p(x, w^k, t^k))$
6: **for all** j **do**
7: $w_j^{k+1} = w_j^k + \lambda(y_i - \hat{y}_i^k)x_{ij}$
8: **end for**
9: **end for**
10: **until** è stata raggiunta una condizione di stop

Dove λ è un parametro dell'algoritmo detto learning rate. le condizioni di stop sono varie e possono essere legate all'errore assoluto sul test set, al miglioramento dell'errore o al numero di iterazioni. Un singolo neurone, in questo modo, produce un iperpiano lineare che divide i record delle due classi in modo non ottimo. Produce quindi un risultato simile ad una SVM ma con una soluzione approssimata. La potenza delle reti neurali sta nel poter collegare diversi nodi di questo tipo in una rete complessa. Uno dei modelli più semplici prevede 3 livelli: un livello di input, un livello nascosto che prende gli input e li rielabora e un livello di output che accoglie gli output del livello nascosto li combina e determina la classificazione. I nodi interni usano, oltre alla combinazione degli input, anche delle funzioni di attivazione f_a diverse. Le più comuni sono:

Funzione Lineare $f_a(z) = z$ la pendenza e l'intercetta di questa funzione vengono date implicitamente da w;

Sigmoide $f_a(z) = \frac{1}{1+e^{-z}}$ w determina l'ampiezza e dove è centrata la curva calcolata partendo da x;

Tangente Iperbolica $f_a(z) = tanh(z)$ omologa alla sigmoide;

Funzione Segno $f_a(z) = sign(z)$ anche in questo caso saranno i pesi a determinarne il comportamento nel domino delle x.

I nodi interni non per forza prendono in input tutte le feature. Nella nostra implementazione useremo solo funzioni Sigmoidali.

Un organizzazione di questo tipo può sfruttare lo stesso algoritmo per il singolo neurone esteso alla catena di neuroni.

La potenza di questi classificatori, per la quale la scelta finale è ricaduta su di essi, è data dalla buona interpretabilità dei risultati e dalla possibilità di personalizzare la topologia della rete in base alle necessità, oltre ad una buona resistenza al rumore e meccanismi studiati apposta per evitare l'overfitting. Inoltre, come verrà mostrato nel prossimo paragrafo, si possono facilmente estendere le reti studiate per un classificatore binario al classificatore multi–classe.

L'implementazione di Weka per le reti neurali, anche per casi multi–classe, è MultilayerPerceptron che, oltre a dare grande libertà nella parametrizzazione degli algoritmi di learning, offre anche un'interfaccia grafica intuitiva per la costruzione di reti neurali personalizzate. In figura 4.5 si può notare come si è scelto di creare neuroni con in input le singole feature e di combinare poi i risultati solo nei nodi di output. Si può notare anche che non tutti i parametri sono stati utilizzati, ma solo quelli specifici delle due specie Pine e Douglas.

L'utilizzo di nodi con in input una sola feature permette di analizzare i risultati e capire se questi nodi si comportano come ci si aspetta. In particolare è stato valutato dove le sigmoidi hanno il flesso rispetto ai valori delle feature e quanto ampie sono le sigmoidi. Ci si aspetta che il flesso sia in corrispondenza delle zone di conflitto fra le due classi, e che la sigmoide saturi dove la zona di conflitto finisce, questo comportamento è maggiormente comprensibile in figura 4.6

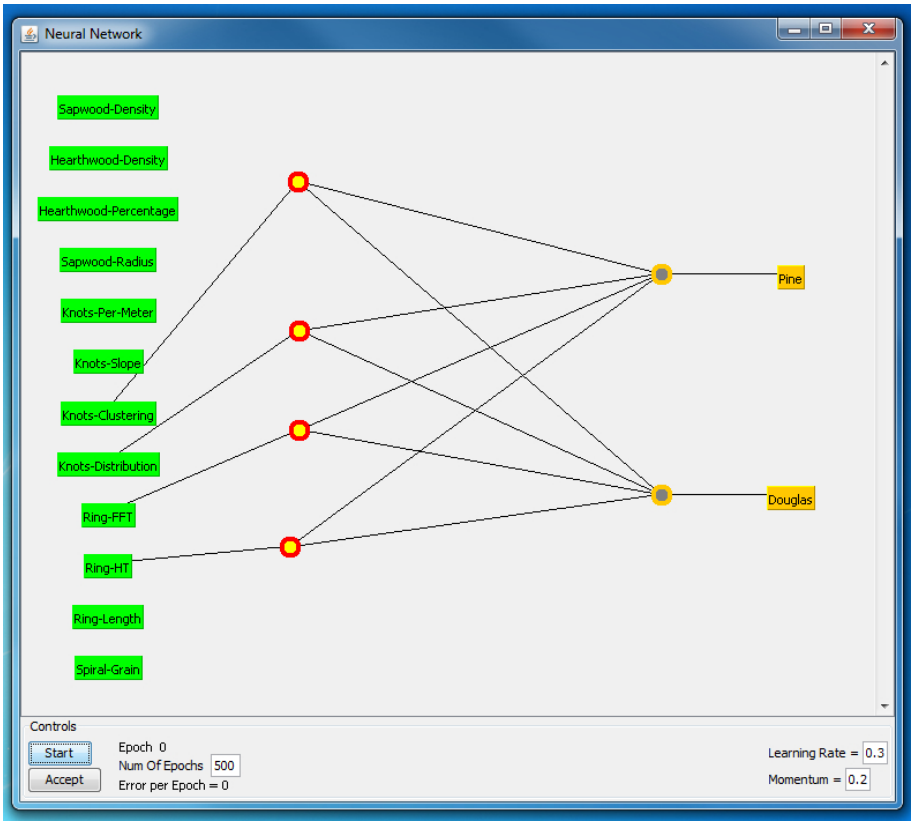


Figura 4.5: Gui di Weka per la creazione di ANN personalizzate

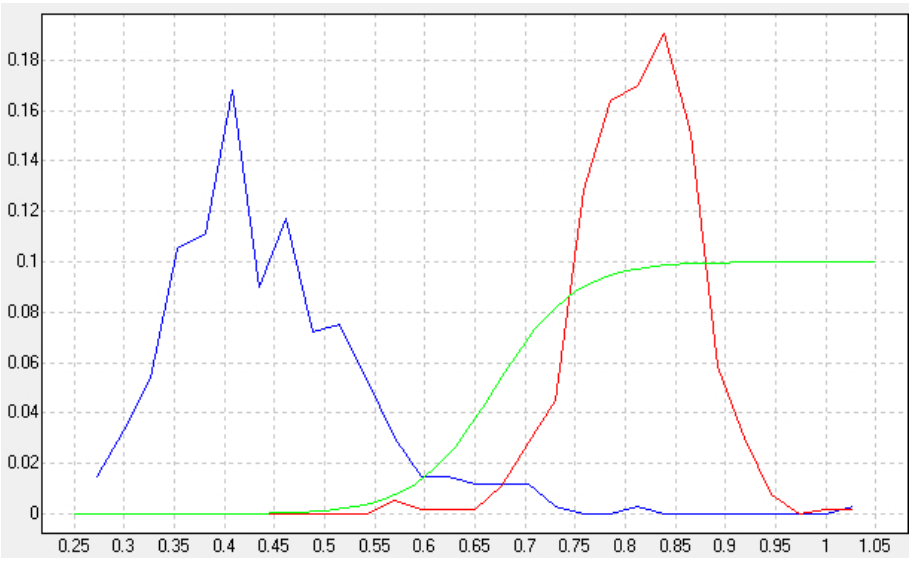


Figura 4.6: Istogramma di Knots–Distribution per i Pine (in blu) e i Douglas (in rosso). La sigmoide, scalata, risultante dal nodo di competenza è in verde

4.4.1 ANN Multi Classe

A differenza delle SVM, il caso Multi–classe per le ANN è una semplice combinazione dei classificatori binari. Infatti si possono affiancare i vari strati nascosti dei singoli classificatori in parallelo e aggiungere un altro livello nascosto che simuli la decisione fra un classe e un'altra.

L'approccio utilizzato è quindi simile ad un approccio One–Vs–One dove viene dato un punteggio positivo ad ogni classe che vince uno scontro e poi questi voti vengono sommati per decidere chi sia il vincitore assoluto. Questo approccio ha una debolezza insita nel fatto che ci possano essere dei pareggi, possibilità resa remota dalla natura dei punteggi. Infatti scegliendo di dare punteggi non interi, ma reali, è molto difficile che due classi abbiano la stessa somma finale di voti. È facile capire che questo passo di assegnare punteggi diversi per ogni classificatore è proprio come aggiungere un ulteriore livello di nodi della rete neurale come spiegato sopra.

4.5 Valutazione e validazione di un classificatore

Sono molte le metriche per valutare la bontà e l'affidabilità di un classificatore, ma nell'ottica commerciale di dover spiegare i risultati ai clienti, abbiamo deciso di affidarci solamente a due metriche: la True Positive Rate e la precision che misurano, rispettivamente, la percentuale di record classificati giusti e la percentuale di record classificati giusti fra quelli classificati nella singola classe.

La tabella 4.1 mostra la terminologia utilizzata per riferirsi al conteggio dei record classificati

TP True Positive, il numero degli esempi positivi correttamente classificati

FN False Negative, il numero degli esempi positivi erroneamente classificati come negativi

		Predetta +	Predetta −
Attuale	+	TP	FN
	−	FP	TN

Tabella 4.1: confusion matrix per un problema di classificazione binaria

FP False Positive, il numero degli esempi negativi erroneamente classificati come positivi

TP True Negative, il numero degli esempi negativi correttamente classificati

Con tale terminologia la True positive rate viene calcolata come

$$TPR = \frac{TP}{TP + FN} \tag{4.12}$$

Mentre la precision P è definita

$$P = \frac{TP}{TP + FN} \tag{4.13}$$

Nonostante sembrino due metriche molto simili hanno significato diverso e possono assumere valori molto dissimili come si vede dalle tabelle 4.2 e 4.3 nelle quali $TPR_1 = TPR_2 = \frac{16}{16+4} = 0.8$ mentre $P_1 = \frac{16}{16+1} \simeq 0.94$ e $P_2 = \frac{16}{16+10} \simeq 0.62$

		Predetta +	Predetta −
Attuale	+	16	4
	−	1	9

Tabella 4.2: confusion matrix per classificatore con $TPR_1 = 0.8$ e$P_1 \simeq 0.94$

		Predetta +	Predetta −
Attuale	+	16	4
	−	10	0

Tabella 4.3: confusion matrix per classificatore con $TPR_1 = 0.8$ e$P_1 \simeq 0.62$

In entrambi i casi la TPR ha valore abbastanza alto, che sta a significare che i classificatori trovano una buona parte dei record positivi e quindi sembrerebbero entrambi buoni classificatori, ma la precision del secondo classificatore, sensibilmente più bassa rispetto a quella del primo, implica che gran parte dei record viene classificata come positiva, anche quelli negativi, e quindi il classificatore non è buono. Stesso discorso si può fare con una precision alta pressochè costante, ma con true positive rate variabile, in questo caso avremo un classificatore buono e uno che ha condizioni per la positività troppo stringenti.

Nel nostro caso, avremo però più classi da gestire e quindi considereremo le medie pesate di queste misure. Sia C la matrice di confusione e sia l'elemento c_{ij} il numero dei record di classe i che il classificatore ha definito di classe j. Definiamo inoltre le somme parziali di righe e colonne come A_i il numero di record effettivamente della classe i, Q_j il totale dei record classificati come j e N il numero di elementi del Test Set. La generalizzazione della (4.12) sarà data dalla media delle TPR pesata sul numero di record di ogni classe

$$\overline{TPR} = \frac{\sum_i TPR_i A_i}{N} = \frac{\sum_i \frac{c_{ii}}{A_i} A_i}{N} = \frac{\sum_i c_{ii}}{N} \tag{4.14}$$

Anche la (4.13) diventerà in ugual modo la media pesata delle precision per ogni singola classe

$$\overline{P} = \frac{\sum_i P_i A_i}{N} = \frac{\sum_i \frac{c_{ii}}{Q_i} A_i}{N} \tag{4.15}$$

La scelta di queste metriche è stata fatta in quanto sono molto più facili da spiegare agli eventuali clienti ed è immediato comprendere se un classificatore compie buone scelte osservando i risultati di tali metriche. Inoltre, in fase di studio dei classificatori, sono state utilizzati i risultati parziali di questi valori, cioè le singole TPR e P per una singola classe, per capire dove migliorare la classificazione, infatti una TPR bassa per una classe implicava che troppi record di quella classe finivano in altre classi cioè c'era un fenomeno di underfitting per quella specifica classe, mentre una P bassa implicava che troppi record di altre classi finivano in quella determinata classe, quindi overfitting.

Nonostante tutto ciò, le due statistiche presentate possono ancora non essere completamente esaustive, quindi, solo in fase di verifica, abbiamo deciso di utilizzare la *Kappa di Cohen*. Questa metrica, di fatto, mette insieme le prime due, fondendo probabilità a priori e a posteriori che il classificatore compia le giuste scelte. La probabilità a posteriori è la media pesata della TPR, che può essere anche interpretata come la percentuale di volte in cui il classificatore e la specie conosciuta concordano che indicheremo con $Pr(a) = \overline{TPR}$, mentre la probabilità a priori è la misura di quanto ci aspettiamo le due statistiche concordino casualmente, e viene definita $Pr(e) = \frac{\sum_i Q_i A_i}{N^2}$

La Kappa misura quindi quanto il classificatore sia in accordo con le classi prestabilite correggendo tale misura rispetto alla probabilità di accordo a priori . Questa misura si calcola così:

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \tag{4.16}$$

Quindi $-1 \leq \kappa \leq 1$. Se assume valori vicini a 0 non c'è correlazione oppure non ce n'è più di quanto ci si aspettasse. Quanto è più vicina a 1 tanto maggiore sarà la concordanza, tipicamente per valori $\kappa \geq 0.8$ il classificatore viene considerato molto buono.

Calcolare le metriche appena presentate sullo stesso insieme di record sul quale è stato addestrato il classificatore non è corretto perché quel classificatore è stato costruito cercando di minimizzare l'errore sul training set stesso.

A questo scopo sono stati usati due approcci per validare i classificatori risultanti da questo studio, e sui risultati di questi due approcci sono state calcolate le metriche.

Il primo e più semplice metodo consiste nel sottocampionare il trainig set, in due parti, tipicamente con un rapporto $2:1$. In questo modo i $\frac{2}{3}$ dei record rimangono per il training, mentre il terzo rimanente viene impiegato per il test. Su qeusto test set vengono calcolate le metriche. Alle volte questo approccio però rischia di limitare troppo il numero di record in uno o nell'altro insieme di una determinata classe. Nel caso in esame, ad esempio, la classe Pine è estremamente sotto rappresentata rispetto alle altre.

Questa difficoltà è aggirabile con l'approccio Cross–Validation. In questo caso si partiziona l'insieme dei record in N parti di ugual numero ed ogni sottoinsieme viene usato $N-1$ volte per il training una sola volta per il test. Il classificatore risultante avrà quindi sfruttato tutti i record a disposizione per essere creato e per essere valutato. Il caso particolare con N uguale alla taglia del training set prende il nome di Leave–one–out. È particolarmente dispendioso in termini di tempo, ma è la migliore misura che si possa ottenere basando training e test sullo stesso insieme.

Il classificatore finale, prodotto al termine di questo studio è stato validato anche con un test in impianto su dati non a disposizione durante lo sviluppo, durante un test generale del CT.LOG.

Capitolo 5

Elaborazione e selezione delle feature

L'ampio dataset a disposiione (sez 3.2) è composto da scansioni di tronchi in tonalità di grigio a 8 bit per la densità con risoluzione di $1mm$ nella sezione trasversale e di $11mm$ nella direzione longitudinale, questo porta ad avere, per un tronco di 8 metri (misura standard delle tavole lavorate) circa 730 slice da 700×700 pixel, che anche se compresso, occuperà circa $50MB$. Queste immagini, se analizzate ad occhio nudo, in una o l'altra sezione, possono dare molte informazioni, è infatti molto semplice individuare i nodi più grossi, vederne la disposizione, individuare il pith, gli anelli di accrescimento, il limite fra heathwood e sapwood, non è altrettanto facile determinare quanto larghi siano gli anelli, quanto il pith si muova, come evolvono i crepi, se l'albero è soggetto ad avvitamento. Inoltre ci sono tutta una serie di valori estremamente difficili da stimare se non attraverso valutazioni statistiche. Per ogni albero bisogna riuscire a ridurre la mole di informazioni in pochi e concisi parametri che ne permettano la classificazione senza dover eccedere con il numero di feature e quindi i gradi di libertà.

Le feature che verranno descritte in seguito non sono state tutte utilizzate per il classificatore implementato e sperimentato in impianto, ma vengono presentate lo stesso in quanto sono state comunque utili alla scelta finale o potrebbero tornare utili per un futuro ampliamento del classificatore ad altre specie.

In [2] Charpentier et al. presentano un loro lavoro sulla classificazione da immagini tomografiche che si basa su dati ad altissima risoluzione ottenuti con scanner medicali molto lenti. I risultati ottenuti nel loro lavoro non sono pienamente compatibili con questo, ma confermano la possibilità di risolvere il problema della classificazione con un approccio di tipo statistico.

5.1 Feature

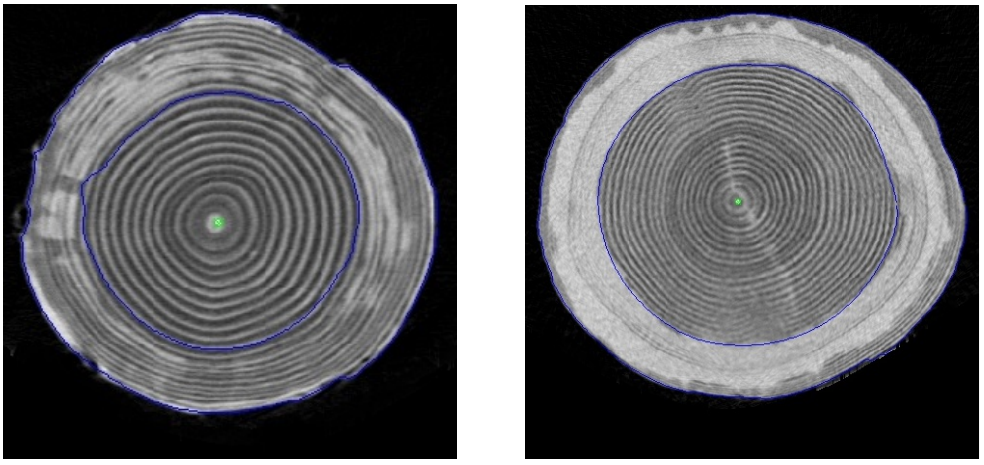
Di seguito verranno presentate le feature utilizzate, come vengono calcolate e come siano distintive per determinate specie.

Le feature calcolate sui nodi, sono calcolate a partire dai nodi individuati tramite l'algoritmo messo appunto da Microtec sulla base dei lavori di Andreu et al. [24] e Johansson et al. [25]. È stata fatta la scelta di pre–elaborare ogni tronco e creare un file a tronco che contenesse tutte le informazione che si estraggono per ogni slice e per ogni nodo. Questo permette di non dover lanciare l'algoritmo di elaborazione dei

dati grezzi che impiega diversi secondi a tronco ogni volta che si modifica il calcolo di qualche feature. A questo punto tutte le informazioni da rielaborare di volta in volta occupano $570MB$ e per ricreare il dataset con le feature basta qualche decina di secondi. Nonostante questo espediente alcune volte è stato necessario lanciare un nuovo batch su tutto l'insieme dei tronchi per raccogliere le informazioni aggiornate dopo il miglioramento di uno o più algoritmi di analisi o rilevazione dei difetti.

5.1.1 Sapwood Density

Questa feature non è usata direttamente dal classificatore vero e proprio per distinguere la specie, ma per determinare se un tronco è secco o ancora fresco. Infatti un tronco secco ha caratteristiche molto differenti e gli stessi algoritmi per trovare i difetti interni del tronco risentono molto di questa condizione. Inoltre i tronchi secchi non vengono lavorati in segheria come gli altri, visto che sono soggetti a crepi e spaccature durante la lavorazione. Oltretutto, il dataset presenta un numero esiguo di tronchi secchi che non permette di sviluppare un classificatore apposito una volta riconosciuta la secchezza o meno. Per riconoscere questi tronchi secchi ci affidiamo alla densità del sapwood, infatti un tronco che si è asciugato troppo presenta una densità del sapwood molto inferiore, visto che ha perso gran parte dell'acqua in favore dell'aria. La densità del sapwood è calcolata come rapporto fra peso e volume



(a) Secco (b) Non secco

Figura 5.1: Tronchi di douglas secco e non. In blu i bordi rilevati del tronco e del sapwood

del sapwood di ogni slice. In seguito viene filtrato lungo la lunghezza del tronco e, al fine di utilizzare questa caratteristica per decidere se un albero è secco o meno, ne viene fatta la mediana su tutti i valori. In accordo con i collaboratori di FVA si è deciso di scartare e classificare come secchi tutti i tronchi che hanno densità del sapwood calcolata in questo modo inferiore a $550\mu g/mm^3$.

5.1.2 Hearthwood Density

È calcolata omologamente alla Sapwood Density nella zona interna al bordo fra hearthwood e sapwood e ne viene preso l'ottantesimo percentile, ma viene utilizzata

per la classificazione, in particolare è una delle caratteristiche distintive fra Spruce e Fir. La scoperta di quanto questa feature sia rilevante è stata fatta in fase di elaborazione del classificatore, infatti, l'algoritmo c4.5 usava questo valore per fare il primo e principale split fra le due specie. Si è scelto un percentile abbastanza alto solo dopo aver fatto diverse prove e aver visto che un valore alto era maggiormente distintivo. Di fatto i fir arrivano ad avere una densità massima dell'hearthwood maggiore di quella degli spruce, viene preso l'ottantesimo percentile per non incappare in errori dovuti a valori spuri o erroneamente troppo alti dovuti a marciume.

5.1.3 Hearthwood Percentage

Anche la percentuale di area occupata dall'hearthwood risulta una buona feature in alcuni casi, anche se nella scelta finale è stata accantonata in quanto non dava risultati migliori di altre feature. Il calcolo è compiuto prendendo l'area dell'hearthwood e di tutto il tronco per ogni slice, ne viene fatto il rapporto e si sceglie il valore mediano per evitare i valori estremi. Infatti il cimale dei tronchi di punta tende ad avere molto poco hearthwood rispetto al sapwood, mentre per le slice molto larghe alla base per tronchi leggermente secchi si fatica ad individuare correttamente il bordo del sapwood con conseguente alterazione delle aree. I Pine e i Douglas tendono ad avere valori estremamente differenti, ma per queste specie sono state studiate differenti feature apposite per far emergere le loro particolarità e quindi questa feature non viene mai selezionata nel classificatore.

5.1.4 Raggio medio del Sapwood

Questa feature, calcolata a partire dall'individuazione del sapwood spiegata nel paragrafo 5.1.1, non è utile di per se per la classificazione, visto che ci sono tronchi di spessore estremamente diverso per varie specie, ma è stata inserita nei classificatori lineari perchè si è dimostrata estremamente correlata a molte altre feature, infatti un sapwood piccolo influisce non poco nella rilevazione dei nodi, dell'avvitamento (par. 5.1.8), e sopratutto della definizione degli anelli (par. 5.1.9). Questa feature si è rivelata particolarmente interessante durante lo sviluppo del classificatore SVM.

5.1.5 Numero di nodi per metro

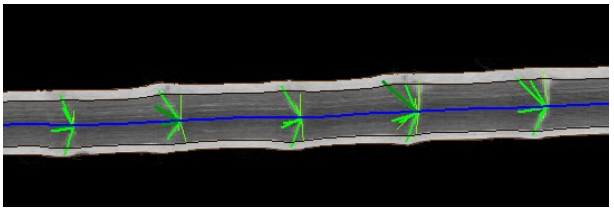
Molto facilmente viene preso il numero di nodi validi e viene diviso per la lunghezza del tronco in esame data dal numero di slice moltiplicata per il reciproco della risoluzione, che nel nostro dataset è di una slice ogni 11mm. Questa feature è la prima che mette in risalto la particolare distribuzione dei nodi nei pini. Infatti i pini, come già spiegato, hanno nodi solo in corone circolari tutti alla stessa altezza. Questa caratteristica è comune anche agli altri alberi, che in più hanno anche altri rami sporadici, questo porta al fatto che i pini hanno mediamente un numero medio di nodi per metro inferiore.

5.1.6 Pendenza media dei nodi

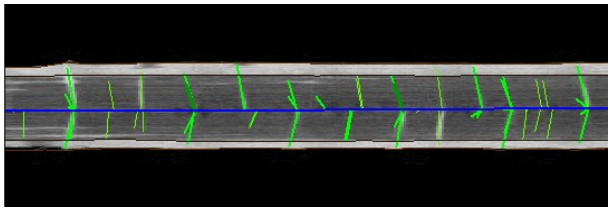
Fra i dati disponibili per i nodi ci sono le coordinate tridimensionali di inizio e fine del pith del nodo. Le coordinate di inizio del nodo corrispondono nelle componenti x e y al pith all'altezza z dove z è la terza coordinata di inizio. Il nodo finisce, invece, in due occasioni: quando arriva al limite del tronco e da vita al ramo, oppure quando muore. Un nodo morto è un ramo che è stato spezzato o tagliato e che quindi da un dato anno non è più cresciuto ed è stato coperto dal resto del tronco. La pendenza del nodo viene calcolata come il rapporto fra quanto è cresciuto in altezza e quanto si è allontanato dal pith, quindi

$$s = \frac{z_e - z_s}{\sqrt{(x_e - x_s)^2 + (y_e - y_s)^2}} \tag{5.1}$$

Da notare che non tutti i nodi salgono verso l'alto anche se la maggioranza di essi ha questo comportamento Della distribuzione delle s teniamo la mediana. Il pino presenta nodi con pendenza molto più accentuata, soprattutto alla base, rispetto alle altre specie, in particolar modo rispetto al fir che invece tende ad avere molti nodi piatti o persino cadenti.



(a) Nodi pendenti



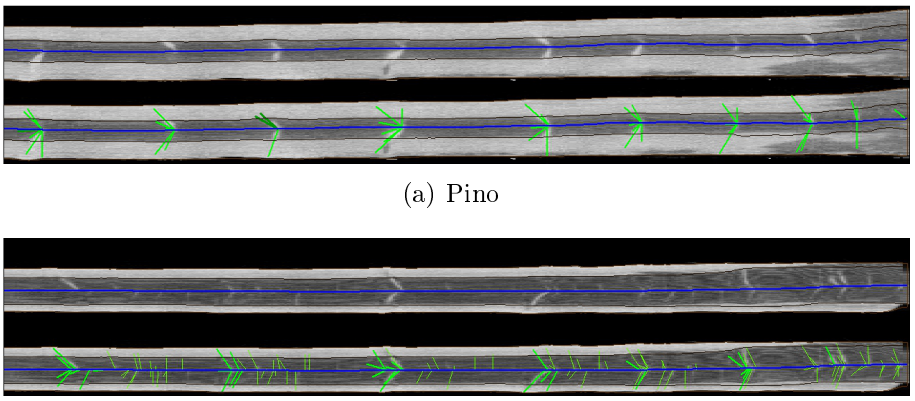
(b) nodi non pendenti

Figura 5.2: Nodi con diversa pendenza

5.1.7 Rilevazione del numero di verticilli

La presenza di nodi esternamente ai verticilli determina inderogabilmente la non appartenenza alla famiglia dei pini. Come si vede in figure 5.3(a) la distribuzione a verticilli dei nodi del pino è molto evidente, mentre in figura 5.3(b) ci sono molti nodi distribuiti a caso senza apparente correlazione fra di essi.

Per cercare di dare un valore a questa diversa distribuzione per poi utilizzarlo nel classificatore, sono stati sviluppati due diversi algoritmi che puntano a evidenziare quanto sia sparsa la collocazione dei nodi usando due approcci diversi, uno di tipo statistico ed uno più somigliante ad un algoritmo di clustering



(a) Pino

(b) Spruce

Figura 5.3: Nodi rilevati su due tronchi di pino e spruce

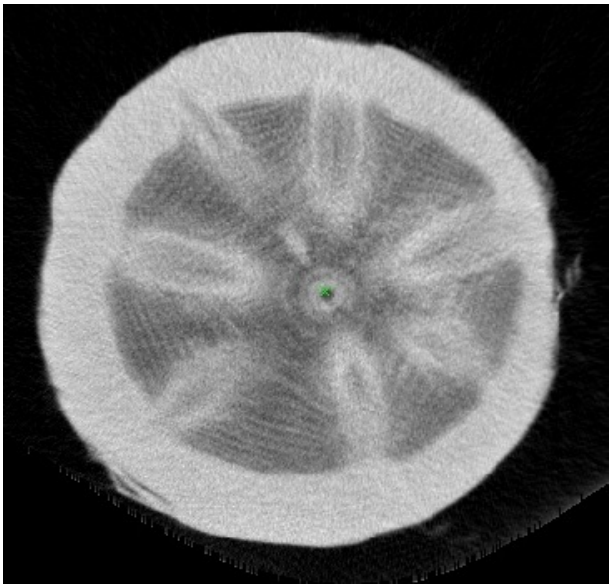


Figura 5.4: Verticillo: si noti come si possono riconoscere almeno 7 nodi che nascono alla stessa altezza

5.1.7.1 Clustering

Questo algoritmo punta a unire in cluster i nodi vicini fra loro e contare quanti nodi in percentuale restano fuori dai cluster. Tutte le operazioni in questo algoritmo sono state fatte tenendo come riferimento l'altezza dell'inizio del nodo z_s. Per determinare quali nodi sono vicini a quali altri si è imposta una soglia di th: due nodi sono considerati vicini se le loro z_s si trovano a meno di th l'una dall'altra. Inoltre consideriamo che non ci possano essere due verticilli a meno di $th1$ l'uno dall'altro.

Algoritmo 3 Clustering per Verticilli

```
1: Clustering(z_s, th, th1)
2: n = z_s.size
3: for r = 0 to n do
4:     adj_(r,r) = 1
5:     adj1_(r,r) = 1
6:     for c = r + 1 to n do
7:         if abs(z_s(r) − z_s,(c)) < th then
8:             adj_(r,c) = adj_(c,r) = 1
9:         end if
10:        if abs(z_s(r) − z_s,(c)) < th1 then
11:            adj1_(r,c) = adj1_(c,r) = 1
12:        end if
13:    end for
14: end for
15:
16: discarded = 0
17: while adj.size> 0 AND adj1.size> 0 do
18:    maxNeigh(adj, indMax, valMax)
19:    if valMax <= 2 then
20:        break
21:    end if
22:    newdiscarded =eraseLineCol(adj, adj1, indMax)
23:    discarded = discarded + newdiscarded
24: end while
25: return (discarded + adj.size)/n
```

Il primo blocco crea le due matrici di adiacenza per le due soglie th e $th1$. Il secondo blocco, tramite la funzione maxNeigh trova l'elemento con maggior numero di vicini, e ne restituisce l'indice in $indMax$ e il numero di vicini in $valMax$. Se il cluster è formato da meno di elementi non è considerato tale e l'algoritmo si ferma. la funzione eraseLineCol(adj, $adj1$, $indMax$) elimina su entrambe le matrici tutte le colonne e le righe relative ai vicini dell'elemento $indMax$ sulla matrice $adj1$ e restituisce il numero di elementi di adj eliminati che non erano segnati come vicini di $indMax$: questi sono i nodi che trovandosi vicini ad un cluster ma non facendone parte non possono far parte neanche di altri cluster e quindi vanno a far parte dei nodi fuori dal cluster. La funzione restituisce la percentuale di nodi fuori dai cluster

sommando quelli scartati a quelli ancora dentro alla matrice quando non ci sono più cluster e dividendo il tutto per il numero di nodi originale.

Tanto più basso sarà il valore, tanto sarà maggiore il raggruppamento in cluster.

5.1.7.2 Studio statistico della distribuzione dei nodi

In questo algoritmo dopo aver creato la distribuzione (5.2) lungo z delle coordinate dei nodi, ponendo un impulso per ogni nodo, si fa la convoluzione (5.3) per una campana con σ studiata approfonditamente. In questo modo si avrà un segnale tanto più alto quanto più nodi saranno vicini fra loro. Per ogni nodo si prende poi l'inverso del valore che la distribuzione assume sulla sua z_s e si fa la media di tutti i valori (5.4).

$$I(z, z_s) = \sum_{i=1}^{N} \delta(z - z_{s,i}) \tag{5.2}$$

$$p(z) = I(z, z_s) * f(z) \tag{5.3}$$

$$r(z_s) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{p(z_{s,i})} \tag{5.4}$$

Dove $\delta(z - z_{s,i})$ è l'impulso centrato in $z_{s,i}$ e $f(z)$ è la gaussiana di media nulla e varianza σ^2 Anche in questo caso tanto più basso sarà r maggiormente i nodi saranno raggruppati.

5.1.8 Spiral Grain

Molti tronchi sono soggetti ad un avvitamento che può variare verso ed intensità negli anni. Nelle zone di tronco dove c'è forte presenza di nodi grossi questo fenomeno si affievolisce. Per rilevare questo genere di comportamento ci si serve di altre imperfezioni del fusto, esclusi chiaramente i nodi, e si cerca di seguire lo spostamento di queste imperfezioni lungo al direzione longitudinale del tronco. Per poter capire bene quanto fosse influenzante per il nostro problema questo genere di comportamento, sono state messe a punto diverse feature, in particolare l'angolo di avvitamento fra una slice e l'altra e quanto affidabile è la misurazione di tale angolo. Durante lo sviluppo ci si è accorti che questo secondo valore poteva assumere anche il significato di quanto forti e riconoscibili erano le imperfezioni che rivelavano la rotazione.
In figura 5.5 sono evidenti le fibre del tronco che si avvitano su di esso.

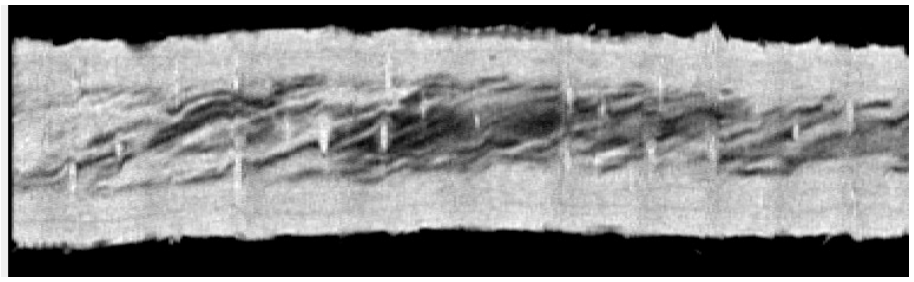
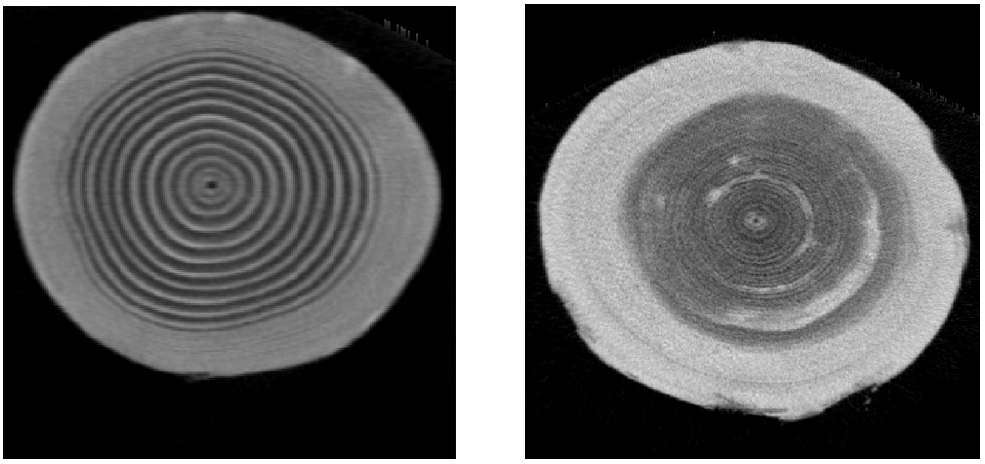


Figura 5.5: sezione longitudinale non assiale di un tronco con spiral grain

5.1.9 Definizione degli anelli

Scorrendo le scansioni del dataset, si può notare come gli anelli di accrescimento degli alberi di douglas siano molto evidenti. Ciò, dovuto principalmente alla veloce crescita che implica anelli ampi, dev'essere quantificato per poterlo usare come feature del classificatore. Sono state pensate diverse soluzioni per trasferire in un singolo valore l'impressione visivamente molto forte.



(a) douglas: anelli definiti (b) spruce: anelli non definiti

Figura 5.6: Differenza di definizione degli anelli

in particolar modo sono state studiate 3 misure che cercano di quantificare questo comportamento.

5.1.9.1 Trasformata di Hough

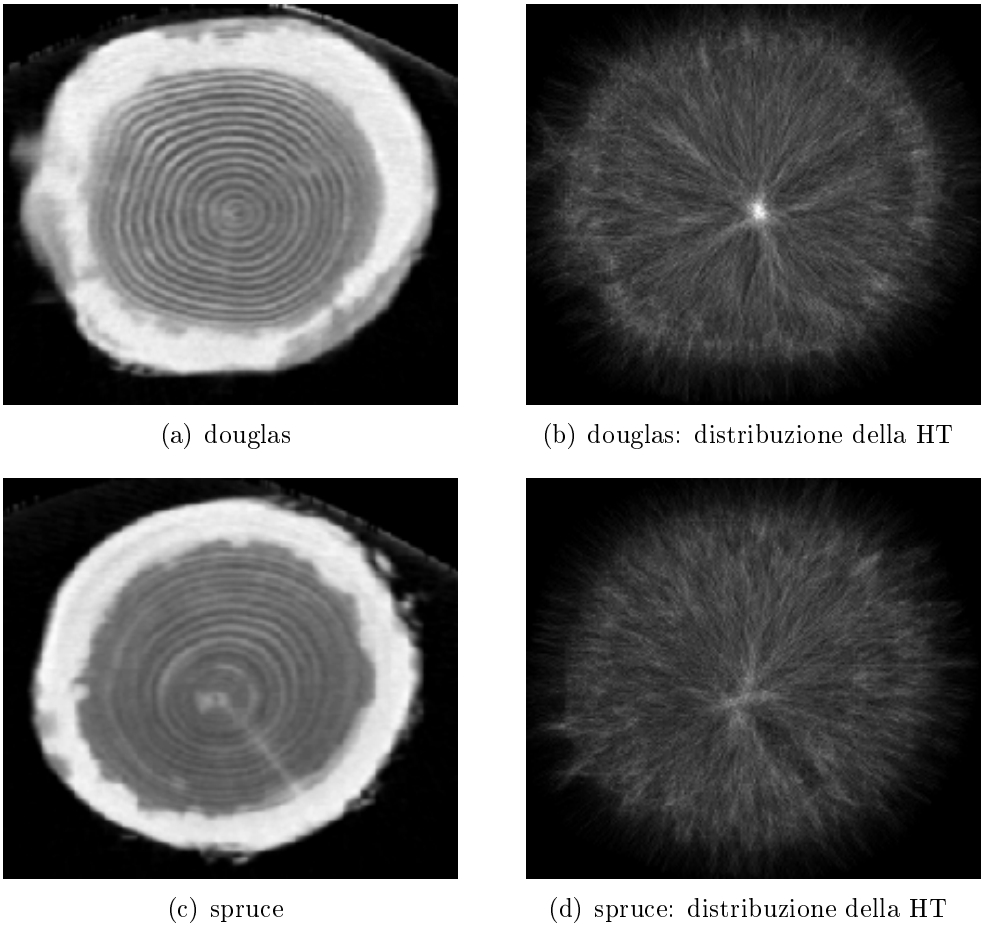
Per la ricerca del pith, particolare fondamentale per poi trovare tutte le altre caratteristiche interne di albero, si utilizza la trasformata di Hough per la ricerca del centro di circonferenze (Andreu et al. [24]). Questa particolare procedura da dei voti per ogni possibile centro, il bucket che riceve più voti è designato come centro delle circonferenze. Visto che le circonferenze ci cui si cerca il centro non sono altro che gli anelli di accrescimento e più gli anelli sono netti e concentrici, più i voti saranno precisi, si può prendere il valore del bucket con valore massimo in relazione alla distribuzione dei voti come misura della bontà della definizione degli

anelli. Questo valore però, in caso di anelli definiti ma ovali o non concentrici tende a diminure, oltre ad essere leggermente influenzato dal numero di anelli rilevabili e quindi dal raggio del sapwood.

5.1.9.2 Trasformata di fourier

Considerando l'immagine polare del tronco, quindi composta da 360 righe, una per grado, e una colonna per ogni possibile raggio, gli anelli di accrescimento, se ben definiti, diventano delle linee verticali alternate di valori alti e bassi. Si nota con facilità che tali alternanze conservano una distanza costante, quindi descrivono delle onde nella direzione orizzontale. Compiendo la trasformata di fourier in orizzontale, dovrebbe spiccare un valore particolarmente alto costante per tutte le righe in corrispondenza di una frequenza relativamente bassa. Anche in questo caso prendendo il valore massimo in relazione al resto della distribuzione, si può ottenere un valore di quanto gli anelli abbiano larghezza costante lungo lo direzione radiale.

Tramite questo calcolo si ottiene anche la terza misura: la larghezza media degli anelli.



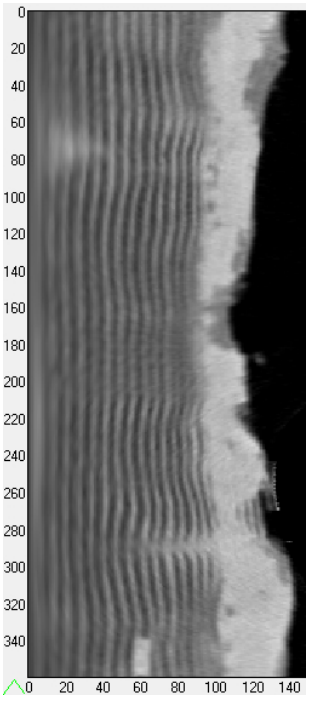
(a) douglas

(b) douglas: distribuzione della HT

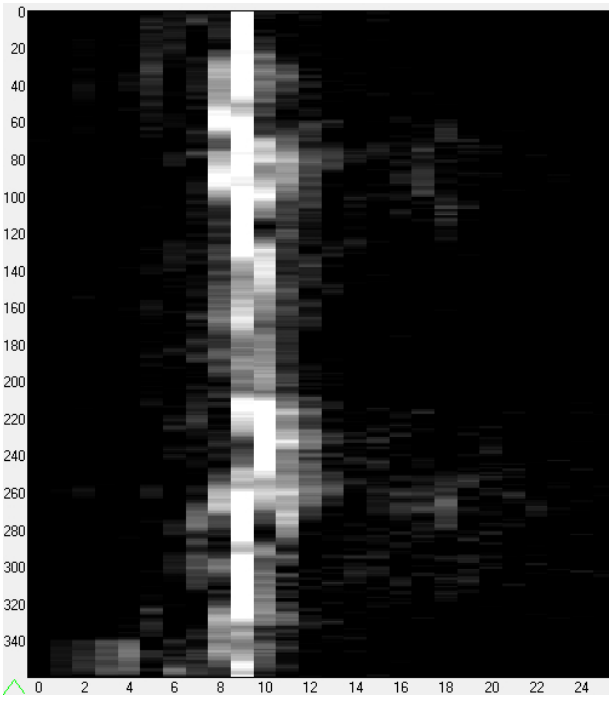
(c) spruce

(d) spruce: distribuzione della HT

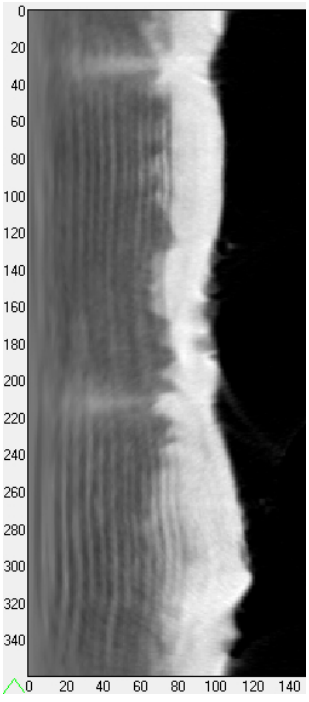
Figura 5.7: Differenza delle distribuzioni delle trasformate di hough per due specie diverse



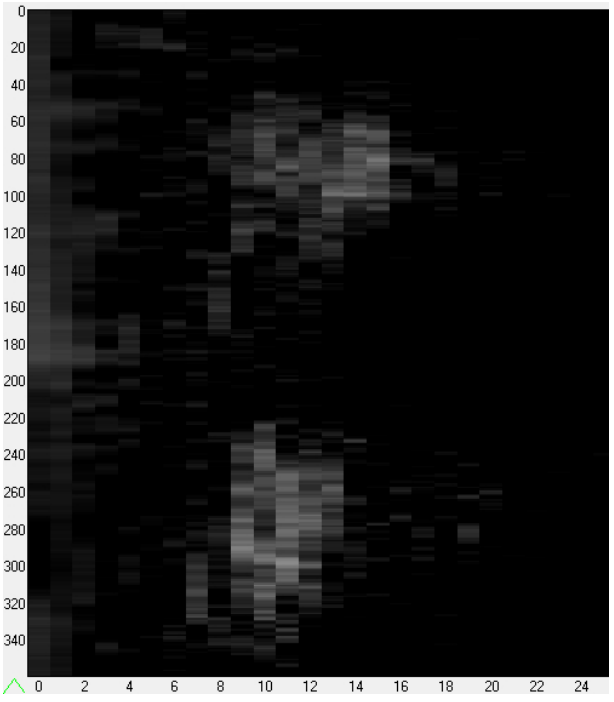
(a) douglas: polare



(b) douglas: distribuzione polare dei coefficienti della FFT



(c) spruce: polare



(d) spruce: distribuzione polare dei coefficienti della FFT

Figura 5.8: Differenza delle distribuzioni dei coefficienti delle trasformate di Fourier per due specie diverse calcolate sulle immagini polari

5.2 Studio e selezione

Come accennato nel paragrafo 4.2 le feature utili alla classificazione sono studiate attraverso la classificazione tramite alberi decisionali coadiuvati dai tool di visualizzazione di weka (figura 4.2 nel paragrafo 4.1). Gli alberi di decisione sono particolarmente adatti scegliere le feature perché specificano esplicitamente, ad ogni nodo, quanti record e di quale classe rispondono positivamente e quali altri negativamente al test. Ciò ha permesso di capire quali feature riuscivano a classificare giusti più record senza bisogno di essere combinate ad altre (primi nodi dell'albero vicino alla radice) e quali altre permettevano di fugare i dubbi su quei record che si trovavano in una zona di confusione. Alle volte, fra le feature di questo secondo gruppo, sono state selezionate alcune feature che erano persino anche troppo specifiche e dividevano pochi record da altri, invertendo persino il significato che veniva dato alla feature. Per capire meglio si pensi alla feature sulla distribuzione dei nodi per la quale i valori dei pini sono mediamente inferiori a quelli dei Douglas. Alle volte, spingendo molto l'algoritmo di Hunt si ottenva un nodo che divideva i record rimanenti con un test su quella feature, del tipo Knot–distribution $\leq val$ mettendo più Douglas che Pine sul nodo relativo ad una risposta positiva al test.

In questo modo sono state trovare le feature distintive per ogni specie e quelle che invece distinguevano una specie dall'altra, per poter implementare le due strategie one–vs–all per SVM (par. 4.3.1) e one–vs–one per ANN (par. 4.4.1)

Di seguito l'elenco completo

5.2.1 Feature distintive per ogni classe

Spruce

- Densità dell'Hearthwood
- Pendenza media dei Nodi
- Spiral Grain

Fir

- Densità dell'Hearthwood
- Pendenza media dei Nodi
- Spiral Grain

Douglas

- Coefficienti massimi della trasformata di Fourier sugli Anelli
- Valori massimi della trasformata di Hough per la ricerca del Pith
- Spiral Grain

Pine

- Distribuzione dei Nodi
- Clustering dei Verticilli

- Numero medio di Nodi per Metro

Come si vede per Douglas e Pine sono distintivi i valori delle feature studiate appositamente per essi sulla base delle particolarità macroscopiche, mentre per Spruce e Fir ci si basa maggiormente sulla distribuzione di feature studiate in generale per tutti e rivelatesi fondamentali per essi. Si noti inoltre che per Spruce e Fir le feature fondamentali sono le stesse, questo implica, evidentemente, che le distribuzioni congiunte di queste siano diverse per le due specie.

5.2.2 Feature distintive per coppie di classi

Spruce vs Fir

- Densità dell'Hearthwood
- Pendenza media dei Nodi
- Spiral Grain

Spruce vs Pine

- Densità dell'Hearthwood
- Distribuzione dei Nodi
- Clustering dei Verticilli

Spruce vs Douglas

- Coefficienti massimi della trasformata di Fourier sugli Anelli
- Valori massimi della trasformata di Hough per la ricerca del Pith
- Spiral Grain
- Pendenza media dei Nodi

Fir vs Pine

- Pendenza media dei Nodi
- Distribuzione dei Nodi
- Clustering dei Verticilli
- Numero medio di Nodi per Metro

Fir vs Douglas

- Coefficienti massimi della trasformata di Fourier sugli Anelli
- Valori massimi della trasformata di Hough per la ricerca del Pith
- Spiral Grain
- Pendenza media dei Nodi

Pine vs Douglas

- Distribuzione dei Nodi
- Clustering dei Verticilli
- Coefficienti massimi della trasformata di Fourier sugli Anelli
- Valori massimi della trasformata di Hough per la ricerca del Pith

Capitolo 6

Risultati

Di seguito verranno riportati i risultati dei classificatori sviluppati. Per ogni famiglia di classificatori verranno riportate le matrici di confusione e le metriche presentate nel paragrafo 4.5 per tutti i classificatori prodotti, specificando il ruolo e il significato dei dati ottenuti. Infine verranno comparati i risultati definitivi ottenuti per ogni famiglia.

6.1 Alberi decisionali

Per la famiglia degli alberi decisionali, sviluppati secondo quanto descritto in 4.2, vengono qui riportati i migliori risultati ottenuti per ogni classificatore One–vs–All, per quelli One–vs–One e per il classificatore unico per le 4 specie. Questo per giustificare le scelte compiute in 5.2.

Spruce vs All

Attuale	Predetta	
	Spruce	Other
Spruce	1085	14
Other	31	1570

Tabella 6.1: confusion matrix: C4.5 – Spruce vs All

$$TPR = 0.987 \qquad P = 0.972$$
$$\overline{TPR} = 0.983 \quad \overline{P} = 0.983 \quad \kappa = 0.966$$

Fir vs All

Attuale	Predetta Fir	Predetta Other
Fir	711	27
Other	28	1934

Tabella 6.2: confusion matrix: C4.5 – Fir vs All

$$TPR = 0.963 \qquad P = 0.962$$
$$\overline{TPR} = 0.980 \quad \overline{P} = 0.980 \quad \kappa = 0.949$$

Pine vs All

Attuale	Classe predetta Pine	Classe predetta Other
Pine	328	5
Other	3	2364

Tabella 6.3: confusion matrix: C4.5 – Pine vs All

$$TPR = 0.985 \qquad P = 0.991$$
$$\overline{TPR} = 0.997 \quad \overline{P} = 0.997 \quad \kappa = 0.986$$

Douglas vs All

Attuale	Classe predetta Douglas	Classe predetta Other
Douglas	494	36
Other	22	2148

Tabella 6.4: confusion matrix: C4.5 – Spruce vs All

$$TPR = 0.932 \qquad P = 0.957$$
$$\overline{TPR} = 0.979 \quad \overline{P} = 0.978 \quad \kappa = 0.931$$

Spruce vs Fir

Attuale	Classe predetta	
	Spuce	Fir
Spruce	1091	8
Fir	17	721

Tabella 6.5: confusion matrix: C4.5 – Spruce vs Fir

$$\overline{TPR} = 0.986 \quad \overline{P} = 0.985 \quad \kappa = 0.972$$

Spruce vs Pine

Attuale	Classe predetta	
	Spuce	Pine
Spruce	1091	8
Pine	8	333

Tabella 6.6: confusion matrix: C4.5 – Spruce vs Pine

$$\overline{TPR} = 0.994 \quad \overline{P} = 0.995 \quad \kappa = 0.984$$

Spruce vs Douglas

Attuale	Classe predetta	
	Spuce	Douglas
Spruce	1092	7
Douglas	7	523

Tabella 6.7: confusion matrix: C4.5 – Spruce vs Douglas

$$\overline{TPR} = 0.991 \quad \overline{P} = 0.991 \quad \kappa = 0.980$$

Fir vs Pine

Attuale	Classe predetta	
	Fir	Pine
Fir	738	0
Pine	2	331

Tabella 6.8: confusion matrix: C4.5 – Fir vs Pine

$$\overline{TPR} = 0.998 \quad \overline{P} = 0.998 \quad \kappa = 0.996$$

Fir vs Douglas

Attuale	Classe predetta	
	Fir	Douglas
Fir	734	4
Douglas	12	518

Tabella 6.9: confusion matrix: C4.5 – Fir vs Douglas

$$\overline{TPR} = 0.987 \quad \overline{P} = 0.987 \quad \kappa = 0.974$$

Pine vs Douglas

Attuale	Classe predetta	
	Pine	Douglas
Pine	333	0
Douglas	0	530

Tabella 6.10: confusion matrix: C4.5 – Pine vs Douglas

$$\overline{TPR} = 1 \quad \overline{P} = 1 \quad \kappa = 1$$

All vs All

Attuale \ Predetta	Spruce	Fir	Pine	Douglas
Spruce	1069	20	3	7
Fir	5	713	11	9
Pine	0	4	329	0
Douglas	11	14	5	500

Tabella 6.11: confusion matrix: C4.5 – Classificatore definitivo

$$TPR_{spruce} = 0.973 \qquad P_{spruce} = 0.985$$
$$TPR_{fir} = 0.966 \qquad P_{fir} = 0.949$$
$$TPR_{pine} = 0.988 \qquad P_{pine} = 0.945$$
$$TPR_{douglas} = 0.943 \qquad P_{douglas} = 0.967$$
$$\overline{TPR} = 0.967 \quad \overline{P} = 0.967 \quad \kappa = 0.953$$

6.2 Support Vector Machine

Con il metodo descritto in 4.3 é stato sviluppato un solo classificatore finale basato, per quanto specificato nel paragrafo 4.3.1, su 4 classificatori One–vs–All i cui risultati sono riportati qui sotto. Da tener conto anche del numero di support vector necessari per una buona classificazione indicato con N_{SV}: un numero troppo alto sarebbe indice di overfitting

Spruce vs All

$$N_{SV} = 96$$

Attuale \ Predetta	Spruce	Other
Spruce	1091	8
Other	15	1586

Tabella 6.12: confusion matrix: SVM – Spruce vs All

$$TPR = 0.993 \qquad P = 0.986$$
$$\overline{TPR} = 0.991 \quad \overline{P} = 0.992 \quad \kappa = 0.982$$

Fir vs All

$$N_{SV} = 108$$

Attuale	Predetta Fir	Predetta Other
Fir	730	8
Other	13	1949

Tabella 6.13: confusion matrix: SVM – Fir vs All

$$TPR = 0.989 \qquad P = 0.983$$
$$\overline{TPR} = 0.992 \quad \overline{P} = 0.992 \quad \kappa = 0.980$$

Pine vs All

$$N_{SV} = 37$$

Attuale	Classe predetta Pine	Classe predetta Other
Pine	333	0
Other	0	2367

Tabella 6.14: confusion matrix: SVM – Pine vs All

$$TPR = 1 \qquad P = 1$$
$$\overline{TPR} = 1 \quad \overline{P} = 1 \quad \kappa = 1$$

Douglas vs All

$$N_{SV} = 90$$

Attuale	Classe predetta Douglas	Classe predetta Other
Douglas	512	18
Other	5	2165

Tabella 6.15: confusion matrix: SVM – Douglas vs All

$$TPR = 0.966 \qquad P = 0.990$$
$$\overline{TPR} = 0.991 \quad \overline{P} = 0.991 \quad \kappa = 0.973$$

Unione dei classificatori

		Predetta			
		Spruce	Fir	Pine	Douglas
Attuale	Spruce	1094	4	0	1
	Fir	5	732	0	1
	Pine	0	0	333	0
	Douglas	3	6	0	521

Tabella 6.16: confusion matrix: SVM – Classificatore definitivo

$$TPR_{spruce} = 0.995 \qquad P_{spruce} = 0.993$$
$$TPR_{fir} = 0.992 \qquad P_{fir} = 0.987$$
$$TPR_{pine} = 1 \qquad P_{pine} = 1$$
$$TPR_{douglas} = 0.983 \qquad P_{douglas} = 0.996$$
$$\overline{TPR} = 0.993 \quad \overline{P} = 0.882 \quad \kappa = 0.990$$

6.3 Artificial Neural Network

Come specificato nel paragrafo 4.4 sono state studiate le reti neurali per tutte le coppie di classi esistenti e poi è stato prodotto un classificatore unico a partire dal training congiunto delle 6 reti studiate.

Spruce vs Fir

Attuale	Classe predetta	
	Spuce	Fir
Spruce	1087	12
Fir	10	728

Tabella 6.17: confusion matrix: ANN – Spruce vs Fir

$$\overline{TPR} = 0.988 \quad \overline{P} = 0.988 \quad \kappa = 0.975$$

Spruce vs Pine

Attuale	Classe predetta	
	Spuce	Pine
Spruce	1099	0
Pine	0	333

Tabella 6.18: confusion matrix: ANN – Spruce vs Pine

$$\overline{TPR} = 1 \quad \overline{P} = 1 \quad \kappa = 1$$

Spruce vs Douglas

Attuale	Classe predetta	
	Spuce	Douglas
Spruce	1094	5
Douglas	10	520

Tabella 6.19: confusion matrix: ANN – Spruce vs Douglas

$$\overline{TPR} = 0.991 \quad \overline{P} = 0.991 \quad \kappa = 0.979$$

Fir vs Pine

Attuale	Classe predetta	
	Fir	Pine
Fir	738	0
Pine	2	331

Tabella 6.20: confusion matrix: ANN – Fir vs Pine

$$\overline{TPR} = 0.998 \quad \overline{P} = 0.998 \quad \kappa = 0.996$$

Fir vs Douglas

Attuale	Classe predetta	
	Fir	Douglas
Fir	728	10
Douglas	11	519

Tabella 6.21: confusion matrix: ANN – Fir vs Douglas

$$\overline{TPR} = 0.983 \quad \overline{P} = 0.983 \quad \kappa = 0.966$$

Pine vs Douglas

Attuale	Classe predetta	
	Pine	Douglas
Pine	333	0
Douglas	0	530

Tabella 6.22: confusion matrix: C4.5 – Pine vs Douglas

$$\overline{TPR} = 1 \quad \overline{P} = 1 \quad \kappa = 1$$

Unione dei classificatori

		Predetta			
		Spruce	Fir	Pine	Douglas
Attuale	Spruce	1086	10	0	3
	Fir	8	719	0	11
	Pine	0	2	331	0
	Douglas	10	6	0	514

Tabella 6.23: confusion matrix: ANN – Classificatore definitivo

$$TPR_{spruce} = 0.988 \qquad P_{spruce} = 0.984$$
$$TPR_{fir} = 0.974 \qquad P_{fir} = 0.976$$
$$TPR_{pine} = 0.994 \qquad P_{pine} = 1$$
$$TPR_{douglas} = 0.970 \qquad P_{douglas} = 0.973$$
$$\overline{TPR} = 0.981 \quad \overline{P} = 0.981 \quad \kappa = 0.974$$

6.4 Confronto

Come si può notare dalle tabelle 6.24, 6.25 e 6.26, il miglior classificatore sembra essere la Support Vector Machine costruita a partire dai classificatore One–Vs–All. Nonostante ciò si è deciso di utilizzare in impianto il classificatore a rete neurale, per diversi motivi

- Nonostante per ANN siano necessari $O(n^2)$ classificatori per n specie diverse, e per SVM ne bastano $O(n)$, per ampliare l'insieme delle specie a $n+1$ bisogna comunque calcolare $O(n)$ classificatori diversi infatti per l'approccio One–vs–One i classificatori già calcolati non vengono modificati., e quindi bisogna elaborare solo gli n classificatori per la nuova specie contro ogni vecchia specie, mentre per l'approccio One–vs–All cambiano i dataset di tutti i classificatori, obbligando a crearne $n + 1$ nuovi;
- La capacità d'interpretazione e la personalizzazione delle reti neurali non ha confronti con le SVM non lineari, che sono sostanzialmente delle black–box;
- Le ANN sono state studiate usando solo le feature necessarie di volta in volta per ogni coppia di classi. Per le SVM tutte e 12 le feature definitive vengono sempre utilizzate;
- I risultati ottenuti con le ANN sono stati accettati dal cliente;
- In alcune SVM il numero di support vector era leggermente eccessivo, con conseguente rischio di overfitting.

	Spruce	Fir	Pine	Douglas
C4.5	0.973	0.966	0.988	0.943
SVM	0.995	0.992	1	0.983
ANN	0.988	0.974	0.994	0.970

Tabella 6.24: True Positive Rate per le singole specie a confronto

	Spruce	Fir	Pine	Douglas
C4.5	0.985	0.949	0.945	0.969
SVM	0.993	0.987	1	0.996
ANN	0.984	0.976	1	0.970

Tabella 6.25: Precision per le singole specie a confronto

Analizzando più in profondità alcuni dati delle sezioni 6.2 e 6.3 si può notare come mediamente le prestazioni migliorino passando dalla classificazione binaria al classificatore finale, questo per quanto spiegato nei paragrafi 4.3.1 e 4.4.1, in quanto non ci si limita a prendere i risultati discreti della classificazione, ma si sfrutta tutta l'informazione a nostra disposizione.

Nel caso dei Douglas con l'SVM, per esempio, nel classificatore One–vs–All (tabella 6.15) ci sono 18 Douglas classificati come di altra specie e 5 alberi delle altre

	$\overline{TPR}$	$\overline{P}$	κ
C4.5	0.967	0.967	0.953
SVM	0.993	0.993	0.990
ANN	0.981	0.991	0.973

Tabella 6.26: Confronto metriche riassuntive

specie etichettati come Douglas che nel classificatore finale (6.16) si riducono a 9 e 2, meno della metà. Questo implica che alcuni di quei douglas che venivano sbagliati, avevano uno score di classificazione comunque più alto per Douglas rispetto a tutte le altre specie, mentre quegli alberi presi erroneamente per Douglas, rispondevano positivamente, con molta più forza, anche ad un'altra classe.

Inoltre, sono stati fatti alcuni test in impianto con alberi la cui specie era stata certificata da esperti in foresta, e i risultati sono stati più che soddisfacenti sia per il cliente che per gli sviluppatori.

Conclusioni

Il classificatore prodotto dopo questo studio è operativo presso i clienti che ne hanno fatto richiesta. Poter lavorare con questo ampio dataset ha permesso di trovare una soluzione al problema che fosse valida e corretta per la grande maggioranza degli alberi analizzati e da analizzare in futuro. Per le eventuali evoluzioni del sistema di classificazione di sono alcune attenzioni da porre:

1. Il dataset è stato addestrato su alberi che provengono da una vasta area ma dal clima pressochè simile, anche se tagliati in diversi periodi dell'anno. Questo classificatore potrebbe non essere valido in altre zone del globo con climi differenti. Restano valide le idee di base e le feature, ma le distribuzioni di valori delle feature potrebbero essere spostate in altri intervalli.

2. Aggiungere una specie comporta dover calcolare nuovi classificatori e probabilmente nuove feature specifiche della specie come nel caso di Pine e Douglas.

3. Per nuove specie, o per un nuovo classificatore per un altro cliente, sono necessari almeno 300 tronchi certificati di quella data specie per comprendere bene il comportamento delle feature per la classe.

Inoltre i risultati del classificatore sono migliorabili con:

1. Studio di nuove feature più specifiche per Spruce e Fir

2. Ampliamento del dataset per le classi sotto rappresentate (Pine)

3. Studio di altri meccanismi di machine learning

4. Collaborazione con il cliente per effettuare test costanti nel tempo per essere sicuri della correttezza

In generale si è visto come i Pine siano facilmente riconoscibili grazie alla disposizione esclusiva dei nodi in verticilli e i Douglas si dividano principalmente dagli altri alberi grazie alla definizione ed ampiezza degli anelli, caratteristica comune anche ad una parte di Fir. Maggiormente problematica è la divisione di Spruce e Fir per i quali, sperabilmente, nel futuro prossimo saranno disponibili nuove feature studiate appositamente.

Bibliografia

[1] Giudiceandrea F., Ursella E., Vicario E., *A high speed CT scanner for the sawmill industry*. Proceedings of the 17th international symposium on nondestructive testing of wood, 1:105–112, 2011.

[2] Charpentier P., Чубинский А. Н., Bombardier V., Longuetaud F., Mothe F.,Тамби А. А., Бахшиева М. А., *Identification of wood species by using computed tomography*. Journal of the Forest academy, Saint–Petersburg, 202:158–167, 2013.

[3] Bouckaert R. R., Frank E. Hall M., Kirkby R., Reutemann P., Seewald A., Scuse D., *WEKA Manual for Version 3–7–10*. University of Waikato, Hamilton, New Zealand 2013.

[4] *WekaWiki*. University of Waikato, Hamilton, New Zealand, http://weka.wikispaces.com gennaio 2014.

[5] Farjon A., *A Handbook of the World's Conifers*. BRILL, 2010.

[6] Buzug T. M., *Computed Tomography*. Springer, 2008.

[7] Tan P. N., Steinbach M., Kumar V., *Introduction to Data Mining*. Pearson, 2006.

[8] Quinlan J. R., *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[9] Quinlan J. R., *Improved use of continuous attributes in c4.5*. Journal of Artificial Intelligence Research, 4:77–90, 1996.

[10] Hunt E. B., Mari, J., Stone P. J., *Experiments in Induction*. New York: Academic Press, 1966.

[11] Pelkmans K., De Brabanter J., Suykens J.A.K., De Moor B., *Handling missing values in support vector machine classifiers*. Neural Networks, 18:684–692, 2005.

[12] Rifkin R., Klautau A., *In Defense of One–Vs–All Classification*. Journal of Machine Learning Research, 5:101–141, 2004.

[13] Lee Y., Lin Y., Wahba G., *Multicategory support vector machines*. Department of Statistics, University of Wisconsin, Technical Report 1043, 2001.

[14] Lee Y., Lin Y., Wahba G., *Multicategory support vector machines*. In Proceedings of the 33rd Symposium on the Interface, 2001.

[15] Crammer K., Singer Y., *On the algorithmic implementation of multiclass kernel–basd vector machines*. Journal of Machine Learning Research, 2:265–292, 2001.

[16] Crammer K., Singer Y., *On the learnability and design of output codes for multiclass problems*. Machine Learning, 47(2):201–233, 2002.

[17] Dietterich T. G. and Bakiri G., *Solving multiclass learning problems via error–correcting output codes*. Journal of Artificial Intelligence Research, 2:263–286, 1995.

[18] J. Fürnkranz, *Solving multiclass learning problems via error–correcting output codes*. Journal of Machine Learning Research, 2:721–747, 2002.

[19] Carletta J., *Assessing Agreement on Classification Tasks: the Kappa Statistic*. Computational Linguistics, 22.2:249–254, 1996.

[20] Liu Y., Zheng Y. F., *One–against–all multi–class SVM classification using reliability measures*. Proc. Int. Joint Conf. Neural Netw., 849–854, 2005.

[21] Magasarian O. L., *Data mining via support vector machines*. Data mining institute report, University of Wisconsin, Madison, 2001.

[22] Kohavi R., *A study of cross–validation and bootstrap for accuracy estimation and model selection*. Proc. 14th Int. Joint Conf. Artificial Intelligence, 338–345, 1995.

[23] Martin J. K., Hirschberg D. S., *Small sample statistics for classification error rates I: Error rate measurements*. Department of Information and Computer Science, UC Irvine, 1996.

[24] Andreu J.P., Rinnhofer A., *Modeling of internal defects in logs for value optimization based on industrial CT scanning*. Fifth International Conference on Image Processing and Scanning of Wood, Bad Waltersdorf, 141–150, 2003.

[25] Johansson E., Johansson D., Skog J., Fredriksson M., *Automated knot detection for high speed computed tomography on Pinus sylvestris L. and Picea abies (L.) Karst. using ellipse fitting in concentric surfaces*. Computers and Electronics in Agriculture, 96:238–2458, 2013.

[26] Katsevich A., *An inversion algorithm for Spiral CT*. Proceedings of the 2001 International Conference on Sampling Theory and Applications, 261–265, 2001.

[27] Katsevich A., *Analysis of an exact inversion algorithm for spiral cone–beam CT*. Phys. Med. Biol., 47:2583–2598, 2002.

[28] Katsevich A., *Theoretically exact filtered backprojection–type inversion algorithm for Spiral CT*. SIAM J. Appl. Math., 62:2012–2026, 2002.

[29] Katsevich A., *Improved exact FBP algorithm for spiral CT*. Adv. Appl. Math, 32:681–697, 2004.

[30] Christopher J. E., *Picea Abies*. The Gymnosperm Database, http://www.conifers.org/pi/Picea_abies.php, novembre 2013.

[31] Christopher J. E., *Abies Alba*. The Gymnosperm Database, http://www.conifers.org/pi/Abies_alba.php, novembre 2012.

[32] Christopher J. E., *Pseudotsuga Menziesii*. The Gymnosperm Database, http://www.conifers.org/pi/Pseudotsuga_menziesii.php, novembre 2012.

[33] Christopher J. E., *Pinus Nigra*. The Gymnosperm Database, http://www.conifers.org/pi/Pinus_nigra.php, febbraio 2013.

[34] Christopher J. E., *Pinus Sylvestris*. The Gymnosperm Database, http://www.conifers.org/pi/Pinus_sylvestris.php, novembre 2012.