Università degli Studi di Milano Bicocca

Scuola di Scienze

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di laurea in Informatica

ENSEMBLE LEARNING: BAYESIAN MODEL AVERAGING

Relatore: *Prof.ssa Vincenzina Messina*

Co-relatore: *Dott.ssa Elisabetta Fersini*

Relazione della prova finale di:

Fabio Cimmino

Matricola 807070

Anno Accademico 2017-2018

Indice

Capitolo 1

Introduzione

In questa tesi viene proposta una tecnica basata sul Bayesian Model Averaging, cioè un metodo di Ensemble Learning che intende superare le limitazioni dei tradizionali algoritmi di apprendimento d'insieme. L'Ensemble Learning utilizza una serie di metodi d'insieme che usano modelli multipli generati da più "learner" per ottenere una miglior prestazione predittiva rispetto ai modelli da cui è costituito. Inoltre è stata anche implementata un'euristica di selezione dei modelli da includere nell'insieme ottimale. Nella teoria del Machine Learning, un "learner" (algoritmo di apprendimento) cerca di identificare un concetto non noto della realtà di interesse. In particolare un algoritmo di apprendimento è impiegato principalmente per la risoluzione dei problemi di classificazione, cioè quando è necessario decidere a quale categoria appartiene un determinato dato. Per compiere tale operazione l'algoritmo di apprendimento si basa sull'analisi di esempi concreti, selezionati casualmente dalla popolazione, e cerca di individuare una funzione che lega una serie di attributi a una o più variabili risposta (o classi). La difficoltà nel selezionare ed implementare il giusto algoritmo di apprendimento sta nel fatto che non si conosce a priori la forma della distribuzione che ha generato il campione di esempi estratto. Per questo motivo vengono sempre più utilizzate tecniche di Ensemble Learning. Come per l'essere umano, tendiamo a cercare diverse opinioni prima di prendere qualsiasi decisione importante. Per questo pesiamo le opinioni individuali e le combiniamo per raggiungere la nostra decisione finale. Ci sono diversi motivi teorici e pratici per cui potremmo preferire un sistema d'insieme. Dal punto di vista statistico, le buone prestazioni sui dati di addestramento non implicano delle buone performance di generalizzazione sui dati, definite come le prestazioni dell'algoritmo sui dati non visti durante l'apprendimento. Una serie di algoritmi con prestazioni di addestramento simili possono avere prestazioni di generalizzazione diverse. In questi casi, combinando gli output di diversi classificatori con una media si può ridurre

il rischio di selezionare un algoritmo con risultati insoddisfacenti. L'apprendimento d'insieme consente anche di analizzare grosse quantità di dati che un singolo algoritmo potrebbe non riuscire a gestire. Ad esempio, l'ispezione delle condotte di trasporto del gas mediante tecniche di dispersione del flusso magnetico genera 10 GB di dati ogni 100 km di conduttura lunga migliaia di chilometri. Addestrare un classificatore con una quantità così grande di dati non è di solito pratico; il training set viene quindi partizionato, ed ogni partizione viene assegnata ad un algoritmo di apprendimento combinando successivamente i risultati.
I sistemi d'insieme possono anche essere utilizzati per risolvere esattamente il problema opposto: avere dati non sufficienti. In assenza di training data adeguati, le tecniche di ricampionamento possono essere utilizzate per creare sottoinsiemi casuali sovrapposti dei dati disponibili, ognuno dei quali può essere utilizzato per addestrare un classificatore diverso. Indipendentemente dalla quantità di dati disponibili, alcuni problemi sono troppo difficili da risolvere per un determinato classificatore. Più in particolare, il limite decisionale che separa i dati dalle diverse classi può essere troppo complesso o non rientrare nello spazio delle funzioni che possono essere implementate dal modello di classificazione scelto. Una combinazione di algoritmi di apprendimento differenti può superare il problema appena introdotto. L'apprendimento d'insieme può essere anche utilizzato nel caso in cui disponiamo di una serie di dati ottenuta da varie fonti in cui la natura delle caratteristiche è diversa (caratteristiche eterogenee), e quindi non è possibile utilizzare un classificatore singolo per apprendere le informazioni contenute in tutti i dati.
Tuttavia, nonostante le tecniche di Ensemble Learning siano ampiamente utilizzate, gli algoritmi tradizionali presenti allo stato dell'arte mostrano delle limitazioni che questa tesi intende superare proponendo una strategia d'insieme basata sul Bayesian Model Averaging. Quest'ultima tecnica consente di non avere modelli indipendenti ed ugualmente affidabili. Verrà anche proposta un'euristica di selezione dei modelli da includere nell'insieme ottimale con l'obiettivo di ottenere una strategia di selezione sia efficiente che efficace.

Di seguito viene riportata una rassegna dei capitoli della tesi ed il loro contenuto:
Capitolo 2: viene introdotto problema della classificazione in generale. Vengono discusse le metriche di prestazione classiche delle tecniche di classificazione. Si discute successivamente della cross-validation, un buon metodo per ottenere misure di prestazione effettive;
Capitolo 3: in questo capitolo viene illustrato il funzionamento delle tecniche d'insieme presenti allo stato dell'arte (Bagging, Boosting, Stacking e Random Forest) descrivendo poi le loro limitazioni;

Capitolo 4: viene spiegato il funziomaneto del Bayesian Model Averaging e la sua implementazione. Successivamente si parla dell'euristica di selezione dell'insieme di modelli ottimale;
Capitolo 5: in questo capitolo vengono mostrati i dataset utilizzati per gli esperimenti, i tipi di insiemi di modelli scelti ed i risultati ottenuti. Questi risultati consistono nel confrontare le accuratezze del Bayesian Model Averaging con quelle delle tecniche tradizionali di Ensemble Learning.
Capitolo 6: si conclude la tesi con i possibili sviluppi futuri di questo lavoro su cui si dovrà concentrare l'attività di ricerca.

Capitolo 2

Classificazione

2.1 Definizione del problema di classificazione

Di seguito vengono introdotte alcune definizioni con lo scopo di formalizzare dal punto di vista matematico il problema di classificazione. É comodo rappresentare i dati con il modello relazionale la cui definizione formale è basata sulla teoria degli insiemi.
Un *training set* T è una relazione composta *n tuple*, chiamate anche *instance*, ognuna delle quali è costituita da un elenco ordinato m *attributi* ed 1 classe. Ogni attributo A_j prende valori sul dominio D_j con $1 \leq j \leq m$. L'insieme di tutti gli attributi (chiamati anche *feature*) viene indicato dall'insieme $A = \{A_1, A_2, \ldots, A_m\}$. La classe C prende valori su $\Gamma = \{\gamma_0, \gamma_1, \ldots, \gamma_{c-1}\}$ che rappresenta l'insieme delle c classi disponibili, con numero c finito. Comunemente ci si riferisce alla classe C anche con il termine *category* oppure *target attribute*.
Sempre in riferimento al modello relazionale, $T \subseteq D_1 \times D_2 \times \cdots \times D_m \times \Gamma$ sullo schema di attributi $A \cup C$ dove C è il *target attribute*. Più in generale invece un *dataset* D può essere costituito da un record la cui classe non è nota, quindi $D \subseteq D_1 \times D_2 \times \cdots \times D_m$ sullo schema di attributi A. La relazione può essere rappresentata graficamente in forma di tabella dato che la rappresentazione tabellare di una relazione è molto intuitiva e facilmente comprensibile.

La tabella per il training set T appena introdotto è costituita da n righe e $m + 1$ colonne ed è raffigurata nella seguente tabella:

A_1	A_2	...	A_m	C
$t_1[A_1]$	$t_1[A_2]$	...	$t_1[A_m]$	$t_1[C]$
$\vdots$				$\vdots$
$t_n[A_1]$	$t_n[A_2]$	...	$t_n[A_m]$	$t_n[C]$

Ora è possibile definire il problema di classificazione in modo formale.

Problema (Classificazione)

INPUT: *Training set T, dove* $T = \{t_1, t_2, \ldots, t_n\}$ *è l'insieme di n record.*

OUTPUT: *Target function o Classification Model M che mappi ogni tupla del prodotto cartesiano* $D_1 \times D_2 \times \cdots \times D_m$ *in* Γ.

$$M : D_1 \times D_2 \times \cdots \times D_m \longrightarrow \Gamma$$
$$t \longmapsto M(t)$$

L'approccio sistematico per risolvere il problema di classificazione appena introdotto è definito *classification technique* o *classifier*. Ogni tecnica utilizza un algoritmo di apprendimento (*learning algorithm*) per identificare il modello che meglio approssima la relazione tra gli attributi in input e la classe. In una *classification technique* è prevista anche una fase di *validazione* che mira a valutare la qualità del modello trovato. Per poterla eseguire è necessario un dataset con la stessa struttura del *training set* utilizzato nella fase di apprendimento. Tale insieme viene comunemente chiamato *test set*. Dei record appartenenti al *test set* $\widetilde{T}$ deve essere nota la classe di appartenenza per poterla confrontare con quella predetta dal modello. Nella Figura 2.1 è riassunto l'approccio generale alla classificazione.

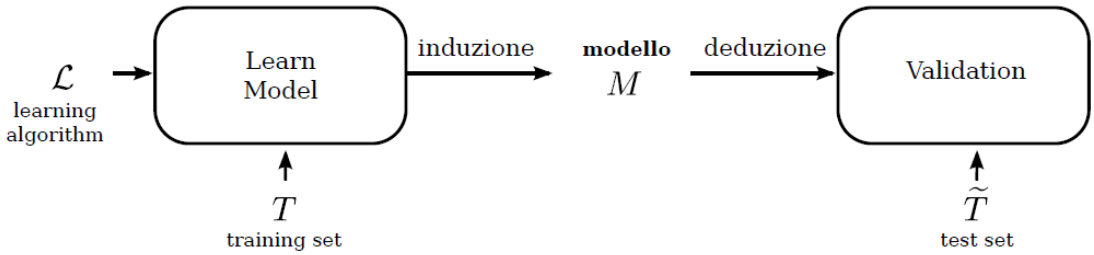

Figura 2.1: Approccio generale alla classificazione

2.2 Confusion matrix e metriche di prestazione

Una *confusion matrix* è una tabella che viene spesso utilizzata per descrivere le prestazioni di un modello di classificazione tramite metriche come *precision, recall, accuracy* e *F-measure* . Ogni riga della matrice rappresenta le istanze in una classe predetta mentre ciascuna colonna rappresenta le istanze di una classe effettiva (o viceversa). È un tipo speciale di tabella di contingenza, con due dimensioni ("predette" e "previste") e insiemi identici di classi in entrambe le dimensioni (ogni combinazione di dimensione e classe è una variabile nella tabella di contingenza). Nella Figura 2.2 di seguito viene proposto un esempio di confusion matrix.

		Actual class		
		Cat	**Dog**	**Rabbit**
Predicted class	**Cat**	**5**	2	0
	Dog	3	**3**	2
	Rabbit	0	1	**11**

Figura 2.2: Esempio di confusion matrix

Per ogni elemento dell'insieme delle classi $\Gamma = \{\gamma_0, \gamma_1, \ldots, \gamma_{c-1}\}$ è possibile calcolare, a partire dalla confusion matrix, i seguenti valori:

- $True\ Positive_{\gamma_i}$ (TP_{γ_i}) - numero dei record di classe prevista γ_i classificati correttamente nella classe γ_i.
- $False\ Negative_{\gamma_i}$ (FN_{γ_i}) - numero dei record di classe prevista diversa da γ_i classificati erroneamente.
- $False\ Positive_{\gamma_i}$ (FP_{γ_i}) - numero dei record classificati erroneamente nella classe γ_i.
- $True\ Negative_{\gamma_i}$ (TN_{γ_i}) - numero dei record con classe prevista diversa da γ_i classificati correttamente.

Assumendo di avere una confusion matrix come quella sottostante, queste quantità possono essere determinate nel seguente modo:

	Actual class		
Predicted class	c_{ii}	$\cdots$	c_{in}
	$\vdots$	$\ddots$	
	c_{n1}		c_{nn}

$$True\ Positive_{\gamma_i} = c_{ii} \tag{2.1}$$

$$False\ Negative_{\gamma_i} = \sum_{l}^{L} c_{li} - TP_i \tag{2.2}$$

$$False\ Positive_{\gamma_i} = \sum_{l}^{L} c_{il} - TP_i \tag{2.3}$$

$$True\ Negative_{\gamma_i} = \sum_{l}^{L}\sum_{k}^{L} c_{lk} - TP_i - FN_i - FP_i \tag{2.4}$$

La *precision* definisce la frazione di record realmente appartenenti alla classe γ_i tra quelli con classe predetta γ_i.

$$precision_{\gamma_i} = \frac{TP_{\gamma_i}}{FP_{\gamma_i} + TP_{\gamma_i}} \tag{2.5}$$

La *recall* indica la frazione di record realmente appartenenti alla classe γ_i tra quelli con classe prevista γ_i.

$$recall_{\gamma_i} = \frac{TP_{\gamma_i}}{TP_{\gamma_i} + FN_{\gamma_i}} \tag{2.6}$$

Più è alta la *precision* meno sono i falsi positivi commessi dal modello, un alto *recall* indica che il numero di falsi negativi è basso.
La *recall* misura la capacità di individuare tutti i record di una determinata classe mentre la precision misura la capacità di individuare soltanto quelli appartenenti ad una determinata classe.
Una misura che racchiude entrambe è la F_1-*measure*:

$$F_1 - measure = \frac{2 \cdot precision \cdot recall}{precision + recall} = \frac{2 \cdot TP_{\gamma_i}}{2 \cdot TP_{\gamma_i} + FP_{\gamma_i} + FN_{\gamma_i}} \tag{2.7}$$

L'*accuracy* indica la percentuale di classificazioni corrette:

$$accuracy_{\gamma_i} = \frac{TP_{\gamma_i} + TN_{\gamma_i}}{TP_{\gamma_i} + FP_{\gamma_i} + TN_{\gamma_i} + FN_{\gamma_i}} \tag{2.8}$$

2.3 Cross-Validation

La *cross-validation* viene utilizzata per valutare la prestazione predittiva dei modelli e per giudicare come si comportano al di fuori del campione in un nuovo dataset noto anche come *test set*. Quando identifichiamo il modello che meglio approssima la relazione tra gli attributi in input e la classe, lo stiamo identificando sul *training set*. Senza la cross-validation abbiamo solo informazioni su come funziona il nostro modello per i nostri dati *in-sample*. Idealmente vorremmo vedere come funziona il modello quando abbiamo un nuovi dati in termini di accuratezza delle sue previsioni.
In [12] è stato effettuato un ampio studio per mettere a confronto metriche di prestazione ottenute con metodi di stima diversi: i risultati suggeriscono che il miglior metodo di stima è una *10-fold cross-validation*.

Dato un *classification model* M e un insieme la cui classe dei record è nota (E) è bene quindi calcolare le metriche di prestazione con la *cross-validation*. Il primo passo è quello di partizionare l'insieme E in k sottoinsiemi:

$$E = E_1 \cup E_2 \cup \cdots \cup E_k$$

tali che

$$E_i \cap E_j = \emptyset \quad \forall i \neq j \;\; 1 \leq i, j \leq k$$

con taglia

$$|E_i| = \frac{1}{k}|E| \;\; \forall i : \; 1 \leq i \leq k$$

Viene quindi costruito un modello M_i sul training set $E \setminus E_i$ per poi utilizzare E_i come test set. Questo procedimento viene ripetuto k volte. Le accuratezze ottenute dai k modelli vengono quindi mediate per ottenere una misura di accuratezza complessiva.
Quindi l'accuratezza stimata nella *cross-validation* è:

$$\frac{1}{k}\sum_{i=1}^{k} \frac{|\{t \in E_i : \; M_i(t) = t[C]\}|}{|E_i|} = \frac{\sum_{i=1}^{k} |\{t \in E_i : \; M_i(t) = t[C]\}|}{|E|} \tag{2.9}$$

$$t[C] \equiv \textit{classe vera del record } t$$

$$M_i(t) \equiv \textit{classe prevista dal modello } M_i \textit{ per } t$$

Capitolo 3

Stato dell'arte

3.1 Metodi di apprendimento

L'apprendimento automatico è strettamente legato al riconoscimento di pattern e alla teoria computazionale dell'apprendimento [10] ed esplora lo studio e la costruzione di algoritmi che possano apprendere da un insieme di dati e fare delle predizioni su questi, costruendo in modo induttivo un modello basato su dei campioni. Insieme all'apprendimento automatico viene introdotto anche il concetto di analisi predittiva come insieme di tecniche derivanti principalmente da esso e dal *data mining* che insieme consentono, partendo da un insieme di dati esistenti, di estrapolare schemi per effettuare predizioni su comportamenti futuri nell'ambito da cui i dati provengono. I problemi di predizione affrontabili con l'apprendimento automatico si possono suddividere in tre categorie principali:

- non supervisionata
- semi-supervisionata
- supervisionata

3.1.1 Apprendimento non supervisionato

Nell'apprendimento senza supervisione non viene data nessuna etichetta agli input dell'algoritmo ed il suo obiettivo è trovare la loro struttura. L'apprendimento non supervisionato ha lo scopo di scoprire nuovi schemi nascosti nei dati oppure può essere utilizzato per il *feature learning* [2]. Un esempio tipico di questi algoritmi lo si ha nei motori di ricerca. Questi programmi, data una o più parole chiave, sono in grado di creare una lista di link relative alle pagine che l'algoritmo ritiene pertinenti con la ricerca. Molti degli

algoritmi di apprendimento non supervisionato sono composti dalle due seguenti fasi: creazione di un lessico del sentiment in modo non supervisionato e valutazione del grado di positività/negatività di una parte testuale usando una funzione basata sugli indicatori di positività e negatività.

3.1.2 Apprendimento semi-supervisionato

Con apprendimento semi-supervisionato [27] ci si riferisce all'insieme di tecniche di apprendimento supervisionato che utilizzano anche dati non etichettati per l'addestramento del classificatore; in genere viene utilizzata una piccola quantità di dati etichettati insieme ad una grande quantità di dati non etichettati. Molti ricercatori di machine learning hanno scoperto che i dati senza una classe usati insieme ad una piccolo numero di dati etichettati possono produrre un notevole miglioramento dell'apprendimento.
La maggior parte degli studi [16, 19] affrontano il problema della classificazione espandendo un insieme iniziale di parole attraverso sinonimi e contrari recuperati dai thesauri.

3.1.3 Apprendimento supervisionato

Per svolgere una classificazione delle istanze di un dataset attraverso dei metodi supervisionati, all'algoritmo di classificazione vengono forniti degli esempi nella forma di possibili input e rispettivi output desiderati e l'obiettivo è quello di estrarre una regola generale che associ l'input all'output corretto.
Nonostante in letteratura vi siano molti rilevanti metodi non supervisionati e semi-supervisionati, la maggior parte degli approcci per la classificazione sfruttano l'apprendimento supervisionato, visto il suo potere predittivo. La caratteristica comune di questi approcci è l'identificazione del modello che classifichi le istanze del dataset con la più alta accuratezza possibile. Purtroppo però, nessun algoritmo di classificazione supervisionata opera costantemente in modo migliore rispetto agli altri. Per superare questa limitazione vengono utilizzate le tecniche di Ensemble Learning che, come indicato in [20], ci permettono di soppesare diversi classificatori individuali e combinarli per ottenerne uno finale migliore rispetto ai singoli classificatori.
Questo viene fatto perchè l'aggregazione di informazioni da più fonti porta a decisioni che sono spesso superiori rispetto a quelle che avrebbe potuto effettuare un singolo classificatore. Facendo un paragone con per l'essere umano, cerchiamo diverse opinioni prima di prendere una decisione importante pesando le opinioni individuali e combinandole per raggiungere la decisione finale. Affinché si arrivi ad una classificazione d'insieme migliore rispetto a quella

dei singoli individui bisogna stabilire uno o più meccanismi per trasformare i giudizi di ogni membro in un una decisione collettiva.

Struttura dell'apprendimendo supervisionato

Lo scopo dell'apprendimento supervisionato è di classificare le istanze in un insieme di categorie che vengono anche chiamate classi o etichette. Comunemente, la classificazione è basata su modelli di classificazione che si ottengono da un insieme di modelli preclassificati. In alternativa, la classificazione utilizza la conoscenza fornita da un esperto nel dominio dell'applicazione.
In una tipica impostazione di apprendimento supervisionato, vengono fornite una serie di istanze, chiamate training set. Le etichette delle istanze nel training set sono conosciute e l'obiettivo è costruire un modello per etichettare nuove istanze.
Un algoritmo che costruisce il modello è chiamato induttore mentre un'istanza dell'induttore per un training set specifico è chiamata classificatore.

3.2 Algoritmi di Ensemble Learning

L'apprendimento d'insieme nell'apprendimento automatico consiste in una serie di metodi d'insieme che usano modelli multipli generati da classificatori diversi per ottenere una miglior prestazione predittiva rispetto ai singoli modelli (Figura 3.1). Lo stato dell'arte comprende le seguenti tecniche tradizionali [24]: Majority Voting, Bagging, Boosting, Stacking e Random Forest. Majority Voting è la tecnica d'insieme più usata; è caratterizzata da un insieme di classificatori che classificano in modo indipendente le istanze di un dataset, dopodiché viene assegnata la classe finale di ogni istanza selezionando quella predetta il maggior numero di volte.

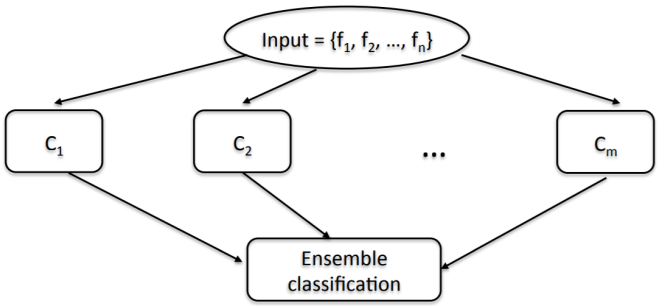

Figura 3.1: Tecniche di ensemble learning

Un algoritmo per l'apprendimento d'insieme, come indicato nella Sezione 3.1.3, è costituito dai seguenti elementi:

1. Training set: un set di dati etichettato utilizzato per addestrare l'insieme. Molto spesso le istanze sono descritte come vettori valore-attributo.

2. Induttore di base: l'induttore è un algoritmo di induzione che riceve un training set e forma un classificatore che rappresenta la relazione generalizzata tra gli attributi di input e l'attributo target.

3. Generatore di diversità: questo componente è responsabile della generazione dei diversi classificatori.

4. Combiner: ha il compito di combinare le classificazioni provenienti da diversi classificatori.

Diversità Il successo di un sistema d'insieme - ovvero la sua capacità di correggere gli errori di alcuni dei suoi membri - si basa direttamente sulla diversità dei classificatori che compongono l'ensemble. Dopotutto, se tutti i classificatori fornissero lo stesso risultato, correggere un possibile errore non sarebbe possibile. Pertanto, i classificatori individuali devono fare errori diversi su istanze diverse. Nello specifico, un sistema d'insieme ha bisogno di classificatori i cui limiti decisionali siano adeguatamente diversi da quelli degli altri. La diversità del classificatore può essere raggiunta in diversi modi. Preferibilmente, gli output del classificatore dovrebbero essere indipendenti dalla classe, o meglio ancora negativamente correlati. Il metodo più diffuso consiste nell'utilizzare differenti training set per addestrare i classificatori individuali. Tali training set sono spesso ottenuti attraverso tecniche di ricampionamento, come il Bagging, in cui i sottoinsiemi di training set sono creati casualmente, di solito con la sostituzione prendendo le istanze dall'intero dataset.

3.2.1 Apprendimento dipendente e indipendente

Un algoritmo di Ensemble Learning può essere caratterizzato da una struttura dipendente o indipendente per costruire insiemi.
In una struttura dipendente l'output di un classificatore viene utilizzato per costruire il prossimo classificatore. Quindi è possibile sfruttare le conoscenze generate nelle iterazioni precedenti per migliorare l'apprendimento nelle prossime iterazioni. In alternativa, ogni classificatore è costruito indipendentemente e i loro output sono combinati con qualche tecnica.
Per quanto riguarda l'apprendimento dipendente possono essere utilizzati i due seguenti approcci:

- Incremental Batch Learning: in questo metodo la classificazione prodotta in un'unica iterazione viene fornita come conoscenza iniziale dell'algoritmo di apprendimento nella seguente iterazione. L'algoritmo di apprendimento utilizza il training set corrente insieme alla classificazione precedente per costruire il prossimo classificatore.

- Model-guided Instance Selection: In questo approccio dipendente, i classificatori che sono stati costruiti in iterazioni precedenti vengono utilizzati per manipolare il training set per la seguente iterazione. Il metodo più conosciuto per la selezione del modello guidato da istanze è il Boosting.

Nella Figura 3.2 possiamo vedere che nell'approccio indipendente il training set originale viene trasformato in diversi training set sui quali vengono addestrati diversi classificatori. I training data creati dal training set originale

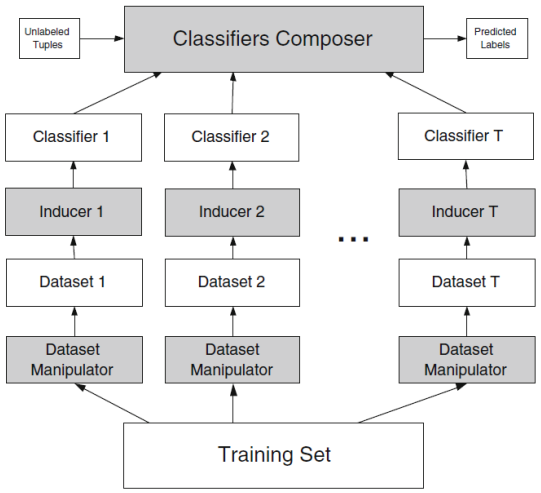

Figura 3.2: Approccio indipendente

possono essere disgiunti (mutuamente esclusivi) o sovrapposti. Viene quindi applicato un metodo di combinazione per produrre la classificazione finale.

3.2.2 Bagging

Il Bagging, che è l'acronimo di Bootstrap Aggregating, è uno dei primi e più intuitivi algoritmi di Ensemble Learning. La diversità dei classificatori nel Bagging si ottiene usando repliche bootstrap dei training set. Il bootstrap (o bootstrapping) è un metodo statistico utilizzato per stimare la distribuzione di campionamento di uno stimatore, sostituendo dei dati nel campione originale, spesso con lo scopo di ottenere stime robuste degli errori standard e degli intervalli di confidenza dei parametri di una popolazione.
Diversi sottoinsiemi di training set sono creati in modo casuale dall'intero training set. Ogni sottoinsieme del training set viene utilizzato per addestrare un classificatore diverso. Infine le singole decisioni dei classificatori vengono unite con un voto di maggioranza. Per ogni istanza data, la classe scelta dal maggior numero di classificatori sarà la decisione finale dell'insieme.
Il processo dell'algoritmo di Bagging è mostrato in figura 3.3 mentre la struttura base di questa tecnica è mostrata nell'Algoritmo 3.1. Il Bagging è particolarmente allettante quando i dati disponibili sono di dimensioni limitate. Per garantire che vi siano campioni di addestramento sufficienti in ciascun sottogruppo, porzioni relativamente grandi dei campioni (dal 75% al 100%) vengono create in ciascun sottoinsieme.

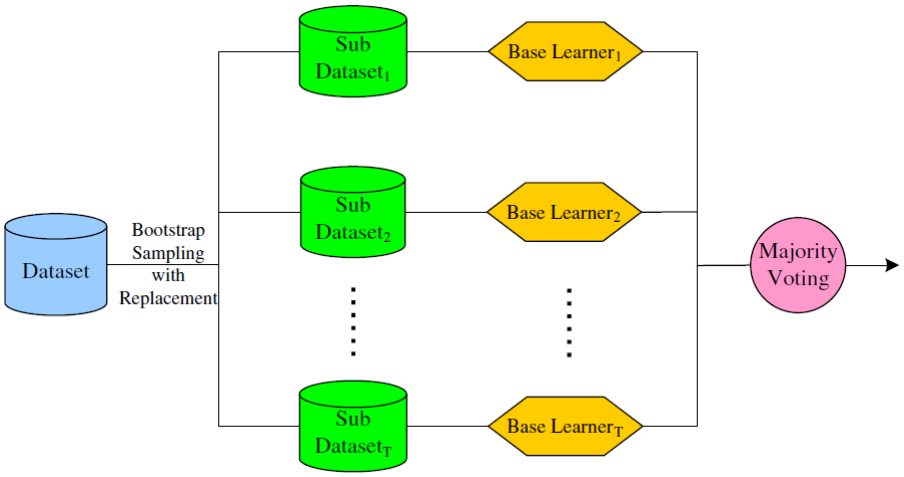

Figura 3.3: Il processo di Bagging

Algoritmo 3.1 Algoritmo di Bagging

Input: I (un induttore), T (il numero di iterazioni), S (il training set), μ (la dimensione del sottocampione)

Output: $M_t; t = 1, \ldots, T$

1: $t \leftarrow 1$
2: **repeat**
3: $S_t \leftarrow$ Campiona μ istanze da S con sostituzione
4: Costruisci un *classification model* M_t usando I su S_t
5: $t++$
6: **until** $t > T$

Il Bagging, come il Boosting, è una tecnica che migliora l'accuratezza di un classificatore generando un modello composito che combina più classificatori derivati dallo stesso induttore. Entrambi i metodi seguono un approccio di voto, che viene implementato in modo diverso, al fine di combinare gli output dei diversi classificatori. Nel Boosting, al contrario del Bagging, ogni classificatore è influenzato dalle prestazioni dei precedenti classificatori. Quindi il nuovo classificatore presta più attenzione agli errori di classificazione commessi dai classificatori precedentemente costruiti. Nel Bagging, ogni istanza viene scelta con uguale probabilità, mentre nel Boosting le istanze vengono scelte con una probabilità proporzionale al loro peso.

3.2.3 Boosting

Per quanto riguarda il Boosting, è un metodo di Ensemble Learning che crea un classificatore fortemente predittivo da un certo numero di classificatori deboli. Ciò avviene creando un modello dal training set, quindi creando un secondo modello che tenta di correggere gli errori del primo modello. Vengono realizzati modelli fino a quando il training set non viene predetto alla perfezione o viene raggiunto un numero massimo di modelli. Perciò l'idea alla base del Boosting è di applicare ripetutamente un learner di base alle versioni modificate del training set, producendo in tal modo una sequenza di learner di base per un numero predefinito di iterazioni.
Per cominciare, tutte le istanze sono inizializzate con pesi uniformi. Dopo questa inizializzazione, ciascuna iterazione mira al potenziamento della predizione adattando un learner di base ai dati di training pesati. Dopodiché viene calcolato l'errore commesso dal modello; il peso delle istanze classificate correttamente viene ridotto mentre quello delle istanze classificate in modo errato viene aumentato. Di conseguenza, il classificatore debole è costretto a concentrarsi sulle istanze difficili da etichettare del training set eseguendo iterazioni aggiuntive. Il modello finale ottenuto dall'algoritmo Boosting è una combinazione lineare di diversi learner di base pesati in base alle proprie prestazioni.

Anche se ci sono diverse versioni degli algoritmi di Boosting, il più usato è quello proposto da Freund e Schapire [17], che è noto come AdaBoost. Il processo è mostrato nella Figura 3.4 e lo pseudo-codice è riportato nell'Algoritmo 3.2. L'idea principale alla base di AdaBoost è concentrarsi maggiormente su schemi che sono più difficili da classificare.
Viene assegnato un peso a ogni classificatore individuale che misura la precisione complessiva del classificatore. Quindi, pesi più alti sono dati a classificatori più accurati. Questi pesi sono usati per la classificazione delle performance di nuovi modelli.

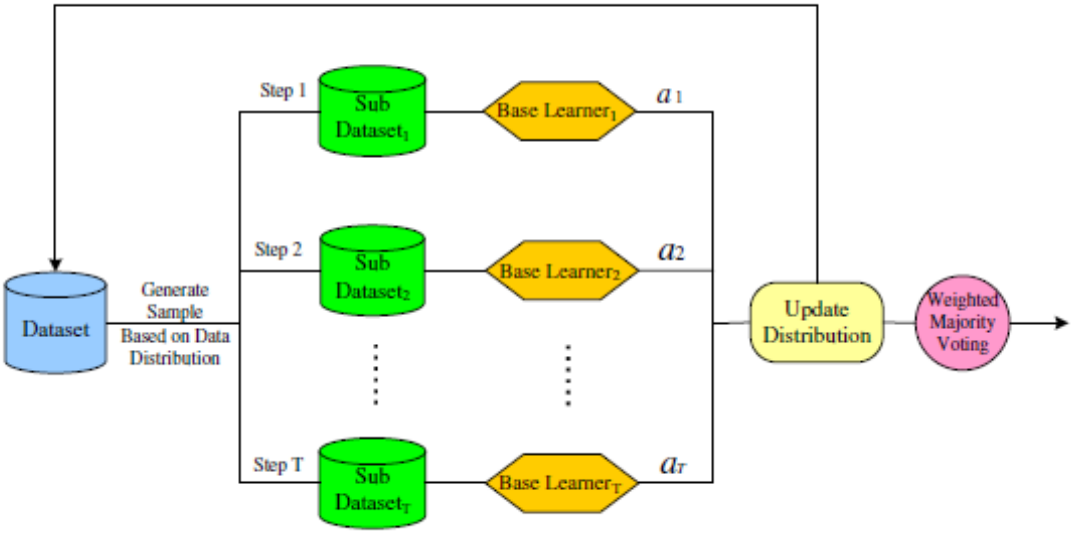

Figura 3.4: Processo di AdaBoost

Algoritmo 3.2 Algoritmo di AdaBoost

Input: I (un induttore di base), T (il numero di iterazioni), S (il training set)

Output: $M_t; \alpha_t; t = 1, \ldots, T$

1: $t \leftarrow 1$
2: $D_1(i) \leftarrow 1/m; i = 1, \ldots, m$
3: **repeat**
4: Costruisci un *classification model* M_t usando I e distribuzione D_t
5: $\epsilon_t \leftarrow \sum_{i:M_t(x_i) \neq y_i} D_t(i)$
6: $t++$
7: **if** $\epsilon_t > 0.5$ **then**
8: $T \leftarrow t - 1$
9: exit Loop
10: **end if**
11: $\alpha_t \leftarrow \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$
12: $D_{t+1}(i) = D_t(i) \cdot e^{-\alpha_t y_t M_t(x_i)}$
13: Normalizza D_{t+1} ad una corretta distribuzione
14: $t++$
15: **until** $t > T$

3.2.4 Random Forest

Una variante dell'algoritmo Bagging è la Random Forest [3], così chiamata perché è costruita da alberi decisionali. Nel machine learning un albero di decisione è un modello predittivo, dove ogni nodo interno rappresenta un attributo ed un arco verso un nodo figlio rappresenta un possibile valore per quella proprietà. Una foglia indica la classe predetta per la variabile obiettivo a partire dai valori delle altre proprietà, che nell'albero è rappresentato dal cammino (path) dal nodo radice (root) al nodo foglia.
Una Random Forest può essere creata da singoli alberi decisionali, i cui parametri di training variano in modo casuale. Tali parametri possono essere repliche bootstrap dei training set, come nel Bagging, ma possono anche essere sottosistemi diversi di attributi come nei metodi di Random Subspace [9] (anche chiamato Attribute Bagging [4] o feature bagging; in questi metodi si tenta di ridurre la correlazione tra gli stimatori in un insieme, addestrandoli su campioni casuali di features anziché sull'intero insieme di features).
I singoli alberi sono costruiti con l'Algoritmo 3.3. Il parametro di input N rappresenta il numero di attributi di input che verranno utilizzati per determinare la decisione su un nodo dell'albero. Questo numero dovrebbe essere molto inferiore al numero di attributi presenti nel training set.
Proprio come il Bagging, le Random Forest campionano anche gli attributi (con sostituzione) per ogni albero. Gli alberi vengono creati con la massima profondità (nessuna potatura) e ogni albero esegue una classificazione indipendente. Quindi ogni albero assegna ad ogni istanza una classe e la foresta sceglie la classe che ha più voti su tutti gli alberi.

Algoritmo 3.3 Algoritmo per la generazione degli alberi

Input: IDT (un induttore dell'albero), T (il numero di iterazioni), S (il training set), μ (la dimensione del sottocampione), N (il numero di attributi usati in ogni nodo)

Output: $M_t; t = 1, \ldots, T$

1: $t \leftarrow 1$
2: **repeat**
3: $\quad S_t \leftarrow$ Campiona μ istanze da S con sostituzione
4: $\quad$ Costruisci un *classification model* M_t usando $IDT(N)$ su S_t
5: $\quad t++$
6: **until** $t > T$

Il processo generale di creazione ed uso di una Random Forest viene descritto di seguito:

Step-1: Applicare K iterazioni di Bagging per creare un totale di K alberi.
Bagging: Dato un training set D di n istanze, il Bagging genera m nuovi training set D_i, ognuno di dimensione n' (circa 2/3 del training set iniziale) effettuando un campionamento da D in modo uniforme e con sostituzione. Campionando con sostituzione, alcune istanze possono essere ripetute in ogni training set.

Step-2: Creare gli alberi di decisione usando i K training set campionati, ma solo usando un sottoinsieme di variabili (colonne) casuali ad ogni nodo (cioè applicando Attribute Bagging).
Attribute Bagging: Ogni istanza $X_i(i = 1, \ldots, n)$ nel training set campionato $X = (X_1, X_2, \ldots, X_n)$ è un vettore p-dimensionale $X_i = (X_{i1}, X_{i2}, \ldots, X_{ip})$ composto da p features. Nell'Attribute Bagging vengono selezionate casualmente $r < p$ features dal training set p-dimensionale X. Si ottiene così un *Random Subspace* r-dimensionale dallo spazio p-dimensionale delle features. Quindi il traning set modificato $X^b = (X_1^b, X_2, \ldots, X_n^b)$ è composto da delle istanze r-dimensionali $X_i^b = (X_{i1}^b, X_{i2}^b, \ldots, X_{ir}^b)$ $(i = 1, 2, \ldots, n)$ dove r componenti $x_{ij}^b(j = 1, 2, \ldots, r)$ sono selezionate in modo casuale dalle p componenti $x_{ij}(j = 1, 2, \ldots, p)$ dell'istanza X_i.

Step-3: Ogni albero di decisione predice la classe delle istanze. Per ogni istanza, verrà assegnata la classe predetta più volte.

3.2.5 Stacking

Per quanto riguarda lo Stacking [26], un insieme di classificatori viene prima addestrato utilizzando campioni bootstrap del training set in modo da creare classificatori di primo e secondo livello. I risultati dei classificatori del primo livello vengono utilizzati per addestrare il classificatore di secondo livello. L'idea alla base dello Stacking è capire se i training data sono stati appresi correttamente. Ad esempio, se un particolare classificatore apprende erroneamente una determinata area dello spazio delle features e quindi classifica in modo scorretto le istanze provenienti da quella regione, allora il classificatore di livello 2 potrebbe essere in grado di apprendere questo comportamento e, insieme ai comportamenti appresi da altri classificatori, può correggere un addestramento inappropriato. Tipicamente è utilizzata la Cross Validation per allenare i classificatori di primo livello: l'intero training set è diviso in T blocchi e ciascun classificatore del primo livello viene addestrato sui $T - 1$

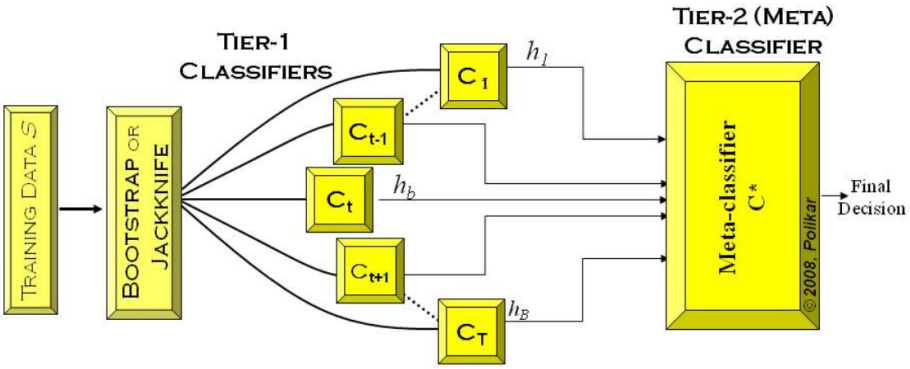

Figura 3.5: Procedura dell'algoritmo di Stacking

blocchi del training set. L'apprendimento di ogni classificatore viene poi valutato sul blocco T-esimo (testing), cioè quello escluso durante la fase di apprendimento del modello. Gli output di questi classificatori insieme alle etichette assegnate correttamente formano il training set per i classificatori di secondo livello (Figura 3.5).
La procedura seguita dall'algoritmo di Stacking è la seguente:

1. Dividere il training set in due gruppi disgiunti.
2. Allenare diversi classificatori sulla prima parte.
3. Fare il testing dei classificatori di base sul secondo gruppo dei training data.
4. Utilizzando le previsioni ottenute al punto 3 come input e le risposte corrette (quelle attese) come output, allenare un classificatore di livello superiore.

3.3 Limitazioni degli algoritmi esistenti

Anche se gli algoritmi di apprendimento automatico proposti sono ampiamente utilizzati nell'apprendimento supervisionato, hanno delle limitazioni che questa tesi intende superare:

Modelli indipendenti e ugualmente affidabili I classificatori tradizionali utilizzati dai metodi di Ensemble Learning sono considerati indipendenti e con la stessa affidabilità. Un classico esempio lo si ha quando un gruppo di cattivi classificatori commette errori altamente correlati ed un buon classificatore che predice nel modo corretto. Dato che si considerano tutti i classificatori ugualmente affidabili, si potrebbe arrivare ad una decisione finale collettiva sbagliata. Nella Sezione 4.4 verrà proposto un metodo che permette di identificare i classificatori con un alto contributo di corrette classificazioni.

Ricerca del modello più accurato ha un costo Nel caso in cui il modello ottimale può essere composto da un numero elevato di classificatori, la complessità computazionale aumenta esponenzialmente. L'euristica di selezione dei modelli da includere nell'insieme ottimale tramite la valutazione del contributo di ogni singolo classificatore ha come obiettivo realizzare una metodologia efficiente ed efficace.

Il singolo classificatore è peggiore di un insieme di classificatori Nel campo della classificazione non vi è una metodologia migliore delle altre: un singolo classificatore può ottenere risultati migliori rispetto ad altri in relazione ad uno specifico dominio applicativo, mentre un ulteriore approccio potrebbe consentire ad un altro classificatore di avere performance migliori. Introducendo una nuova metodologia di ensemble learning in grado si sfruttare le potenzialità di diversi classificatori possiamo non avere più l'incertezza su quale modello rappresenti la scelta ottimale.

Capitolo 4

Bayesian Ensemble Learning

4.1 Teorema di Bayes

L'inferenza bayesiana è un approccio all'inferenza statistica in cui le probabilità non sono interpretate come frequenze, proporzioni o concetti analoghi, ma piuttosto come livelli di fiducia nel verificarsi di un dato evento. Il nome deriva dal teorema di Bayes, che costituisce il fondamento di questo approccio.
Il teorema di Bayes fornisce un metodo per modificare il livello di fiducia in una data ipotesi, alla luce di nuova informazione. Denotando con H_0 l'ipotesi nulla, e con E il dato empirico osservato, il teorema di Bayes può essere enunciato come:

$$P(H_0 \mid E) = \frac{P(E \mid H_0) \cdot P(H_0)}{P(E)}$$

Dove:

- $\mid$ significa "evento condizionale" (quindi $(A \mid B)$ significa A dato B)
- H_0 sta per qualsiasi ipotesi la cui probabilità può essere influenzata dai dati
- $P(H_0)$ è detta probabilità a priori di H_0 ed è la stima della probabilità dell'ipotesi H_0 prima che i dati E siano osservati
- E corrisponde a nuovi dati che non sono stati utilizzati nel calcolo della probabilità a priori
- $P(H_0 \mid E)$, la probabilità a posteriori, è la probabilità di H_0 dato E dopo l'osservazione di E

- $P(E \mid H_0)$ è la probabilità di osservare E dato H_0 ed è chiamata funzione di verosimiglianza (likelihood)
- $P(E)$ è detta probabilità marginale, la probabilità di osservare E senza alcuna informazione pregressa; è una costante di normalizzazione. Questo fattore è lo stesso per tutte le possibili ipotesi prese in considerazione, quindi non influenza le probabilità relative di diverse ipotesi.

4.2 Voting Democratico

L'idea alla base di un meccanismo di voting, che permette di unire più decisioni individuali in modo da prendere una decisione finale, è quella di sfruttare le caratteristiche di diversi classificatori indipendenti combinandoli per ottenere prestazioni migliori rispetto al miglior classificatore singolo.
La tecnica di voting più popolare è il Majority Voting (MV), chiamato anche voting democratico, ed è caratterizzato da un insieme di "esperti" che assegnano una classe ad un insieme di istanze. I voti dei classificatori hanno tutti lo stesso peso e viene determinata la classe finale selezionando l'etichetta più votata per quella istanza.
Sia C un insieme di n classificatori indipendenti e $l_i(s)$ l'etichetta assegnata ad una tupla del training set s dal classificatore $i \in C$. Quindi la classe ottimale $l^{MV}(s)$ è assegnata come segue:

$$\begin{cases} \text{positiva se } \sum_{i \in C} l_i(s)_+ > \sum_{i \in C} l_i(s)_- \\ \text{negativa se } \sum_{i \in C} l_i(s)_+ < \sum_{i \in C} l_i(s)_- \\ \hat{l}(\text{s}) \text{ altrimenti} \end{cases} \tag{4.1}$$

dove $l_i(s)_+ = 1$ se l'etichetta assegnata da i è positiva (0 altrimenti), $l_i(s)_- = 1$ se l'etichetta assegnata da i è negativa e $\hat{l}(s)$ se è l'etichetta assegnata a s dal classificatore più esperto (accurato).

4.3 Bayesian Model Averaging

La limitazione più importante dei metodi ensemble esistenti è che i modelli da includere nella composizione hanno pesi uguali indipendentemente dalla loro affidabilità (Sezione 3.3). Tuttavia, l'incertezza dei modelli può essere filtrata considerando un meccanismo di voting bayesiano. In particolare, tutti i modelli presi in considerazione possono essere sfruttati considerando le

loro capacità di previsione marginale e le loro affidabilità. Questo è possibile utilizzando il contributo pesato di ciascun classificatore per effettuare la previsione dell'etichetta finale.
Data un'istanza s composta da features, un training data D e un insieme di classificatori indipendenti C, la probabilità della classe $l^{BMA}(s)$ è calcolata dal Bayesian Model Averaging (BMA) come segue:

$$P(l(s) \mid C, D) = \sum_{i \in C} P(l(s) \mid i, D) P(i \mid D) \tag{4.2}$$

dove $P(l(s) \mid i, D)$ è la distribuzione marginale dell'etichetta predetta dal classificatore i e $P(i \mid D)$ è la probabilità a posteriori del modello i.
$P(i \mid D)$ può essere calcolata come segue:

$$P(i \mid D) = \frac{P(D \mid i) P(i)}{\sum_{j \in C} P(D \mid j) P(j)} \tag{4.3}$$

dove $P(i)$ è la probabilità a priori di i e $P(D \mid \cdot)$ è la verosimiglianza del training data D.
Nella equazione 4.3 $P(i)$ e $\sum_{j \in C} P(D \mid j) P(j)$ si presume che siano costanti e quindi possono essere omessi. Quindi il BMA assegna l'etichetta ottimale $l^*(s)$ con la seguente regola di decisione:

$$\begin{aligned} l^*(s) = \arg\max P(l(s) \mid C, D) &= \sum_{i \in C} P(l(s) \mid i, D) P(i \mid D) \\ &= \sum_{i \in C} P(l(s) \mid i, D) P(D \mid i) P(i) \\ &= \sum_{i \in C} P(l(s) \mid i, D) P(D \mid i) \end{aligned} \tag{4.4}$$

La misura implicita $P(D \mid i)$ può essere sostituita da una stima esplicita chiamata $F_1 - measure$, ottenuta da una valutazione preliminare del classificatore i. In particolare, eseguendo una cross validation ogni classificatore può produrre una misura che indica quanto bene un classificatore generalizza a dati inosservati. Considerando una *k-fold cross validation* per un classificatore i, la misura $P(D \mid i)$ può essere approssimata nel seguente modo:

$$P(D \mid i) \approx \frac{1}{\iota} \sum_{\iota=1}^{k} \frac{2 \times P_{i\iota}(D) \times R_{i\iota}(D)}{P_{i\iota}(D) + R_{i\iota}(D)}$$

dove $P_{i\iota}(D)$ e $R_{i\iota}(D)$ denotano le misure conosciute come *precision e recall* del classificatore i nel fold ι.
In questo modo possiamo tenere conto del voto di ciascun classificatore sfruttando la previsione ottenuta dalla distribuzione marginale e regoliamo questa affermazione probabilistica in base alla capacità del classificatore di adattarsi ai training data ($F_1 - measure$). Questo approccio consente di prendere in considerazione l'incertezza di ogni classificatore, evitando le predizioni troppo sicure.

4.4 Selezione del modello ottimale

Un aspetto molto importante e discusso riguarda la selezione della serie di modelli da includere nell'insieme ottimale dato che questa operazione è molto dispendiosa dal punto di vista delle risorse e del tempo.
La selezione dell'insieme ottimale è un problema di ottimizzazione combinatoria su $\sum_{p=1}^{N} \frac{N!}{p!(N-p)!}$ possibili soluzioni dove N è il numero di possibili classificatori e p rappresenta la dimensione di ogni potenziale insieme.
Anche se in letteratura vengono proposti diversi metodi([15]) per valutare il contributo di un classificatore rispetto all'insieme, questi non sono adatti per un insieme in cui viene applicato il Bayesian Model Averaging dato che ad ogni classificatore viene assegnato lo stesso peso.
Perciò utilizzo un'euristica che permette di calcolare il contributo di ciascun classificatore fornito rispetto ad un determinato insieme. Consideriamo di avere due classificatori, i e j. Per valutare il contributo fornito dal classificatore i rispetto a j bisogna introdurre i seguenti casi:

- j non etichetta in modo corretto l'istanza del training set s mentre i la etichetta correttamente. Questo rappresenta il più importante contributo di i nel meccanismo di votazione e rappresenta quanto i è in grado di correggere j.
  Rappresentiamo questo caso come $P(i = 1 \mid j = 0)$.

- Sia i che j etichettano correttamente l'istanza del training set. In questo caso il classificatore i conferma la predizione di j.
  Rappresentiamo questo caso come $P(i = 1 \mid j = 1)$.

Il classificatore j può influenzare la predizione nei seguenti casi:

- j etichetta correttamente l'istanza del training set s ma i la etichetta incorrettamente. Questo caso rappresenta il contributo peggiore che i può dare in quanto modifica negativamente l'etichetta (corretta) fornita da j. Rappresentiamo questo caso come $P(i = 0 \mid j = 1)$.

- Sia i che j etichettano in modo errato l'istanza del training set. In questo caso c'è una doppia classificazione errata dato che i convalida l'ipotesi sbagliata di j.
  Rappresentiamo questo caso come $P(i = 0 \mid j = 0)$.

Il contributo r_i^s di ogni classificatore i appartenente ad un determinato insieme $S \subseteq C$ può essere calcolato come:

$$r_i^s = \frac{\sum_{j \in \{S \setminus i\}} \sum_{q \in \{0,1\}} P(i = 1 \mid j = q) P(j = q)}{\sum_{j \in \{S \setminus i\}} \sum_{q \in \{0,1\}} P(i = 0 \mid j = q) P(j = q)}$$

dove $P(j = q)$ è la probabilità a priori del classificatore j di predire in modo corretto o errato le etichette. In particolare $P(j = 1)$ è la percentuale di istanze classificate correttamente e come misura esplicita utilizziamo la *precision* del classificatore, mentre $P(j = 0)$ rappresenta la percentuale di istanze etichettate incorrettamente (come misura utilizziamo l'*error rate*).

Dopo aver scelto questa euristica per calcolare il contributo di ogni classificatore rispetto un insieme in modo da non avere dei pesi uniformemente distribuiti, un ulteriore problema è quello di decidere una strategia di selezione della composizione di classificatori ottimale. In letteratura vengono proposti i seguenti due approcci: *forward selection* ([14]) e *backward elimination* ([5]).
Nella eliminazione *backward* l'insieme S contiene inizialmente tutti i classificatori dell'insieme completo C e iterativamente viene rimosso un classificatore $i \in C$ con il contributo minore; questa è la strategia che viene adottata. Il vantaggio dell'eliminazione *backward* consiste nell'identificare facilmente i modelli irrilevanti consentendo di ridurre le possibili combinazioni da $\sum_{p=1}^{N} \frac{N!}{p!(N-p)!}$ a $N-1$ potenziali candidati per determinare l'insieme ottimale. Ad ogni passo il classificatore con il contributo minore viene ignorato fino a quando non viene raggiunta la combinazione più piccola. Questo processo di ricerca termina il contributo del classificatore medio (ACC) di un sottoinsieme è minore rispetto all'insieme principale. Più formalmente, un insieme S viene indicato come composizione ottimale se viene soddisfatta la seguente condizione:

$$\frac{ACC(S)}{|S|} \geq \frac{ACC(S \setminus x)}{|S - 1|}$$

dove $ACC(S)$ è stimata come la media dei contributi r_i^s di ogni classificatore appartenenti all'insieme S. Il contributo di ogni classificatore è calcolato tenendo conto dell'insieme S, che è iterativamente aggiornato una volta

che il peggior classificatore viene rimosso. Ciò porta alla definizione di S caratterizzata da una dimensione decrescente $|S| = N, N-1, \ldots, 1$.

4.5 Implementazione

4.5.1 Input del Bayesian Model Averaging

Il BMA può prendere in input un qualsiasi numero di file *.csv* che contengono le classificazioni effettuate dai vari modelli. Un esempio di file di input è riportato in Figura 4.1.

```
inst#,actual,predicted,error,distribution,
1,2:0,2:0,,0,*1
2,2:0,1:1,+,*0.999,0.001
3,2:0,2:0,,0.001,*0.999
4,2:0,2:0,,0.002,*0.998
5,2:0,2:0,,0,*1
6,2:0,1:1,+,*0.809,0.191
7,2:0,2:0,,0.286,*0.714
8,2:0,2:0,,0.223,*0.777
9,2:0,2:0,,0.001,*0.999
10,2:0,2:0,,0.016,*0.984
11,2:0,2:0,,0.002,*0.998
12,2:0,2:0,,0.433,*0.567
13,2:0,2:0,,0.024,*0.976
14,2:0,2:0,,0.174,*0.826
15,2:0,1:1,+,*0.65,0.35
16,2:0,2:0,,0.004,*0.996
17,2:0,2:0,,0.013,*0.987
18,2:0,2:0,,0.102,*0.898
```

Figura 4.1: Esempio file .csv di input

Ogni riga rappresenta un'istanza del dataset in cui è possibile ricavare le seguenti informazioni:

- numero dell'istanza
- classe attesa
- classe predetta
- presenza o assenza di un errore di classificazione
- distribuzione di probabilità delle relative classi

Durante l'operazione di classificazione delle tuple del dataset viene sempre eseguita una *10-fold cross validation*.
Questi file di input possono essere ottenuti tramite *Weka* [25] che contiene una raccolta di strumenti di visualizzazione e algoritmi per l'analisi dei dati e la modellazione predittiva, insieme a interfacce utente grafiche per un facile accesso a queste funzioni.
Il formato utilizzato in Weka per la lettura dei dataset è l'ARFF (Attribute Relationship File Format), è simile al più famoso CSV (Comma-separated values) ed è equivalente alla tabella di un database relazionale.
Di seguito viene riportato un esempio della struttura di un dataset in formato ARFF:

@relation golfWeather	Qui viene stabilito il nome del dataset
@attribute outlook {*sunny, overcast, rainy*} *@attribute windy* {*TRUE, FALSE*}	Qui definiamo due attributi nominali, outlook e windy. Il primo ha tre valori: soleggiato, nuvoloso e piovoso; il secondo ne ha due: VERO e FALSO
@attribute temperature real *@attribute humidity real*	Queste linee definiscono due attributi numerici
@attribute play {*yes, no*}	L'ultimo attributo è la variabile di destinazione o classe predefinita utilizzata per la previsione. Nel nostro caso è un attributo nominale con due valori, rendendo questo un problema di classificazione binaria.
@data *sunny,FALSE,85,85,no* *sunny,TRUE,80,90,no* *overcast,FALSE,83,86,yes* *rainy,FALSE,70,96,yes* *rainy,FALSE,68,80,yes*	Il resto del dataset è costituito dal token @data, seguito da valori separati da virgola per gli attributi: una riga per istanza. Nel nostro caso ci sono cinque istanze.

4.5.2 Logica di funzionamento

Nell'Algoritmo 4.1 di seguito viene presentato lo schema generale di funzionamento del BMA:

Algoritmo 4.1 Struttura del Bayesian Model Averaging

Input: Classification Models $M = \{m_1, m_2, \ldots, m_i\}$ con $i > 1$, possibili combinazioni $\mathcal{P}(M)$ dei Classification Models, insieme $\Gamma = \{\gamma_0, \gamma_1, \ldots, \gamma_n\}$ delle classi

Output: Accuratezza del voting bayesiano e democratico delle combinazioni

```
1:  for each Classification Model m ∈ M do
2:    for i = 1 to 10 do
3:      for each istanze ∈ fold i do
4:        CM_{classe predetta,classe attesa} ← CM_{classe predetta,classe attesa} + 1
5:      end for
6:      F-MEASURE_{m,i} ← Calcola f-measure del classification model m al
        fold i
7:    end for
8:  end for
9:  CALCOLA F-MEASURE(M) {per stimare la P(D|i) dell'Equazione 4.4}

10: for each X ∈ P(M) tale che |X| > 1 do
11:   for i = 1 to 10 do
12:     for each classe γ ∈ Γ = {γ_0, γ_1, ..., γ_n} do
13:       BAYESIAN DEMOCRATIC VOTE(X, i-esimo fold, γ)
14:     end for
15:   end for
16: end for
```

Dalla riga 1 alla riga 8 vengono determinate le *f-measure* di ogni fold per ogni classificatore ricostruendo la confusion matrix (linea 4) e successivamente calcolando questa misura di performance (linea 6) come introdotto nella Sezione 2.2.

La funzione CALCOLA F-MEASURE alla riga 9 consente di calcolare, per ogni modello, la misura che indica quanto bene un classificatore generalizza a dati inosservati, cioè quelli non inclusi nel test set (questa misura è contenuta nella matrice FM). Lo pseudo-codice di questa funzione è riportato nell'Algoritmo 4.2.Si ricorda che nella *10-fold cross validation* viene iterativamente costriuto un modello M_i sul training set escludendo a turno un fold per poi

utilizzare quel fold come test set (Sezione 2.3). Perciò la misura di quanto siano buone le previsioni su nuovi dati di un classificatore, per ogni fold che a turno è un test set, è la media delle *f-measure* dei fold che compongono il traning set, cioè 9 fold.

Algoritmo 4.2 CALCOLA F-MEASURE

Input: Classification Models $M = \{m_1, m_2, \ldots, m_i\}$ con $i > 1$
Output: F-measure dei dati inosservati per ogni fold

```
1: for each Classification Model m ∈ M do
2:   for i = 1 to 10 do
3:     for j = 1 to 10 do
4:       if i ≠ j then
5:         FM_{m,i} ← FM_{m,i} + F-MEASURE_{m,i} {sommo le f-measure di
           ogni fold escludendo il fold del test set}
6:       end if
7:     end for
8:     FM_{m,i} ← FM_{m,i} / 9
9:   end for
10: end for
```

La funzione BAYESIAN DEMOCRATIC VOTE permette di effettuare il voting bayesiano e democratico per ogni istanza (Algoritmo 4.3). La probabilità bayesiana che un'istanza abbia la classe predetta uguale a $\gamma_i \in \Gamma = \{\gamma_0, \gamma_1, \ldots, \gamma_n\}$ è data dal prodotto tra la distribuzione di probabilità associata alla classe γ_i (nota nel file csv di ogni classificatore) e la f-measure $FM_{m,j}$ del $j - esimo$ fold appartenente al classificatore m. Sommando i prodotti, per ogni classe γ_i, di tutti i classificatori compresi nella possibile combinazione otteniamo le probabilità finali di avere la classe predetta uguale a γ_i. La classe con probabilità maggiore sarà la classe assegnata come predetta (righe 14-15-16).
Le righe 6-7-8 servono per realizzare il voting democratico. In questo caso, per ogni classificatore compreso nella possibile combinazione, viene contato quante volte la classe predetta per una specifica istanza è uguale a γ_i (informazione ricavabile dai file csv di input). La classe predetta il maggior numero di volte sarà la classe assegnata a quell'istanza (righe 18-19-20). Viene effettuato anche un voting democratico, oltre a quello bayesiano, in modo da confrontare le accuratezze dei due metodi di voting.

Algoritmo 4.3 Voting bayesiano e democratico

Input: possibile combinazione X $\in \mathcal{P}(M)$, j-esimo fold del Classification Models, classe $\epsilon \in \Gamma = \{\gamma_0, \gamma_1, \ldots, \gamma_n\}$ da ricercare come classe prevista in ogni istanza del fold

Output: Classe predetta di ogni istanza di un fold con voting bayesiano e democratico

1: **for each** istanza i $\in$ j-esimo fold **do**
2: **if** classe prevista dell'istanza == ϵ **then**
3: **for each** classification models $m \in X$ **do**
4: **for** $k = 0$ **to** $|\Gamma|$ **do**
5: $prob\_classe_k \leftarrow prob\_classe_k + distribuzione\_prob_{m,i} * FM_{m,j}$
6: **if** classe predetta dell'istanza da $m \in X == \gamma_k$ **then**
7: $cont\_democratic_k \leftarrow cont\_democratic_k + 1$ {Effettuo voting democratico}
8: **end if**
9: **end for**
10: **end for**
11: max bayesian $\leftarrow 0$
12: max democratic $\leftarrow 0$
13: **for** $k = 0$ **to** $|\Gamma|$ **do**
14: **if** $prob\_classe_k >$ max bayesian **then**
15: max bayesian $\leftarrow prob\_classe_k$
16: class bayesian $\leftarrow \gamma_k$
17: **end if**
18: **if** $cont\_democratic_k >$ max democratic **then**
19: max democratic $\leftarrow cont\_democratic_k$
20: class democratic $\leftarrow \gamma_k$
21: **end if**
22: **end for**
23: $CM^{bayesiana}_{class\ bayesian,\epsilon} \leftarrow CM_{class\ bayesian,\epsilon} + 1$ {Confusion matrix bay}
24: $CM^{democratica}_{class\ democratic,\epsilon} \leftarrow CM_{class\ democratic,\epsilon} + 1$ {Confusion matrix dem}
25: Calcola accuratezza voting bayesiano e democratico
26: **end if**
27: **for** $k = 0$ **to** $|\Gamma|$ **do**
28: $prob\_classe_k \leftarrow 0$
29: $cont\_democratic_k \leftarrow 0$
30: **end for**
31: **end for**

Essendo ora a conoscenza della classe ricercata all'interno del fold come classe prevista (classe ϵ) e della classe predetta dal voting bayesiano e democratico, per ogni possibile combinazione di modelli di classificazione, è possibile ricreare la *confusion matrix* (righe 23 e 24) e calcolare la misura di accuratezza di ogni insieme di classificatori per il voting bayesiano e democratico.
Le accuratezze vengono riportate in un file Excel simile a quello nella Figura 4.2.

	Democratic	Bayesian	Max	Mean	Product	
1	0,70523245	0,730467	0,710038	0,727204	0,727204	(nb, svm)
2	0,67572869	0,727038	0,678818	0,722401	0,722401	(mlp, svm)
3	0,66525526	0,728068	0,681394	0,721718	0,721718	(mlp, nb)
4	0,72497775	0,731843	0,68105	0,732532	0,732016	(mlp, nb, svm)
5	0,73150165	0,719483	0,744548	0,741113	0,741113	(lr, svm)
6	0,71519248	0,73699	0,735791	0,733558	0,733558	(lr, nb)
7	0,73767808	0,733898	0,73682	0,74008	0,740423	(lr, nb, svm)
8	0,68173861	0,732013	0,728928	0,725839	0,725839	(lr, mlp)
9	0,736137	0,733729	0,730474	0,737164	0,734591	(lr, mlp, svm)
10	0,72824176	0,737336	0,722575	0,737339	0,734763	(lr, mlp, nb)
11	0,72772571	0,738364	0,723261	0,73974	0,738023	(lr, mlp, nb, svm)
12	0,70008842	0,715707	0,695453	0,699746	0,699746	(dt, svm)
13	0,68085474	0,726527	0,698369	0,708497	0,708497	(dt, nb)

Figura 4.2: Report delle accuratezze bayesiane e democratiche per ogni ensemble

Capitolo 5

Risultati Degli Esperimenti

5.1 Dataset utilizzati

Il Bayesian Model Averaging è stato testato su 31 dataset presi dall'UCI database [8] ed i suoi risultati sono stati paragonati con quelli ottenuti da Shaohua Wan e Hua Yang nel paper *Comparison among Methods of Ensemble Learning* [23]. Nella Tabella 5.1 sottostante viene riportato per ogni dataset utilizzato il numero di istanze, di attributi e di classi.

Dataset	# Istanze	# Features	# Classi
anneal	898	39	5
autos	205	26	6
audiology	226	70	2
balance-scale	625	5	3
breast-cancer	286	10	2
breast-w	699	10	2
colic	368	23	2
credit-rating	690	16	2
german-credit	1000	21	2
pima-diabetes	768	9	2
glass	214	10	6
heart-c	303	14	2
heart-h	294	14	5
heart-statlog	270	14	2
hepatitis	155	20	2
hypothyroid	3772	30	4
ionosphere	351	35	2
iris	150	5	3
kr-vs-kp	3196	37	2

Dataset	# Istanze	# Features	# Classi
labor	57	17	2
lymph	148	19	4
mushroom	8124	23	2
primary-tumor	339	18	9
segment	2310	20	7
sick	3772	30	2
sonar	208	61	2
soybean	683	36	19
vehicle	846	19	4
vote	435	17	2
vowel	990	13	11
zoo	101	18	7

Tabella 5.1: Numero di istanze, features e classi per ogni dataset

É stato scelto un gruppo di dataset il più vario possibile in modo da verificare successivamente quali insiemi di modelli riescono ad avere performance migliori su dataset con un numero elevato di classi e/o di attributi.

5.2 Classificatori utilizzati

Gli esperimenti sono stati condotti utilizzando i seguenti 5 tipi di classificatori:

- K-Nearest Neighbors
- Decision tree
- Multilayer Perceptron
- Naive Bayes
- Support Vector Machines

K-Nearest Neighbors [1] è un algoritmo utilizzato nel riconoscimento di pattern per la classificazione di oggetti basandosi sulle caratteristiche degli oggetti vicini a quello considerato. Un oggetto è classificato in base alla maggioranza dei voti dei suoi k vicini. k è un intero positivo tipicamente non molto grande. Se k=1 allora l'oggetto viene assegnato alla classe del suo

vicino. Un punto (che rappresenta un oggetto) è assegnato alla classe C se questa è la più frequente fra i k esempi più vicini all'oggetto sotto esame, la vicinanza si misura in base alla distanza fra punti. I vicini sono presi da un insieme di oggetti per cui è nota la classificazione corretta.
Decision tree [21] è un modello predittivo, dove ogni nodo interno rappresenta una $feature$, un arco verso un nodo figlio rappresenta un possibile valore per quella proprietà ed una foglia rappresenta il valore predetto per la feature obiettivo a partire dai valori delle altre proprietà, che nell'albero è rappresentato dal cammino (path) dal nodo radice (root) al nodo foglia. Normalmente un albero di decisione viene costruito utilizzando tecniche di apprendimento a partire dall'insieme dei dati iniziali, il quale può essere diviso in due sottoinsiemi: training set sulla base del quale si crea la struttura dell'albero e il test set che viene utilizzato per testare l'accuratezza del modello predittivo così creato.
Multilayer Perceptron [22] è un modello di rete neurale artificiale che mappa insiemi di dati in ingresso in un insieme di dati in uscita appropriati. È fatta di strati multipli di nodi in un grafo diretto, con ogni strato completamente connesso al successivo. Questo classificatore usa una tecnica di apprendimento supervisionato chiamata *backpropagation* per l'allenamento della rete.
Naive Bayes [13] è un classificatore basato sull'applicazione del teorema di Bayes. Il classificatore bayesiano richiede la conoscenza delle probabilità a priori e condizionali relative al problema, quantità che in generale non sono note ma sono tipicamente stimabili. Nel gergo della classificazione di testi o *Text Categorization*, con il termine classificatore bayesiano ci si riferisce convenzionalmente al classificatore bayesiano naif (Naive Bayes Classifier), ossia un classificatore bayesiano semplificato con un modello di probabilità che fa l'ipotesi di indipendenza delle features, ovvero assume che la presenza o l'assenza di una particolare feature in un documento testuale non sia correlata alla presenza o assenza di altre features.
Support Vector Machines [7] sono modelli di apprendimento supervisionati con algoritmi di apprendimento associati che analizzano i dati utilizzati per l'analisi di regressione e classificazione. Dato un insieme di esempi nel training set, ciascuno contrassegnato come appartenente ad una categoria, un algoritmo di addestramento SVM crea un modello che assegna nuovi esempi a una categoria o all'altra, rendendolo un classificatore lineare binario non probabilistico. Un modello SVM è una rappresentazione degli esempi come punti nello spazio, mappati in modo tale che gli esempi delle diverse categorie siano divisi da un vuoto chiaro il più ampio possibile. I nuovi esempi vengono quindi mappati in quello stesso spazio e si prevede che appartengano a una categoria in base al lato su cui cadono.

5.3 Tipi di insiemi e performance

I classificatori introdotti nella sezione precedente sono stati utilizzati per comporre 6 insiemi di modelli; 5 di questi sono omogenei ed uno è eterogeneo. Per insieme omogeneo si intende un insieme che comprende dei modelli di classificazione provenienti tutti dallo stesso classificatore ma ottenuti facendo variare dei parametri di configurazione interni del classificatore. Mentre l'insieme eterogeneo sarà composto da modelli ognuno dei quali ottenuto con un classificatore diverso.
Per quanto riguarda il voting bayesiano, è stata selezionata l'accuratezza migliore di tutte le possibili combinazioni dei vari modelli. Di seguito viene riportata la composizione dei vari insiemi e le tabelle riassuntive dei risultati in cui verrà sottolineata l'accuratezza della tecnica d'insieme migliore.

Ensemble omogeneo k-nearest neighbors composto da 10 modelli ottenuti facendo variare il parametro k da 1 a 10. k indica il numero dei neighbors più vicini utilizzati nella classificazione. Per la descrizione completa dei parametri fare riferimento all'articolo [1] presente in bibliografia. I risultati sono riportati nella Tabella 5.2.

Ensemble omogeneo Decision Tree composto da 10 modelli ottenuti facendo variare il parametro che indica la soglia di confidenza per l'eliminazione dei rami, da 0.1 a 1.0 con passo 0.1. Per la descrizione completa dei parametri fare riferimento all'articolo [18] presente in bibliografia. I risultati sono riportati nella Tabella 5.3.

Ensemble omogeneo Multilayer Perceptron composto da 10 modelli ognuno dei quali ottenuto facendo variare il parametro che indica il tasso di apprendimento per l'algoritmo di *backpropagation*, con i seguenti valori 0.0001; 0.001; 0.01; 0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7. Per la descrizione completa dei parametri fare riferimento all'articolo [22] presente in bibliografia. I risultati sono riportati nella Tabella 5.4.

Ensemble omogeneo Naive Bayes composto da 3 modelli. Il primo è ottenuto con i parametri standard del classificatore Naive Bayes, il secondo utilizzando lo stimatore della densità del kernel piuttosto che la distribuzione normale per gli attributi numerici ed il terzo utilizzando la discretizzazione supervisionata per elaborare gli attributi numerici. Per la descrizione completa dei parametri fare riferimento all'articolo [11] presente in bibliografia. I risultati sono riportati nella Tabella 5.5.

Ensemble omogeneo Support Vector Machines composto da 40 modelli ottenuti facendo variare il parametro che indica il parametro di penalità del termine di errore da 0.1 a 1.0 con passo 0.1, ed il parametro che indica il grado per una funzione del kernel polinomiale da 2 a 5 con passo 1. Dato che le possibili combinazioni dei modelli sono circa 2^{40}, generare tutte possibili combinazioni risulta dispendioso dal punto di vista computazionale. Perciò è stata applicata l'euristica di selezione del modello ottimale presentata nella sezione 4.4. Per la descrizione completa dei parametri fare riferimento all'articolo [6] presente in bibliografia. I risultati sono riportati nella Tabella 5.6.

Ensemble eterogeneo composto da 5 modelli, ognuno ottenuto rispettivamente utilizzando K-Nearest Neighbors, Decision tree, Multilayer Perceptron, Naive Bayes e Support Vector Machine. La configurazione dei parametri interni dei classificatori è quella di default ed è visionabile tramite i riferimenti alla bibliografia situati nella descrizione dei precedenti insiemi. I risultati sono riportati nella Tabella 5.7.

Dataset	Boosting	Bagging	Stacking	RandomForest	Democratic	Bayesian
anneal	0.8363	0.9822	0.7617	0.9933	0.9811	0.9911
autos	0.4488	0.6976	0.3268	0.8341	0.7305	0.7843
audiology	0.4646	0.7655	0.2522	0.7699	0.7067	0.7429
balance-scale	0.7272	0.8288	0.4576	0.8048	0.9008	0.9008
breast-cancer	0.7028	0.6783	0.7028	0.6923	0.7484	0.7488
breast-w	0.9485	0.9557	0.6552	0.9614	0.9699	0.9728
colic	0.8125	0.8533	0.6404	0.8614	0.8398	0.8399
credit-rating	0.8464	0.8507	0.5505	0.8507	0.8681	0.8754
german-credit	0.695	0.744	0.70	0.725	0.7570	0.7560
pima-diabetes	0.7435	0.7461	0.651	0.7383	0.7435	0.7396
glass	0.4486	0.6963	0.3551	0.729	0.7242	0.7190
heart-c	0.8218	0.8218	0.5446	0.8152	0.8444	0.8413
heart-h	0.7789	0.7857	0.6395	0.7789	0.8339	0.8339
heart-statlog	0.80	0.7926	0.5556	0.7815	0.8296	0.8185
hepatitis	0.8258	0.8452	0.7935	0.8258	0.8838	0.8392
hypothyroid	0.9321	0.9955	0.9229	0.991	0.9353	0.9374
ionosphere	0.9088	0.9088	0.641	0.9288	0.9030	0.8660
iris	0.9533	0.94	0.3333	0.9533	0.9667	0.9667
kr-vs-kp	0.9384	0.9912	0.5222	0.9881	0.9659	0.9722
labor	0.8772	0.8596	0.6491	0.8772	0.9500	0.9500
lymph	0.7432	0.7838	0.5473	0.8108	0.8367	0.8571
mushroom	0.962	1	0.518	1	1	1
primary-tumor	0.2891	0.4513	0.2478	0.4248	0.4865	0.4746
segment	0.2857	0.9697	0.1429	0.9766	0.9667	0.9719
sick	0.9719	0.9849	0.9388	0.9838	0.9655	0.9642
sonar	0.7163	0.774	0.5337	0.8077	0.8848	0.8702
soybean	0.2796	0.8682	0.1318	0.9165	0.9179	0.9180
vehicle	0.3995	0.727	0.2565	0.7707	0.7412	0.7294
vote	0.954	0.9586	0.6138	0.9586	0.9358	0.9336
vowel	0.1737	0.8576	0.909	0.9606	0.9838	0.9929
zoo	0.604	0.4257	0.4059	0.8911	0.9509	0.9709

Tabella 5.2: Migliore accuratezza per ogni dataset con ensemble omogeneo K-Nearest Neighbors

Dataset	Boosting	Bagging	Stacking	RandomForest	Democratic	Bayesian
anneal	0.8363	0.9822	0.7617	0.9933	0.9988	0.9988
autos	0.4488	0.6976	0.3268	0.8341	0.8429	0.8479
audiology	0.4646	0.7655	0.2522	0.7699	0.8532	0.8532
balance-scale	0.7272	0.8288	0.4576	0.8048	0.8000	0.8112
breast-cancer	0.7028	0.6783	0.7028	0.6923	0.7554	0.7554
breast-w	0.9485	0.9557	0.6552	0.9614	0.9556	0.9556
colic	0.8125	0.8533	0.6404	0.8614	0.8584	0.8612
credit-rating	0.8464	0.8507	0.5505	0.8507	0.8594	0.8623
german-credit	0.695	0.744	0.70	0.725	0.7130	0.7150
pima-diabetes	0.7435	0.7461	0.651	0.7383	0.7449	0.7410
glass	0.4486	0.6963	0.3551	0.729	0.6952	0.6952
heart-c	0.8218	0.8218	0.5446	0.8152	0.7917	0.7788
heart-h	0.7789	0.7857	0.6395	0.7789	0.8106	0.8106
heart-statlog	0.80	0.7926	0.5556	0.7815	0.7815	0.7741
hepatitis	0.8258	0.8452	0.7935	0.8258	0.8317	0.8379
hypothyroid	0.9321	0.9955	0.9229	0.991	0.9958	0.9958
ionosphere	0.9088	0.9088	0.641	0.9288	0.9089	0.9089
iris	0.9533	0.94	0.3333	0.9533	0.9467	0.9467
kr-vs-kp	0.9384	0.9912	0.5222	0.9881	0.9972	0.9972
labor	0.8772	0.8596	0.6491	0.8772	0.8100	0.8267
lymph	0.7432	0.7838	0.5473	0.8108	0.7838	0.7771
mushroom	0.962	1	0.518	1	1	1
primary-tumor	0.2891	0.4513	0.2478	0.4248	0.4189	0.4101
segment	0.2857	0.9697	0.1429	0.9766	0.9753	0.9758
sick	0.9719	0.9849	0.9388	0.9838	0.9889	0.9889
sonar	0.7163	0.774	0.5337	0.8077	0.7117	0.7117
soybean	0.2796	0.8682	0.1318	0.9165	0.9282	0.9253
vehicle	0.3995	0.727	0.2565	0.7707	0.7423	0.7518
vote	0.954	0.9586	0.6138	0.9586	0.9610	0.9633
vowel	0.1737	0.8576	0.909	0.9606	0.8768	0.8758
zoo	0.604	0.4257	0.4059	0.8911	0.9418	0.9418

Tabella 5.3: Migliore accuratezza per ogni dataset con ensemble omogeneo Decision Tree

Dataset	Boosting	Bagging	Stacking	RandomForest	Democratic	Bayesian
anneal	0.8363	0.9822	0.7617	0.9933	0.9933	0.9933
autos	0.4488	0.6976	0.3268	0.8341	0.7938	0.7986
audiology	0.4646	0.7655	0.2522	0.7699	0.8401	0.8314
balance-scale	0.7272	0.8288	0.4576	0.8048	0.9247	0.9152
breast-cancer	0.7028	0.6783	0.7028	0.6923	0.7452	0.7384
breast-w	0.9485	0.9557	0.6552	0.9614	0.9642	0.9685
colic	0.8125	0.8533	0.6404	0.8614	0.8559	0.8478
credit-rating	0.8464	0.8507	0.5505	0.8507	0.8696	0.8696
german-credit	0.695	0.744	0.70	0.725	0.7710	0.7690
pima-diabetes	0.7435	0.7461	0.651	0.7383	0.7683	0.7722
glass	0.4486	0.6963	0.3551	0.729	0.7190	0.7100
heart-c	0.8218	0.8218	0.5446	0.8152	0.8483	0.8548
heart-h	0.7789	0.7857	0.6395	0.7789	0.8575	0.8508
heart-statlog	0.8	0.7926	0.5556	0.7815	0.8407	0.8407
hepatitis	0.8258	0.8452	0.7935	0.8258	0.8396	0.8329
hypothyroid	0.9321	0.9955	0.9229	0.991	0.9510	0.9531
ionosphere	0.9088	0.9088	0.641	0.9288	0.9317	0.9203
iris	0.9533	0.94	0.3333	0.9533	0.9800	0.9800
kr-vs-kp	0.9384	0.9912	0.5222	0.9881	0.9950	0.9947
labor	0.8772	0.8596	0.6491	0.8772	0.8967	0.9333
lymph	0.7432	0.7838	0.5473	0.8108	0.8519	0.8519
mushroom	0.962	1	0.518	1	1	1
primary-tumor	0.2891	0.4513	0.2478	0.4248	0.4840	0.4721
segment	0.2857	0.9697	0.1429	0.9766	0.9680	0.9684
sick	0.9719	0.9849	0.9388	0.9838	0.9751	0.9761
sonar	0.7163	0.774	0.5337	0.8077	0.8374	0.8326
soybean	0.2796	0.8682	0.1318	0.9165	0.9458	0.9487
vehicle	0.3995	0.727	0.2565	0.7707	0.8583	0.8570
vote	0.954	0.9586	0.6138	0.9586	0.9609	0.9679
vowel	0.1737	0.8576	0.909	0.9606	0.9636	0.9677
zoo	0.604	0.4257	0.4059	0.8911	0.9618	0.9618

Tabella 5.4: Migliore accuratezza per ogni dataset con ensemble omogeneo Multilayer Perceptron

Dataset	Boosting	Bagging	Stacking	RandomForest	Democratic	Bayesian
anneal	0.8363	0.9822	0.7617	0.9933	0.9666	0.9565
autos	0.4488	0.6976	0.3268	0.8341	0.6381	0.6679
audiology	0.4646	0.7655	0.2522	0.7699	0.7342	0.7123
balance-scale	0.7272	0.8288	0.4576	0.8048	0.9087	0.9072
breast-cancer	0.7028	0.6783	0.7028	0.6923	0.7170	0.7346
breast-w	0.9485	0.9557	0.6552	0.9614	0.9757	0.9757
colic	0.8125	0.8533	0.6404	0.8614	0.8099	0.7989
credit-rating	0.8464	0.8507	0.5505	0.8507	0.8681	0.8464
german-credit	0.695	0.744	0.70	0.725	0.7630	0.7560
pima-diabetes	0.7435	0.7461	0.651	0.7383	0.7631	0.7644
glass	0.4486	0.6963	0.3551	0.729	0.5567	0.6825
heart-c	0.8218	0.8218	0.5446	0.8152	0.8480	0.8378
heart-h	0.7789	0.7857	0.6395	0.7789	0.8441	0.8541
heart-statlog	0.8	0.7926	0.5556	0.7815	0.8444	0.8444
hepatitis	0.8258	0.8452	0.7935	0.8258	0.8446	0.8513
hypothyroid	0.9321	0.9955	0.9229	0.991	0.9626	0.9682
ionosphere	0.9088	0.9088	0.641	0.9288	0.9203	0.9203
iris	0.9533	0.94	0.3333	0.9533	0.9600	0.9600
kr-vs-kp	0.9384	0.9912	0.5222	0.9881	0.8789	0.8783
labor	0.8772	0.8596	0.6491	0.8772	0.9000	0.9167
lymph	0.7432	0.7838	0.5473	0.8108	0.8505	0.8371
mushroom	0.962	1	0.518	1	0.9583	0.9581
primary-tumor	0.2891	0.4513	0.2478	0.4248	0.5013	0.4777
segment	0.2857	0.9697	0.1429	0.9766	0.9160	0.9195
sick	0.9719	0.9849	0.9388	0.9838	0.9745	0.9714
sonar	0.7163	0.774	0.5337	0.8077	0.7267	0.7740
soybean	0.2796	0.8682	0.1318	0.9165	0.9296	0.9282
vehicle	0.3995	0.727	0.2565	0.7707	0.6076	0.6229
vote	0.954	0.9586	0.6138	0.9586	0.9014	0.9014
vowel	0.1737	0.8576	0.909	0.9606	0.6889	0.7061
zoo	0.604	0.4257	0.4059	0.8911	0.9509	0.9609

Tabella 5.5: Migliore accuratezza per ogni dataset con ensemble omogeneo Naive Bayes

Dataset	Boosting	Bagging	Stacking	RandomForest	Democratic	Bayesian
anneal	0.8363	0.9822	0.7617	0.9933	0.9244	0.8988
autos	0.4488	0.6976	0.3268	0.8341	0.4036	0.3800
audiology	0.4646	0.7655	0.2522	0.7699	0.6854	0.6636
balance-scale	0.7272	0.8288	0.4576	0.8048	0.9952	0.9952
breast-cancer	0.7028	0.6783	0.7028	0.6923	0.7341	0.7063
breast-w	0.9485	0.9557	0.6552	0.9614	0.9485	0.9528
colic	0.8125	0.8533	0.6404	0.8614	0.8015	0.8096
credit-rating	0.8464	0.8507	0.5505	0.8507	0.5928	0.5551
german-credit	0.695	0.744	0.70	0.725	0.7030	0.70
pima-diabetes	0.7435	0.7461	0.651	0.7383	0.6627	0.6510
glass	0.4486	0.6963	0.3551	0.729	0.5281	0.50
heart-c	0.8218	0.8218	0.5446	0.8152	0.4555	0.4555
heart-h	0.7789	0.7857	0.6395	0.7789	0.3605	0.3605
heart-statlog	0.80	0.7926	0.5556	0.7815	0.7667	0.7741
hepatitis	0.8258	0.8452	0.7935	0.8258	0.8013	0.7942
hypothyroid	0.9321	0.9955	0.9229	0.991	0.9669	0.9647
ionosphere	0.9088	0.9088	0.641	0.9288	0.8717	0.8690
iris	0.9533	0.94	0.3333	0.9533	0.9733	0.9733
kr-vs-kp	0.9384	0.9912	0.5222	0.9881	0.9443	0.9440
labor	0.8772	0.8596	0.6491	0.8772	0.9333	0.8967
lymph	0.7432	0.7838	0.5473	0.8108	0.8110	0.8043
mushroom	0.962	1	0.518	1	0.9850	0.9842
primary-tumor	0.2891	0.4513	0.2478	0.4248	0.2951	0.3128
segment	0.2857	0.9697	0.1429	0.9766	0.9182	0.8835
sick	0.9719	0.9849	0.9388	0.9838	0.9594	0.9531
sonar	0.7163	0.774	0.5337	0.8077	0.6776	0.6826
soybean	0.2796	0.8682	0.1318	0.9165	0.8623	0.8139
vehicle	0.3995	0.727	0.2565	0.7707	0.8158	0.8193
vote	0.954	0.9586	0.6138	0.9586	0.9564	0.9564
vowel	0.1737	0.8576	0.909	0.9606	0.8061	0.8192
zoo	0.604	0.4257	0.4059	0.8911	0.8118	0.8218

Tabella 5.6: Migliore accuratezza per ogni dataset con ensemble omogeneo Support Vector Machine

Dataset	Boosting	Bagging	Stacking	RandomForest	Democratic	Bayesian
anneal	0.8363	0.9822	0.7617	0.9933	0.9922	0.9922
autos	0.4488	0.6976	0.3268	0.8341	0.8181	0.8529
audiology	0.4646	0.7655	0.2522	0.7699	0.8449	0.8225
balance-scale	0.7272	0.8288	0.4576	0.8048	0.9087	0.9024
breast-cancer	0.7028	0.6783	0.7028	0.6923	0.7590	0.7558
breast-w	0.9485	0.9557	0.6552	0.9614	0.9699	0.9714
colic	0.8125	0.8533	0.6404	0.8614	0.8559	0.8586
credit-rating	0.8464	0.8507	0.5505	0.8507	0.8638	0.8638
german-credit	0.695	0.744	0.70	0.725	0.7690	0.7640
pima-diabetes	0.7435	0.7461	0.651	0.7383	0.7800	0.7800
glass	0.4486	0.6963	0.3551	0.729	0.7201	0.7472
heart-c	0.8218	0.8218	0.5446	0.8152	0.8347	0.8415
heart-h	0.7789	0.7857	0.6395	0.7789	0.8540	0.8506
heart-statlog	0.8	0.7926	0.5556	0.7815	0.8556	0.8593
hepatitis	0.8258	0.8452	0.7935	0.8258	0.8508	0.8638
hypothyroid	0.9321	0.9955	0.9229	0.991	0.9690	0.9960
ionosphere	0.9088	0.9088	0.641	0.9288	0.9517	0.9460
iris	0.9533	0.94	0.3333	0.9533	0.9800	0.9800
kr-vs-kp	0.9384	0.9912	0.5222	0.9881	0.9947	0.9956
labor	0.8772	0.8596	0.6491	0.8772	0.9333	0.9333
lymph	0.7432	0.7838	0.5473	0.8108	0.8710	0.8581
mushroom	0.962	1	0.518	1	1	1
primary-tumor	0.2891	0.4513	0.2478	0.4248	0.4807	0.4837
segment	0.2857	0.9697	0.1429	0.9766	0.9801	0.9805
sick	0.9719	0.9849	0.9388	0.9838	0.9812	0.9852
sonar	0.7163	0.774	0.5337	0.8077	0.8702	0.8657
soybean	0.2796	0.8682	0.1318	0.9165	0.9458	0.9473
vehicle	0.3995	0.727	0.2565	0.7707	0.8217	0.8264
vote	0.954	0.9586	0.6138	0.9586	0.9701	0.9678
vowel	0.1737	0.8576	0.909	0.9606	0.9707	0.9929
zoo	0.604	0.4257	0.4059	0.8911	0.9718	0.9809

Tabella 5.7: Migliore accuratezza per ogni dataset con ensemble eterogeneo

5.4 Risultati

Nella Tabella 5.8 è riportata la percentuale di volte in cui il Bayesian Model Averaging, per ogni insieme, risulta migliore rispetto a tutte le tecniche di ensemble learning (Boosting, Bagging, Stacking, Random Forest e Majority Voting) con il numero di modelli utilizzati. Al fine di verificare le performance del BMA sui dataset con un numero sufficientemente alto di attributi o classi, sono stati presi in considerazione anche i soli dataset con almeno 20 attributi o almeno 5 classi.

Tipo insieme	**# modelli**	**% successo BMA**	**% successo BMA (attributi $\geq$ 20)**	**% successo BMA (classi $\geq$ 5)**
K-Nearest Neighbors	10	39%	21%	33%
Decision Tree	10	42%	57%	22%
Multilayer Perceptron	10	45%	21%	55%
Naive Bayes	3	29%	7%	11%
Support Vector Machine	40	10%	0%	0%
Eterogeneo	5	65%	57%	88%

Tabella 5.8: Percentuale di successo del BMA per ogni insieme considerando tutti i dataset, dataset con almeno 20 attributi, dataset con almeno 5 classi

Dai risultati riportati nella Tabella 5.8 è possibile notare che le performance peggiori sono state ottenute con l'insieme omogeneo realizzato con il classificatore Support Vector Machine, nonostante l'elevato numero di modelli generati. Dato che per l'insieme di Support Vector Machines è stata applicata l'euristica di selezione dell'insieme di modelli ottimale proposta nella sezione 4.4, quest'ultima si è rivelata efficiente ma non efficace. Sono invece abbastanza soddisfacenti i risultati ottenuti dall'insieme omogeneo generato dal classificatore Multilayer Perceptron che risulta il secondo insieme con i migliori risultati sia su tutti i dataset, sia su quelli con almeno 20 attributi che su quelli con almeno 5 classi.
Le performance migliori si ottengono con l'insieme eterogeneo con cui su più della metà dei dataset si riesce ad ottenere la migliore accuratezza, fino ad arrivare all'88% di successo sui dataset con almeno 5 classi.

Capitolo 6

Conclusioni e sviluppi futuri

In questa tesi ho analizzato e approfondito il potenziale dell'Ensemble Learning confrontando le principali tecniche d'insieme presenti allo stato dell'arte (Bagging, Boosting, Random Forest, Stacking e Majority Voting) con un metodo d'insieme basato sul Bayesian Model Averaging. Quest'ultima tecnica ha l'obbiettivo di superare le limitazioni degli algoritmi tradizionali. Ciò è possibile sfruttando le potenzialità di diversi algoritmi di apprendimento in modo da non avere più l'incertezza su quale modello rappresenti la scelta ottimale. Inoltre, dato che i classificatori utilizzati dalle tecniche d'insieme tradizionali sono considerati indipendenti e ugualmente affidabili, è stata proposta un'euristica finalizzata a valutare il contributo a priori dei singoli modelli nell'operazione di classificazione. Questo contributo viene utilizzato per la selezione dell'insieme di modelli ottimale.
Per confrontate il Bayesian Model Averaging con gli algoritmi tradizionali sono stati formati 6 insiemi di modelli, 5 omogenei ed 1 eterogeneo. I risultati sperimentali sono soddisfacenti: il BMA risulta avere la migliore accuratezza utilizzando l'insieme eterogeneo. Quest'ultimo è l'insieme con l'accuratezza predittiva maggiore anche sui dataset con un numero sufficientemente alto di features o classi. Posso concludere che l'insieme ideale di modelli fortemente predittivo deve essere composto da modelli generati da algoritmi di apprendimento diversi tra loro.

Come futuro sviluppo è possibile introdurre una strategia di *forward selection* [14] per la selezione dell'insieme di modelli ottimale, dato che la strategia di *backward elimination* si è rivelata poco efficace, e paragonare le performance ottenute con quelle della *backward elimination*.
Un altro possibile miglioramento può essere la parallelizzazione della generazione di tutte le possibile combinazioni dei modelli. In questo modo l'algoritmo del Bayesian Model Averaging può essere reso più efficiente dal punto di vista computazionale.

Infine l'approccio attuale di apprendimento può essere modificato utilizzando delle tecniche di *multi-task learning* in cui sono presenti più task messi in relazione tra di loro. In particolare l'obbiettivo è realizzare una struttura di voto gerarchica in cui viene prima affrontata la discriminazione tra "oggettivo" e "soggettivo" per poi affrontare la classificazione delle istanze del dataset considerate soggettive.

Bibliografia

[1] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Robert Bryll, Ricardo Gutierrez-Osuna, and Francis Quek. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern recognition*, 36(6):1291–1302, 2003.

[5] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18. ACM, 2004.

[6] Chih-Chung Chang and Chih-Jen Lin. Libsvm - a library for support vector machines, 2001. The Weka classifier works with version 2.82 of LIBSVM.

[7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[8] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[9] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

[10] WL Hosch. Machine learning artificial intelligence. *Encyclopeadia Britannic*, 2004.

[11] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Mateo, 1995. Morgan Kaufmann.

[12] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[13] Pat Langley, Wayne Iba, Kevin Thompson, et al. An analysis of bayesian classifiers. In *Aaai*, volume 90, pages 223–228, 1992.

[14] Gonzalo Martínez-Muñoz and Alberto Suárez. Pruning in ordered bagging ensembles. In *Proceedings of the 23rd international conference on Machine learning*, pages 609–616. ACM, 2006.

[15] Ioannis Partalas, Grigorios Tsoumakas, and Ioannis Vlahavas. An ensemble uncertainty aware measure for directed hill climbing ensemble pruning. *Machine Learning*, 81(3):257–282, 2010.

[16] Wei Peng and Dae Hoon Park. Generate adjective sentiment dictionary for social media sentiment analysis using constrained nonnegative matrix factorization. *Urbana*, 51:61801, 2004.

[17] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[18] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[19] Delip Rao and Deepak Ravichandran. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 675–682. Association for Computational Linguistics, 2009.

[20] Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.

[21] Lior Rokach and Oded Z Maimon. *Data mining with decision trees: theory and applications*, volume 69. World scientific, 2008.

[22] C Van Der Malsburg. Frank rosenblatt: Principles of neurodynamics: perceptrons and the theory of brain mechanisms. In *Brain Theory*, pages 245–248. Springer, 1986.

[23] Shaohua Wan and Hua Yang. Comparison among methods of ensemble learning. In *Biometrics and Security Technologies (ISBAST), 2013 International Symposium on*, pages 286–290. IEEE, 2013.

[24] Gang Wang, Jianshan Sun, Jian Ma, Kaiquan Xu, and Jibao Gu. Sentiment classification: The contribution of ensemble learning. *Decision support systems*, 57:77–93, 2014.

[25] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[26] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[27] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.