Università degli Studi di Ferrara

Università degli Studi di Ferrara

Corso di Laurea in Informatica

Il mio titolo della tesi in italiano

Relatore:
Prof. Nome Cognome

Laureando:
Luca Greggio

Anno Accademico 2020 – 2021

Indice

Page

Introduzione **5**

1 Machine Learning **7**
1.1 Apprendimento supervisionato 7
1.2 Alberi di decisione 8
1.3 I passaggi necessari 12

2 Il problema **15**
2.1 La discretizzazione 15
2.1.1 Formati WAV e AIFF 17
2.2 Features utilizzate 17

3 L'intelligenza **19**
3.1 Prima fase 19
3.2 Seconda fase 21

Introduzione

Elementi essenziali nella musica sono importanti il tempo, la sincronia tra i diversi musicisti e come questi si interfacciano gli uni agli altri. Tra musicisti (principalmente in generi come il Jazz) si dice *"avere groove"*, termine usato per definire un portamento ritmico che provoca nella musica un'empatia tra musicisti e ascoltatori.
Sembrerebbe che questa carica di emozioni possa trovare origine in un comportamento dei musicisti, i quali appaiono suonare non perfettamente a metronomo, ma gli stessi suonano introducendo delle micro-variazioni di tempo (di millisecondi) le singole note, originando del movimento all'interno della musica, questo fenomeno è chiamato *microtiming*, e apparirebbe essere collegato al groove.
Illustratomi questo problema dal mio tutore di tirocinio, il mio compito è stato quello di costruire un applicativo in grado di aiutare nell'analisi di questo problema. Dunque il risultato finale è un software in grado di collocare temporalmente le note suonate da uno strumento in una traccia audio dello stesso. Dopo un'analisi attenta, il percorso scelto per la risoluzione è stato quello dell'intelligenza artificiale, precisamente il ML (machine learning), tecnica molto usata in questo ambito per approcciarsi a questa analisi. Lo scopo era dunque insegnare a una macchina la differenza tra una nota e una non-nota all'interno delle tracce audio fornite.
Una volta costruita l'intelligenza del software usando il linguaggio di programmazione python, e appoggiandosi a weka, software open source per l'apprendimento automatico, è risultato necessario aggiungere uno scheletro per facilitarne l'utilizzo a utenti meno esperti con linea di comando, dunque creare una GUI (Graphical User Interface), realizzata anch'essa in linguaggio Python e con un framework chiamato Qt.

Capitolo 1

Machine Learning

Si intende, per ML, una tecnica utilizzabile per risolvere problemi nei quali si è in grado di specificare degli output dati determinati input senza però essere in grado di comprendere la relazione esistente tra i valori. Spesso problemi del tipo appena descritto portano con se un quantitativo di dati decisamente troppo vasto per essere analizzato soltanto da una o più persone, questa particolare tecnica offre dunque un modo più efficiente per svolgere analisi sui dati e rendere automatici determinati procedimenti consentendo anche di diminuire errori di tipo casuale dati dall'uomo, o addirittura catturare più informazioni rispetto a un operatore umano.
Il ML possiede un dominio di applicazione molto ampio, che può andare dalla medicina alla psicologia, e proprio per questo motivo esistono vari approcci e modalità. Nel mio lavoro di tirocinio in particolare è stato usato un approccio supervisionato.

1.1 Apprendimento supervisionato

Lo scopo di questo sistema è come dice la parola stessa, di supervisionare una macchina, questo costruendo un *data-set* [1] di valori, solitamente dei tipi che se-

[1] Il data-set è un insieme di dati raccolti da misurazioni svolte in fase di preparazione per lo studio del problema, classificati in modo opportuno.

guono: numerici, che devono essere normalizzati [2] prima di essere usati, nominali, stringhe oppure date, che serviranno come punto di riferimento alla macchina per costruire le regole che permetteranno di restituire precisi output dati determinati input.
Il ML con apprendimento supervisionato si divide in problemi di classificazione o di regressione. Il Caso in questione rientra nel tipo di classificazione, dunque un problema che dato un valore, o insieme di valori, restituisce un risultato discreto appoggiandosi a un *albero di decisione* (DT) che la macchina ha costruito a partire da un meta-algoritmo, ciò dopo essere stata allenata con i dati di interesse.

1.2 Alberi di decisione

Un albero di decisione è una tecnica di ML supervisionato, questo si divide in due tipologie: di classificazione e di regressione. Un DT di classificazione permette di classificare i dati in modo discreto, le sue variabili infatti sono delle categorie come possono essere maschio e femmina, giallo e rosso e via dicendo. Un DT di regressione viene usato per restituire delle predizioni dati dei valori di input, solitamente appartenenti all'insieme dei reali.
Come il nome suggerisce sono delle strutture ad albero dove ogni nodo in base a un valore consente di continuare in una direzione piuttosto che un'altra fino ad arrivare al termine di questo albero e dunque della decisione, termine che si trova nei nodi foglia, ovvero quelli finali. Questa struttura decisionale viene costruita tramite dei *meta-algoritmi*[3] ad esempio J48, usato nel nostro caso, un'estensione del suo predecessore ID3 che prevede i seguenti passaggi:

[2] Per normalizzazione si intende un processo che rende paragonabili dati tra loro di diversa natura fornendo valori compresi tra 0 e 1, ciò ottenuto nel modo seguente:
$\forall x_i, i \subset \mathbb{N}\ x_{norm} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$

[3] Ovvero si tratta di un algoritmo che non risolve il problema, ma è in grado di costruire lui stesso un nuovo algoritmo.

1. Iterazione di ogni attributo su cui avviene il calcolo dell'*entropia*, o, valore compreso tra 0 e 1.

2. Selezione dell'attributo con l'entropia[4] minore.

3. Divisione (split)[5] del data-set sull'attributo con minore entropia.

I passaggi appena descritti sono la sintesi del seguente pseudocodice:

---

procedure ID3(*Esempi*, *AttributoTarget*, *Attributi*)
Crea un nodo *Radice*.
if Se tutti gli esempi sono positivi **then**
restituisci un albero con un unico nodo *Radice* ed etichetta = +
else if Se tutti gli esempi sono negativi **then**
restituisci un albero con un unico nodo *Radice* ed etichetta = -
else if Se Attributi è vuoto **then**
restituisci un albero con un unico nodo *Radice* ed etichetta = il valore di *AttributoTarget* più comune tra le istanze di Esempi.
else
$A \leftarrow$ L'elemento di Attributi che riduce maggiormente l'entropia.
for all v in A **do**
Aggiungi un nuovo ramo sotto *Radice* corrispondente al test $A = v$.
$Esempi(i) \leftarrow$ sottoinsieme di *Esempi* che hanno valore v per A.
if *Esempi* è vuoto **then** Aggiungi una foglia con etichetta = valore *AttributoTarget* più comune tra gli esempi.
else
Aggiungi sotto *Radice* il sottoalbero ID3 ($Esempi(v)$, *AttributoTarget*, *Attributi* – A).

---

[4]Misura che indica la quantità di incertezza di un valore, misurabile nel modo che segue $E(X) = \sum_{x \in X} -p(x) log_2 p(x)$ con S il data-set su cui si vuole calcolare l'entropia, X il set di classi in S e in fine $p(x)$ il numero di elementi di classe x nel set S

[5]Il data-set su cui vengono svolti i calcoli, viene diviso in due a partire dall'attributo con una minor entropia. Ciò porterà ad avere il primo sub-set conterrà tutte le istanze che avranno un determinato valore della classe osservata, il secondo possederà invece tutte le tuple contenenti il valore altro dell'attributo, se si ha un terzo valore che può assumere l'attributo si avrà un terzo sub-set, e così via per tanti quanti sono i valori possibili che questo può assumere.

Questi passaggi teoricamente andrebbero ripetuti per ogni sub-set fino a ottenerne di puri[6], operazione però non sempre possibile per le dimensioni spesso molto grandi dei set di dati. Vengono dunque presi in esame vari fattori come la dimensione del data-set, o il livello di precisione accettabile per interrompere l'esecuzione dell'algoritmo. Un altro caso possibile di interruzione si ha nel momento in cui viene calcolata un'entropia pari a 1.

Per comprendere meglio è possibile considerare il seguente data-set che consentirà di decidere se una partita può o meno essere giocata in base alle condizioni meteorologiche.

tempo	temperatura	umidità	vento	si gioca
sole	caldo	alta	no	no
sole	caldo	alta	si	no
nuvoloso	caldo	alta	no	si
pioggia	media	normale	no	si
pioggia	freddo	normale	no	si
pioggia	freddo	normale	si	no
nuvoloso	freddo	normale	si	no
sole	media	alta	no	no
sole	freddo	normale	no	si
pioggia	media	normale	no	si
sole	media	normale	si	si
nuvoloso	media	alta	si	si

Si procede dunque calcolando l'entropia per il data-set con 7 esiti positivi e 5 negativi il risultato risulterà la seguente informazione:
$I(D) = E\left(\frac{7}{12}; \frac{5}{12}\right) = -\left(\frac{7}{12} log_2\left(\frac{7}{12}\right) + \frac{5}{12} log_2\left(\frac{5}{12}\right)\right) = 0.98$
Ora è possibile calcolare l'entropia dei singoli attributi.

[6]Dove ogni uno di essi contiene una sola sola tupla

Si osserva il tempo su tutto il set per cominciare
$I(tempo, D) = \frac{|D(tempo=sole)|}{|D|} + \frac{|D(tempo=pioggia)|}{|D|} + \frac{|D(tempo=nuvoloso)|}{|D|}$
$= \frac{5}{12} I(D(tempo = sole)) + \frac{4}{12} I(D(tempo = pioggia)) + \frac{3}{12} I(D(tempo = nuvoloso))$
$= \frac{5}{12} E\left(\frac{2}{5}; \frac{3}{5}\right) + \frac{4}{12} E\left(\frac{3}{4}; \frac{1}{4}\right) + \frac{3}{12} E\left(\frac{2}{3}; \frac{1}{3}\right) = 0.74$
Questo procedimento andrà eseguito per ogni attributo, quindi temperatura, umidità e vento ancora. Se vengono svolti i calcoli per i restanti si avrà che il l'entropia minore si ha proprio per la temperatura, risulteranno dunque i seguenti data-set:

tempo	temperatura	umidità	vento	si gioca
sole	caldo	alta	no	no
sole	caldo	alta	si	no
sole	media	alta	no	no
sole	freddo	normale	no	si
sole	media	normale	si	si

tempo	temperatura	umidità	vento	si gioca
pioggia	media	normale	no	si
pioggia	freddo	normale	no	si
pioggia	freddo	normale	si	no
pioggia	media	normale	no	si

tempo	temperatura	umidità	vento	si gioca
nuvoloso	caldo	alta	no	si
nuvoloso	freddo	normale	si	no
nuvoloso	media	alta	si	si

uno con tutti tutti i valori dell'attributo tempo pari a "sole", uno con "pioggia", e uno con "nuvoloso" generando così la prima porzione di albero, la sua radice che presenterà 3 possibili percorsi, 1 per ogni tipo di meteo. A questo punto bisognerà ripetere i passaggi per ogni set che è stato ottenuto e fino a che ogni set sia puro.

1.3 I passaggi necessari

Per costruire un sistema di classificazione intelligente con l'utilizzo del ML supervisionato, precisamente con weka sono dunque necessari vari passaggi. Il primo passaggio è la creazione di un data-set, questo potrà essere fatto in diversi modi, nel caso corrente il risultato è un file ARFF[7] contenente le features[8] dei file audio che sono stati etichettati con le rispettive classi.
Segue poi una fase di allenamento dove viene costruito l'albero di decisione con un meta-algoritmo che può variare in base alle diverse esigenze e al tipo di problema in analisi. Generato il modello di classificazione creato ci si può comportare poi in diversi modi e decidere se validare o meno il risultato ottenuto, scegliendo dunque tra *full train*, *cross validation*, e *training più test*. Scegliendo di lavorare in modalità full train si decide non effettuare validazioni rischiando così l'overfitting[9]. Per ridurre questo rischio è possibile dunque ricorrere a una tecnica di validazione come la modalità training più test andando così a dividere il data-set in due ottenendo così un sottoinsieme per il training e uno per il test, metodologia principalmente utilizzata con un elevato quantitativo di dati. Se per il problema che si intende risolvere la numerosità non è sufficiente per ottenere buoni risultati con la tecnica appena riportata, può allora essere considerata la metodologia di tipo cross validation consistente nella divisione del data-set in k sottoinsiemi, $k-1$ di questi verranno utilizzati per l'allenamento e 1 invece per il testing, operazione ripetuta k volte.
Il classificatore che è stato ottenuto dovrà essere poi valutato per le sue capacità di classificare in modo corretto, passo generalmente svolto appoggiandosi a una *matrice di confusione* (CM), questa fornirà delle informazioni riguardo ai valori che sono i *true positive* (TP), *true negative* (TN), *false positive* (FP), *false negative* (FN) e quindi comprendere le performances dell'algoritmo. Ogni sua colonna rappresenta i valori predetti, mentre e righe i valori reali.

[7] File di tipo testuale che presenta due sezioni, header e data. In header viene specificata la struttura e il tipo di dati per ogni tupla. In data vengono riportate le misurazioni o valori ordinati come specificato in header con i rispettivi valori.

[8] Sono le caratteristiche di interesse del fenomeno che deve essere studiato

[9] Condizione per la quale un classificatore ha imparato a riconoscere casi troppo specifici e ha difficoltà in quelli più generali. Solitamente dato da un allenamento con una durata maggiore di quella necessaria oppure un data-set troppo piccolo.

Data la CM sarà possibile poi calcolare la *accuracy* come $ACC = \frac{TP+TN}{TP+TN+FP+FN}$, oppure *sensitivity*) e *specificity* dette anche *true positive rate* (TPR) e *true negative rate* (TNR), nei seguenti modi: $TPR = \frac{TP}{TP+FN}$ e $TNR = \frac{TN}{TN+FP}$.

Capitolo 2

Il problema

Come già illustrato il dominio del problema è il suono. Per poterne svolgere delle analisi e ricavare i dati necessari deve prima di tutto essere catturato, questo tramite microfoni collegati ad degli appositi ADC[1] che porteranno il segnale a un computer in grado, tramite appositi software, di salvare l'audio catturato in file audio WAV o AIFF, i audio non compressi in grado di preservare tutte le informazioni catturate con la strumentazione usata, a differenza di formati come mp3[2].

2.1 La discretizzazione

Essendo dunque il suono riconvertito in digitale la sua onda non è più continua come quella catturata da un microfono, ma viene discretizzata passando in digitale. Questo significa che l'onda risultante sul computer svolgente le registrazioni sarà un insieme di punti, detti anche *campioni* o *samples*, distanti tra loro intervalli di tempo sufficientemente brevi e costanti, che verranno poi interpolati tra loro per ricreare quella che è l'onda sonora inizialmente catturata. Introduciamo dunque una caratteristica del suono registrato in ambiente digitale, il *samplerate*(SR)

[1] Strumenti in grado di convertire il segnale analogico fornito da componenti analogiche, microfoni audio per questo particolare caso, in digitale. Nello specifico per il suono questi convertitori sono chiamati schede audio

[2] Formato audio compresso per essere di dimensioni inferiori ma di qualità inferiore rispetto ai due appena descritti.

(ovvero frequenza di campionamento) tipicamente pari a 44100Hz[3][4], o più fino a 192KHz.

Nel passare da analogico a digitale particolare attenzione deve essere prestata anche alla *risoluzione* dell'onda, infinita per un segnale analogico e non per uno digitale, infatti un segnale digitale ha una risoluzione tanto maggiore tanto quanto grande il valore di *bit depth*. Il segnale audio viene discretizzato non solo nel tempo, ma anche in ampiezza, e la risoluzione indica quanto quanti valori possibili può assumere in ampiezza un segnale audio digitale, tanto è maggiore la bit depth tanto più la risoluzione di un segnale digitale si avvicinerà a quella di uno analogico. Solitamente il valore di registrazione, a livello professionale, è di 24 bit, il che equivale ad avere 2^{24} valori possibili in ampiezza per il segnale.

Come possibile osservare nell'immagine 2.1 l'input che riceve un ADC è un segna-

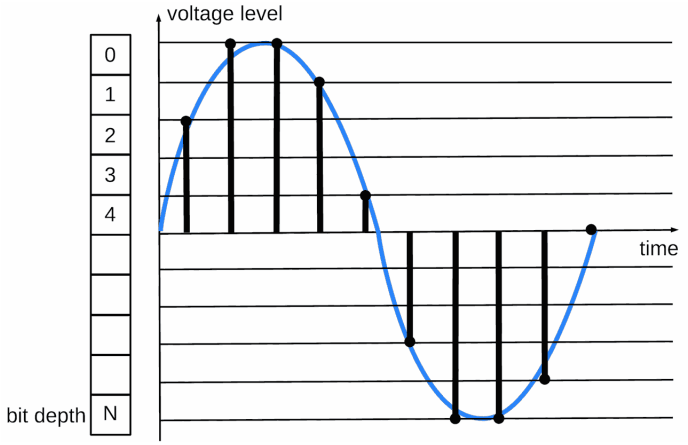


Figura 2.1: Conversione segnale audio analogico digitale

le analogico, un voltaggio identificato con il sinusoide colorato di blu, continuo e senza interruzioni. Questo voltaggio verrà registrato a intervalli regolari di tempo dal convertitore, andando ad approssimare il segnale rilevato al bit che più si avvicina ad esso, ad esempio vediamo he alla prima rilevazione l'ADC approssima il segnale analogico al secondo bit, dal momento che era il bit a cui si era più vicino.

[3] Ovvero 44100 campioni rilevati ogni secondo

[4] Questa è la frequenza minima di campionamento per registrazioni che interessano lo spettro dell'udibile all'uomo 20-22KHz, ovvero $SR \geq 2f_{max}$, questo per evitare la generazione di onde fantasma nei segnali registrati.

Il secondo come il terzo vengono approssimati al bit 0, e così via fino a che non termina il segnale. Questi punti verranno poi interpolati tra loro dai programmi che permettono di effettuare le registrazioni e salvarli nei formati di preferenza, nel nostro caso ARFF o WAV.

2.1.1 Formati WAV e AIFF

Questi formati audio sono molto importanti, come detto all'inizio del capitolo, perché consentono di salvare tracce audio senza alterare le caratteristiche sonore e dinamiche dell'audio. Questo quantitativo di informazioni deve però essere pagato in termini di memoria, infatti i file salvati in questi formati possono arrivare ad essere di grandi dimensioni anche per registrazioni di pochi minuti. A questo proposito viene introdotto negli anni '90, con l'avvento dell'audio digitale, il formato mp3, che consente di effettuare un *downsampling* dell'audio riducendo lo spazio di memoria necessario per l'archiviazione. Questa procedura di compressione delle informazioni, riduce però la qualità, non strettamente percepibile da rovinare l'ascolto, ma a sufficienza da rendere questo tipo di file inadatto ad analisi attente come missaggio, o mixing in inglese[5] e mastering[6] per gli studi di registrazione.

2.2 Features utilizzate

Visto come sono formati i file audio da utilizzare, WAV o AIFF, sono state scelte le seguenti features da utilizzare per l'approccio deciso di ML per descrivere un'onda sonora: la *media*, la *varianza*, la *media della derivata prima* e *della derivata seconda*.

[5]Fase di miscelazione dei suoni in produzione audio

[6]Fase successiva al missaggio svolta su una singola traccia ottenuta dal missaggio per consentire la migliore riproduzione sonora dell'audio su diversi dispositivi

Capitolo 3

L'intelligenza

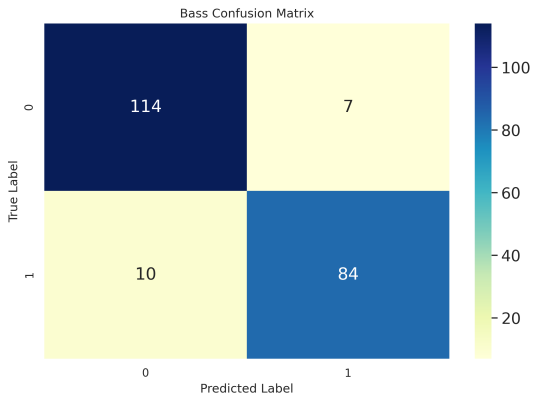
La fase di costruzione dell'intelligenza in grado di comprendere quando vengono suonate le note in una traccia audio si è divisa in due fasi. La prima è stata quella di riconoscere la differenza tra nota e non nota dati dei audio selezionati manualmente senza considerare la lunghezza. In un secondo momento si è poi proceduto dividendo i campioni di note in più parti di uguale lunghezza, associando a ogni divisione del campione un valore che va da 1 a 5, ovvero delle classi che indicano, essere certamente una nota associandovi la classe 5, o essere certamente non una nota accoppiando il campione alla classe 1.
Le tracce di strumenti fornitemi dal tutore e utilizzate per gli esperimenti sono state quelle di:

- Contrabbasso
- Spazzolata di rullante
- Colpo di rullante
- Colpo di cassa

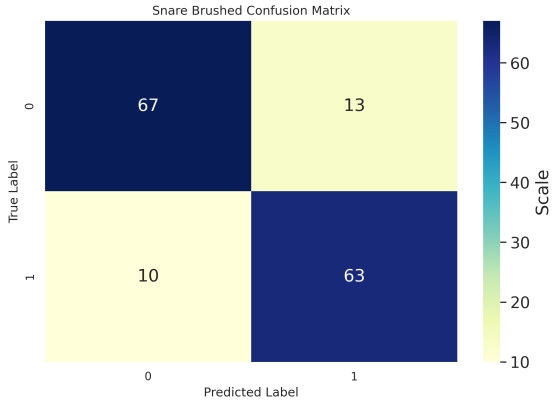
3.1 Prima fase

In questa fase lo scopo è stato appunto comprendere se una macchina con le features selezionate fosse in grado di riconoscere la differenza tra nota e non nota. Una volta creato il file arff contenente le informazioni necessarie tramite un apposito

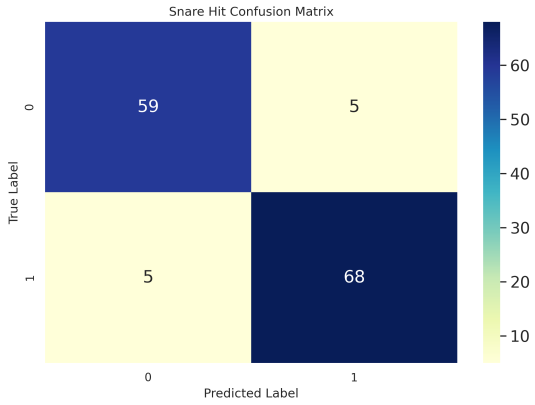
script python lo stesso file è stato usato per costruire l'albero decisionale utilizzando l'algoritmo J48 (versione più fine di ID3 descritto nel Capitolo 2) ottenendo come risultato le seguenti matrici di confusione[1] per gli strumenti sopra elencati:



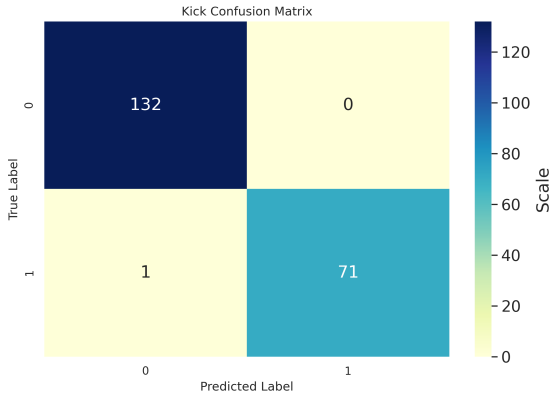
(a) Matrice di confusione per il basso



(b) Matrice di confusione per il rullante spazzolato



(c) Matrice di confusione per il colpo di rullante



(d) Matrice di confusione per la cassa

Figura 3.1: Matrici di confusione della prima fase

È possibile notare degli ottimi risultati per tutti gli strumenti nel riconoscimento, in particolar modo per la cassa e il colpo di rullante, vista la natura dell'onda sonora da loro prodotta, un'impulso con una crescita molto rapida. Lo strumento

[1]Le colonne di queste matrici di confusione indicano il valore atteso, sulle righe vi sono invece i valori risultanti

che è riconosciuto peggio è invece il rullante spazzolato, anch'esso per la natura dell'onda sonora da lui prodotto, distinguibile meno facilmente da un possibile rumore per ampiezza e impulso con crescita molto minore.
Essendo questi risultati molto buoni, è possibile affermare che con le features utilizzate risulta possibile distinguere tra una nota e una non nota.

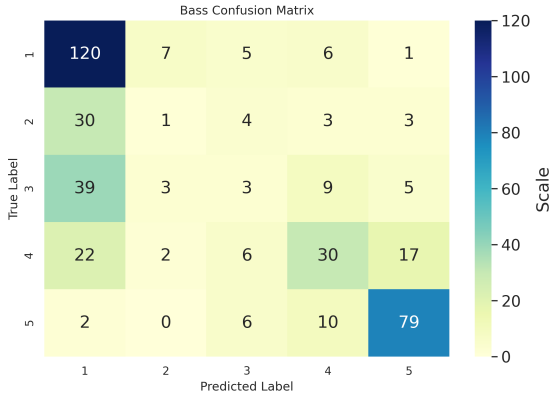
3.2 Seconda fase

In questa fase si vuole rendere più fine il riconoscimento delle note, consentendo di accettare in input un'intera traccia audio e non solo dei singoli frammenti di note o non per il riconoscimento. Partendo dalla classificazione, viene sempre utilizzato J48 per costruire un albero di decisione che non porterà più a decidere se un frammento analizzato è una nota o meno, questo infatti porterà a classificare in 5 classi diverse ogni elemento di input, come appunto descritto a inizio capitolo. L'algoritmo di classificazione è molto simile al passo precedente, l'unica differenza è che vengono scelti dei campioni più lunghi per poter essere spezzati in più sub-campioni, il primo classificato come 5 (certamente una nota), l'ultimo come 1 (certamente non una nota). Ogni sub-campione avrà la stessa lunghezza, e proprio così facendo è stato possibile fornire in input delle intere tracce, perché suddivise in sub-campioni, di lunghezza costante, al quale si associa a ognuno una classe da 1 a 5 che consente così di localizzare temporalmente le note, e risolvere dunque il problema posto inizialmente.

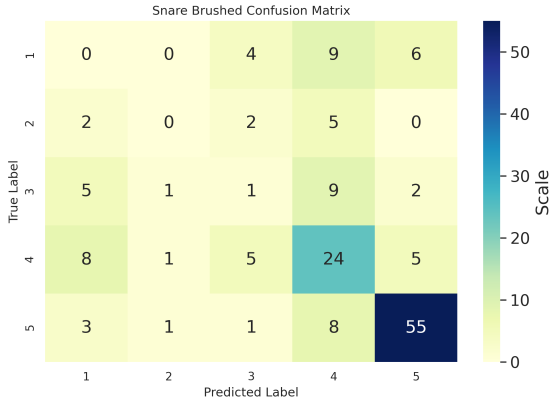
Questi risultati sono stati ottenuti effettuando una classificazione dove ogni sub campione è esattamente 8820 campioni, quindi 0.2s, per il campione successivo si esegue uno shift di 4410 campioni, ottenendo 0.1s di granularità e di overlap[2]

Nei risultati ottenuti è possibile notare un peggioramento delle predizioni in modo particolare nel rullante, precisamente con un 51% di istanze classificate correttamente per il rullante spazzolato e un 41% per il colpo semplice di rullante. Con la cassa è stato ottenuto invece quello che è il risultato migiore con l'83% di

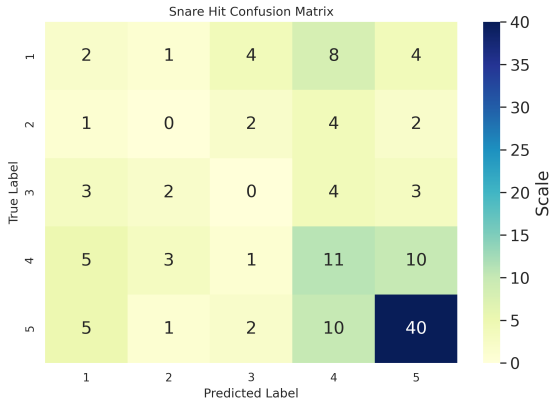
[2]Il tempo di sovrapponimento tra un sub-campione e l'altro



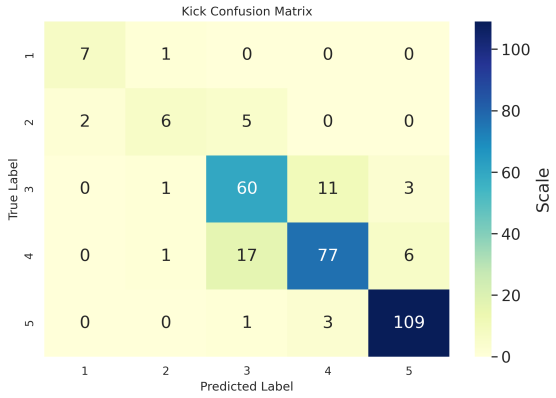
(a) Matrice di confusione per il basso



(b) Matrice di confusione per il rullante spazzolato



(c) Matrice di confusione per il colpo di rullante



(d) Matrice di confusione per la cassa

Figura 3.2: Matrici di confusione della prima fase

istanze classificate in modo corretto. Questa così ampia divergenza, così come per i risultati ottenuti precedentemente, è data dalla forma e dell'onda sonora in se. Infatti la cassa genera un onda con transiente molto marcato, e facilmente riconoscibile con le features utilizzate, che decade in modo molto rapido differentemente da quanto invece succede con il rullante, in particolar modo quello spazzolato. Un altro aspetto che influisce sull'abilità di riconoscere o meno un'onda sonora da un rumore è dato dall'intensità del segnale generato, infatti il rullante spazzolato genera un'onda con un ampiezza molto inferiore, difficilmente distinguibile da quello che è il rumore. L'ultimo aspetto ma non meno importante riguarda come sono

stati selezionati i campioni per allenare la macchina, la numerosità dei campioni estratti e la qualità delle registrazioni utilizzate, tutti questi parametri possono influenzare pesantemente quelli che sono i risultati. Un numero maggiore di campioni potrebbe insegnare in modo migliore alla macchina la quale sarà in grado di costruire un albero di decisione migliore, come anche delle registrazioni migliori o con volumi maggiori per alcuni strumenti dove possibile.

Bibliografia

[1] Groove, "Groove (music) — Wikipedia, the free encyclopedia," 2021.

[2] C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch, "A review of automatic drum transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1457–1483, 2018.

[3] M. Wright and E. Berdahl, "Towards machine learning of expressive microtiming in brazilian drumming," 2006.

[4] N. J. Nilsson, "Introduction to machine learning. an early draft of a proposed textbook," 1996.