Università degli Studi di Ferrara

UNIVERSITÀ DEGLI STUDI DI FERRARA

CORSO DI LAUREA IN INFORMATICA

Machine learning per lo studio del groove musicale: un primo approccio

Relatore:

Prof. Luciano FADIGA

Correlatore:

Prof. Guido SCIAVICCO

Laureando:

Luca GREGGIO

ANNO ACCADEMICO 2020 – 2021

Indice

Introduzione

Elementi essenziali nella musica sono 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.
Questa carica di emozioni sembrerebbe essere data da un'imperfetta sincronia tra i musicisti, i quali parrebbero suonare non perfettamente a tempo. I musicisti stessi suonano le singole note introducendo delle micro-variazioni di tempo (millisecondi), originando in tal modo, del *movimento* all'interno del brano musicale. Questo fenomeno è chiamato *microtiming*, e questo, potrebbe essere alla base del groove.
Partendo da questi presupposti, che il mio tutore di tirocinio aveva provveduto ad illustratimi, ho lavorato alla costruzione di un applicativo che potesse costituire uno strumento utile all'analisi di tale fenomeno. 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'attenta analisi, il percorso scelto per la risoluzione è stato quello dell'intelligenza artificiale e precisamente il ML (machine learning), tecnica molto usata in questo ambito per approcciarsi a questa ambito. Lo scopo è stato dunque, quello di 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 che facilitasse l'utilizzo a utenti meno esperti nell'utilizzo della linea di comando, dunque creare una GUI (Graphical User Interface), realizzata anch'essa in linguaggio Python utilizzando un framework chiamato Qt.

Oggi giorno esiste una quantità innumerabile di soluzioni e programmi che consentono di portare a termine i compiti più disparati e, proprio per questo, il primo passo posto ancora prima di cominciare è stato quello di documentarsi a riguardo, cercando se esistessero già dei SW in grado di svolgere ciò che mi è stato richiesto e, una volta verificato che nessuna delle soluzioni esistenti soddisfaceva il mio tutore ho intrapreso il percorso di documentazione per la realizzazione del SW. Il fenomeno è molto noto nell'ambito musicale ma, sfortunatamente, la letteratura in materia, nonostante sia vasta per il lato informatico è poco sviluppata. Gli articoli fino ad ora pubblicati hanno permesso di comprendere come affrontare il problema, utilizzando appunto il ML. Ciò nonostante le informazioni presenti non erano sufficienti, soprattutto per un utente inesperto in questa branca. Non vi erano codici o spiegazioni dettagliate e, proprio per questo motivo, è stato deciso di tentare un primo approccio alla risoluzione di questo problema con il codice reso disponibile a chi volesse approfondire alla mia pagina github partendo da zero

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

Tabella 1.1: Data-set di esempio iniziale.

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.
Si osserva il tempo su tutto il set per cominciare
$I(tempo, D) = \frac{|D(tempo=sole)|}{|D|} + \frac{|D(tempo=pioggia)|}{|D|} + \frac{|D(tempo=nuvoloso)|}{|D|}$

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

$= \frac{5}{12}I(D(tempo = sole)) + \frac{4}{12}I(D(tempo = pioggia)) + \frac{3}{12}I(D(tempo = nuvoloso))$
$= \frac{5}{12}E\left(\frac{2}{5}; \frac{3}{5}\right) + \frac{4}{12}E\left(\frac{3}{4}; \frac{1}{4}\right) + \frac{3}{12}E\left(\frac{2}{3}; \frac{1}{3}\right) = 0.74$
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

Tabella 1.2: caption.

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

Tabella 1.3: caption.

Tempo	**Temperatura**	**Umidità**	**Vento**	**Si Gioca**
nuvoloso	caldo	alta	no	si
nuvoloso	freddo	normale	si	no
nuvoloso	media	alta	si	si

Tabella 1.4: caption.

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 albero1.1, 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.

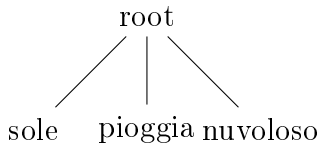


Figura 1.1: DT per il data-set di esempio

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

[7] Attribute-Relation File Format è un 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.

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.
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: $TPR = \frac{TP}{P}$ e $TNR = \frac{TN}{N}$. In aggiunta a questi parametri per comprendere l'efficacia di un classificatore vengono utilizzati anche la *Precision* ottenuta come $PPV = \frac{TP}{TP+FP}$ ovvero la proporzione tra il numero di classi effettivamente appartenenti a una classe sopra il totale delle istanze classificati come tale, il *recall* invece, analogo alla sensitivisty, ritorna la proporzione tra istanze classificate correttamente e diviso l'effettivo numero di valori che dovrebbero essere contenuti in quella classe. Gli ultimi due parametri rilevanti utilizzati per comprendere la qualità di una classificazione sono *F-Measure* avente scopo di rappresentare i livello di accuracy in un test misurata come $F - Measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$, e *ROC Area* un valore compreso tra 0, nel caso pessimo, e 1, nel caso di un classificatore perfetto, rappresentante l'area sottostante a una curva data dal plot del TPR contro il *False Positive Rate* (FPR) ovvero $FPR = 1 - TNR$, il valore dell'area in un solo valore è in grado di indicare la qualità del classificatore in modo molto esaustivo. Importante è che l'accuracy viene calcolata una volta per ogni relazione, invece per quanto riguarda i parametri elencati in seguito, devono essere questi calcolati per ogni classe presente nella relazione.

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 di esempio 2.1 l'input che riceve un ADC

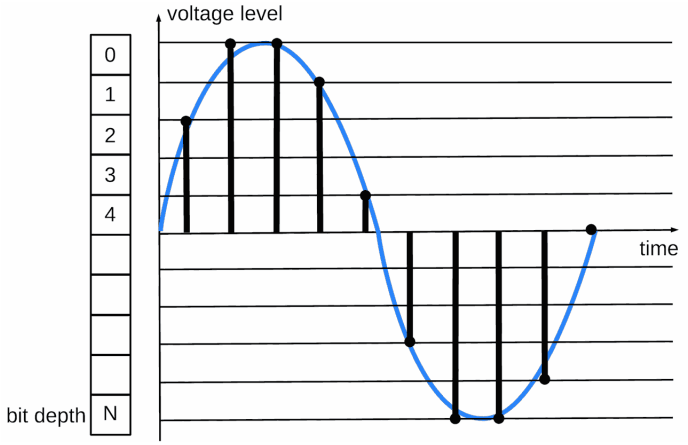


Figura 2.1: Conversione segnale audio analogico digitale

è un segnale 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 il

[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.

segnale si avvicinava maggiormente. 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. I valori di SR e bit depth sono solitamente limitati dall'hardware, in questo caso la scheda audio.

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 che dovrebbe permettere 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
- Rullante spazzolato
- 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 è 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 matrici di confusione[1] per gli strumenti sopra elencati in figura 3.1

I campioni da classificare sono stati divisi in due classi: "si" e "no" identificanti note e non note. Nella tabella 3.1 si può osservare la numerosità di campioni per ogni classe, la classe con meno campioni è quella del colpo di rullante vista la natura delle tracce audio, è stato complesso estrarre dei campioni adatti vista la natura dello strumento e della registrazione, infatti le note erano spesso molto vicini tra loro e confusi oltre che essere molto inferiori rispetto a quelle degli altri strumenti.

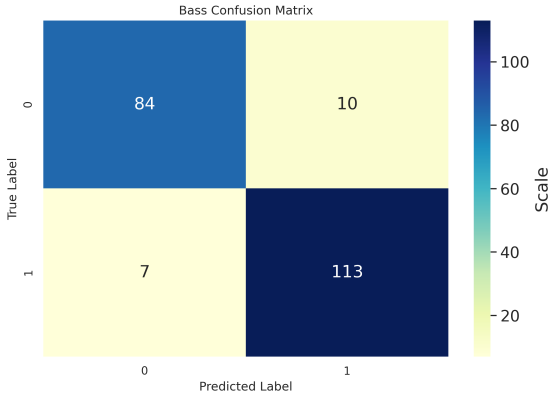
Classe	**Si**	**No**
Basso	121	94
Rullante Spazzolato	80	73
Colpo di Rullante	64	73
Cassa	132	72

Tabella 3.1: Numerosità campioni prima fase.

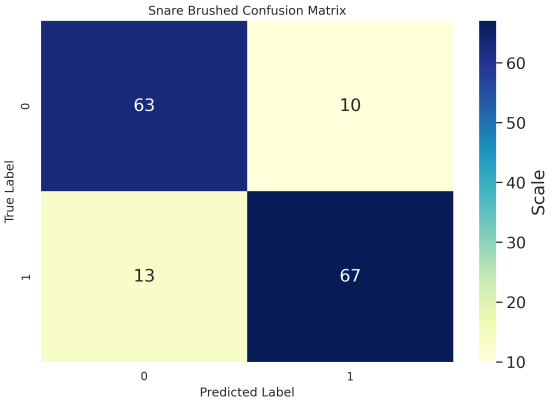
Gli esperimenti sono stati condotti alla stessa maniera utilizzando l'approccio di cross validation con un $k = 10$ numero di sotto insiemi in cui viene diviso il data-set 1.3. Osservando dunque numerosità dei dati e i risultati delle matrici di confusione possiamo ora valutare la qualità dei risultati ottenuti calcolando i parametri presentati nel paragrafo 1.3.

Procedendo per ordine osservando il basso si ha un'accuracy pari a 0.92 seguita dai valori illustrati nella tabella 3.2. In seguito sono presenti le misure delle due tipologie di note provocate dal rullante, spazzolato e colpo normale, avneit accuracy rispettivamente di 0.84 e 0.92 con i valori presenti nelle tabelle 3.3 e 3.4. Per quanto riguarda i risultati ottenuti dalla cassa sono un accuracy pari a 0.99 e la tabella 3.5 mostra i valori a essa relativi.

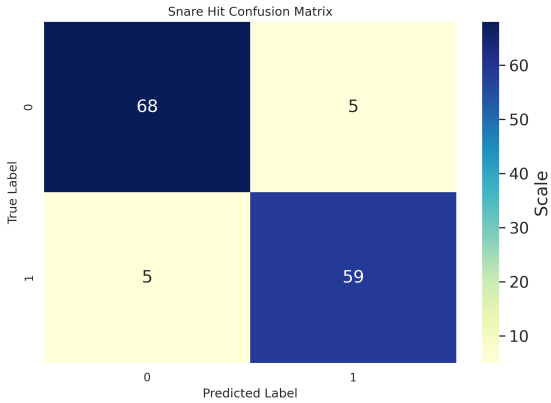
[1]Le colonne di queste matrici di confusione indicano il valore atteso, sulle righe vi sono invece i valori risultanti



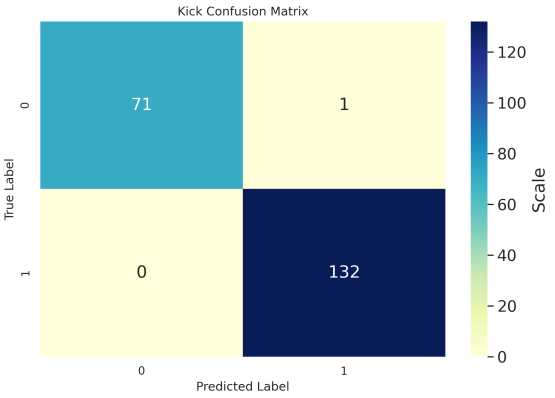
(a) Matrice di confusione per il basso



(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

Basso	**TPR**	**FPR**	**Precision**	**Recall**	**F-Measure**	**ROC Area**
si	0,942	0,106	0,919	0,942	0,931	0,946
no	0,894	0,058	0,923	0,894	0,908	0,946

Tabella 3.2: Risultati basso per la 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 sono questi gli

Rull. Spazz.	TPR	FPR	Precision	Recall	F-Measure	ROC Area
si	0,838	0,137	0,870	0,838	0,854	0,861
no	0,863	0,163	0,163	0,863	0,846	0,861

Tabella 3.3: Risultati rullante spazzolato per la prima fase.

Rull. Colpo	TPR	FPR	Precision	Recall	F-Measure	ROC Area
si	0,922	0,068	0,922	0,922	0,922	0,939
no	0,932	0,078	0,932	0,932	0,932	0,939

Tabella 3.4: Risultati colpo di rullante per la prima fase.

Cassa	TPR	FPR	Precision	Recall	F-Measure	ROC Area
si	1,000	0,014	0,992	1,000	0,996	0,993
no	0,986	0,000	1,000	0,986	0,993	0,993

Tabella 3.5: Risultati colpo di cassa per la prima fase.

strumenti più semplici da riconoscere per la macchina. Lo strumento che è riconosciuto peggio è invece il rullante spazzolato, anch'esso per la natura dell'onda sonora da lui prodotto, meno distinguibile da un possibile rumore per ampiezza d'onda e impulso con crescita molto minore.
Essendo questi risultati molto buoni, soprattutto per quanto riguarda la ROC Area, è possibile affermare che con le features utilizzate risulta possibile distinguere tra una nota e una non nota, e dunque passare a una fase di riconoscimento più fine illustrata nella sezione che segue.

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 utilizza-

to 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.

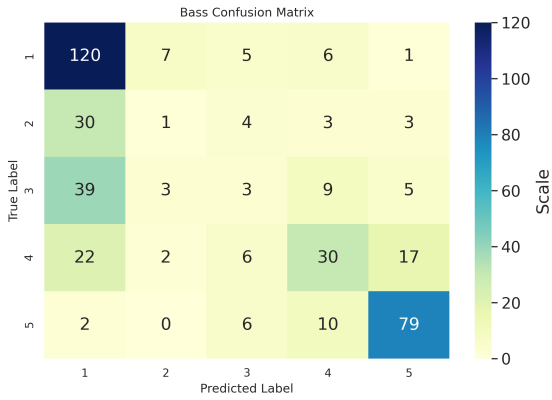
Questi risultati sono stati ottenuti effettuando una classificazione dove ogni sub campione è esattamente 8820 campioni, quindi 0.2s, per il campione successivo si esegue uno shift di 4410 campioni, ottenendo 0.1s di granularità e di overlap[2]. Anche qui in modo analogo alla prima fase il DT viene condotto l'esperimento con un k=10 numero di sotto-insiemi in cui viene diviso il data-set. Possiamo vedere la numerosità dei campioni selezionati dalle tracce audio nella tabella 3.6 e la numerosità delle classi da 1 a 5 per i sub-campioni derivati dai campioni nella tabella 3.7

Classe	**Campioni**
Basso	103
Rullante Spazzolato	115
Colpo di Rullante	146
Cassa	119

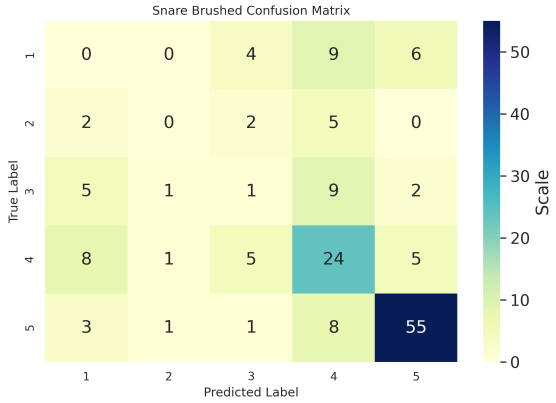
Tabella 3.6: Numerosità campioni seconda fase.

I numeri dei campioni audio selezionati nella prima tabella sono molto omogenei, ma data la natura delle registrazioni e degli strumenti oltre al modo in cui questi vengono suonati porta ad avere nella seconda tabella valori molto eterogenei di sub-campioni per ogni classe. Osserviamo infatti che il per il basso e la cassa

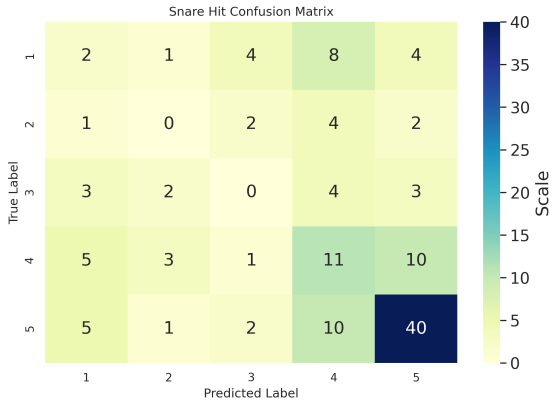
[2]Il tempo di sovrapponimento tra un sub-campione e l'altro



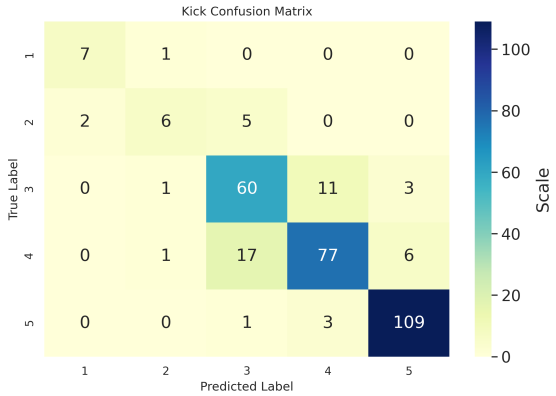
(a) Matrice di confusione per il basso



(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 seconda fase

Classe	**1**	**2**	**3**	**4**	**5**
Basso	139	41	59	77	97
Rullante Spazzolato	19	9	18	43	68
Colpo di Rullante	20	9	13	30	58
Cassa	8	13	75	101	113

Tabella 3.7: Numerosità campioni seconda fase.

si possiedono il maggior numero di sub-campioni a differenza delle due tipologie di rullanti, con un numero molto minore a confronto. Il rullante è uno strumento che in questo contesto produce note molto ravvicinate tra loro e per questo motivo molti campioni non possono essere usati perché troppo corti rispetto alla dimensione che è stata utilizzata per l'esperimento, lo stesso vale per i sotto campioni, questi infatti sono molto meno dal momento che effettuando dei tagli di campioni che già di natura sono corti, il risultato è di conseguenza un numero inferiore di sotto-campioni, in particolar modo appartenenti alla classe "1", maggiore è il numero invece per la classe "5", appunto per come vengono etichettati si sub-campioni.

I risultati ottenuti mostrano un'accuracy pari a 0.56 per il basso, un valore molto inferiore rispetto al risultato più che positivo ottenuto nella prima fase. La tabella 3.8 ci mostra in modo più preciso che cosa succede per le singole classi, da subito si possono notare dei risultati molto positivi nel riconoscimento delle classi 5 e 1 con dei TPR vicini a 0.9, accompagnati da valori positivi, anche rispetto a quelli delle altre classi d ROC Area. La prestazione peggiore è osservata invece per la classe 3, sembra che crei molta confusione per il riconoscimento. I risultati analizzati in seguito sono quelli del rullante spazzolato, e del colpo di rullante, i risultati peggiori in assoluto in questo set di strumenti con accuracy rispettivamente di 0.50 e 0.41, questa imprecisione molto elevata è probabile che sia originata dal modo in cui vengono estratti i campioni, oltre agli aspetti definiti nella prima fase. Le rispettive tabelle 3.9 e 3.10 mostrano in modo molto chiaro il peggioramento, ma solo per alcune classi. Partendo dal TPR di tutte e due si può evincere l'incapacità della macchina nel riconoscere le classi 1 e 2, questo probabilmente dato da numero di sub-campioni troppo piccolo, come anche dalla natura del suono da esso generato nella prima fase. Quello che è però il TPR delle classi 5 dei due rullanti, si vede un miglioramento, in particolar modo per quanto riguarda il rullante spazzolato, con un valore che si avvicina molto di più a 1 di quanto le altre classi non facciano. Per comprendere meglio la qualità dei classificatori ottenuti consideriamo la ROC Area, che per la classe 5 del rullante spazzolato ha un valore molto buono, che va però a scendere con lo scendere delle classi. In modo simile si comporta il classificatore per il colpo di rullante, con la differenza però che la classe con il risultato peggiore è la 3. In fine per quanto riguarda la cassa è stato ottenuto un risultato più che positivo con un valore di accuracy pari a 0.83, il

valore migliore per questa fase, accompagnato da metriche di valutazione molto buone presentate nella tabella 3.11 partendo da un valore di TPR molto buono per la classe, leggermente inferiore per le altre, con un esito invece non ideale nella classe 2. Portiamo maggiore attenzione alla ROC area, con valori molto vicini a 1, mostrando che il classificatore ottenuto è in grado di lavorare molto bene. Anche in questo caso influiscono, però positivamente, le caratteristiche dell'onda sonora prodotta dalla cassa discusse precedentemente.

Basso	**TPR**	**FPR**	**Precision**	**Recall**	**F-Measure**	**ROC Area**
1	0,863	0,339	0,563	0,863	0,682	0,790
2	0,024	0,032	0,077	0,024	0,037	0,604
3	0,051	0,059	0,125	0,051	0,072	0,609
4	0,390	0,083	0,517	0,390	0,444	0,676
5	0,814	0,082	0,752	0,814	0,782	0,890

Tabella 3.8: Risultati basso per la seconda fase.

Rull. Spazz.	**TPR**	**FPR**	**Precision**	**Recall**	**F-Measure**	**ROC Area**
1	0,000	0,130	0,000	0,000	0,000	0,479
2	0,000	0,020	0,000	0,000	0,000	0,531
3	0,056	0,086	0,077	0,056	0,065	0,509
4	0,558	0,272	0,436	0,558	0,490	0,695
5	0,809	0,146	0,809	0,809	0,809	0,856

Tabella 3.9: Risultati rullante spazzolato per la seconda fase.

A questo punto si hanno due classificatori, uno utilizzato nella prima fase, e uno per la seconda. I risultati ottenuti nella prima fase sono più che positivi, mostrano un approccio al problema corretto, a differenza della seconda fase. Questa seconda è stata trattata come un problema di classificazione, probabilmente un approccio non sufficientemente adeguato al tipo di problema considerato tanto quanto lo sarebbe un approccio per regressione. Oltre a migliorare il riconoscimento, uno degli

Rull. Colpo	TPR	FPR	Precision	Recall	F-Measure	ROC Area
1	0,105	0,128	0,125	0,105	0,114	0,556
2	0,000	0,059	0,000	0,000	0,000	0,536
3	0,000	0,078	0,000	0,000	0,000	0,388
4	0,367	0,265	0,297	0,367	0,328	0,505
5	0,690	0,271	0,678	0,690	0,684	0,693

Tabella 3.10: Risultati colpo di rullante per la seconda fase.

Cassa	TPR	FPR	Precision	Recall	F-Measure	ROC Area
1	0,875	0,007	0,778	0,875	0,824	0,996
2	0,462	0,010	0,667	0,462	0,545	0,898
3	0,800	0,098	0,723	0,800	0,759	0,899
4	0,762	0,067	0,846	0,762	0,802	0,910
5	0,965	0,046	0,924	0,965	0,944	0,972

Tabella 3.11: Risultati colpo di cassa per la seconda fase.

obiettivi è di portare la granularità al millesimo di secondo in futuro, dal momento che in musica, quelli che sono intervalli di tempo per noi brevi, come 1/10 di secondo, troppo ampi, all'interno di essi possono occorrere un numero molto elevato di eventi. Gli strumenti che sono ritenuti interessanti per gli studi da condurre sono principalmente il basso e la cassa, che come visto, vengono riconosciuti in modo discreto, il rullante, soprattutto quello spazzolato portano delle difficoltà non indifferenti che devono essere superate.
Avendo a questo punto un sistema funzionante, il passo successivo è stato la costruzione di una *graphical user interface* (GUI) presentata nel capitolo seguente.

Capitolo 4

GUI

Il passo successivo alla costruzione del sistema intelligente è stato lo sviluppo della GUI da rendere l'utilizzo più facile a utenti non esperti da riga di comando, facilitando quello che è il workflow.
L'interfaccia è stata costruita in modo tale da avere un'unica pagina con molteplici tab, posizionate in ordine di utilizzo, per rendere meno dispersivo e più intuitivo possibile il sistema creato. Le sezioni in ordine sono le seguenti:

- Home
- Classify
- Predict
- Merge
- Settings

Come specificato nell'introduzione per questa parte si è utilizzato il linguaggio di programmazione Python con un modulo per la realizzazione della grafica chiamato PyQt. Tale modulo altro non è se non un binding del noto framework cross-platform QT per C++. A differenza del linguaggio appena citato, Python non gode della stessa velocità di esecuzione dal momento che si tratta di un linguaggio interpretato, ma ha come punto di forza la facilità di sviluppo.

4.1 Home

Appena avviato il programma si presenta nel modo seguente fornendo una breve introduzione su che cosa sia il programma in utilizzo e illustrando brevemente le sue modalità di lavoro.

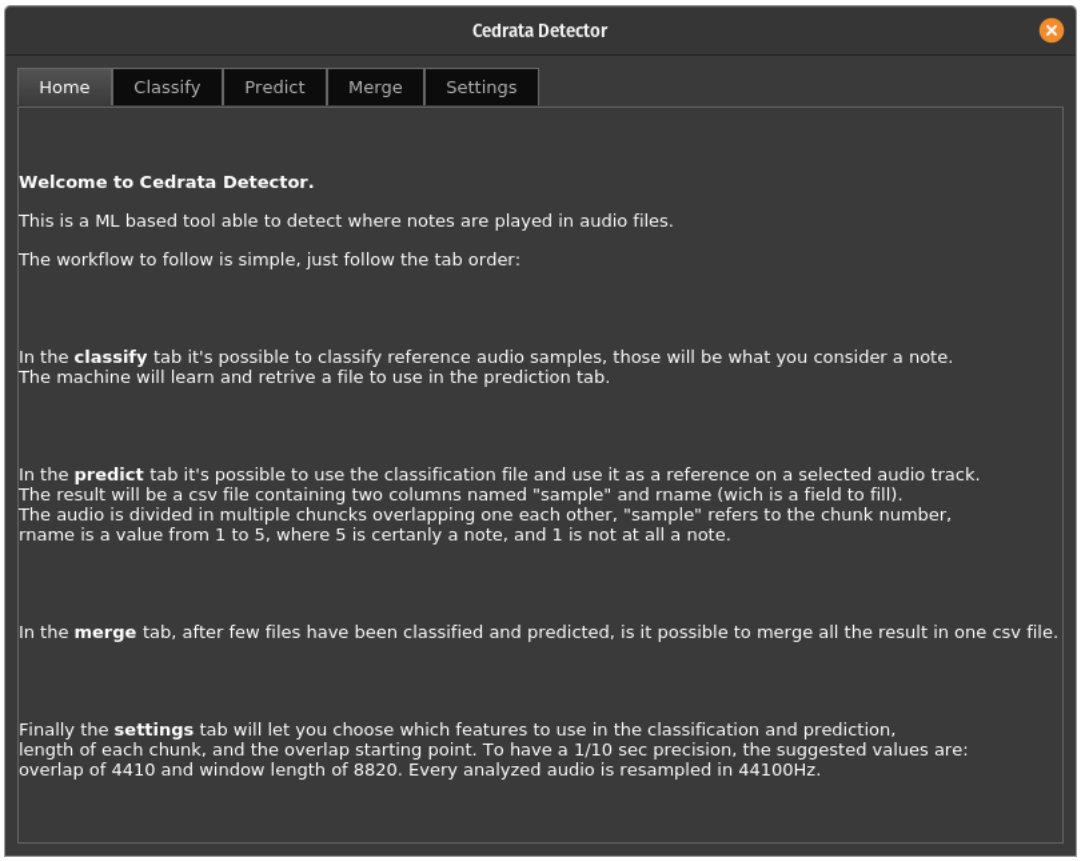


Figura 4.1: Schermata di home

4.2 Classify

Procedendo con quello che l'ordine delle sezioni arriviamo alla prima vera e propria sezione, quella per lo svolgimento della classificazione, dove viene costruito un file ARFF che verrà, nel punto successivo, utilizzato per la costruzione di un classificatore. Sono richiesti in input: una cartella contenente tutti i campioni

di note per cui si vorranno estrarre le features, il nome della relazione, in questo caso lo strumento di cui si stanno effettuando le estrazioni, e in fine una cartella di destinazione per salvare il file ARFF creato avente come nome il nome della relazione.

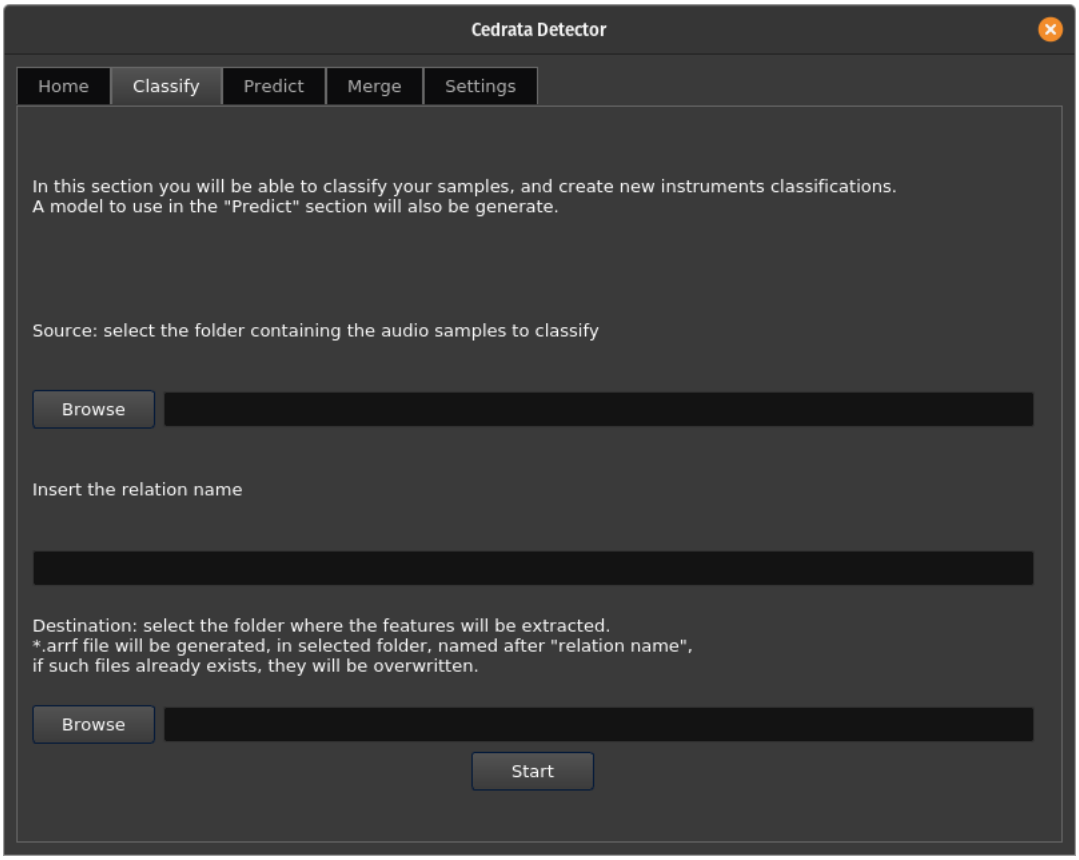


Figura 4.2: Schermata di home

4.3 Predict

A questo punto l'utilizzatore avrà ottenuto un file ARFF dalla sezione precedente, tale viene richiesto in input per costruire un classificatore assieme a un file audio nei formati AIFF o WAV. Per terminare sarà necessario indicare la directory dove salvare i risultatati scritti in un file di tipo CSV (Comma-Separated value) il quale presenterà due colonne, sub-sample e note. Le tracce audio verranno infatti

trattate allo stesso modo di quelle usate per la classificazione, creando così dei sub-sample, di lunghezza e overlap specificati nelle impostazioni del software, con i rispettivi valori delle features associati. A ogni sub-sample verrà associata una classe di nota, da 1 a 5, e quindi stampata nel file CSV risultante.

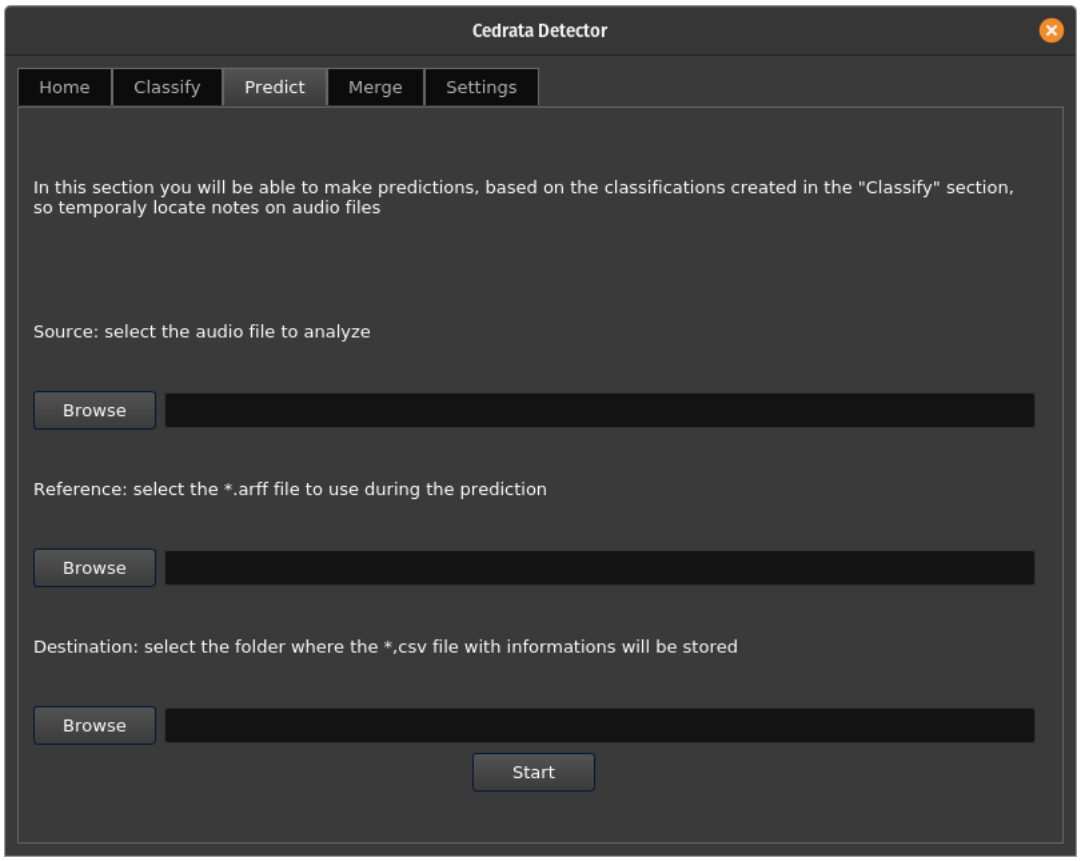


Figura 4.3: Schermata di home

4.4 Merge

Questo progetto ha come obiettivo il fornire un modo per confrontare il tempismo tra i diversi strumenti e musicisti, ma i risultati ottenuti ora forniscono una visione per i singoli strumenti. Per avere una panoramica la soluzione adottata è stata quella di effettuare un merge di molteplici file, in un unico CSV. Questo è il compito di questa tab, è dunque , selezionare più risultati CSV di strumenti diversi

appartenenti che suonano la stessa canzone, e specificare dove si vuole salvare il risultato. A questo punto si avrà un file contenente la colonna dei sample e per ogni strumento a ogni sample viene associato il valore da 1 a 5.

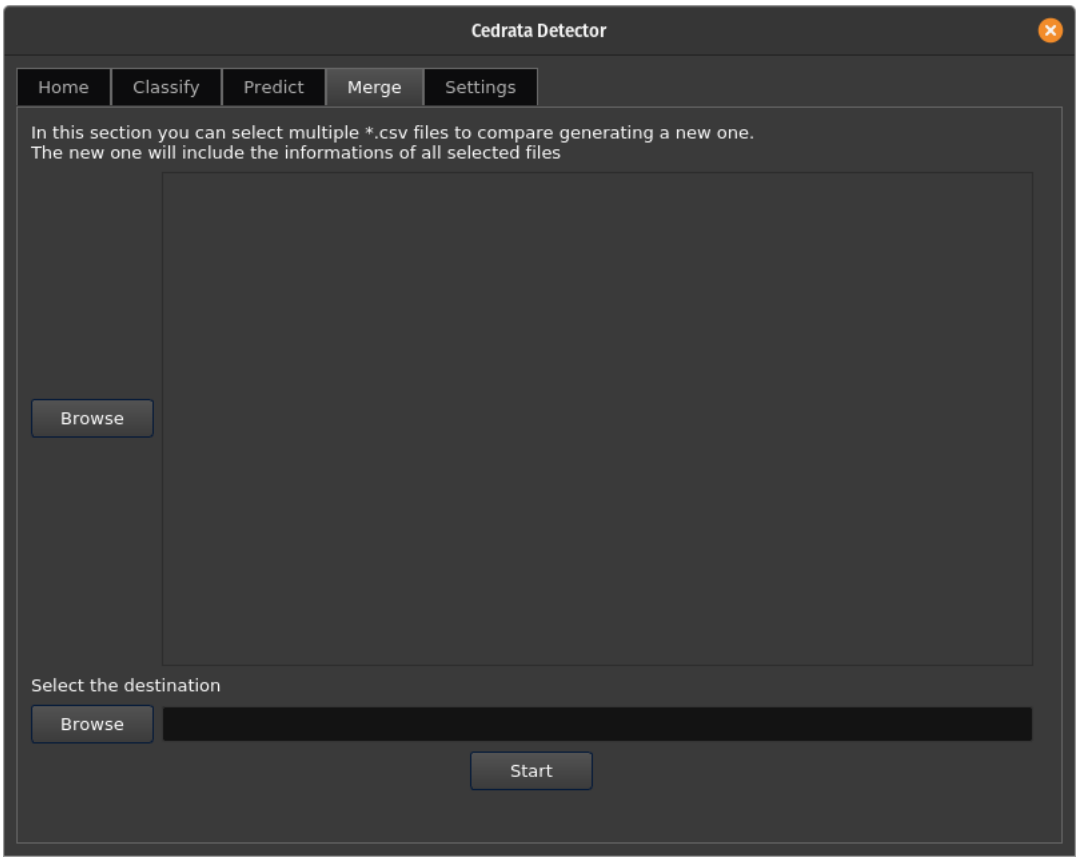


Figura 4.4: Schermata di home

4.5 Settings

A differenza degli esperimenti, nei quali sono stati utilizzati dei parametri di riferimento descritti nel terzo capitolo, è fondamentale che un utente possa provare differenti configurazioni e impostazioni per portare a termine la configurazione e predizione. Per le due sezioni appena citate sono presenti due gruppi di impostazioni separati con gli stessi elementi: la selezione, tramite checkbox, delle features da utilizzare (media, varianza, media della derivata prima e seconda) la selezione

della lunghezza di ogni sub-sample e il tempo di overlap che si vuole tra un sub-sample e l'altro. Sono stati inseriti poi i bottoni per applicare le impostazioni e ripristinare a default.

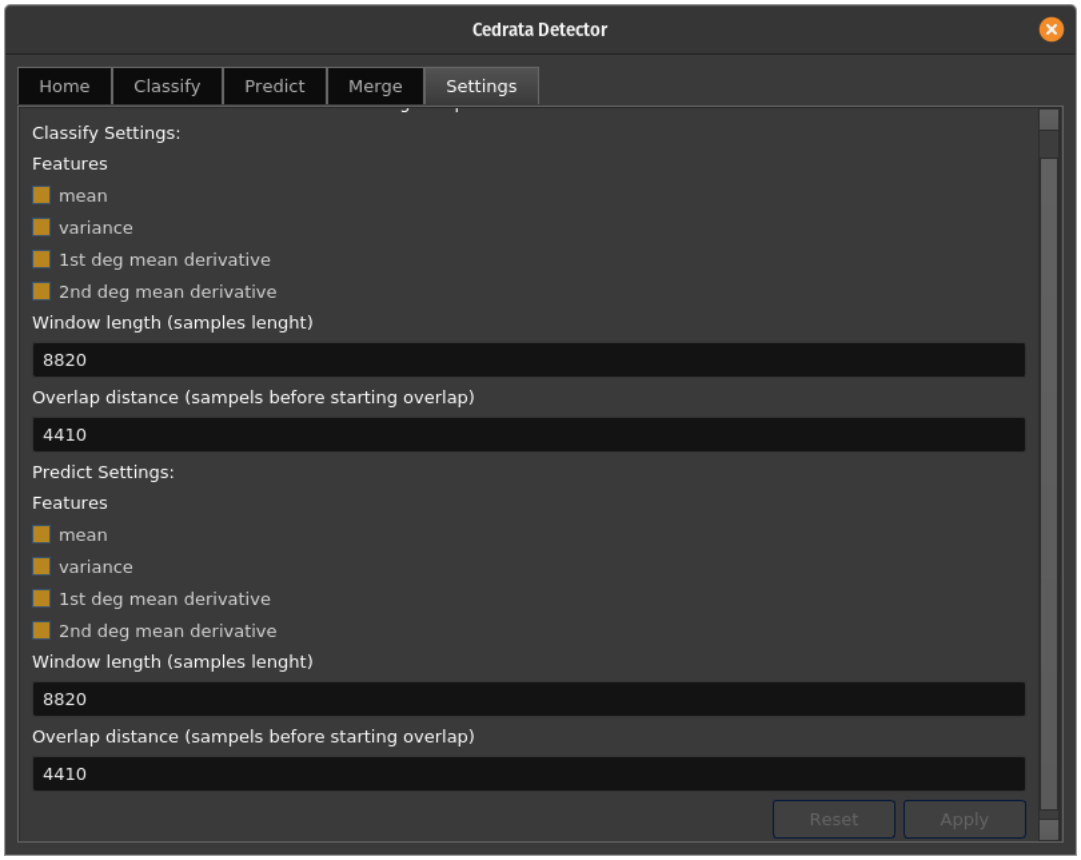


Figura 4.5: Schermata di home

Conclusioni

In questa tesi l'obiettivo è stato fornire un primo approccio, come scritto nell'introduzione, all'utilizzo del ML per la costruzione di uno strumento che permettesse di effettuare studi sul groove musicale provando dunque a insegnare a una macchina la differenza tra una nota da una non-nota, per poterle collocare temporalmente permettendo di svolgere un'analisi tra la correlazione che esiste tra i vari strumenti e come i musicisti si incastrino temporalmente tra loro creando una sensazione di inerzia nella musica. Dopo aver analizzato il problema e trovato un modo teorico per la risoluzione, si è passati alla pratica con l'esperimento descritto nel terzo capitolo agendo dunque in due fasi: la prima nella quale si prova a comprendere l'abilità di una macchina di portare a termine questo tipo di riconoscimento, e una seconda nella quale viene tentato un approccio più fino. I risultati che sono stati ottenuti nella prima parte dell'esperimento sono molto incoraggianti, vediamo infatti che si tratta di un problema di classificazione come intuito inizialmente dal momento che l'output deve essere un valore di tipo discreto "si" o "no" e i numeri ottenuti supportano quest'affermazione. In particolar modo notiamo dei valori di TPR e di ROC Area superiori addirittura al 90% mostrando dunque un'elevata efficacia dei classificatori costruiti per i diversi strumenti.
Per quanto riguarda la seconda parte dell'esperimento risulta evidente che il problema non è stato trattato nel modo ideale, e differentemente da quanto ipotizzato inizialmente si tratta di un problema di regressione e non di uno di classificazione. Analizzando i risultati ottenuti si nota immediatamente che i valori ricavati sono decisamente più eterogenei rispetto alla prima fase dell'esperimento con dei valori di ROC Area che variano da circa il 0.5 fino al 0.9 circa, il che implica un algoritmo poco consistente che non permette di trovare classificatori con performance buone in modo consistente al variare dello strumento analizzato. In modo errato è stato

deciso di assegnare delle classi con valori da 1 a 5 rappresentanti una percentuale di essere nota o meno, ma questo è chiaramente sconveniente, sarebbe possibile trattare il problema come un tipo di regressione, e probabilmente già questo potrebbe contribuire a un miglioramento del programma, assieme ad altre features e strumenti di analisi del suono più precisi che consentano di riconoscere la timbrica e altri aspetti legati al suono.
Ciò detto, come appunto ripetuto più volte, questa vuole essere la presentazione di un primo approccio che utilizza il ML in questo modo che si spera possa essere migliorato in futuro.

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] M. R. L. W. P. B. H. C. C. S. E. J. P. B. Magdalena Fuentes, Lucas S. Maia, "Tracking beats and microtiming in afro-latin american music using conditional random fields and deep learning."

[5] G. M. Kahl Hellmer, "Quantifying microtiming patterning and variability in drum kit recordings: a method and some data."

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