UNIVERSITAT DE BARCELONA

Treball de Fi de Grau (TFG)

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica

Universitat de Barcelona

Comparació entre diferents sistemes d'extracció de característiques per la classificació d'Instruments (MIR)

Joaquim Comes

Director: Sergio Escalera

Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 24 de gener de 2023

Index

Abstract

In this Final Degree Project, we conducted an experiment comparing two sound feature extraction methods, the first of which is based on the Mel scale (MFCC), and the second on Wavelet Transforms (WTCC). The experiment will determine which system is better at categorizing musical instruments, a job that may be extremely useful in areas such as audio post-production, recommendation algorithms, music analysis, and so on.

This study is a continuation of Angel Bergantiños and Chan Yoon's TFG, in which algorithms were constructed to classify musical songs based on their musical genre using supervised and unsupervised machine learning approaches, respectively.

Resum

En aquest Treball de Final de Grau s'ha realitzat un experiment on s'han comparat dos sistemes d'extracció de característiques del so, el primer, el més comú de tots basat en l'escala de Mel (MFCC), el segon, per altra part, està basat amb les transformades de Wavelet (WTCC). L'experiment se centrarà en quin sistema és millor a l'hora de classificar instruments musicals, tasca que pot ajudar molt en tasques com la postproducció d'àudio, algoritmes de recomanació, anàlisi musical, etc.

Aquest treball és la continuació dels TFG dels Angel Bergantiños i del Chan Yoon en els quals es van fer realitzar sistemes en els quals es podien classificar pistes musicals segons el seu gènere musical amb mètodes d'aprenentatge automàtic de forma supervisada i no-supervisada respectivament.

Capítol 1: Introducció

1.1 Introducció

La música és probablement un dels elements culturals que ha acompanyat la humanitat en tots els moments i certs instruments han aconseguit que la música adquireixi molt més protagonisme en les nostres vides, fins al punt que hi ha certs gèneres musicals o inclús moviments culturals que s'identifiquen amb un instrument com pot ser el cas de la guitarra elèctrica en el Rock & Roll..

Avui en dia existeixen una gran varietat d'instruments, els quals poden sonar de formes molt diverses i com que no estan lligades als gèneres, es poden utilitzar en multitud d'àmbits, a més que el nombre d'instruments està en constant creixent.

A causa de la quantitat de combinacions d'instruments que poden existir i la necessitat en certs àmbits com el de les discogràfiques, produccions audiovisuals, telecomunicacions, etc. El fet de poder identificar els instruments involucrats en un arxiu (únicament d'àudio) de forma eficient, ja sigui per construir un motor de recomanació o de cerca, separar instruments en diferents pistes, edició musical, entre altres aplicacions cada cop és més important.

A part aquesta tasca també és molt rellevant en camps d'investigació com el de Music Information Retrieval (en endavant MIR). El MIR dintre de la informàtica és el camp encarregat de l'estudi del processament de dades musicals, els principals objectius del MIR són identificar i segmentació musical, generació d'algoritmes per processar dades musicals, generar contingut musical, etc.

Per aquest motiu l'objectiu del treball és fer un prova de concepte per veure quina és la forma més adient a l'hora d'extreure característiques amb l'objectiu de poder identificar els instruments que participen dintre d'una mostra àudio amb instruments composta únicament per les dades sonores dels instruments, com podria ser un fitxer MP3 o WAV.

1.2 Antecedents

En els últims anys s'han produït grans avanços en el fet de poder identificar instruments aconseguint grans resultats en especial amb l'ús de sistemes com les Neural Networks (NN), K-Nearest Neighbors (KNN), Support-Vector Machines (SVM) [9][10].

Tots aquests sistemes tenen els seus contres i pros, però pel que s'ha vist indagant en publicacions científiques és que la gran majora d'ells tenen resultats similars a l'hora d'executar tasques d'identificació instrumental, però en la gran majoria de vegades l'extractor de característiques utilitzat és la Mel Frequency Cepstral Coefficients (en endavant MFCC) [9][10][4].

S'ha vist que hi ha dues estratègies predominants a l'hora d'identificar instruments, la primera conta de millorar els resultats dels algoritmes de classificació, per altra banda, la segona, es basa en modificar els sistemes d'extraccions de característiques per obtenir dades que representin millor el problema a solucionar..

A part es considera necessari fer un cerca sobre que és el timbre musical pel que fa al tractament de senyals.

Els documents amb els quals s'ha elaborat la recerca estan indexats a la bibliografia.

Per aquest motiu els antecedents es divideixen en tres categories:

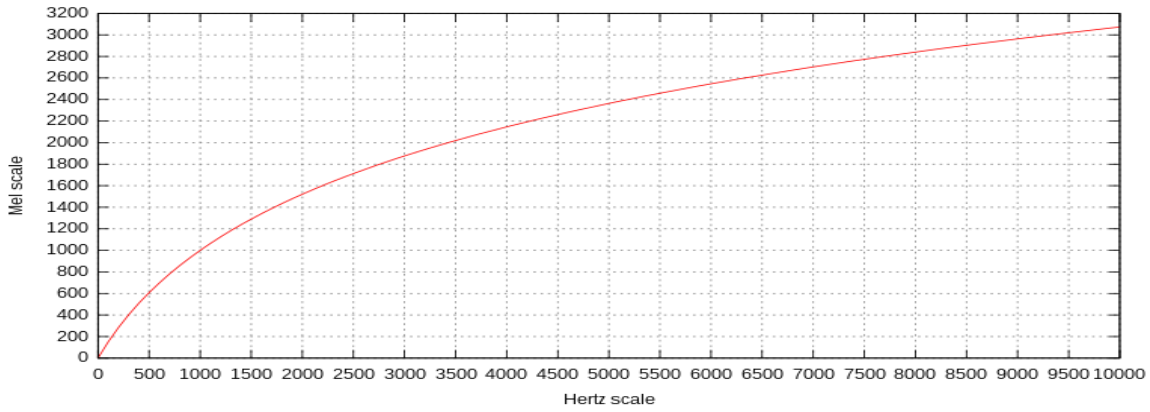
- Extractor de característiques MFCC.
- Sistemes de classificació KNN.
- El Timbre com a senyal.

1.2.1 Extractor de característiques MFCC

Els MFCC són un tipus de coeficients que originalment es van crear per representar les característiques de la parla, ja que va sorgir de la necessitat d'extreure característiques de les components d'un senyal d'àudio que siguin adequades per identificar contingut rellevant com les de la parla entre d'altres, això es deu al fet que l'extractor MFCC es basa en extreure les característiques més estables durant un període determinat [17].

Perquè l'extractor MFCC pugui extreure les característiques més estables fa ús de l'escala de MEL, aquesta escala permet transforma les freqüències que participen en un fitxer d'àudio i convertir-les en una escala logarítmica, d'aquesta manera l'extractor pot transformar les dades en un conjunt de característiques únicament amb les components estables. Per convertir una freqüència (f) en mel (m) es fa servir la següent fórmula:

$$m = 1127{,}01048 \log_e(1 + \frac{f}{700})$$



Il·lustració 1. Escala MEL

Pel fet d'utilitzar l'escala de MEL a l'hora de calcular el conjunt de característiques, les característiques resultants són les més estables, aquest fet fa que l'extractor MFCC basat en l'escala de MEL no tingui en compte moltes informacions, una d'elles és el timbre, ja que és una característica local i precisament és la que dona més informació a l'hora d'identificar instruments.

En investigar també s'ha vist que existeixen altres sistemes d'extracció basats en el sistema de codificació MPEG-7 [18] i són àmpliament utilitzats en tasques com identificar cançons, generes o la generació de nous arxius, però a diferència dels MFCC, aquests tipus de sistemes utilitzen les metadades de la codificació MPEG-7, que ja inclou quin tipus d'instruments participar en aquell moment, aquestes dades s'acostumen a generar en estudis de gravació i per aquest motiu no s'utilitzen a l'hora d'identificar instruments en fitxers d'àudio purs com MP3 o WAV.

Durant la tasca de recerca s'han trobat les transformades de Wavelet [19][20][21] les quals són similars a les transformades de Fourier, però tenint en compte el component temporal, permeten analitzar i descompondre un senyal en diferents escales temporals i de freqüència, això és útil per moltes tasques, específicament per detectar tipus de senyals, a més a més, existeixen alternatives que també tenen en compte les dades temporals, de

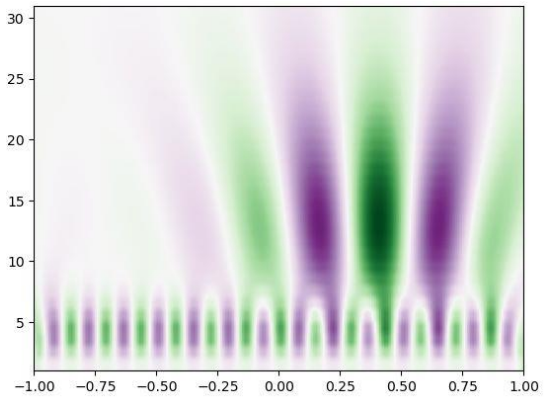
totes les alternatives disponibles ens centrarem en la Continue Wavelet Transform (en endavant CWT) i la Discrete Wavelet Transform (en endavant DWT):

- CWT → La transformada continua de Wavelet utilitzar unes funcions mare de Wavelet, la qual es pot escalar o transforma amb l'objectiu d'ajustar-se a la regió analitzada, la CWT és la versió més flexible i generalista però també té un cost computacional major.
- DWT → La transformada discreta de Wavelet en contra utilitzar una funció fixa de Wavelet, tot i que aquesta funció també es pot escalar i transforma per adaptar-se a la regió analitzada, la DWT és una versió més simple i amb un cost computacional menor a costa de ser menys flexible i generalista.

Les funcions mare de Wavelet són funcions matemàtiques que s'utilitzen en la CWT, com a norma general aquestes funcions són específiques per a una determinada aplicació i se seleccionen en funció de les característiques de senyal que es vol utilitzar, les més utilitzades són:

- La funció de Morlet és una funció gaussiana, on les seves aplicacions habituals són en àmbits on els senyals contenen components de freqüència tenen un pes moderadament alt (ones de ràdio, sismes, etc.).
- La funció de México és una funció en la qual els components de freqüència tenen menys pes que les temporals (dades meteorològiques, control de sensors, etc.).

La gran diferència entre les diferents alternatives de les funcions mare que es poden utilitzar en les CWT se la diferencia de pes entre les components temporals o les de freqüència.



Il·lustració 2 Representació transformada de Wavelet

1.2.2 Sistemes de classificació KNN

Avui en dia existeixen multitud de sistemes per categoritzar dades com els esmentats anteriorment els quals depenen de l'objectiu poden donar pitjors o millors resultats, en aquest cas, ens centrarem en el classificador K-Nearest Neighbords (en endavant KNN), ja que és un algoritme especialitzar a classificar grans conjunts de dades en diferents categories basats en tècniques d'aprenentatge automàtic.

L'algoritme KNN segueix la premissa que els elements que conformen les dades s'han de classificar segons com són de pròxims respecte als seus veïns, aquesta premissa funciona tant amb dades de text, com d'imatge o dades que representen sensors en el món real.

Per tal de poder calcular la distancia entre dos elements s'utilitzen mètriques de distància com Euclidiana, Manhattan o Minkowski. Aquests mètodes permeten crear una mètrica que permet decidir a l'algoritme quins elements són més similars.

Les mètriques esmentades anteriorment són les més utilitzades a l'hora de treballar amb KNN i tenen les següents diferències:

- La distancia Euclidiana és la mètrica més utilitzada per mesurar la distancia entre dos elements d'un conjunt de dades, aquesta distancia és sensible a les variacions de les observacions de tots els components que formen l'element, es representa de la següent forma matemàticament:

$$d_E(P_1, P_2) = \sqrt{\sum_{i=n}^{n} (|x_i - y_i|)^2}$$

- La distancia Manhattan és una mètrica que basa el càlcul de la distancia entre dos punts fent la suma de les diferencies absolutes entre els punts. Aquesta mètrica a diferencia de l'Euclidiana és una mètrica menys sensible a les variacions de les observacions, es representa amb la següent fórmula:

$$d_M(P_1, P_2) = \sum_{i=n}^{n} |x_i - y_i|$$

- La distancia de Minkowski és una mètrica que es basa en una combinació de les distancies anteriors, per tal d'aconseguir això es defineix "p", en aquest cas serveix per alternar entre les dues distancies, si $p = 1$ actua com la distancia Manhattan si $p = 2$ actua com la distancia Euclidiana, es representa amb la següent fórmula:

$$d_{Mink}(P_1, P_2) = \left(\sum_{i=n}^{n} |x_i - y_i|^p\right)^{\frac{1}{2}}$$

A l'hora de crear els KNN avui dia existeixen multitud de llibreries que permeten crear el teu propi classificador amb facilitat, les més utilitzades són SkLearn, PyTorch, TensorFlow, StatsModel, entre altres.

1.2.3 El timbre com a senyal

El timbre és una característica del so que permet distingir diferents fonts sonores, independentment de si aquestes comparteixen freqüències, es pot definir com una firma dactilar dels sons, o en aquest cas instruments.

Per altra banda, en l'àmbit de senyal, el timbre es pot descriure com un conjunt de característiques espectrals les quals pertanyen al grup de característiques Locals. Tot i que les Locals no són el mateix que les Temporals sí que tenen una relació.

1.3 Objectiu

Durant l'elaboració dels antecedents s'ha vist que tot i utilitzar diferents sistemes de classificació sempre s'assoleixen resultats similars i quasi sempre el mètode utilitzat per extreure característiques és el MFCC, el qual ignora característiques locals com pot ser el timbre, quan és precisament una de les característiques més importants d'un instrument, es proposa la hipòtesi que la forma de millorar els resultats pot anar més encaminat a com s'extreuen les característiques de les dades que no tant com en millorar la predicció en el sistema de classificació.

Per tant en aquest treball es proposar el desenvolupament d'un mètode d'extracció de característiques en el que es tinguin en compte el timbre i l'evolució temporal d'aquest basat en les transformades de Wavelet, ja que pel que s'ha vist tenen en compte tant dades de la freqüència com de temporals, les quals estan relacionades amb les variables Locals

com es el cas del Timbre. En endavant aquest sistema el nomenarem Wavelet Transform Cepstral Coefficients (en sigles WTCC).

A part del desenvolupament de l'extractor de característiques WTCC, també es desenvoluparà un extractor de característiques MFCC amb l'objectiu de poder fer beure quin dels dos mètodes d'extracció dona millors resultats a l'hora de classificar 10 instruments diferents.

Per poder medi els resultats es faran diferents proves amb diferents distancies i configurant els paràmetres fins a obtenir el resultat òptim. Si en la comparativa es veu una millora substancial de la precisió en el sistema de classificador es podrà dir que la hipòtesi era correcta i que el problema resideix en com s'extreuen les característiques, en cas contrari la hipòtesi no seria vàlida.

En el cas que la hipòtesi no fos vàlida es tindrien que planteja diferents sistemes d'extracció utilitzant altres variacions de la transformada de Wavelet o un canvi d'estratègia.

En el cas que la hipòtesi si fos vàlida es podria facilitar moltes de les tasques relacionades amb la classificació d'instruments dintre del MIR des d'algoritmes de recomanació o per segmentar pistes, entre moltes altres aplicacions.

Capítol 2: Metodologia

Per poder contrastar la hipòtesi es proposa un conjunt d'experiments en els quals s'anirà augmentant la complexitat de les dades, per aquest motiu l'experiment constarà de dues fases.

La primera fase constarà de realitzar l'experiment amb pistes que continguin únicament els sons dels instruments, per altra banda, la segona fase es compon per les mateixes dades, però amb un componen de soroll.

D'aquesta manera podrem analitzar si el sistema d'extracció de característiques WTCC ofereix millors o pitjors resultats que el MFCC i gràcies als resultats de cada fase poder aplicar correccions a la fase posterior o en la mateixa fase per tal de millorar els resultats..

A l'hora de resoldre el problema s'identifiquen tres metodologies, una per tal de poder generar les dades, una altra per poder dissenyar el sistema d'extracció de característiques i la tercera per poder avaluar quina dels dos sistemes ofereix millors resultats.

2.1 Metodologia: Generador de les Dades

La generació de les dades s'ha fet de tal forma que permeti emular un conjunt de dades compost per pistes musicals en les quals participen 2 instruments i on la durada d'aquests pot ser modificada a més de les dues fases de les dades (amb soroll o sense), per tal d'aconseguir això s'ha dividit el problema en 4 subgrups:

2.1.1 Definició de les dades

Per poder avaluar la precisió dels dos sistemes d'extracció seleccionats tenint en compte el context, primer s'han de definir les dades amb les quals es faran l'experiment, per això s'han seleccionat 10 instruments amb els quals es faran les proves, els instruments seleccionats són els següents:

- Acordió (instrument de vent)
- Clarinet (instrument de vent)
- Guitarra Elèctrica (instrument de corda)
- Flauta (instrument de vent)
- Guitarra Acústica (instrument de corda)

- Arpa (instrument de corda)
- Piano (instrument de corda)
- Saxofon (instrument de vent)
- Trompeta (instrument de vent)
- Violi (instrument de corda)

En aquest cas en concret es busca poder diferenciar el timbre dels instruments segons el seu timbre, per aquest motiu s'han omès els instruments que entren dintre del grup de la percussió, ja que aquest com a tal no tenen un timbre associat, espectralment es tracta d'un cop de freqüència net.

Tenint en compte els instruments esmentats s'obtindran un mínim de 100 mostres per cada una d'elles amb únicament un sol tipus instrument, d'aquesta forma podrem etiqueta les mostres des del primer moment.

Les 100 mostres dels 10 instruments han de complir les següents restriccions per tal que el resultat de l'experiment sigui el més fidel a l'objectiu, les restriccions a complir són les següents:

- Els fitxers d'àudio han de contenir un total de 10 instruments diferents.
- Cada instrument ha de compondre's per un mínim en 100 fitxers d'àudio.
- Els fitxers han de tindre una duració mínima de 25 segons.
- Només han de contenir l'instrument desitjat.

Amb aquestes dades que en endavant anomenarem Dades Crues es generant les 100 mostres de 25 segons de cada instrument, en total 1.000, les quals conformen les Dades Preparades.

L'objectiu de les Dades Preparades és combinar-les entre elles sense repetir-les i on cada pista resultant estigui composta per dos instruments de tal forma que amb les 1.000 mostres es poden arribar a crear un total de 499.500 pistes musicals, aquest conjunt el qual anomenarem Dades Finals es amb el que es realitzaran els experiments.

2.1.2 Obtenció de les Dades Crues

En l'actualitat obtindré pistes musicals que només continguin un instrument i complexi els requisits esmentats, a més a més, que aquestes tinguin una llicència "Free Use" pot arribar a ser una tasca complexa, per aquest motiu s'ha optat per realitzar descàrregues de les dades crues en portals amb pistes musicals sense drets d'autor i de coneguts amb accés als instruments esmentats.

Després de realitzar una recerca de portals que permetin la descàrrega de mostres musicals sense drets d'autor, per les restriccions definides en la metodologia només s'ha trobat un portal [14], amb el qual no s'ha pogut completar totes les mostres necessàries per generar les 100 mostres de la Guitarra Elèctrica, amb els altres instruments si que s'han pogut completar les 100 mostres.

El portal utilitzat per descarregar mostres s'ha seleccionat perquè és un lloc on es comparteixen peces formades únicament per un tipus d'instrument, per exemple, un fitxer de saxofon pot contindré des d'1 a diversos saxofons.

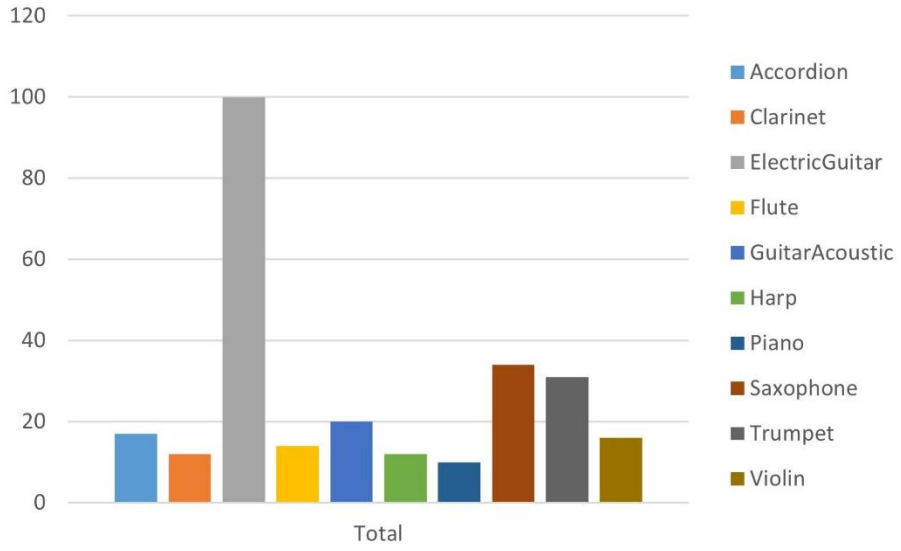
Per obtenir les dades restants s'ha utilitzat un equip de gravació semiprofessional facilitat per un contacte personal el qual també disposa d'una Guitarra Elèctrica.

Per tal de poder tindre etiquetades les dades des del principi s'ha generat un fitxer CSV anomenat "pre_track_info.csv" amb les següents columnes:

- ID: Identificador únic de la taula (Primary Key)
- NAME: Nom del fitxer
- DURATION: Duració del fitxer, aquesta com a mínim es de 25 segons.
- N_LOOPS: Numero de mostres de 25 segons que es poden extreure del fitxer.
- INTRUMENT: Etiqueta que conte el nom del instrument al que pertany el fitxer.
- PATH: Ruta al fitxer

Aquest CSV a part de contenir les dades necessàries per generar les Dades Preparades també està ordenat de tal forma que les primeres files són les pistes amb menys mostres, amb aquesta estratègia aconseguim que en generar les mostres s'utilitzin el màxim de fonts diferents possibles.

Per poder generar la quantitat requerida en la metodologia de 100 mostres per 10 instruments ha estat necessari descarregar 166 fitxers i 100 s'han hagut de generar en un estudi de gravació, el nombre de fonts per cada instrument és el següent:



Il·lustració 3. Distribució fonts per instrument

2.1.3 Transformació de les Dades Crues a les Dades Preparades

En aquesta fase s'han de transformar les dades crues obtingudes (formades per les pistes musicals descarregades o enregistrades) en dades que compleixin les restriccions donades en la definició de les 100 mostres per instrument, les quals anomenarem Dades Preparades.

Per poder generar les Dades Preparades es proposa dissenyar un algoritme que sigui capaç de crear tantes mostres per instrument com es desitgi amb un màxim de 100 mostres.

L'algoritme ha de permetre modificar la duració de les mostres amb l'objectiu de poder reduir la mida que ocupa en el disc.

Aquesta decisió es pren perquè tot i que la idea és realitzar els experiments amb la totalitat de les dades amb el Hardware del qual es disposa si el conjunt de dades és molt gran els temps d'execució poden ser desproporcionats o inclús provoca la fallada del programa.

El mètode encarregat de la segmentació les Dades Crues a Dades Preparades és "transform_data_to_clean_data" que té els següents paràmetres:

- segment_dur_secs = duració de la mostra, en aquest la durada del segment que s'extreu del fitxer, el màxim valor es 25 segons.
- Rate = Frquencia de mostreig a la que es vol treballar, pel que s'ha vist a diferents foros sobre MIR el mes recomanable es treballar amb una frequencia de mostreig de 22.050 Hz.
- number_loops = Numero de mostres que es vol tindre per instrument, en aquest cas el numero màxim de mostres es 100.
- number_instruments = Numero d'instruments amb el que es vol fer el experiment.

Els paràmetres "number_intruments" i "number_loops" estan pensats per si en cas que el cost computacional dels 10 instruments o de les 100 mostres fos molt elevat es pogués generar un conjunt de dades més petit complint les restriccions sense necessitat de modificar el codi.

La funció "transform_data_to_clean_data" llegeix el CSV "pre_track_info.csv" i permet generar conjunts de dades amb diferents configuracions d'instruments i fitxers mitjançant la segmentació de fitxers en mostres de X segons, totes les mostres generades són emmagatzemades en un directori prèviament definit en el que cada instrument té la seva pròpia carpeta.

Per poder veure amb més detall el funcionament de la funció "transform_data_to_clean_data" anar a l'ANEX I, allà hi ha el pseudocodi.

Per tal d'evitar problemes entre els diferents conjunts i per reduir el consum de Disc Dur s'ha implementat un sistema per a eliminar les dades prèvies de tal forma que només pot coexistir una versió final de les dades.

Un cop creades les dades és necessari generar un CSV de control, anomenat "loop_info.csv". Aquest CSV és crear utilitzant el mètode "save_track_info" el qual explora el directori on s'han emmagatzemat les mostres generades anteriorment, aquest CSV consta de les següents columnes:

- ID = Identificador únic de la taula (Primary Key)

- TAG = Conte el nom del directori on es guarda el document, en aquest cas correspon al nom del instrument.
- PATH = Ruta a la mostra.

El CSV "loop_info.csv" és un fitxer de control el qual és principalment creat per poder treballar en cada una de les fases en paral·lel, ja que menters es busca millorar els resultats d'una es pot continuar fent proves a les altres fases amb les dades prèviament calculades, a més a més, permet tant reduir memòria RAM com els costos computacionals. Durant el transcurs de l'experiment es crearan CSV de controls entre la gran majoria de les fases.

2.1.4 Transformació de les Dades Preparades a les Dades Finals

L'objectiu d'aquesta fase és la creació d'un algoritme que sigui capaç de combinar tots els fitxers generats amb l'algoritme de la fase anterior on el fitxer resultant ha de complir les següents restriccions:

- Els fitxers resultants han d'estar compostos per dues pistes diferents, per tant, en una pista pot sonar un únic instrument però de pistes diferents.
- Les dades estaran en canal mono (menys dades a processar, per tant, reducció del cost computacional).

Per tal de poder fer computa totes les combinacions possibles tenint en compte el nombre total de mostres (es a dir, si tenim 100 mostres i 10 instruments el total és 1.000) aplicarem la següent fórmula on "n" = número total mostres i "m" = nombre d'instruments:

$$numero\ combinacions\ posibles: \frac{n!}{m! * (n - m)!}$$

Seguint el cas d'utilitzar 100 mostres per cada un dels 10 (100 * 10 = 1.000) instruments tenim que:

$$numero\ combinacions\ posibles: \frac{1.000!}{2! * (1.000 - 2)!} = 499.500$$

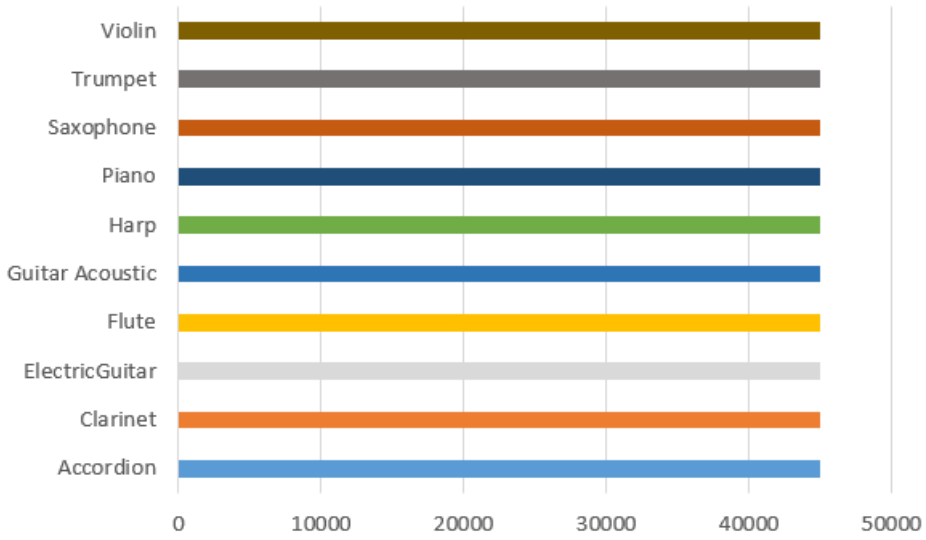
El mètode encarregat de calcular totes les combinacions possibles de les mostres es l'anomenat "create_data" i té els següents paràmetres:

- PHASE = Nom que rep la fase actual, amb aixo podem identificar quin tipus de dades estem creant, si es la Fase 1 son dades sense soroll, si es la Fase 2 son dades que contenen un soroll blanc
- TRACK_INFO = La informació de les mostres creada en el mètode "save_track_info", pot utilitzar la variable que retorna (no es poden fer canvis a la fase anterior, o el CSV "loop_info.csv").
- NOISE = Variable que ens permet definir si volem o no soroll a les combinacions de les mostres.

En aquest cas l'únic paràmetre que ens permet jugar amb la funció és "noise", ja que les altres sempre contindran les dades de les mostres a combinar i el nom de la fase.

En aquest com en l'anterior aquest mètode també és generar un CSV de control anomenat "phase_name_loop_info.csv" i en aquest cas també segueix el principi de segmentar fases amb l'objectiu de poder millorar diferents fases en paral·lel, això si la tasca "create_data" no es pot executar al mateix temps que s'extreuen dades, ja que treballen sobre el mateix directori. Això es deu al fet que per reduir l'ús del Disc Dur disponible aquesta estructura de fitxers ha de ser comú.

Un cop generades les dades els instruments queden segons la següent distribució:



Per poder veure amb més detall el funcionament de la funció "create_data" anar a l'ANEX I, allà hi ha el pseudocodi.

2.2 Metodologia: Extractor de Característiques

Per dur a terme l'extracció de les característiques seguint l'objectiu de l'experiment s'han de desenvolupar dos algoritmes, un centrat en les MFCC i l'altre centrat en les WTCC.

Aquests algoritmes tindran l'objectiu d'extreure les característiques de les Dades Finals seguin les següents restriccions:

- Els dos sistemes extractors han d'agafar la quantitat de característiques necessàries per representar la variància de les característiques.
- En dos sistemes han de permetre modificar la finestra, la finestra es l'espai que analitzar l'extractor a l'hora de calcular les característiques dintre del fitxer musical, la finestra ha de ser d'una duració inferior a la del fitxer.
- Els dos sistemes s'han de construir de tal forma que tinguin el menor impacte en la memòria RAM

Aquestes restriccions tenen l'objectiu de millorar la predicció al mateix temps que s'aprofiten millor els recursos de hardware dels quals es disposa.

2.2.1 Extractor MFCC

Per tal de poder implementar l'extractor MFCC podem utilitzar el mètode que rep el mateix nom inclòs en la llibreria LibRosa esmentada anteriorment, el qual permet agafar automàticament el nombre de característiques necessàries per representar la variància de la mostra, això si s'han de normalitzar els valors.

L'extractor MFCC és realitzar en el mètode "extract_mfcc" el qual té els següents tres paràmetres:

- PHASE = Nom que rep la fase actual, amb això podem identificar quin tipus de dades utilitzem dintre del extractor, si es la Fase 1 son dades sense soroll, si es la Fase 2 son dades que contenen un soroll blanc
- PHASE_CSV = Es el nom que se li han ficat als CSV de control que s'han creat durant la fase on es creen les combinacions de totes les mostres "phase_name_loop_info.csv".
- SEGMENT_DURS_SECS = Es la mida de la finestra que s'utilitzarà per extreure característiques de les mostres

Aquest mètode utilitzar totes les combinacions creades per una fase especifica (amb soroll, sense soroll) i per tal de poder treure el màxim partit s'ha de jugar amb la finestra.

Per poder veure amb més detall el funcionament de la funció "extract_mfcc" anar a l'ANEX I, allà hi ha el pseudocodi.

La finestra en aquest cas és un concepte que representa quina part de la mostra s'utilitzarà per calcular les característiques MFCC, es a dir si una combinació té la durada de 10 segons i la finestra és d'1 segons, aquesta mostra es pot dividir en 10 finestres, com més gran és la finestra menys dada local representaran les característiques extretes i viceversa.

Depenen de la quantitat de mostres l'execució d'aquesta tasca pot romandre durant molt de temps, i per això tot el codi està adaptat per funcionar independentment de la quantitat de combinacions a processar, ja que aquesta informació va donada pel CSV de control anterior, "phase_name_loop_info.csv" i per modificar el nombre d'elements i instruments s'ha d'utilitzar la funció "create_data" de la fase de combinació de Dades Preparades a Dades Finals.

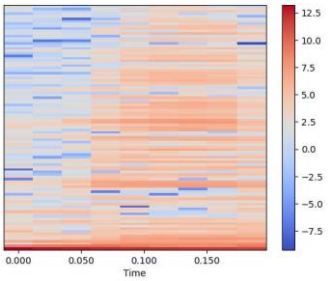
A part d'extreure les característiques de les finestres que componen la mostra aquest mètode també és l'encarregat de generar el diccionari en el qual es guarden les etiquetes de cada una de les finestres, aquest diccionari és necessari per a poder després entrenar i predir utilitzant el classificador KNN.

Pel fet que aquesta tasca depenen de quin sistema s'utilitzi pot consumir una gran quantitat de memòria RAM s'ha guardat les dades generades a un diccionari, ja que tant el cost per crear un element com per accedir-hi és O(1), de tal forma que al finalitzar disposen de dos diccionaris, un amb les dades i l'altre amb les etiquetes, els dos diccionaris segueixen el mateix ordre pel que es poden mapar entre ells.

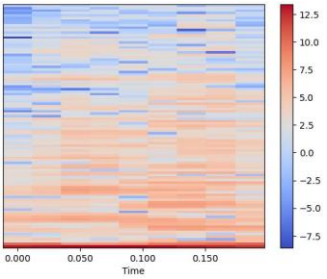
Per poder avançar de fase les dades han d'estar en una array de NumPy (és necessari per a poder relacionar les dades amb les etiquetes, per fer això definim la mida de l'array abans d'afegir les dades i després afegim les dades del diccionari 1 a 1, d'aquesta manera el cost d'afegir les dades a l'array és O(1).

S'ha decidit utilitzar aquest mètode, ja que utilitzar un mètode com "vStack" o "Concatenate" de NumPy consumeixen molta memòria i poden provocar una fallada del programa.

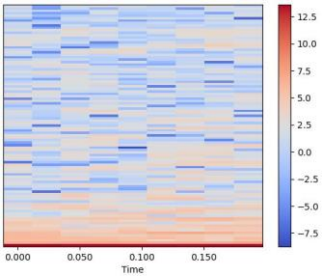
Un cop finalitzat el programa ja tenim les dades necessàries per fer proves de classificació amb característiques MFCC. A continuació hi ha una mostra de les característiques MFCC dels diferents instruments:



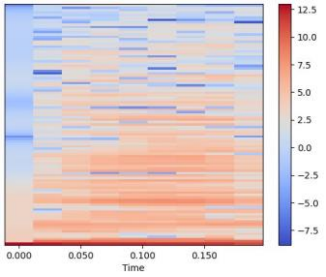
Il·lustració 4 Característiques MFCC d'un Acordió



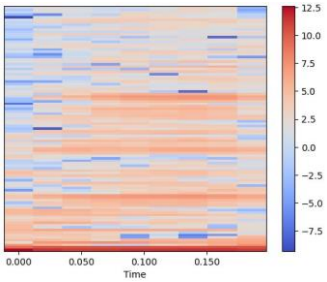
Il·lustració 5 Característiques MFCC d'un Clarinet



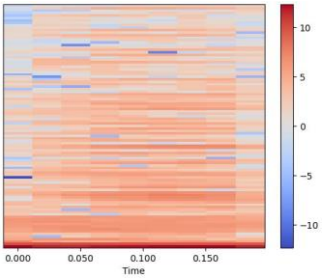
Il·lustració 6 Característiques MFCC d'una Guitarra Elèctrica



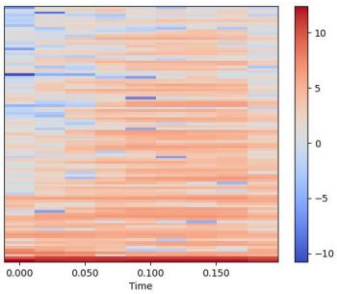
Il·lustració 7 Caracteristiques MFCC d'una Flauta



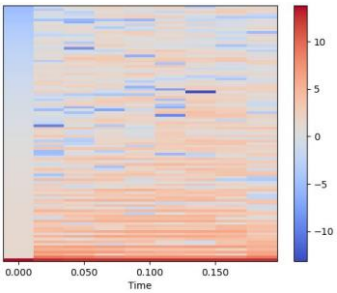
Il·lustració 8 Característiques MFCC d'una Guitarra Acústica



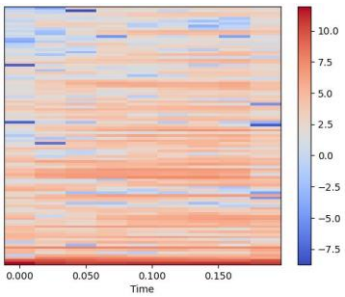
Il·lustració 9 Caracteristiques MFCC d'un Piano



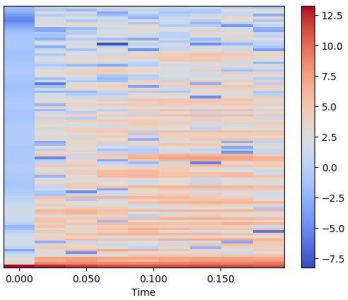
Il·lustració 10 Característiques MFCC d'una Arpa



Il·lustració 11 Característiques MFCC d'una Trompeta



Il·lustració 12 Característiques MFCC d'un Saxofon



Il·lustració 13 Característiques MFCC d'un Violi

2.2.2 Extractor WTCC

Per tal de poder implementar l'extractor WTCC no s'ha trobat com llibreria que el tingui com a tal per tal s'ha elaborat la següent estratègia per desenvolupar un extractor de característiques:

- Primer pas: Extreure les Transformades de Wavelet d'una finestra, en aquest el concepte finestra té la mateixa definició que en el MFCC.
- Segon pas: Realitzar un anàlisis de la variància explicada sobre la transformada amb l'objectiu de reduir les característiques originals de la Transformada de Wavelet.
- Tercer pas: Calcular el coeficients cepstrals de la Transformada de Wavelet reduïda, per tal de obtenir les característiques WTCC.

Primer pas: Extreure les transformades de Wavelet

Per poder extreure les transformades de Wavelet s'ha fet una recerca i s'ha seleccionat PyWT que és una de les llibreries més utilitzades i té una documentació detallada.

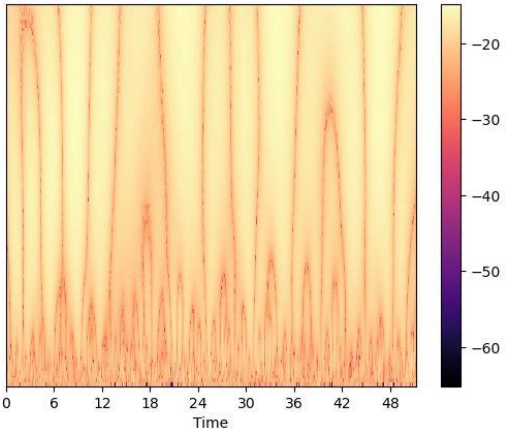
Aquesta llibreria permet calcular les transformades de Wavelet sobre un conjunt de dades, a més, permet calcular quasi tots els tipus de Transformades, DWT i CWT incloses, la diferencia entre elles és que la primera li dona més pes a les components de freqüència mentre que la segona li dona més pes a les components temporals.

Per aquest motiu es realitzarà l'experiment amb les CWT, ja que el timbre dels instruments és una component més relacionada amb les dades temporals que amb les components de la freqüència.

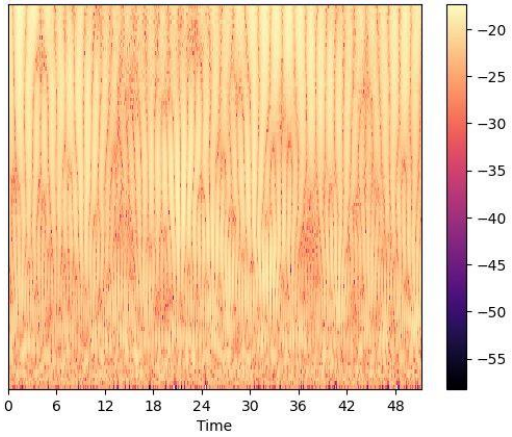
Ja que la CWT permet utilitzar diferents alternatives de les Transformades de Wavelet es realitzaran les proves de l'experiment tant amb les dues que s'han esmentat en l'apartat dels "Antecedents", la funció Morlet i de México.

Per evitar que el rang de valors de cada transformada calculada sigui diferent s'han normalitzat els valors de les Transformades de Wavelet entre 0 i 1.

Aquesta és una mostra de la transformada de Wavelet aplicada sobre una mostra de les pistes del conjunt que formen les dades:



Il·lustració 14 Transformades de Wavelet México



Il·lustració 15 Transformada de Wavelet Morlet

Segon pas: Anàlisi variància explicada

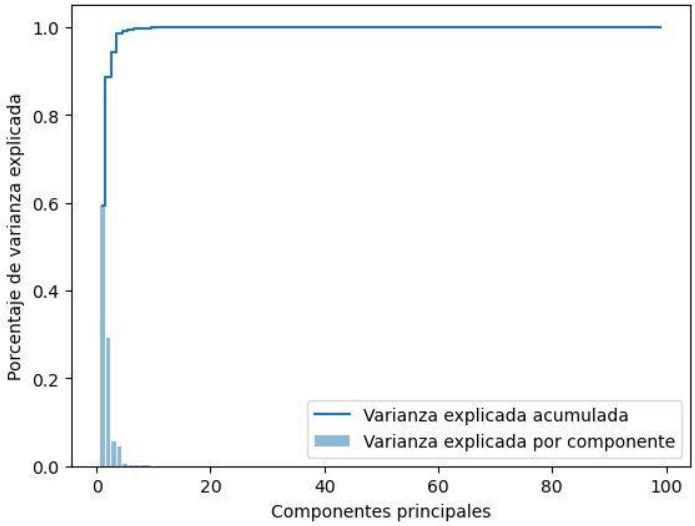
Un cop calculades les Transformades de Wavelet, es pot observar que retorna un total de 3307 components, aquest nombre és molt elevat i representaria uns temps de computació molt elevat, per tant, s'ha de reduir el nombre de components però sense perdre informació rellevant de les característiques.

Per tal de reduir el nombre de característiques s'ha fet un anàlisi de la variància explicada amb l'objectiu de deduir el nombre de característiques són necessàries per mantenir la variància de la mostra.

Per poder descobrir el nombre de components per mantenir la variància s'han calculat les Transformades de Wavelet d'algunes possibles finestres de tots els instruments, això s'ha fet segmentant les Dades Finals en finestres i aplicant la funció.

Un cop calculades calculem el percentatge de la variància explicada, la qual ens permet observa el nombre de components necessaris per mantenir la informació de les dades reduint el seu nombre de característiques, és a dir aquest càlcul ens permet mantenir la variància de les dades reduint el nombre de components.

Al calcular diferents percentatges de la variància explicada podem observar com en tots els casos analitzats la distribució dels pesos és molt similar, la distribució dels pesos del percentatge de la variància explicada es pot veure en la següent gràfica:



Il·lustració 16. Percentatge de variància explicada per component

A l'interpretar la gràfica anterior podem veure com tindre més de 10 components no és necessari, ja que a partir de la 5 component el percentatge de la variància explicada és ridículament superior al 0 %, per aquest motiu l'extractor de característiques es realitzarà tenint en compte 10 components..

Per poder fer la transformació de les 3077 components originàries del mètode CWT a les 10 d'interès s'ha utilitzat el paràmetre "scales" del mateix mètode empleat per calcular les transformades, aquest paràmetre ens permet definir amb quantes components ha d'estar calculat la transformada. D'aquesta forma obtenim les transformades de Wavelet Reduïdes amb el nombre de components desitjat.

Tercer pas: Calcular WTCC

Un cop tinguem les Transformades de Wavelet Reduïdes es pot calcular els coeficients cepstrals, aquests seran els coeficients que conformaran les característiques WTCC.

Per poder transformar els valors de les Transformades de Wavelet Reduïdes a coeficients ens és necessari utilitzar un sistema que ens permeti transformar les noves característiques a una nova dimensió menor sense perdre dades rellevants, per poder fer això s'ha utilitzat la llibrera SkLearn, concretament el mètode PCA.

PCA (Principal Component Anilysis) és una tècnica estadística per analitzar i visualitzar dades en un espai de característiques de menor dimensió tot i que també es pot utilitzar

per calcular els coeficients cepstrals, ja que matemàticament són una combinació lineal de les característiques.

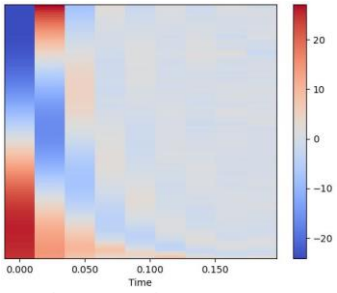
Un cop reduïdes per PCA ja s'ha acabat el còmput de les característiques per finalitzar l'extractor seria adaptar l'algoritme creat per MFCC perquè ara calcules totes les característiques WTCC del conjunt de dades.

Per tal de poder computar les característiques WCTT de tot el conjunt de dades, s'ha implementat el mètode "extract_wtcc" el qual té els següents paràmetres:

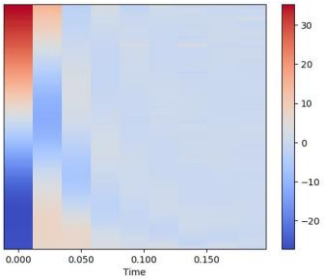
- PHASE = Nom que rep la fase actual, amb això podem identificar quin tipus de dades utilitzem dintre de l'extractor, si és la Fase 1 són dades sense soroll, si és la Fase 2 són dades que contenen un soroll blanc.
- PHASE_CSV = És el nom que se li han ficat als CSV de control que s'han creat durant la fase on es creen les combinacions de totes les mostres "phase_name_loop_info.csv".
- SEGMENT_DURS_SECS = És la mida de la finestra que s'utilitzarà per extreure característiques de les mostres.
- WAVELET = Permet modificar la Transformada de Wavelet amb la que utilitzar, ja sigui Morlet o México, entre altres.

La lògica de la funció és exactament la mateixa que de l'extractor de MFCC adaptada per tal de fer tots els càlculs extra necessaris per a obtenir els coeficients WTCC.

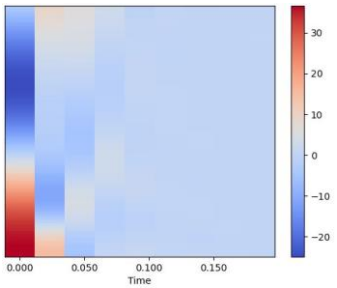
Un cop finalitzat el programa ja tenim les dades necessàries per fer proves de classificació amb característiques WTCC. A continuació hi ha una mostra de les característiques WTCC dels diferents instruments:



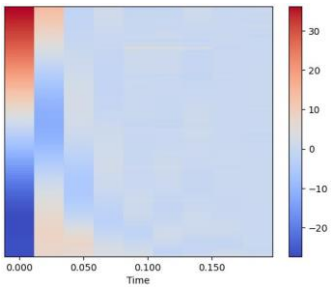
Il·lustració 17 Característiques WTCC d'un Acordió



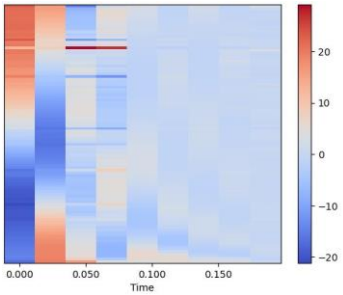
Il·lustració 18 Característiques WTCC d'un Clarinet



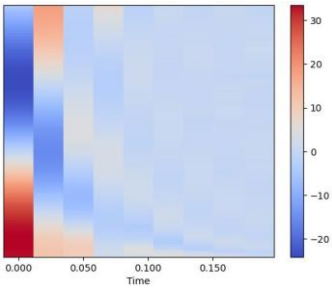
Il·lustració 19 Característiques WTCC d'una Guitarra Elèctrica



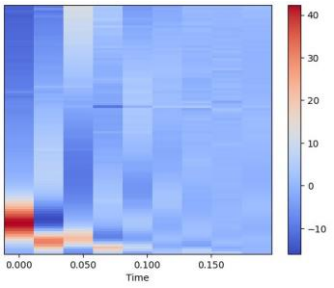
Il·lustració 20 Característiques WTCC d'una Flauta



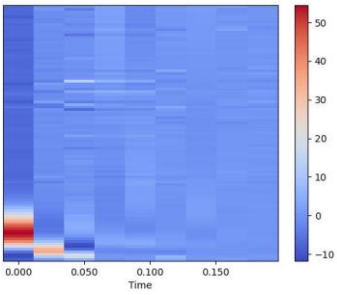
Il·lustració 21 Característiques d'una Guitarra Acústica



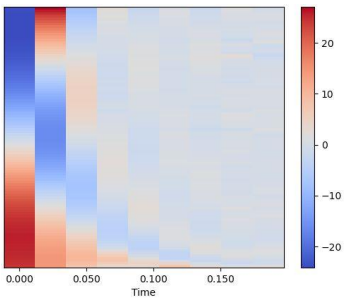
Il·lustració 22 Característiques WTCC d'una Arpa



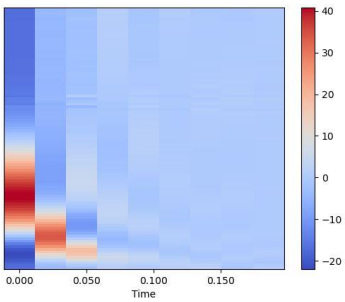
Il·lustració 23 Característiques WTCC d'un Piano



Il·lustració 24 Característiques WTCC d'un Saxofon



Il·lustració 25 Característiques WTCC d'una Trompeta



Il·lustració 26 Característiques WTCC d'un Violi

2.3 Metodologia: Classificador KNN

Per dur a terme la validació de quin sistema d'extracció de característiques ofereix millors resultats es proposa la construcció d'un algoritme que permeti la creació de diferents classificadors KNN permeten modificacions en els paràmetres amb l'objectiu de poder millorar la precisió del model.

Per tal de dur a terme el classificador KNN s'ha fet una recerca de les possibles llibreries per poder construir una KNN personalitzant els seus paràmetres, per tant, s'ha escollit la llibreria SkLearn la qual disposa de la classe KNeighborsClassifier la qual ens permet configurar i realitzar tota mena de proves i té una documentació molt detallada.

S'ha implementat la funció "knn_class" la qual ens permet crear i configurar classificadors KNN per realitzar totes les proves desitjades, està conformada pels següents paràmetres:

- DATA = És una array de NumPy de N x M on N són el número de finestres i M és el nombre coeficient sigui MFCC o WTCC.
- LABELS = És un diccionari on existeixen totes les finestres amb la seva etiqueta corresponent i està en el mateix ordre que a DATA.
- N_NEIGHBORS = Indica el nombre de veïns a l'hora de classificar un punt de les dades, com a norma general com més veïns més precisa serà la classificació.
- WEIGTHS = Defineix com s'han d'aplicar els pesos als veïns, per defecte és "uniform" és a dir que tots els veïns tenen el mateix pes, però es pot configurar com "distance" on els veïns més propers tenen més pes que els llunyans.
- LEAF_SIZE = Defineix la finestra de cerca de veïns que l'algoritme definit utilitzarà, fet que un valor molt elevat augmentarà la precisió a costa de la velocitat i viceversa.
- ALGORITHM = Especificar el tipus d'algoritme que s'utilitzarà per buscar els veïns més propers, per defecte en conjunts grans un algoritme ràpid com Ball Tree, però es poden utilitzar d'altres més simples.
- P = Defineix el valor de p si utilitzem la distancia de Minkowski, si p = 1 és l'equivalent a utilitzar la distancia de Manhattan i si és 2 es com si es fes servir l'Euclidiana.

- METRIC = Tipus de funció per calcular la distancia entre dos punts, per defecte es la distancia de Minkowski, però es pot modificar per utilitzar-ne d'altres.
- TEST_SIZE = Defineix el percentatge de les dades que s'utilitzarà com a test a l'hora d'avaluar la precisió del classificador KNN.

Aquesta funció està conformada pels següents passos:

- Primer pas: Crear un DataFrame que contingui les dades de totes les finestres (DATA) conjuntament amb les seves etiquetes (LABELS), pel fet que les dades resultants en els extractors sigui MFCC o WTCC tenen el mateix ordre es pot utilitzar la funció "map" de NumPy.
- Segon pas: Un cop tenim el DataFrame creat es procedeix a crear les variables X (conjunt de dades) i Y (conjunt d'etiquetes per cada una de les dades).
- Tercer pas: Segmentació del conjunt de dades X i Y per tal de crear els conjunts d'entrenament ("X_train" i "y_train") i les necessàries per poder realitzar una prova de predicció ("X_test" i "y_test"). Aquest paràmetre pot ser configurat amb la variable "TEST_SIZE" segons la prova que es vulgui fer però en la gran majoria el conjunt de dades per entrenar és del 80% i d'un 20% per les proves.
- Quart pas: Crear un classificador KNN amb la configuració dels paràmetres anteriors (menys les ja esmentades en els passos anteriors).
- Cinquè pas: Un cop definit el classificador, aquest s'ha d'alimentar amb les dades d'entrenament i les etiquetes respectives ("X_train" i "y_train"), això es fa mitjançant el mètode "fit" que incorpora la classe KNeighboursNearest de la llibreria SkLearn.
- Sisè pas: Un cop alimentat el model amb les dades d'entrenament es procedeix a realitzar la predicció dels elements de la prova ("X_test" i "y_test"), en aquesta part és on realitzen els càlculs necessaris per predir les etiquetes tenint en compte les dades i les etiquetes d'entrenament.
- Sete pas: Un cop finalitzi la predicció es procedeix a calcular la precisió de la predicció realitzada.

Amb aquest mètode es poden crear diferents classificadors KNN modificant els valors dels paràmetres del mètode "knn_class".

Capítol 3: Resultats

3.1 Proves realitzades

Un cop finalitzada la implementació i validació de tots els codis necessaris tant per poder generar les Dades Finals partint de les Dades Crues, com el sistema que combina totes les mostres, els dos sistemes d'extracció de característiques i el mètode que permet generar KNN segons la prova que es vulgui realitzar, s'han de realitzar prou nombre de proves per a poder avaluar la hipòtesi de l'experiment.

Com que en aquest experiment es busca veure quin dels dos sistemes d'extracció és més eficient, totes les proves que es realitzin a un sistema com el MFCC s'han de realitzar al WTCC i viceversa, d'aquesta forma es podrà crear una taula en la qual es podran avaluar i comparar els resultats. Això si les mètriques internes de l'extractor són pròpies de cada sistema d'extracció, ja que en aquell punt es busca obtenir el menor nombre de característiques sense perdre informació rellevant.

Per poder valorar quin sistema d'extracció ofereix millors resultats s'han realitzat les següents proves:

3.1.1 Prova 1: 10 instruments amb 100 mostres de 25 segons

Per poder fer aquesta prova és necessari utilitzar tot el conjunt de dades del qual es disposa, per tal de poder fer això un cop tenim les Dades Finals (Mostres de 25 segons que només contenen un únic tipus d'instrument) es procedeix a crear totes les possibles repeticions de les 1.000 (10 instruments * 100 mostres) mostres sense repetir-les, això fa un total de 499.500 pistes en les quals participen dos instruments que en alguns casos serà un altre instrument del mateix tipus:

$$numero\ pistes: \frac{1.000!}{2! * (1.000 - 2)!} = 499.500$$

Això s'ha computat dues vegades, un cop amb dades purament sonores (fase 1) i l'altra amb dades sonores que tenen una component de soroll (fase 2).

En el moment de calcular totes les dades tenim el problema de què no es disposa d'espai suficient en el disc i no es poden guardar el total de les dades. Per aquest motiu es pren la

decisió de cancel·lar aquesta prova, ja que no es disposen dels suficients recursos d'emmagatzematge, en aquest cas concret es requerien aproximadament 11 TB per cada una de les fases i el Hardware actual només disposa d'1 TB.

3.1.2 Prova 2: 10 instruments amb 5 mostres de 25 segons

En aquest cas el nombre de mostres s'ha anat reduint de 5 en 5, començant amb un total de 100 fins a arribar al punt de què el Hardware disponible és capaç de treballar amb la quantitat de dades amb uns temps d'execució raonables, per aquest motiu el nombre de mostres és de 5.

Per poder fer aquesta prova és necessari utilitzar una part del conjunt de dades del qual es disposa, això és possible pel fet que el mètode "transform_data_to_clean_data" s'ha pensat per generar diferents nombres de mostres per instruments i que el mètode "save_track_info" generar el CSV de control necessari per "create_data".

Un cop generades les Dades Finals (Mostres de 25 segons que només contenen un únic tipus d'instrument) es procedeix a crear totes les possibles repeticions de les 50 (10 instruments * 5 mostres) mostres sense repetir-les, això fa un total de 499.500 pistes en les quals participen dos instruments que en alguns casos serà un altre instrument del mateix tipus:

$$numero\ pistes: \frac{50!}{2! * (50-2)!} = 1.225$$

Això s'ha computat dues vegades, un cop amb dades purament sonores (fase 1) i l'altra amb dades sonores que tenen una component de soroll (fase 2).

Un cop generades les combinacions es disposa a extreure les característiques tant MFCC com WTCC de les dades, un cop finalitzin s'executarà el classificador KNN amb tots els ajustos per defecte amb una k de 2. Els valors per defecte són:

- WEIGHTS = uniform, en aquest cas tots els components tenen el mateix pes.
- ALGORITHM = auto, al tractar-se d'un conjunt de dades gran fem que sigui el propi sistema que seleccioni el algoritme de cerca de veïns.
- LEAF_SIZE = 30, defineix la finestra del algoritme de cerca.

- METRIC = Minkowski, ja que segons quin valor tingui p el podem alterar el seu comportament i varia entre distancia Euclidiana o Manhattan.
- P = 2, d'aquesta forma la funció de Minkowski es comporta com una distancia Euclidiana.
- TEST_SIZE = 0,2, ens permet definir el tamany del conjunt de dades a predir i per tant el d'entrenament (1 – TEST_SIZE).

El classificador KNN s'ha executat tant amb les característiques MFCC com WTCC per les dues fases tant amb la component de soroll com sense.

En el moment de validar els resultats s'han obtingut les següents precisions:

- MFCC → 0.006 % (Tant amb soroll com sense)*
- WTCC → 0.008 % (Tant amb soroll com sense)*

* La precisió no és exactament la mateixa tenint soroll o amb l'absència d'ell, però si arrodonim els valors al tercer decimal obtenim el mateix valor.

En aquest cas podem deduir que el resultat de la precisió és tan petit pel fet que el conjunt de dades estava conformat per molt poques dades i el classificador KNN no ha estat capaç de generalitzar o perquè el nombre de veïns és molt reduït, ja que s'ha fet la prova amb només 2 veïns.

Ara per ara no es pensa que la distancia Euclidiana (s'utilitza la distancia Minkowski, però amb la p = 2, per tant, actua com si fos una distancia Euclidiana)

Veient que així el classificador KNN no donarà resultats comparables es reduirà el nombre d'instruments en comptes del nombre de mostres.

3.1.3 Prova 3: 2 instruments amb 100 mostres de 10 segons

Veient els resultats de la prova anterior i les limitacions de Hardware de què es disposen s'ha reduït el nombre d'instruments de 10 a 2 i de mostres per instrument de 100 a 60 i de 10 segons de duració per tal de poder complir amb els següents objectius:

- No superar els 400 GB de pistes un cop ja combinades amb altres mostres. El disc dur del qual es disposa és d'1 TB, però aquest no disposa del 100 % de l'espai.

- Permetre realitzar proves sense ficar l'ordinador a més del 80 % de processament, ja que el Hardware disponible té certa antiguitat i el sistema de refrigeració no és l'adient per tasques exhaustives.
- Que el temps d'execució de les diferents parts del codi, sigui generant dades o extreien característiques o computant el classificador KNN no demori més de 12 hores.

* El perquè d'aquesta reducció tan abrupta està detallat en el següent apartat de la memòria.

Seguint les restriccions de només 60 mostres per 2 instruments s'ha utilitzat una part del conjunt de dades del qual es disposa, això és possible pel fet que el mètode "transform_data_to_clean_data" s'ha pensat per generar diferents nombres de mostres per instruments i que el mètode "save_track_info" generar el CSV de control necessari per "create_data".

Un cop generades les Dades Finals (Mostres de 120 segons que només contenen un únic tipus d'instrument) es procedeix a crear totes les possibles repeticions de les 50 (10 instruments * 5 mostres) mostres sense repetir-les, això fa un total de 1.770 pistes en les quals participen dos instruments que en alguns casos serà un altre instrument del mateix tipus, les pistes tenen una duració de 20 segons:

$$numero\ pistes: \frac{60!}{2! * (60-2)!} = 1.770$$

Ha diferencia de l'experiment anterior s'ha decidit només computa les pistes una única vegada únicament amb dades sonores, s'ha prescindit de les dades amb components de soroll per tal de centrar-nos a obtenir una millor precisió amb les dades pures.

Un cop generades les combinacions es disposa a extreure les característiques tant MFCC com WTCC de les dades, aquesta vegada les WTCC s'han extret dues vegades un cop utilitzant l'alternativa de Morlet i l'altra amb la de México.

En total es disposen de 3 conjunts de característiques amb els quals es podran alimentar diferents classificadors KNN. Les execucions s'han executat en sèrie de tal forma que després s'han agrupat els resultats per tal de poder comparar els resultat.

Un cop finalitzin les execucions dels diferents classificadors KNN s'han agrupat els resultats amb l'objectiu de contrastar els resultats, ja que aquest cop és disposar de més temps per realitzar configuracions, s'han realitzat les diferents proves en les quals únicament es modifiquen els paràmetres que rep el classificador KNN.

En la següent taula es poden veure els resultats de totes les proves utilitzant el conjunt de característiques WTCC extretes utilitzant l'alternativa de México i els paràmetres que s'han utilitzat:

Sistema d'extracció	Paràmetres	Descripció	Precisió
WTCC México	n_neighboirs = 2 weights = "uniform" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns molt petita.	0,505
WTCC México	n_neighboirs = 2 weights = "uniform" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan ja que p es igual a 1 i amb una mostra de veïns molt petita.	0.502
WTCC México	n_neighboirs = 20 weights = "uniform" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns mes gran respecte la primera prova.	0.494
WTCC México	n_neighboirs = 20 weights = "uniform" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan, ja que p es igual a 1 i amb una mostra de veïns idèntica a la prova 3.	0.493
WTCC México	n_neighboirs = 1.000 weights = "distance" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns gran respecte. També s'ha modificat el pes dels veïns. També s'ha modificat el pes a dels veïns a "distance".	0.467
WTCC México	n_neighboirs = 1.000 weights = "distance" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan ja que p es igual a 1 i amb una mostra de veïns molt mes gran respecte la prova anterior. També s'ha modificat el pes a dels veïns a "distance".	0.467
WTCC México	n_neighboirs = 1.000 weights = "uniform" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan ja que p es igual a 1 i amb una mostra de veïns molt mes gran respecte la prova anterior.	0.49
WTCC México	n_neighboirs = 1.000 weights = "uniform" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns molt mes gran respecte la prova anterior.	0.494

En la següent taula es poden veure els resultats de totes les proves utilitzant el conjunt de característiques WTCC extretes utilitzant l'alternativa de Morlet i els paràmetres que s'han utilitzat:

Sistema d'extracció	Paràmetres	Descripció	Precisió
WTCC Morlet	n_neighboirs = 2 weights = "uniform" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns molt petita.	0,503
WTCC Morlet	n_neighboirs = 2 weights = "uniform" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan ja que p es igual a 1 i amb una mostra de veïns molt petita.	0.503
WTCC Morlet	n_neighboirs = 20 weights = "uniform" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns mes gran respecte la primera prova.	0.498
WTCC Morlet	n_neighboirs = 20 weights = "uniform" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan, ja que p es igual a 1 i amb una mostra de veïns idèntica a la prova 3.	0.494
WTCC Morlet	n_neighboirs = 1.000 weights = "distance" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns molt mes gran respecte la prova anterior. També s'ha modificat el pes dels veïns. També s'ha modificat el pes a dels veïns a "distance".	0.468
WTCC Morlet	n_neighboirs = 1.000 weights = "distance" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan ja que p es igual a 1 i amb una mostra de veïns molt mes gran respecte la prova anterior. També s'ha modificat el pes a dels veïns a "distance".	0.468
WTCC Morlet	n_neighboirs = 1.000 weights = "uniform" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan ja que p es igual a 1 i amb una mostra de veïns molt mes gran respecte la prova anterior.	0.491
WTCC Morlet	n_neighboirs = 1.000 weights = "uniform" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns molt mes gran respecte la prova anterior.	0.491

En la següent taula es poden veure els resultats de totes les proves utilitzant el conjunt de característiques MFCC i els paràmetres que s'han utilitzat:

Sistema d'extracció	Paràmetres	Descripció	Precisió
MFCC	n_neighboirs = 2 weights = "uniform" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns molt petita.	0,506
MFCC	n_neighboirs = 2 weights = "uniform" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan ja que p es igual a 1 i amb una mostra de veïns molt petita.	0.504
MFCC	n_neighboirs = 20 weights = "uniform" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns mes gran respecte la primera prova.	0.495
MFCC	n_neighboirs = 20 weights = "uniform" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan, ja que p es igual a 1 i amb una mostra de veïns idèntica a la prova 3.	0.495
MFCC	n_neighboirs = 1.000 weights = "distance" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns molt mes gran respecte la prova anterior. També s'ha modificat el pes dels veïns. També s'ha modificat el pes a dels veïns a "distance".	X
MFCC	n_neighboirs = 1.000 weights = "distance" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan ja que p es igual a 1 i amb una mostra de veïns molt mes gran respecte la prova anterior. També s'ha modificat el pes a dels veïns a "distance".	X
MFCC	n_neighboirs = 1.000 weights = "uniform" metric = "minkowski" p = 1	Aquesta prova es realitzar amb la distancia Manhattan ja que p es igual a 1 i amb una mostra de veïns molt mes gran respecte la prova anterior.	X
MFCC	n_neighboirs = 1.000 weights = "uniform" metric = "minkowski" p = 2	Aquesta prova es realitzar amb la distancia Euclidiana ja que p es igual a 2 i amb una mostra de veïns molt mes gran respecte la prova anterior.	X

* Les precisions definides com a "X" es deu pel fet que els càlculs no han finalitzat.

Tal com es pot veure en les taules cada prova a constat en augmenta el nombre de veïns (n_neighbors) que es tenen en compte, modificant la distancia (mètric) per calcular els pesos o l'algoritme (weights) que permet modificar els pesos que tenen els veïns. Aquests resultats són els que s'han utilitzat per elaborar la comparativa de les conclusions.

3.2 Problemes durant l'execució de l'experiment

En aquest apartat s'exposaran els problemes que s'han trobat durant l'execució de l'experiment i quines estratègies o solucions s'han aplicat per tal de poder prosseguir. No es tenen tots aquells problemes resultants de la validació de codis, ja que tots aquests s'han resolt amb l'ús de la documentació oficial i analitzant l'execució del codi..

3.2.1 Problema 1: Memòria RAM

El problema més gran amb diferencia durant l'experiment ha estat la quantitat de limitacions donades pel Hardware disponible, en aquest cas ens centrarem amb el problema de no disposa suficient memòria RAM per a processar totes les dades.

Per tal de solucionar això s'ha desenvolupat tot el codi de tal forma que les diferents fases del codi puguin executar-se sense la necessitat de tenir la fase anterior en la memòria RAM.

Les úniques fases que si que necessiten tindre les dades carregades en memòria es la de l'extractor de dades per poder realitzar les classificacions i en el moment que es computen les característiques necessàries pel KNN.

Tot i alliberar la memòria entre fases segons el nombre d'elements o segons quina configuració del classificador s'utilitza es pot arribar a generar una demanda superior a la memòria RAM disponible en el Hardware.

Per tal de poder solucionar això s'ha reduït el nombre d'elements a processar, per poder trobar la configuració d'elements ideal s'han fet proves de càlcul amb diferents conjunts començant amb el màxim de dades i anar reduint les mostres de cada instrument de 10 en 10 fins, en el moment que es tenien 50 mostres en comptes de reduir més les mostres es redueix el nombre d'instruments i tornem a les 100 mostres originals dels instruments restants.

Seguint aquest mètode s'ha vist com el màxim de dades amb les quals es podria fer l'experiment és únicament amb 2 instruments amb un total de 60 mostres per cada un d'ells d'un màxim de 20 segons. Totes les altres combinacions quan es tractava de llança el classificador KNN amb nombres veïns elevats, acabar generant una demanda de RAM molt superior als 16 GB disponibles.

Amb la reducció del conjunt de dades a part de solucionar el problema de la RAM s'ha vist que en cap moment l'ús del processador sobrepassa el 80 % (exceptuant pics), cosa que ens permet calcular diferents proves sense problemes de temperatura.

3.2.2 Problema 2: Obtenció de les dades

A causa de les restriccions de com han de ser les dades crues per realitzar l'experiment i la dificultat d'obtenir pistes musicals sense drets d'autor, no ha estat possible generar un conjunt amb totes les mostres desitjades de tal forma que s'ha recorregut a generar les mostres amb diferents segments d'una pista de més duració a la desitjada, de tal forma que una pista d'1 minut podem extreure dues mostres de 25 segons.

Així i tot, no es va poder completar el data set i es va recórrer a la gravació de totes les dades crues restant en un estudi semiprofessional d'un conegut.

A part aquest problema ha estat el que ha fet dissenyar el codi de forma que es puguin generar diferents mides de conjunt de dades segons uns paràmetres, cosa que ens ha permès resoldre amb molta més facilitat el problema anterior i la realització de diferents proves.

3.2.3 Problema 3: Antecedents

En el moment que es va realitzar la cerca d'antecedents es va poder veure ràpidament que tot i que l'objectiu que es persegueix en aquest experiment generar molt d'interès no hi ha molta documentació, en el cas dels algoritmes d'aprenentatge automàtic sense supervisió encara hi ha menys informació relacionada.

Per aquest motiu la recerca de informació s'ha fet en diferents àmbit amb l'objectiu de poder crear una estratègia utilitzant alguna tecnologia o tècnica que de normal no es s'utilitzés per resoldre el tipus de problema com a tal.

3.3 Conclusions

Per tal de poder realitzar una comparativa entre els diferents sistemes utilitzats s'ha elaborat la següent taula en la qual es poden comparar els resultats de precisió utilitzant diferents paràmetres, aquesta taula ens permet veure de forma visual quin sistema ha donat una millor precisió utilitzant els mateixos paràmetres:

Paràmetres utilitzats	Precisió WTCC México	Precisió WTCC Morlet	Precisió MFCC
n_neighboirs = 2 weights = "uniform" metric = "minkowski" p = 2	0,505	0,503	0,506
n_neighboirs = 2 weights = "uniform" metric = "minkowski" p = 1	0.502	0.503	0.504
n_neighboirs = 20 weights = "uniform" metric = "minkowski" p = 2	0.494	0.498	0.495
n_neighboirs = 20 weights = "uniform" metric = "minkowski" p = 1	0.493	0.494	0.495
n_neighboirs = 1.000 weights = "distance" metric = "minkowski" p = 2	0.467	0.468	0.464
n_neighboirs = 1.000 weights = "distance" metric = "minkowski" p = 1	0.467	0.468	X
n_neighboirs = 1.000 weights = "uniform" metric = "minkowski" p = 1	0.49	0.491	X
n_neighboirs = 1.000 weights = "uniform" metric = "minkowski" p = 2	0.494	0.491	X

Partint dels resultats de la Prova 3 plasmats en la taula anterior, no podem assegurar que el sistema WTCC sigui millor que el sistema MFCC, ja que en cap moment cap dels dos sistemes a donant uns resultats significativament superiors respecte a l'altre.

Per aquest motiu s'ha de descartar la hipòtesi, ja que l'única forma de donar per vàlida la hipòtesi és que les classificacions efectuades amb les característiques WTCC fos d'una precisió significativament alta respecte a la de les MFCC.

Tot i descartar la hipòtesi tampoc es pot dir que el sistema MFCC sigui millor que el WTCC, ja que els resultats obtinguts són molt similars als del WTCC.

Dels resultats també es pot extreure que el classificador KNN no doni millors resultats a causa de la baixa qualitat de les dades, això es pot donar al fet que moltes de les mostres poden tindre el mateix fitxer d'àudio d'origen, que tot i contindré múltiples instruments aquests participen en diferents mostres.

Aquesta és la causa més probable, ja que durant la investigació s'han vist experiments amb una precisió del 90 % aproximadament utilitzant únicament les característiques MFCC [9].

Per poder solucionar aquest problema es podria recórrer a un estudi de gravació i enregistra les 100 mostres dels 10 instruments, però en aquest cas registrant 100 instruments diferents per cada instrument, d'aquesta forma la qualitat de les dades augmentaria i podríem estar més segurs a l'hora de realitzar les proves.

Un motiu pel qual el classificador KNN no ha pogut donar diferents precisions en les proves que s'han realitzat, sobretot en el conjunt de proves 3 segurament es deu al fet que el nombre de veïns que s'ha pogut arribar a computar és molt petit o que les funcions de les distancies Euclidiana o de Manhattan no era la correcta, que la mida de la finestra del cercador de veïns fos molt petita, etc.

Per tal de poder solucionar els problemes anteriors es podria adquirir un servidor en el qual poder computar tot el conjunt de dades i poder realitzar un nombre superior de proves, això si a un cost econòmic considerable.

Una altra possible modificació de l'experiment perquè fos més robust seria incloure diferents sistemes de classificació basats en diferents tecnologies, com una NN o SVM.

Capítol 4: Bibliografia

- [1]https://dl.acm.org/doi/abs/10.1145/3243274.3243311
- [2]https://ieeexplore.ieee.org/document/4436069
- [3]https://ieeexplore.ieee.org/document/4959924
- [4]https://link.springer.com/chapter/10.1007/11790853_35
- [5]https://www.researchgate.net/publication/260387308_LIBXTRACT_A_LIGHTWEIGHT_LIBRARY_FOR_AUDIO_FEATURE_EXTRACTION
- [6]https://www.sciencedirect.com/science/article/abs/pii/S0003682X19308795
- [7]https://scikit-learn.org/stable/
- [8]https://librosa.org/doc/main/index.html
- [9]https://www.researchgate.net/publication/343683111_Classification_of_Muscal_Instruments_using_SVM_and_KNN
- [10]https://www.researchgate.net/publication/355290860_KNN_Classification_with_One-step_Computation
- [11]https://www.upf.edu/web/mtg/music-information-retrieval
- [12]https://www.researchgate.net/publication/261254016_Using_perceptually_defined_music_features_in_music_information_retrieval
- [13]https://ianring.com/musictheory/scales/
- [14]https://www.free-scores.com/index_uk.php
- [15]http://cursodeacusticamusical.blogspot.com/2016/03/capitulo-11-el-timbre.html
- [16]http://www.revistacubanadefisica.org/RCFextradata/OldFiles/2012/Vol29-No1/RCF-29-1-37.pdf
- [17]http://lorien.die.upm.es/barra/pfcs/2007-carmenr/docs/proyecto.pdf
- [18]https://www.researchgate.net/figure/Instrument-Hierarchy-Tree-Categorized-by-the-MPEG-7-audio-descriptor-LogAttackTime-We_fig3_225462978
- [19]https://en.wikipedia.org/wiki/Wavelet_transform
- [20]https://towardsdatascience.com/the-wavelet-transform-e9cfa85d7b34
- [21] https://www.nature.com/articles/s43588-021-00183-z

Annexos

En aquests annexos es poden ubicar els pseudocodis de les funcions implementades per tal de complir els objectius.

Annex I

En aquest annex es detallen les proves que s'han realitzat per efectuar les decisions oportunes a l'hora de realitzar l'experiment, tots els codis esmentats en aquest annex estan ubicats en el fitxer Jupyter NoteBook anomenat "tests".

Aquest NoteBook s'ha utilitzat per generar tot un conjunt de proves que ens han permès detectar des que era necessari fer un anàlisis descriptiu dels components que conformen el resultat de la Transformada de Wavelet per tal d'agafar els components estrictament necessaris.

També ens ha servit per efectuar proves sobre el codi de l'Annex II de tal forma que en els altres codis només s'introduïa codi validat amb un conjunt de dades molt reduït, per tal d'agilitzar el desenvolupament.

Annex II

En aquest annex es pot trobar el pseudocodi de totes les funcions esmentades durant el transcurs de l'experiment.

Annex II.I Mètode: transform_data_to_clean_data

En aquest apartat es pot veure el pseudocodi utilitzar per poder passar les Dades Crues a Dades Preparades tenint en compte les restriccions de la duració dels segments, el nombre de mostres o el nombre d'instruments que participen el conjunt de dades:

1. S'eliminen totes les dades que puguin existir en el directori on es guarden els resultats
2. Utilitza un bucle for per recórrer a tots els fitxers musicals que conformen les Dades Crues.
    a. S'utilitza un bucle for per recórrer les dades de cada un dels fitxers.

i. Utilitza un condicional per comprovar si tenim el nombre de bucles necessaris i si l'instrument existeix en el diccionari.
ii. Es segmentar el fitxer, es transforma en mono i es redimensionar per tindre una freqüència de mostreig de 22050.
iii. Es guarda en la carpeta del instrument en curs

Aquesta funció està ubicada al fitxer Jupyter Notebook "data_generator".

Annex II.II Mètode: save_track_info

Aquesta funció en permet crear un CSV de control anomenat “loop_info.csv” que s’utilitzarà per generar el conjunt de Dades Finals, partint de les dades Preparades:

1. S’utilitza un bucle for per recórrer a través de cada arxiu en els subdirectoris en el que s’han guardat les dades generades per la funció del Annex II.I
    a. Afegeix informació sobre cada arxiu al diccionari loop_dict
2. Utilitza Pandas per convertir el diccionari a un Dataframe
3. Utilitza Pandas per desar el Dataframe a un arxiu CSV anomenat “loop_info.csv”

Aquesta funció està ubicada al fitxer Jupyter Notebook "data_generator".

Annex II.III Mètode: create_data

Aquesta funció permet combinar totes les Dades Preparades i generar les Dades Finals utilitzant la informació emmagatzemada en el CSV de control anomenat “loop_info.csv”:

1. Es Crea un diccionari buit anomenat dict_loops amb les claus "id", "id_inst_1", "tag_inst_1", "tag_inst_1_start", "tag_inst_1_end", "id_inst_2", "tag_inst_2", "tag_inst_2_start", "tag_inst_2_end" i "path". Aquestes claus contenen una llista dels valors pertinents de cada combinació i ens permeten etiquetar les dades posteriorment.
2. Utilitza un bucle for per eliminar tots possibles arxius de proves anteriors en el directori especificat.
3. Utilitza un doble bucle for per recórrer a través de totes les mostres del CSV “loop_info”
    a. Utilitza un condicional per comprovar si la connexió entre dos elements del CSV ja existeix, per tal d’evitar repeticions.

b. Utilitza LibRosa per carregar dos instruments, transformar-los a mono i eliminar els silencis de les mostres.
c. Concatena les mostres dels dos instruments actuals
d. Si volem una component de soroll s'afegirà en aquest moment.
e. S'emmagatzema el nou arxiu de so
f. Afegeix informació sobre els arxius concatenats al diccionari dict_loops

Aquesta funció està ubicada al fitxer Jupyter Notebook "data_generator".

Annex II.IV Mètode: extract_mfcc

Aquesta funció permet extreure les característiques MFCC de totes les finestres que conformen les Dades Finals, de tal forma que retorna un diccionari amb totes les etiquetes i una matriu amb totes les dades MFCC.

1. Llegir el fitxer CSV anomenat "nom_de_la_fase_loop_info"
2. Crear un diccionari per a les etiquetes de tots els instruments
3. Crear un diccionari per a les dades dels MFCC
4. Per cada pista de la llista de pistes:
    a. Calcular la longitud de la finestra i establir-ho com el primer bucle
    b. Per cada bucle dins de la pista:
        i. Comprovar si la finestra té només un instrument
            1. Si ho fa, tallar la secció i calcular el MFCC amb 10 característiques
            2. Normalitzar les característiques
            3. Si no hi ha valors NaN,
                a. Afegir l'etiqueta correcta al diccionari d'etiquetes i afegir les dades del MFCC al diccionari de dades del MFCC
5. Crear una matriu buida per a les dades del MFCC
6. Per cada clau en el diccionari de dades del MFCC:
    a. Afegir les dades del MFCC a la matriu
7. Retornar el diccionari d'etiquetes i la matriu de dades del MFCC

Aquesta funció està ubicada al fitxer Jupyter Notebook "data_extratcor_and_knn".

Annex II.V Mètode: extract_wtcc

Aquesta funció permet extreure les característiques WTCC de totes les finestres que conformen les Dades Finals, de tal forma que retorna un diccionari amb totes les etiquetes i una matriu amb totes les dades WTCC.

1. Llegir el fitxer CSV anomenat “nom_de_la_fase_loop_info”
2. Crear un diccionari per a les etiquetes de tots els instruments
3. Crear un diccionari per a les dades dels CWT
4. Per cada pista de la llista de pistes:
    a. Calcular la longitud de la finestra i establir-ho com el primer bucle
    b. Per cada bucle dins de la pista:
        i. Comprovar si la finestra té només un instrument
            1. Si ho fa, tallar la secció i calcular el CWT
            2. Normalitzar les característiques
            3. Si no hi ha valors NaN
                a. Crear un objecte PCA amb 10 components
                b. Ajustar els dades al objecte PCA i transformar-los en les noves característiques de menor dimensió
                c. Afegir l'etiqueta correcta al diccionari d'etiquetes i afegir les dades del CWT al diccionari de dades del CWT
5. Crear una matriu buida per a les dades del CWT
6. Per cada clau en el diccionari de dades del CWT:
    a. Afegir les dades del CWT a la matriu
7. Retornar el diccionari d'etiquetes i la matriu de dades del CWT

Aquesta funció està ubicada al fitxer Jupyter Notebook "data_extratcor_and_knn".

Annex II.VI Mètode: class_knn

Aquesta funció permet crear un classificador KNN amb les dades desitjades i amb la configuració necessària per realitzar l'experiment:

1. Crear un DataFrame amb les dades i les etiquetes utilitzant el diccionari desitjades.
2. Dividir el DataFrame en les dades i les etiquetes
3. Dividir les dades en conjunts d'entrenament i prova
4. Crear un objecte de classificador KNN amb K veïns i la configuració definida pels paràmetres de la funció
5. Entrenar el classificador amb les dades d'entrenament
6. Fer prediccions en el conjunt de prova
7. Avaluar el rendiment del model
8. Mostrar la precisió del model.

Aquesta funció està ubicada al fitxer Jupyter Notebook "data_extratcor_and_knn".

Annex II.VII Estructura del directori

En aquest apartat es mostra l'estructura de directoris amb els fitxers corresponents necessaris per a l'execució del programa:

1. _Code/
    a. _Data_Folder
        i. _Clean_data
            1. Accordion
            2. Clarinet
            3. ElectricGuitar
            4. Flute
            5. GuitarAcoustic
            6. Harp
            7. Pianon
            8. Saxophone
            9. Trumpet
            10. Violin
        ii. _Data

1. Accordion
2. Clarinet
3. ElectricGuitar
4. Flute
5. GuitarAcoustic
6. Harp
7. Pianon
8. Saxophone
9. Trumpet
10. Violin

iii. PH1_Tracks
1. Totes les combinacions creades sense soroll

iv. PH2_Tracks
1. Totes les combinacions creades amb soroll

b. 0_Data_Generator/
i. data_generator.ipynb
ii. loop_info.csv
iii. PH1_loop_info.csv
iv. PH2_loop_info.csv
v. pre_track_info.csv
vi. tests.ipynb

c. 1_Data_Extractor/
i. data_extractor_and_knn.ipynb
ii. PH1_Track_Labels.csv
iii. PH1_Tracks_mfcc_data.csv
iv. PH1_Tracks_mfcc_labels.csv
v. PH1_Tracks_wtcc_data.csv
vi. PH1_Tracks_wtcc_labels.csv

Tots els directoris que tenen els noms dels instruments contenen les Dades Crues (_Data) o les Dades Preparades (_Clean_Data).