Report

Machine Learning Project

Traditional Iranian Music Dastgāh Prediction

Sina Pirmoradian 810101125

Zahra Reihanain 810101177

Fatemeh Taherinejad 810101219

Fatemeh Nadi 810101285

January, 2023

Contents

1 Introduction

In traditional Persian music, it is not only the genre that can affect people's choice of listening but also the dastgāh. Dastgāh is the standard music system in Persian art music and is mainly categorized to seven categories: Shur, Segah, Mahur, Homayun, Rast-panjgah, Nava and Chahargah. Many music listeners create paylists based on their favorite dastgāh without the use of recommender applications. Therefore, music dastgāh classification forms a basic step for creating a strong recommendation system. The idea behind this project is to see how to handle music files in python, compute and extract different features from them, run machine learning algorithms on the extracted features to see the results. In other words, the main aim of the project is to create a machine learning model, which classifies music samples into different dastgāhs.

2 Dataset

To run different machine learning model and to build the explained model, there is the need for data. A collection of 949 audio files were collected (this is what we got until 8/11/1401) with different lengths and in different dastgāhs. These music pieces were not only sorted based on dastgāh, but also categorized into different instruments. These instruments are the most popular traditional instruments of Persian music: tar, Kamancheh, Santur, Setar, nay and multiple instruments. Therefore, the collected dataset can not only be used to demonstrate the effect of instrument in classifying each dastgāh, but also to detect each instrument based on other features of a music piece. Music itself, can not be used to build models and in order to train and test models, useful features must be extracted from each audio signal.

3 Introduction to dastgāh

۱ مقدمه

شناخت دستگاه‌های ایرانی را به دو بخش می‌توان تقسیم کرد. در وحله اول اصلا چه المان‌هایی در شکل‌گیری مفهوم دستگاه نقش دارند، و دوم این‌که هر کدام از دستگاه‌ها چه ویژگی‌هایی دارند و چطور می‌توان از هم تمیز داده شوند.

وقتی می‌گوییم مثلاً اگر آهنگ به "بادا بادا مبارک بادا" بخورد "چهارگاه" است؛ یا اگر به "لی لی لی لی حوضک" بخورد "افشاری" و یا اگر به مرغ سحر، "ماهور" است. مسئله این‌جاست که وقتی می‌گوییم بخورد منظور چیست؟

اگه هر دستگاه را مثل یک خانواده ببینیم، زمانی که چند آهنگ در یک خانواده هستند، لزوما بدین معنا نیست که سریع می‌توان آن‌ها را از هم تشخیص داد. ممکن است در ظاهر مشخص نباشد که دو آهنگ برای یک دستگاه هستند. در کل می‌توان گفت تشخیص این هم‌خانواده بودن آن‌قدرها سخت نیست. البته آن‌قدرها آسان هم نیست.

در ادامه به بررسی مفهوم دستگاه می‌پردازیم اما ساده‌ترین تعریف این است که "دستگاه یکی از چندین نظام طبقه‌بندی موسیقی‌هاست"

توجه به این نکته ضروری است که طبقه‌بندی یک مفهوم نسبی است. چون خیلی از مفاهیم از یک سری جهات با هم متفاوت هستند و از برخی جهات شبیه به هم. به همین دلیل است که به طور ناگزیر، همیشه طبقه‌بندی از زاویه‌دید معیّنی صورت می‌گیرد.

زمانی که موسیقی را بر اساس چشم‌انداز دستگاه‌ی تفکیک می‌کنیم، نمی‌توانیم در مورد محتوای عاطفی آن صحبت کرد. باید توجه داشت زمانی که یک آهنگ حس غم یا شادی را در ما القا می‌کند، بیشتر به‌خاطر شعر و متن آن موسیقی است که به آن ضمیمه شده است. برای مثال وقتی می‌گوییم مثلا دستگاه الف شاد است یا دستگاه ب عاشقانه است، بیشتر این موضوع به تکرار تجربیاتی برمی‌گردد که در آن دستگاه صورت گرفته‌است و این موضوع باعث شده‌است هر کدام از دستگاه‌ها هاله‌ی عاطفی تداعی‌گری پیدا کنند.

همان‌طور که پیش‌تر گفتیم، طبقه‌بندی مطلقی وجود ندارد. مثلا بیات ترک از متعلقات دستگاه شور شناخته می‌شود اما بسیاری از خصوصیات آن شبیه دستگاه ماهور است و یا مثلا از یک زاویه‌ی دید دیگر ممکن است شباهت‌های زیادی بین سه‌گاه و چهارگاه قابل بیان باشد اما از یک زاویه‌ی دیگر اصلا به هم شباهتی نداشته باشند. برای همین است که گفته می‌شود در این موضوع نباید صفر و یک به قضیه نگاه کرد.

طبقه‌بندی‌ها همیشه بازتاب‌دهنده‌ی دانش و معرفت انسان‌های یک جامعه در یک دوره‌ی معین به موضوعاتی است که در آن دوره برای آن‌ها از اهمیت بالایی برخوردار بوده است و به فراخور نیاز به وجود می‌آیند. از طبقه‌بندی دستگاه‌های ایران و تقسیم آن‌ها به هفت دستگاه اصلی و پنج شعبه‌ی فرعی که غالباً آواز نامیده می‌شوند، ۱۵۰ سال بیشتر نمی‌گذرد.

این طبقه‌بندی دستگاه‌ی نشانگر نوع موسیقی‌ی می‌باشد که در آن زمان رایج بوده است؛ و فواید زیادی داشته است؛ ازجمله این که به موضوعات نظم داده است و باعث شده، مفاهیم جدیدی به وجود بیایند. به دین معنا که هم به موسیقی و روند آموزش آن روال داده و هم این‌که به آفرینش موسیقی جدید کمک کرده است.

۲ دستگاه

دستگاه که از ترکیب دو واژه "دست" و "گاه" به معنای مکان/ زمان و نغمه تشکیل شده است، اصطلاحی در موسیقی سنتی ایرانی است که به مجموعه‌ایی از چند نغمه (گوشه) اتلاق می‌شود که با هم در گام، فواصل و نُت هماهنگی دارند.

به بیانی دیگر می‌توان گفت دستگاه از به هم پیوستن چند مقام شکل می‌گیرد و از این چند مقام، مقامی را به عنوان مقام اصلی، نقش شروع، خاتمه و هم‌چنین رابطه میان مقام‌ها را بر عهده دارد. گوشه‌ها درمقام‌های مختلف ساخته‌شده‌اند و براساس نظم خاصی به هم پیوند خورده‌اند و حس و حالی را به شنونده القا می‌کند. یکی ازنتایج این نظم خاص این است که گذر از یک مقام به مقامی دیگر در درون یک دستگاه، به شکل خوشایند صورت می‌گیرد. در همهٔ دستگاه‌ها یک مقام مادر وجود دارد که نام خود را به دستگاه می‌دهد. از طرفی کلمه‌ی "دستگاه" می‌تواند به مفهوم محل قرار گرفتن دست نوازنده روی دسته ساز باشد[۱۰].

قدمت واژه دستگاه به دوران قاجار، زمانی که علی اکبر خان فراهانی گوشه‌های مختلف را جمع‌آوری و به نظم درآورد برمی‌گردد [۱]. به‌طور سنتی، این گوشه‌ها معمولاً در یک قالب دایره‌ای پنج قسمتی اجرا می‌شوند که شامل پیش‌درآمد، چهارمضراب، آواز، تصنیف و رنگ است [۲]. موسیقی سنتی شامل هفت دستگاه شور، سه‌گاه، ماهور، همایون، راست پنج‌گاه، نوا و چهارگاه می‌باشد که در ادامه به شرح مختصری از هرکدام پرداخته خواهد شد.

۳ دستگاه شور

دستگاه شور را مهم‌ترین دستگاه موسیقی ایرانی می‌دانند [۳] چرا که از آن پنج آوار مهم منشعب شده و اثر آن در دستگاه‌هایی نظیر نوا و سه‌گاه نیز مشهود است. در تقسیم‌بندی ارائه شده توسط داریوش طلایی که تمام دستگاه‌ها را با کمک چهار دانگ اصلی توضیح می‌دهد، دانگ اول دستگاه شور یکی از این دانگ‌های بنیادین است [۴]. آوازهای شور از کمرکش دستگاه آغاز شده و هر کدام با ایست و تاکید بر درجه‌ای خاص شکل گرفته و به لحاظ فضای نغمگی و سیر مدال، استقلال اجرایی خاص خود را دارا می‌باشند [۵]. این آوازها هر کدام به نوعی با جغرافیا و فرهنگی خاص در ارتباط است. این آواز ها شامل ابوعطا، بیات ترک، افشاری، بیات کرد و دشتی می‌شوند.

دستگاه شور شامل سه جایگاه اجرایی بوده که به ترتیب شور پایین، شور وسط، و شور بالا (یا اوج) نامیده می‌شوند. شور پایین وسعتی برابر یک فاصله چهارم درست دارد، یعنی همان دانگ اول دستگاه شور. درآمد شور در این منطقه اجرا می‌شود. شور وسط دانگ دوم دستگاه شور را می‌سازد و به دانگ اول متصل است به گونه‌ای که نت آخر دانگ اول، نت اول دانگ دوم است. شور بالا (اوج) از درجه هشتم آغاز می‌شود و دانگ سوم دستگاه شور را تشکیل می‌دهد. بین دانگ سوم (شور بالا) و دانگ دوم (شور وسط) یک پرده فاصله است؛ به عبارتی دیگر، دانگ سوم همان دانگ اول است با این تفاوت که یک اکتاو بالاتر اجرا می‌شود.

گام شور از نظر مُد کوچک بوده چرا که در درجه سوم، ششم و هفتم در مد شور به نسبت نت پایه دارای دارای فواصل کوچک‌تری هستند و می‌توان آن را با گام مینور غرب مقایسه نمود. گام شور پایین‌رونده بوده به این معنا که روند ملودی‌های شور اکثراً از بالا به پایین است [۶]. درجه پنجم گام شور، نت متغیر است. این درجه با فاصله یک پرده از درجه چهارم تعریف می‌شود اما در برخی گوشه‌های این درجه می‌تواند یک ربع پرده پایین‌تر آورده شود. اما در برخی گوشه‌ها و ملحقات دیگر شور نت پنجم متغیر نیست و هرگز پایین آورده نمی‌شود. هرگاه نت پنجم گام شور را ربع پرده کم کنیم، گام شور تبدیل به گام سه‌گاه می‌شود.

۴ دستگاه سه‌گاه

دستگاه سه‌گاه را به لحاظ تاثیر حسی و عاطفی حزن انگیز و مغموم دانسته‌اند که در از نظر درجات با مقام راست ارتباط دارد. بااهمیت‌ترین گوشه‌ها در دستگاه سه‌گاه عبارت‌اند از درآمد، زنگ شتر، زابل، مویه، حصار، مخالف و مغلوب که در دستگاه چهارگاه نیز گوشه‌هایی با همین نام وجود دارد اما مد در این دو دستگاه کاملا متفاوت است. از منظر مد دستگاه سه‌گاه با آواز افشاری اشتراکات و مشابهت‌هایی دارد اما تفاوت آن‌ها در استقلال ردیف است. هم‌چنین گوشه‌های مخالف و مغلوب اوج دستگاه سه‌گاه را تشکیل می‌دهند و از نظر مد به بیات اصفهان نزدیک هستند [۳]. گوشه زابل در دستگاه راست‌پنج‌گاه و همایون نیز به کار گرفته می‌شود و امکان پرده‌گردانی به دستگاه شور را نیز فراهم می‌کند [۶]. سه‌گاه در درجات خود دارای دو ربع پرده است که در دستگاه های همایون و شور، یک ربع پرده است. هم‌چنین، تونیک این گام همیشه ربع پرده است.

۵ دستگاه همایون

دستگاه همایون با دستگاه شور از لحاظ فواصل دانگ اول ارتباطی نزدیک دارد. هم‌چنین با دستگاه‌های نوا، سه‌گاه، چهارگاه نیز به واسطه پرده گردانی گوشه‌های دیگر خود دارای وجوه اشتراک است. در شناخت دستگاه همایون همین بس که بزرگان موسیقی آن را تداوم زندگی، اتحاد عشق و عاشق و معشوق و توحید خوانده‌اند و به دلیل این حس زیبای دستگاه همایون، آن را " دستگاه عشاق " می‌نامند [۷].

گام در دستگاه همایون با گام دستگاه شور اندکی نزدیکی دارد. با این تفاوت که گام همایون در درجه سوم از گام شور در همین درجه، نیم‌پرده بالاتر بوده و در همین نت حالت همایون بیان می‌گردد. از آنجا که نت شاهد همایون در درجه دوم است و از چند نت زیرگام برای اجرای آواز استفاده می‌شود ، پس حالت صعود محسوسی ندارد و از این جهت نیز مانند گام شور پایین رونده است [۷].

موسیقی‌دانان دستگاه همایون را از لحاظ فواصل و دانگ‌ها به دستگاه موسیقی سه گاه نیز شبیه می‌دانند. از نظر ساختارشناسی دستگاه همایون، دانگ های گام همایون یک‌سان نیستند زیرا دانگ اول همایون از یک دوم نیم بزرگ و یک دوم بیش بزرگ و یک دوم کوچک و دانگ دوم همایون از یک دوم کوچک و دو دوم بزرگ تشکیل شده اند.

دستگاه همایون از لحاظ پرده گردانی نیز با دستگاه موسیقی چهارگاه بی‌شباهت نیست. در گام ششم دستگاه همایون با افزودن نیم پرده به نت ششم و کم کردن ربع پرده از نت پنجم تداعی کننده برخی از گوشه های دستگاه چهارگاه است که عکس همین عمل در دستگاه چهارگاه امکان پذیر است و به گذر از چهارگاه به همایون منتهی می‌شود. دستگاه همایون از منظر مرکب خوانی (یعنی ورود به دستگاه‌های موسیقی دیگر) نیز همانند دستگاه شور و دستگاه سه‌گاه است [۷].

۶ دستگاه ماهور

دستگاه ماهور با دارا بودن حدود ۵۰ گوشه‌ی متنوع در میان دستگاه‌های موسیقی ایران شاخص است. تکثر گوشه‌های این دستگاه این امکان را برای هنرمند ایجاد می‌کند تا بتواند از گوشه ای به دستگاهی دیگر وارد شود و به اصطلاح پرده گردانی به تناوب در اجرای دستگاه ماهور رخ دهد.

ویژگی «پرده گردانی» از خصوصیات اساسی دستگاه ماهور است و گوشه ها اجزاء اصلی این دستگاه موسیقی را تشکیل می دهند [۸]. در ردیف میرزا عبدالله به روایت نورعلی برومند دستگاه ماهور ۴۴ گوشه دارد. حال آن که در ردیف موسی معروفی ۵۸ گوشه برای آن آمده است.

دستگاه ماهور با دستگاه‌های دیگر به ویژه راست‌پنج‌گاه و نوا چند گوشه مشترک دارد که اصفهانک یکی از آن‌هاست [۸]. دستگاه ماهور در دانگ اول دارای دو پرده و یک نیم‌پرده است. در دانگ دوم نیز دو پرده و یک نیم‌پرده وجود دارد که یک پرده‌ی کامل (دوی اکتاو) آن را تکمیل می‌کند [۹].

۷ دستگاه نوا

دستگاه نوا دارای ملودی‌های گوناگون است هر چند گام و فواصل آن از جهاتی کاملا با شور مطابق است ولی در نهایت استقلال ملودی دارد آواز نوا در شروع از ناحیه بم آغاز می‌شود و کم کم به قسمت‌های زیر می‌رود. روند ملودی نوا مایل به طرف نت‌های پایین رونده گام (قبل از تونیک) است. [۱۱]

نوا آوازی است در حد اعتدال و آهنگی ملایم و متوسط دارد، نه زیاد شاد و نه زیاد حزن انگیز. نوا را آواز خوب گفته‌اند و معمولا در اخر مجلس می‌نواختند. معمولا اشعار عاشقانه مثل اشعار حافظ را برای نوا انتخاب می کنند، زیرا تاثیر بسیار زیادی در شنونده ایجاد می کند. [۱۰]

گوشه های مهم دستگاه نوا گردانیه، بیات راجع، عشاق، نیشابورک، نهفت می‌باشد.

بستر صوتی دستگاه نوا به صورت زیر است: [۱۲]

شکل ۱: بستر صوتی دستگاه نوای سل

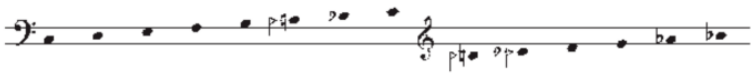
۸ دستگاه راست – پنج‌گاه

راست‌پنج‌گاه دست کم از نظر نام با هیچ‌یک از مقام‌های موسیقی سنتی ایران مطابقت ندارد و به عقیده برخی، نام راست و پنج‌گاه هر دو نام‌های تازه‌تری بین مقام‌های موسیقی هستند. دانگ اول دستگاه راست‌پنج‌گاه با مقام راست ارتباط و با دستگاه ماهور مطابقت دارد. این دستگاه، امکان پرده‌گردانی به تمام دستگاه‌های دیگر را فراهم می‌کند و برخی معتقدند که هدف از ایجاد این دستگاه، آموزش مراحل عالی موسیقی از جمله مُرَکّب‌نوازی و مرکب‌خوانی بوده‌است. [۱۳]

راست‌پنج‌گاه در بین دستگاه‌ها از همه کمتر اجرا می شود. راست‌پنج‌گاه ترکیبی است از سایر مقام‌ها و در این آواز می‌توان به تمام مقام‌های مقام‌های ایرانی وارد شد. از این رو می‌توان تمام احساساتی که در دستگاه‌های موسیقی ایرانی است را با راست‌پنج‌گاه ایجاد نمود. راست‌پنج‌گاه آواز کاملی است، زیرا دارای تمام حالات و صفات و آوازهای دیگر است. [۱۰]

گوشه‌های مهم دستگاه راست‌پنج‌گاه، پنج‌گاه، قرچه، ماوراء النهر، بیات عجم، طرز، نفیر و فرنگ می‌باشد.

بستر صوتی دستگاه راست‌پنج‌گاه به صورت زیر است: [۱۲]



شکل ۲: بستر صوتی دستگاه راست‌پنج‌گاهِ فا

۹ دستگاه چهارگاه

این فقط اسم سه‌گاه و چهارگاه نیست که به هم نزدیک است مشابهت‌های فراوانی به هم دارند ولی در عین حال هیچ شباهتی میان‌شان نیست.

چهارگاه تنها دستگاه موسیقی ایرانی است که هیچ جای آن ساختار نیرومند شور حضور ندارد لذا از حیث نظام فواصل شباهتی با هیچ مایه دیگری ندارد. البته در چهارگاه‌های کاملا کلاسیک گوشه مویه تا حدی تداعی‌گر حضور شور است اما در عمل این مقام تا حال چنان نقش دسته چندمی برعهده داشته که می‌توان آن را در عرصه چهارگاه‌های رایج محذوف انگاشت.

در ذهن بسیاری از ایرانیان میان چهارگاه و حماسه ارتباط تنگانگی برقرار است و بهترین انتخاب برای ساخت قطعات و تصانیف ملی است.

گام آن مانند شور و همایون، پایین رونده و مثل گام ماهور و اصفهان بالارونده می‌باشد، چرا که در دو حالت محسوس است. یعنی می‌توان گفت که این گام، مخلوطی از گام سه‌گاه و همایون است و اگر نت دوم و ششم گام ماهور را ربع پرده کم کنیم، تبدیل به چهارگاه می‌شود. در گام چهارگاه همیشه دو علامت نیم پرده برشو و دو علامت ربعی فرو شو با هم وارد شده‌اند و فواصل درجات این گام نسبت به تونیک عبارت‌اند از: دو نیم بزرگ، سوم بزرگ، چهارم درست، پنجم درست، ششم نیم بزرگ، هفتم بزرگ و هنگام، که دانگ‌های آن هم با یک‌دیگر برابرند. نت شاهد (تونیک) این دستگاه نیز در راست کوک «دو» است. حالت آغازین درآمدهای چهارگاه، با نت «لا» بسیار واضح و مشخص است و به این وسیله به راحتی می‌توان آن را از سایر گام‌ها تشخیص داد [۱۲].

4 Cleaning dataset and extracting features

Audio feature extracting is an essential step in audio signal processing. It deals with the processing and utilizing audio signal, removing noise and balancing the time-frequency ranges by conversion. Features that can be extracted from audio signals consist of the most revealing and informative aspects of audio. There are two main types of features that can be extracted from each audio piece:

1. Time domain: these are the features extracted directly from raw audio waveforms.

2. Frequency domain: these features focus on the frequency components of the audio signal. Audio signals are generally converted from time domain to frequency domain using Fourier transforms.

There is also the type where time and frequency domain components are combined. The time-frequency domain representation is obtained by applying Short-Time Fourier transform on the waveform of the audio signal.

4.1 Time domain features

Time domain features mainly use the waveform of the audio. Plotting the waveforms of the different dastgāhs show that there is quiet a difference in some dastgāhs with each other. Some are very dense, such as Shur, Homayun and Chahargah, while others such as Segah and Nava are thinly scattered. Extracting features from these waveforms might end up building a good classifier to distinguish some dastgāhs from each other but might not be as accurate in distinguish some.

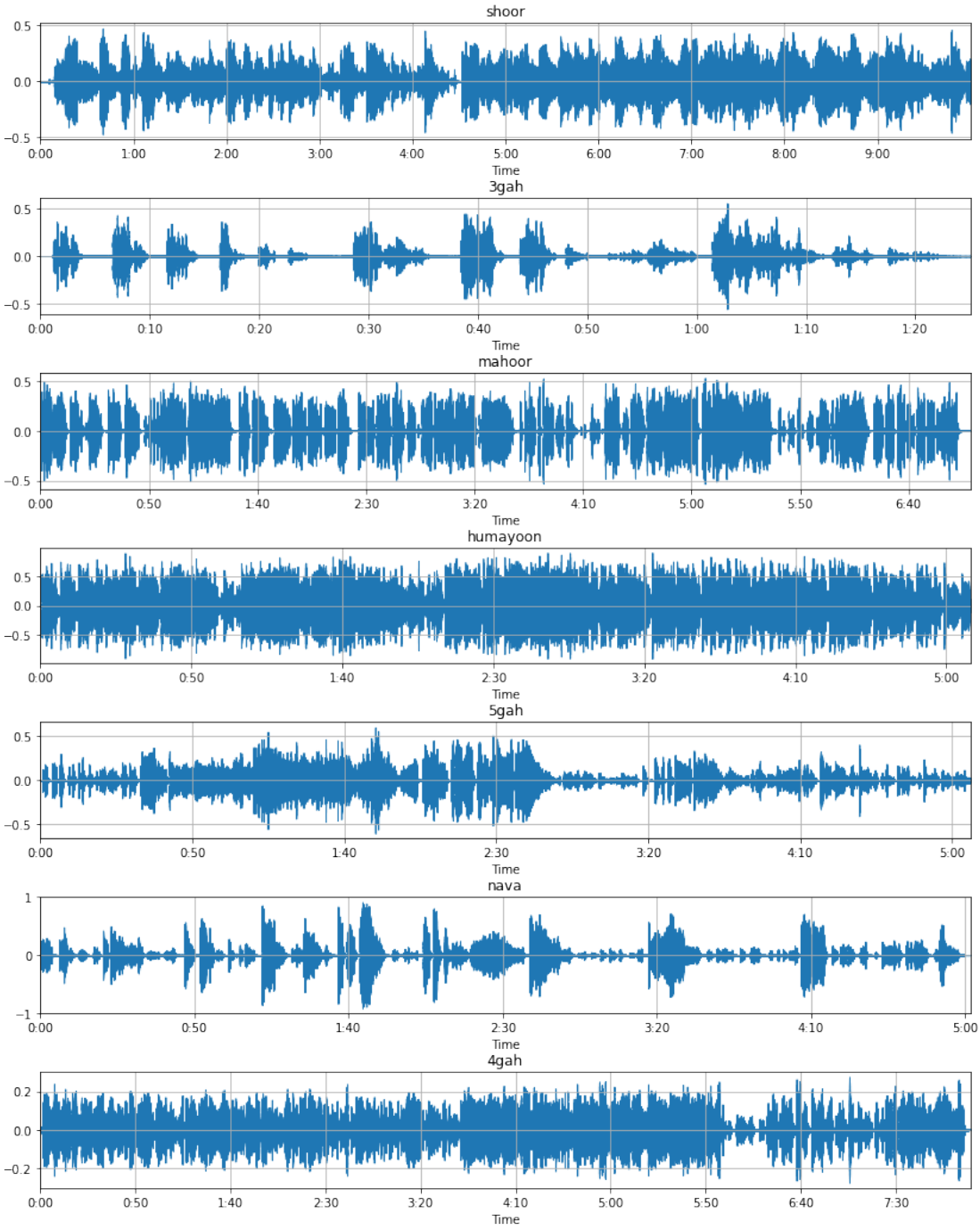


Figure 1: waveform of music signals

4.1.1 Zero crosses

it is the rate of sign-changes along a signal. This is the rate at which the waveform fluctuates the zero line, changing the sign of taken samples from negative to positive or reverse. It usually has higher values for sound produced by percussion. For the five waveforms plotted in figure 1, tthe below is only a short piece of each to better demonstrate the meaning of zero-crossing. This feature is extracted for the whole audio and

total number of zero crosses is saved as a feature.

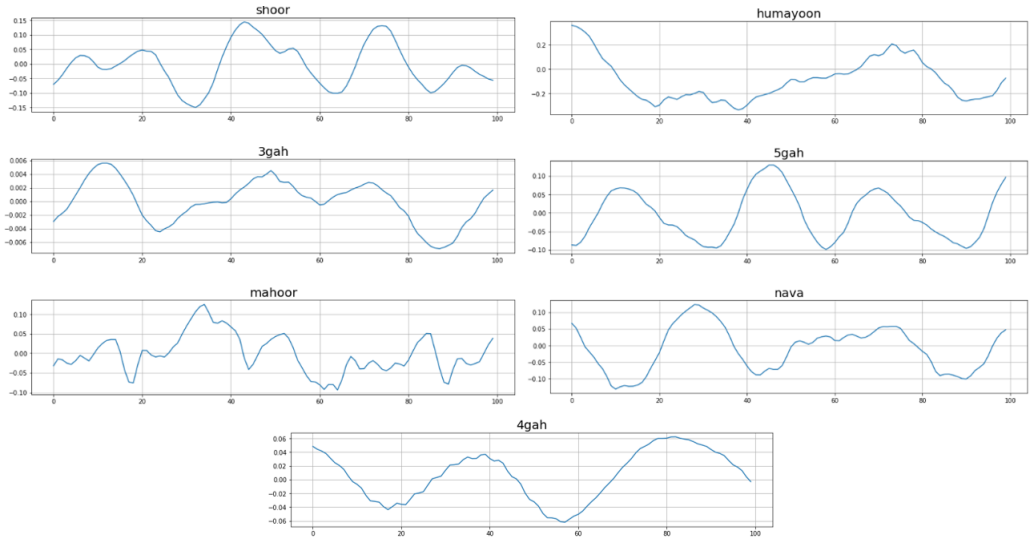


Figure 2: zero crossing of a music in each dastgāh

For the rest of the time domain features, extraction is done based on frames on the sampled audio and frames are selected based on hop length. These two values are the same for all the following features, frame size of 65536 and hop length of 16384. This choice of frame size and hop length results 27 features for each of the following features, for a music of length 20 seconds.

4.1.2 Amplitude Envelope

consists of the maximum amplitudes value among all samples in each frame. This is the type of feature that can best demonstrate the loudness of the audio. It is however, sensitive to outliers. Amplitude envelope is also one of the most used features in music genre classification.

4.1.3 Mean acceleration

each sample taken from a music signal has a value in a range. This feature counts the mean value for the frames of the music piece.

4.1.4 RMS acceleration

Root Mean Square Energy contributes all the samples in a frame and acts as an indicator of the loudness since the higher the energy, the louder the music. It is less sensitive to outliers.

4.1.5 Standard deviation

Is calculated based on the values of samples in a frame and is the square root of the variance.

4.1.6 Variance

It is also calculated based on values in a frame.

4.1.7 Peak acceleration

Defined as the time over which the largest accelerations occur.

4.1.8 Skewness

Calculated the amount of skewness in a frame

4.1.9 Kurtosis

Can be caused by nonlinearity or there's a fault or interfering signal which occurs as not so usual values time after time so that the distribution pattern becomes wider or narrower than what belongs to the expected noise.

4.1.10 Crest factor

Showing the ratio of peak values to the effective value. In other words, crest factor indicates how extreme the peaks are in a waveform. Crest factor is the peak amplitude of the waveform divided by the RMS value of the waveform.

4.1.11 Shape factor

It is dependent of the shape of the signal in the frame but independent of its dimension.

4.1.12 Impulse factor

Compare the height of a peak to the mean level of the signal.

4.1.13 A factor

This features calculated the maximum value of signals within a frame, over the multiplication of standard deviation and variance of frame.

4.1.14 B factor

This feature combines the values of skewness, kurtosis, standard deviation and variance in a similar way to A-factor. This is a total function of features extracted before and may not have much of an impact in classification.

4.2 Frequency domain features

Frequency domain features are the type that can better be demonstrated using plots and therefore give a sense of differences in different dastgāhs. They are usually extracted from the Fourier transform of a time signal and reveal the underlying structures of audios. A spectrogram is a graphic representation of the time-varying spectrum of sound or other signal frequencies. In figure 3, spectrogram of a music piece from each dastgāh is plotted in the log transformed manner as values are mostly close to zero and plotting the log form is more reasonable. Similar and different patterns can be seen within the spectrogram of each music but the one that really stands out is the spectrogram of segah. It takes much lower values compared to other dastgāhs. This difference was not at all obvious in the waveform plots. Other frequency domain features can also be sort of plotted and the differences can be observed via the plots. It is also worthy to mention that the mean and variance of the mentioned features will be saved.

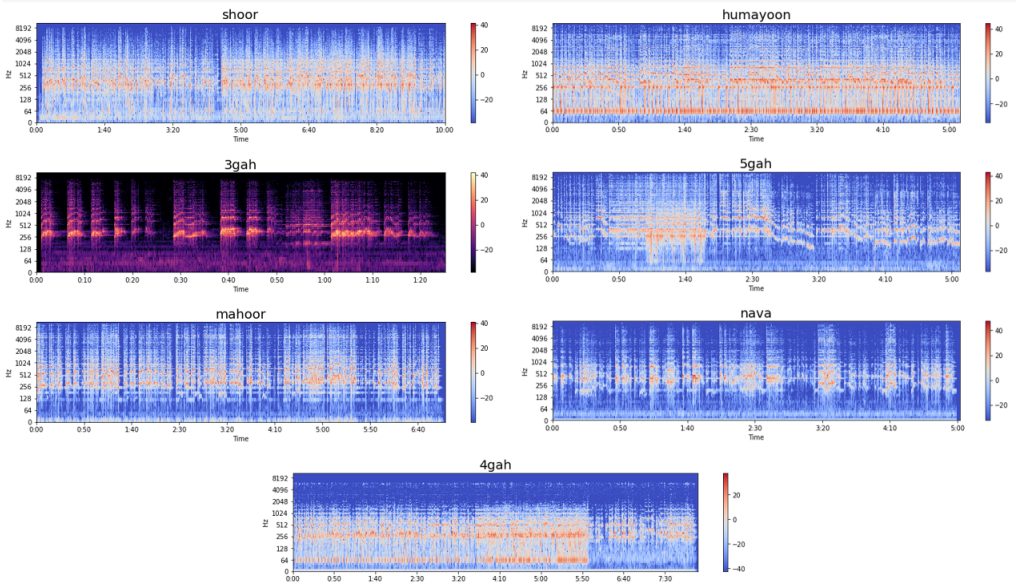


Figure 3: spectrogram of music from different dastgāhs

4.2.1 Spectral centroids

It displays different frequencies in a signal and indicates the centre of the mass for sound by taking a weighted mean over the frequencies. The spectral centroid is plotted on top of the wave plot in figure 4, and as can be seen, it is quite different for different dastgahs. It has taken low values for Nava while taking high values for Mahur. If it is this way for all music in the dastgahs, this is quite a good feature for classification. This will add two features to the dataset, mean and variance over the whole music.

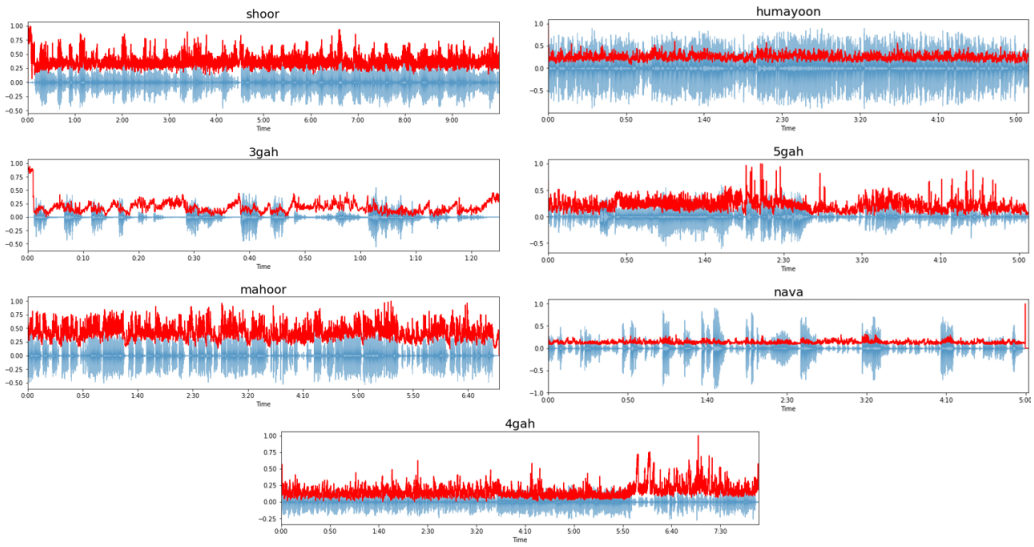


Figure 4: spectral centroids

4.2.2 Spectral rolloff

It is a measure of the shape of the signal. It denotes the frequency below which a particular proportion of the total spectral energy, such as 80 percent, falls. The roll-off frequency can be used to differentiate between noise (above roll-off value) and harmonic sounds (below roll-off value). This feature also seems different in distinct dastgāhs and can be used in classification. This will add the next two features to the dataset, mean and variance.

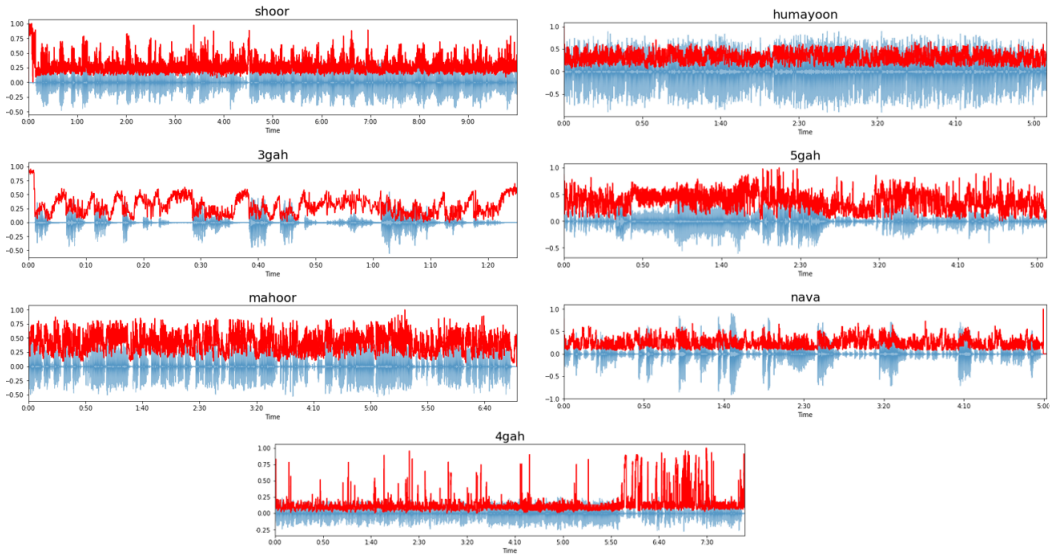


Figure 5: central rolloff

4.2.3 Spectral contrast

The frame of a spectrogram is divided into sub-bands, and spectral peaks, valleys, and their differences in each sub-band are extracted. Spectral contrast represents the relative spectral attributes in audio. It can demonstrate the harmonic and non-harmonic relative distribution and be used for music genre classification. The warmer the colour on the spectral contrast, the higher power is in those areas. The output of this function in Libros is a vector of shape (. . . , sub-bands + 1), which is seven by default and has not been changed. Then the mean and variance are calculated along the first axis, and two vectors of length seven are added to the feature vector.

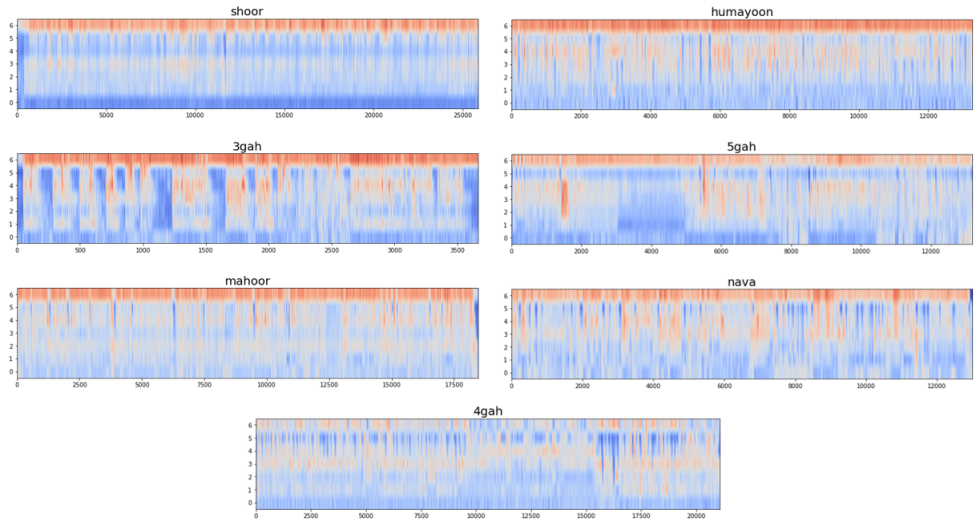


Figure 6: spectral contrast

4.2.4 Chroma-based features

Also referred to as "pitch class profiles", they are a powerful tool for analyzing music with categorizable pitches. One leading quality of chroma-based features is that it captures harmonic and melodic attributes with robustness to instrument and timber. This could be good for this task as there are different instruments for one dastgāh, but the pitches don't differ very much in different categories. Pitch can differ in music that contains singing of men and women. The output is a chroma vector of length twelve, indicating how much energy of each pitch class is present in the signal. This vector is based on all samples taken from the audio, and therefore the mean and variance of each twelve axes are calculated and added to the feature vector.

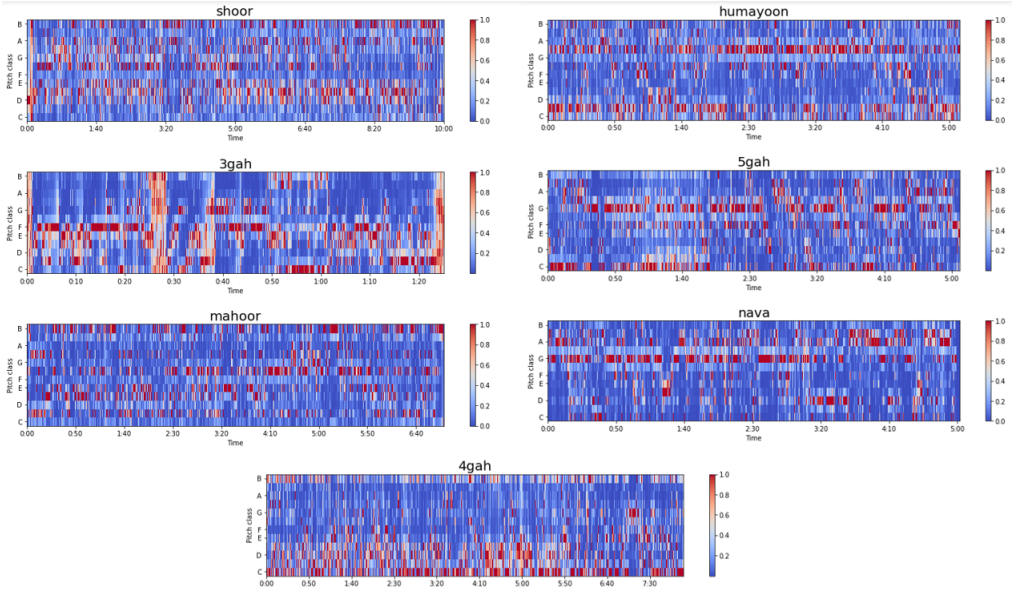


Figure 7: chroma based features

4.2.5 Mel-Frequency Cepstral Coefficients

Based on a linear cosine transform of a log power spectrum on a nonlinear MEL scale of frequency, MFC represents the short-term power spectrum of a sound used in sound processing. The output is a vector of the calculated result for each frame in the number of MEL bands generated which is 20. Same as the previous features, mean and variance are calculated over audio samples and added to the feature dataset.

Based on the explanations above, the frequency domain analysis will result in 82 features along with the feature of the instrument. Time-frequency features will also be 352 with the instrument. Music files must be read and passed to a function to generate these features to calculate each. It is necessary for time features to be calculated on music pieces of the same length, and extracting frequency features on music with the same length would be much more sensible. Also, classification performance on frequency and time domain features will be compared. These features must be for the same piece of music to result in a reasonable and reliable comparison.

To split audio files, a function is first written to check if all available files are in the correct format and if there are no corrupted files in the provided dataset. This function uses the mutgen.mp3 to read files in the format of mp3s, and if there are files that can not be read, it puts the full path in a list to be used later. For splitting purposes, the length of correct files in seconds is also kept in a list and returned by the function. The resulting list of harmful files is saved to a CSV file to be used along the feature extraction process. There were 149 harmful files in the dataset, leaving 800 music files to perform the rest of the project. A list of the length of audio files is also inspected to see the length of the ten shortest audio records. This is done to obtain the best length audio records that can be split without the loss of many records and also much information. The length of the ten shortest audio records is shown in figure 8.

```
sorted(result)[:10]
```

```
[4, 4, 7, 11, 15, 16, 21, 21, 21, 23]
```

Figure 8: ten shortest audio lengths

Although there was not supposed to be music with a length below 10 in the dataset, three music records are shorter than 10 seconds. There are also two pieces with lengths below 20 seconds. Twenty seconds is a reasonable time to split music pieces. This way, a comparison of different records would be more sensible.

A function is written to split the music into 20 seconds pieces, and if there is a part left at the end with a length of less than 20, it is ignored. These 20 seconds pieces are saved one by one (with replacement to avoid extra memory consumption) in .wav format and then read with librosa and passed to the feature extraction function. All time and frequency features are extracted and returned in separate vectors to be stored in the dataset. It is also worth saying that features were extracted separately for each dastgāh; the datasets were concatenated to one CSV file for frequency features and one CSV file for time domain features. These datasets will be used for model fitting and prediction in the upcoming parts.

If a brand new music piece comes to the system and its dastgāh is to be predicted, it must be passed to the explained functions to take the expected features out of it and then passed to the classifiers to detect the label.

4.3 Distribution of target values in training and testing data

Distribution of different classes in training data is as below. As Figure 9 shows, Nava, Mahoor, and Homayoon has the most frequency respectively.

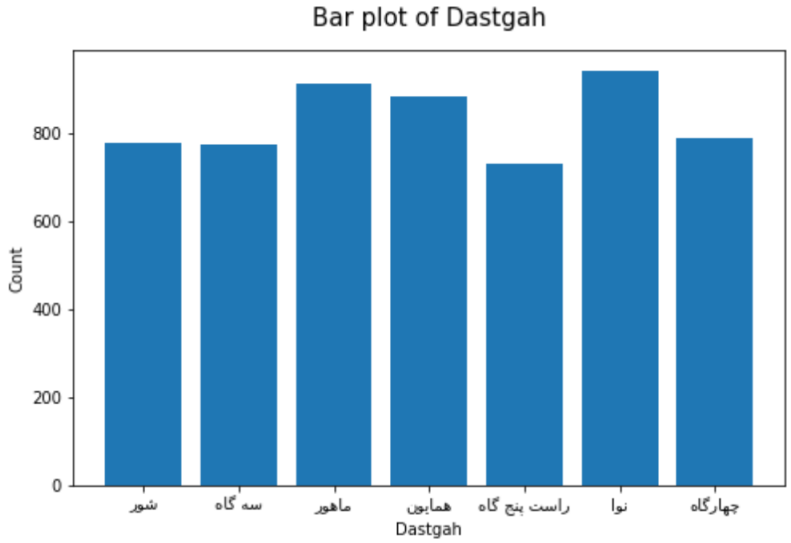


Figure 9: Distribution of different classes in training data

Distribution of different classes in testing data is as below. As you see, the ratio of frequency of each class is equal in both training and testing data.

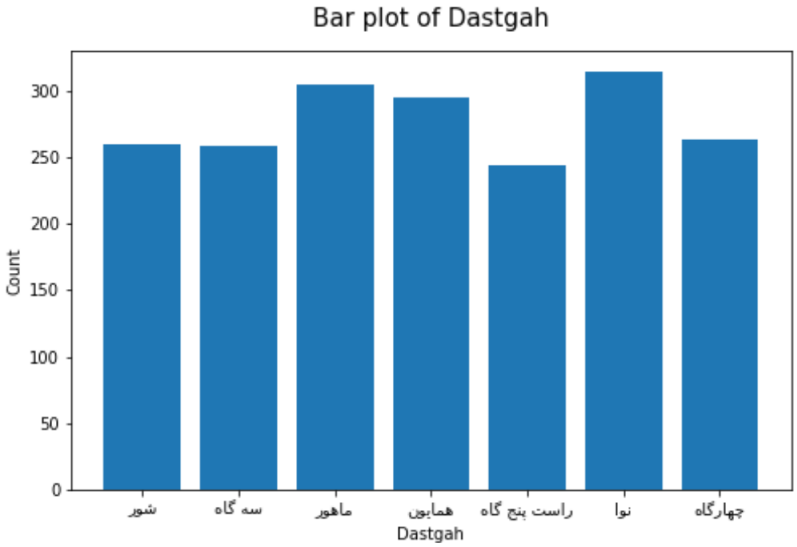


Figure 10: Distribution of different classes in testing data

5 Classification

As explained in the first part, the models were trained with two sets of features, the frequency and time domains. Each frequency or time domain is compared with the normalized and raw form of the data. Different results were obtained from each classifier that will be explained.

5.1 Logistic Regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. This classifier uses the sigmoid function to give classes a score of belonging. It is suitable for binary classification and in case of training on a multiclass problem, methods such as one-to-one correspondence or one-to-rest must be applied. Another weakness of logistic regression is that classes must be linearly separateable, otherwise, the results will not be as good as when using other flexible classifiers. Logistic regression is one of the simplest classification algorithms which is a suitable choice for starting. Features are not 2 or 3-dimensional so plotting them to observe if classes are linearly separateable is not possible. Therefore logistic regression is used as a starting point.

1. First of all, the frequency form is examined along with data standardization.
   For this classifier, default values of all hyperparameters, such as l_2 penalty, $C = 1.0$ and ect, were used from the scikit learn library were used and for the first part, it has been trained on frequency domain features. The time taken to train the model is approximately 1584.17 milliseconds. Evaluation metrics such as precision, recall and $F1$ *score* for each class and the overall model have been calculated. The metrics were calculated on both the training and test sets to evaluate model bias and variance errors.
   Looking at the results, it seems like class 0 which is the Shur dastgāh, has the lowest value for precision, recall and F1 score. In contrast, Nava has been classified better than the rest. The overall accuracy is 48% on training and 44% on test set.

Train Data on frequency features

	precision	recall	f1-score	support
0	0.43	0.39	0.41	777
1	0.50	0.51	0.50	775
2	0.47	0.50	0.48	914
3	0.45	0.48	0.46	883
4	0.46	0.39	0.42	733
5	0.54	0.58	0.56	942
6	0.51	0.51	0.51	790
accuracy			0.48	5814
macro avg	0.48	0.48	0.48	5814
weighted avg	0.48	0.48	0.48	5814

Figure 11: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.36	0.32	0.34	259
1	0.48	0.48	0.48	258
2	0.42	0.46	0.44	305
3	0.43	0.41	0.42	295
4	0.37	0.37	0.37	244
5	0.51	0.54	0.53	314
6	0.47	0.47	0.47	263
accuracy			0.44	1938
macro avg	0.44	0.44	0.44	1938
weighted avg	0.44	0.44	0.44	1938

Figure 12: Test Data on frequency features

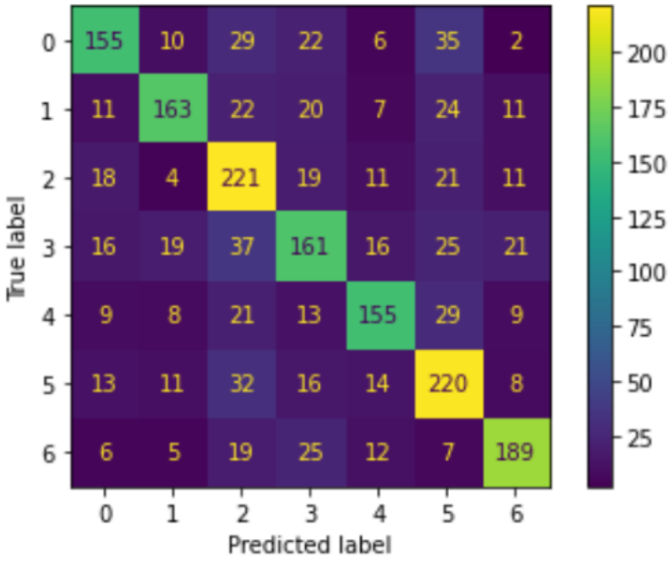


Figure 13: Confusion matrix of frequency domain logistic regression with normalization

The results show that although accuracy is much better than random assignment, the model is still simple for the classification task. This is probably due to the fact that classes are not linearly separateable. This has resulted in a bit of bias in the model. Comparing train and test accuracy shows that there is no sign of overfit and this is pretty much logical as in the case of the presence of bias in a model, there would be no variance error.
Figure 14 shows the ROC curve for each class. As mentioned above, it seems that this model has the highest performance at predicting class 5 with 0.72 area. However, it has the poorest performance at predicting class 0 with 0.62 area in comparison to other classes.

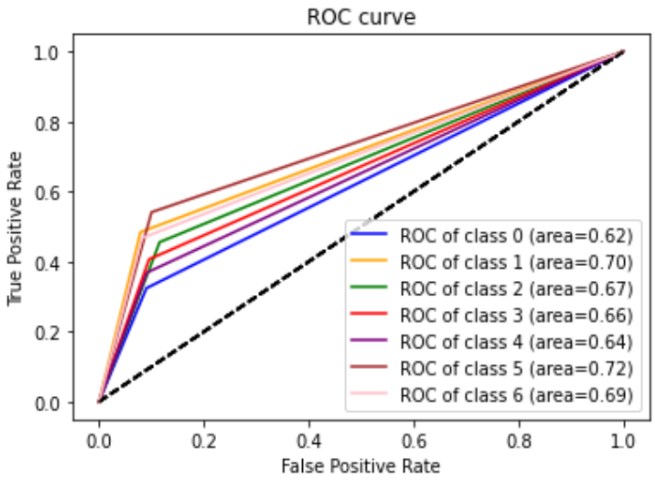


Figure 14: ROC curve of frequency domain logistic regression with normalization

Figure 15 shows the learning curve of the model. It is demonstrated from the learning curve that by increasing the number of samples in the training set, the accuracy of test data increases. In contrast, the accuracy of training data drops down slightly. This means that by increasing the number of samples in the training data, overfitting is prevented and this is shown by the decrease in training accuracy. On the other hand, as the model is simple, increasing the number of data points it sees before prediction, will result in higher test accuracy. The best accuracy for training data is about 0.65 while the best one for test data is near 0.45.

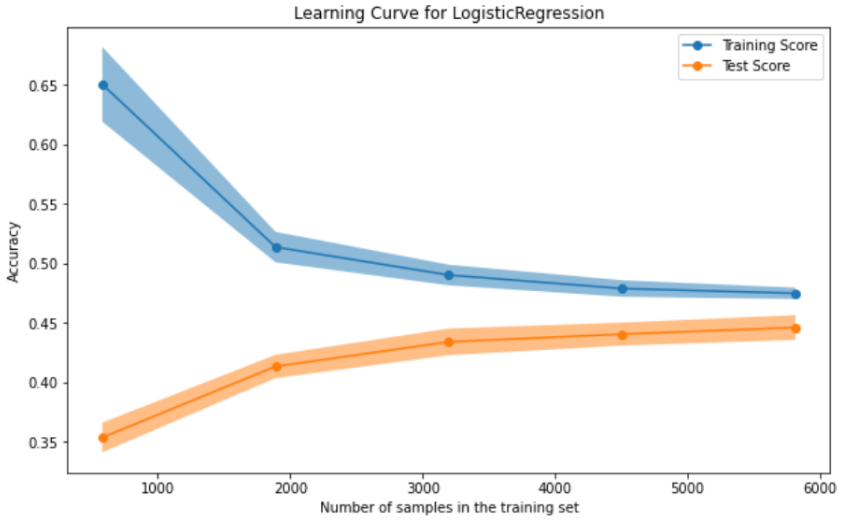


Figure 15: Learning curve for frequency domain logistic regression model with normalization

2. The frequency form without data standardization is examined in the second stage. Looking at the results, it seems like class 4 which is the Rast-panjgah dastgāh,

has the lowest value for precision, recall and F1 score. Comparatively, Shur has been classified better than the rest. The overall accuracy is 15% on training and 11% on test sets.

Train Data on frequency features

	precision	recall	f1-score	support
0	0.25	0.04	0.06	777
1	0.00	0.00	0.00	775
2	0.16	0.47	0.24	914
3	0.21	0.25	0.23	883
4	0.00	0.00	0.00	733
5	0.22	0.44	0.30	942
6	0.20	0.00	0.01	790
accuracy			0.19	5814
macro avg	0.15	0.17	0.12	5814
weighted avg	0.15	0.19	0.13	5814

Figure 16: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.17	0.02	0.03	259
1	0.00	0.00	0.00	258
2	0.16	0.49	0.24	305
3	0.24	0.27	0.25	295
4	0.00	0.00	0.00	244
5	0.22	0.42	0.29	314
6	0.00	0.00	0.00	263
accuracy			0.19	1938
macro avg	0.11	0.17	0.12	1938
weighted avg	0.12	0.19	0.13	1938

Figure 17: Test Data on frequency features

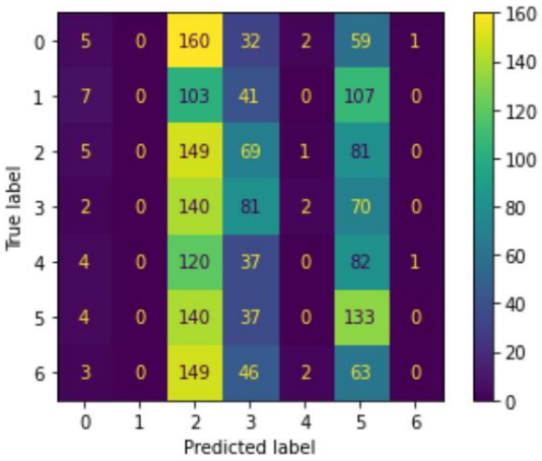


Figure 18: Confusion matrix of frequency domain logistic regression without normalization

In this case, bias error exists in the model for 2 reasons. One is that the model is too simple and the classes cannot be separated linearly. As a result the model has a lot of bias. Also, because the data are not normalized, there is a strong dispersion in each class of data, as a result, the model bias is high.
Figure 19 shows the ROC curve for each class. Just as mentioned above, it seems that this model has the highest performance in predicting class 5 and 3 respectively with 0.57 and 0.56 area. The rest of the classes have the worst efficiency.

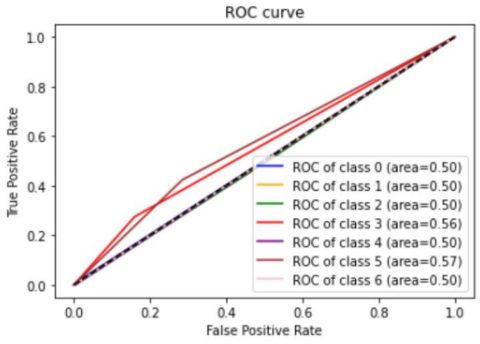


Figure 19: ROC curve of frequency domain logistic regression without normalization

Learning curve of the model can be seen in figure 20. It is demonstrated from the learning curve that by increasing the number of samples in the training set, accuracy of test data increases and accuracy of training data drops slightly. Finally, the accuracy of the training and test data converges and becomes equal.
As it was said before, due to the simplicity of the model and the excessive complexity of the data and the lack of standardization of the data, this model is not suitable. Therefore, the accuracy of the model on the test data is not satisfactory. The best accuracy for training data is about 0.2 while the best one for test data

is near to 0.17.

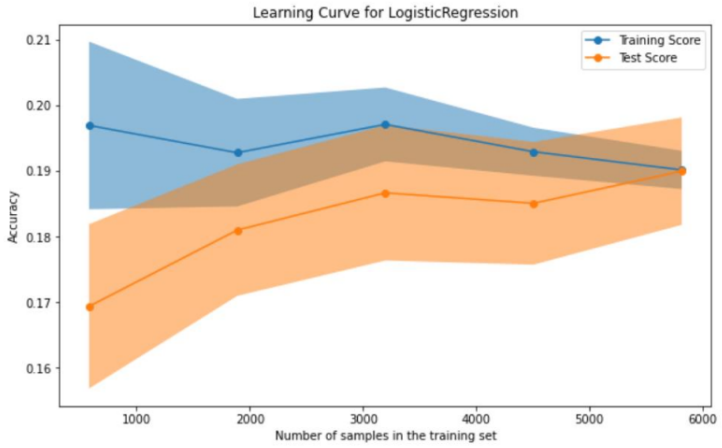


Figure 20: ROC curve of frequency domain logistic regression without normalization

3. In the third stage, the time domain form with data standardization is examined. Looking at the results, it seems like class 0 which is the Segah dastgāh, has the lowest value for precision, recall and F1 score. On the other hand, Chargah has been classified better compared to the rest. The overall accuracy is 22% on training and 16% on test set.

Train Data on time features

	precision	recall	f1-score	support
0	0.21	0.16	0.18	777
1	0.18	0.11	0.14	775
2	0.24	0.25	0.24	914
3	0.22	0.32	0.26	883
4	0.21	0.08	0.12	733
5	0.21	0.40	0.28	942
6	0.25	0.15	0.18	790
accuracy			0.22	5814
macro avg	0.22	0.21	0.20	5814
weighted avg	0.22	0.22	0.21	5814

Figure 21: Train Data on time features

Test Data on time features

	precision	recall	f1-score	support
0	0.17	0.11	0.14	259
1	0.10	0.06	0.07	258
2	0.17	0.17	0.17	305
3	0.18	0.27	0.21	295
4	0.15	0.06	0.09	244
5	0.18	0.35	0.24	314
6	0.17	0.10	0.12	263
accuracy			0.17	1938
macro avg	0.16	0.16	0.15	1938
weighted avg	0.16	0.17	0.15	1938

Figure 22: Test Data on time features

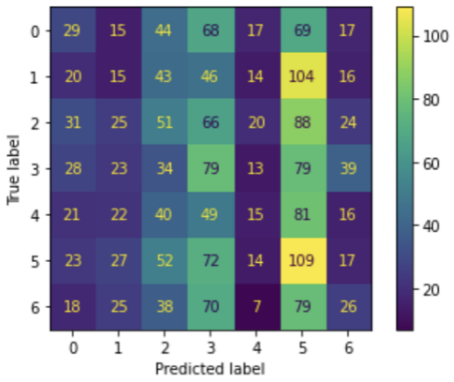


Figure 23: Confusion matrix of time domain logistic regression with normalization

The results show that although accuracy is a bit better than random assignment, the model is still simple for the classification task. This is probably due to the fact that classes are not linearly separable. This has resulted in a lot of bias in the model. Comparing train and test accuracy shows that there is no sign of overfit. As test and train data are close to each other and follow each other, we do not have variance error. If there is bias in a model, there is no variance error. Figure 24 shows the ROC curve for each class. Just as mentioned above, it seems that this model has the best performance on predicting class 3,5 with 0.51 area and the rest of the classes have the worst efficiency by 1%.

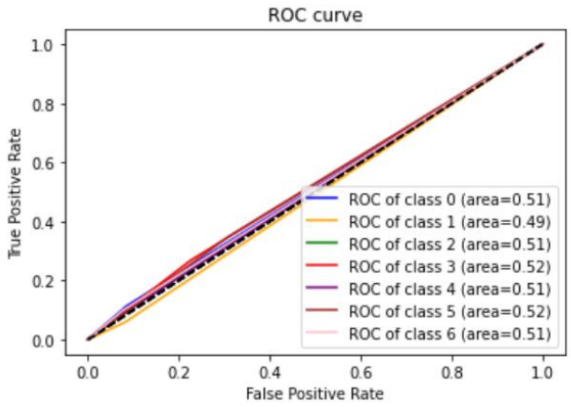


Figure 24: ROC curve of time domain logistic regression with normalization

Learning curve of the model can be seen in figure **??**. It is demonstrated from the learning curve that by increasing the number of samples in the training set, accuracy of test data increases and accuracy of training data drops slightly. Finally, the accuracy of the training and test data approximately converges. As it was said before, due to the simplicity of the model and the excessive complexity of the data, the accuracy of the model on the test data is not satisfactory.
The best accuracy for training data is about 0.38 while the best one for test data is near 0.16.

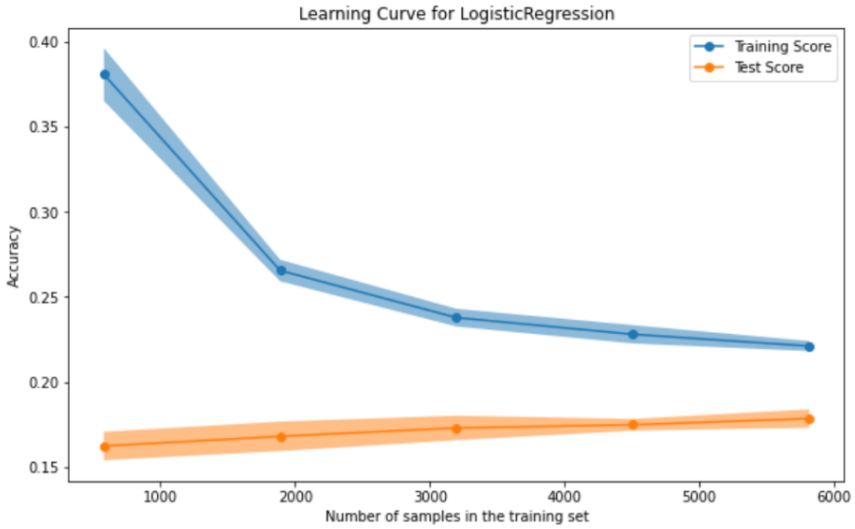


Figure 25: Learning curve for time domain logistic regression model with normalization

4. In the fourth stage, the time domain form without data standardization is examined.
   Because the data is not standardized and the dispersion of the data is large and in a non-linear form, therefore, when calculating the confusion matrix, most of the parameters are zero, so all the parameters such as precision, recall and f1-score

for most classes, except Homayun dastgāh, which slightly scatters its data in a linear form, become zero.
It is demonstrated from the ROC curve that the dashed line indicates that in this case the classifier randomly classified the data.

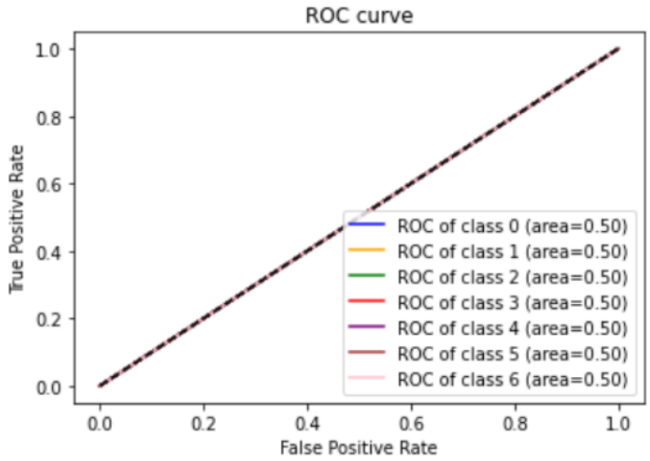


Figure 26: ROC curve of time domain logistic regression without normalization

Among the four models above, the best results occurred using frequency domain features along with normalization. The overall outcome shows that normalization has helped model performance significantly in both time and frequency domain features and that frequency domain features are better used in classification. These features tend to reveal more information about the structure of each dastgāh. Least performance is also for using time domain features without data standardization. The resulting model predicts randomly.

	Data standardization	Train Accuracy %	Test Accuracy %	Fit Time milliseconds
Frequency domain features	Standard	48	44	1584.17
Frequency domain features	Not standard	19	19	625.75
Time domain features	Standard	22	17	501.64
Time domain features	Not standard	15	15	73.22

Table 1:

Comparing the results of model fitting time shows that logistic regression has taken less time to train with time domain features. Data normalization has ended up with longer training time.

5.2 KNN

Figure 27 was drawn to select best number of k in KNN classifier. This plot is based on frequency domain data. For training data, the accuracy of knn algorithm is almost constant, but for testing data, the accuracy of knn algorithm decreases by increasing number of k. So, k = 2 was selected that has the best accuracy.

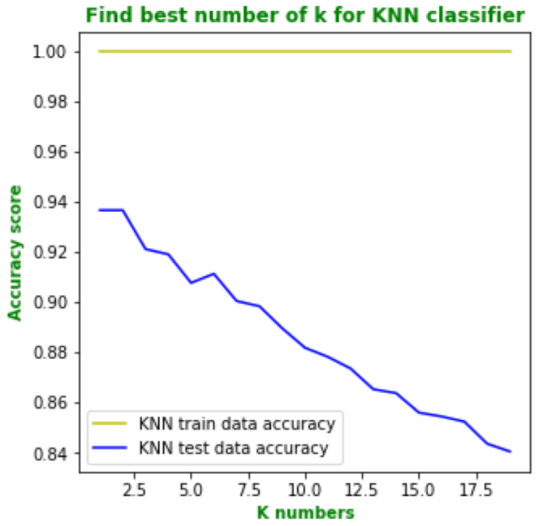


Figure 27: plot of accuracy score by number of k in knn

KNN is a supervised learning classification algorithm used to solve both classification and regression problems. KNN assumes that similar data points are close together. It is simple to implement and robust to noisy training data. It can be more effective if the training data is large. Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm. On the other hand, the K parameter should be determined at each step. The computation cost is a bit high, because of calculating the distance between the data points for all the training samples in each epoch. The higher the number of examples, the slower the performance of KNN is achieved. As KNN has uses distance as the metric of classification, it has been proved to work best in lower dimensions as well as with standard data.

1. First of all, the frequency domain form is examined along with data standardization.
   For this classifier, the time taken to train the model is approximately 34.37 milliseconds. Evaluation metrics such as precision, recall and F1 score for each class and the overall model have been calculated. The metrics were calculated on both the training and test sets to evaluate model bias and variance errors.
   Looking at the results, it seems like all classes have the highest values for precision, recall and F1 score. The overall accuracy is 100% on training and 94% on test sets.

Train Data on frequency features

	precision	recall	f1-score	support
0	1.00	1.00	1.00	777
1	1.00	1.00	1.00	775
2	1.00	1.00	1.00	914
3	1.00	1.00	1.00	883
4	1.00	1.00	1.00	733
5	1.00	1.00	1.00	942
6	1.00	1.00	1.00	790
accuracy			1.00	5814
macro avg	1.00	1.00	1.00	5814
weighted avg	1.00	1.00	1.00	5814

Figure 28: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.93	0.91	0.92	259
1	0.96	0.95	0.95	258
2	0.92	0.95	0.93	305
3	0.96	0.92	0.94	295
4	0.92	0.94	0.93	244
5	0.93	0.93	0.93	314
6	0.94	0.95	0.95	263
accuracy			0.94	1938
macro avg	0.94	0.94	0.94	1938
weighted avg	0.94	0.94	0.94	1938

Figure 29: Test Data on frequency features

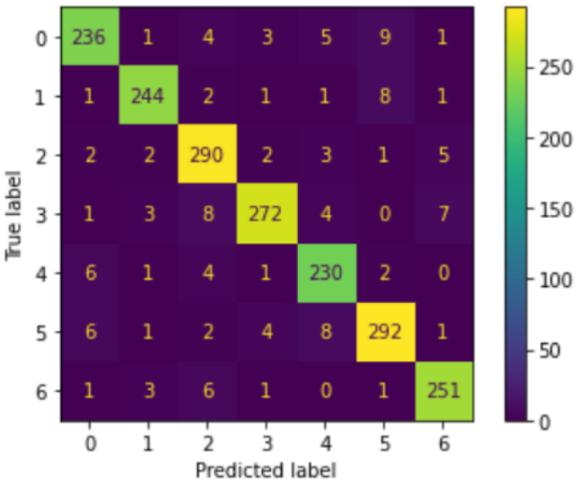


Figure 30: Confusion matrix of frequency domain KNN with normalization

The results show that the model has done a good task of classification and it discriminates well the classes that are distributed non-linearly. Because the accuracy of the train data is 100%, there is no bias error, but variance error is unavoidable. The comparison of train and test accuracy shows that the model is well trained on the train data. However, it still cannot completely separate the test data, so we have a very small variance error. This is almost reasonable because, in the case of the presence of variance in a model, there is no bias error.

Figure fi12 shows the ROC curve for each class. As mentioned above, it seems that this model has the highest performance at predicting all classes with approximately 0.97 area.

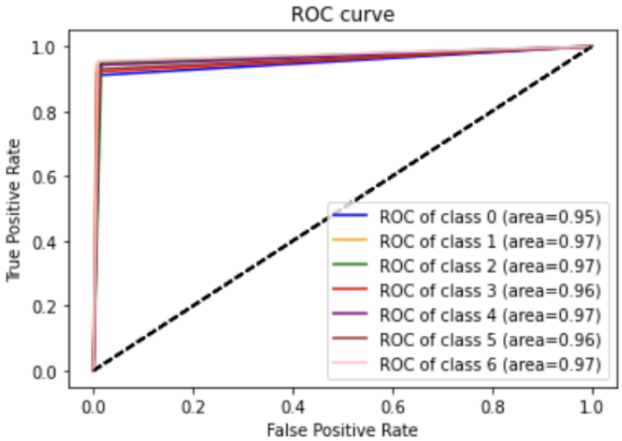


Figure 31: ROC curve of frequency domain KNN with normalization

Learning curve of the model can be seen in figure 32. It has been shown from the learning curve that with the increase in the number of samples in the training

set, the accuracy of the test data increases drastically and the accuracy of the training data decreases slightly. This means that with the increase in the number of samples in the training data, overfitting occurs and this is shown by a slight decrease in the accuracy of the training and a sharp increase in the accuracy of the test data. On the other hand, since the model is a bit complex, increasing the number of data points it sees before making predictions leads to higher test accuracy.
The best accuracy for the training data is around 100% while the best accuracy for the test data is close to 95%

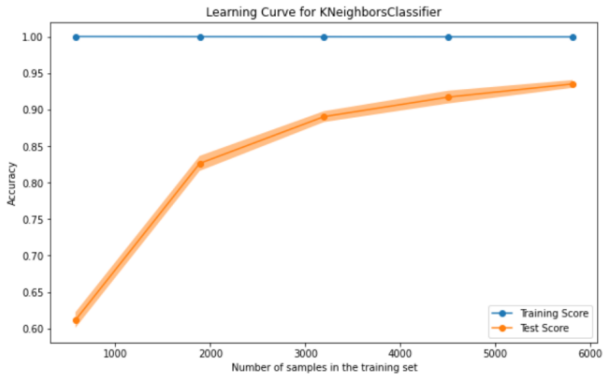


Figure 32: Learning curve for frequency domain KNN model with normalization

2. In the second stage, the frequency form without data standardization is examined.
   The time taken to train the model is approximately 45.41 milliseconds.

   Looking at the results, it seems like all class have the best value for precision, recall and f1 score on training data. But the accuracy of the test data has dropped drastically, and the best accuracy of the test data is related to the Mahur and Segah with 28% accuracy, and the lowest accuracy is related to the Rast panjgah with 19% accuracy. The overall accuracy is 100% on training and 25% on test set.

Train Data on frequency features

	precision	recall	f1-score	support
0	1.00	1.00	1.00	777
1	1.00	1.00	1.00	775
2	1.00	1.00	1.00	914
3	1.00	1.00	1.00	883
4	1.00	1.00	1.00	733
5	1.00	1.00	1.00	942
6	1.00	1.00	1.00	790
accuracy			1.00	5814
macro avg	1.00	1.00	1.00	5814
weighted avg	1.00	1.00	1.00	5814

Figure 33: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.23	0.25	0.24	259
1	0.28	0.26	0.27	258
2	0.28	0.25	0.26	305
3	0.25	0.24	0.25	295
4	0.19	0.19	0.19	244
5	0.32	0.36	0.34	314
6	0.20	0.21	0.21	263
accuracy			0.25	1938
macro avg	0.25	0.25	0.25	1938
weighted avg	0.25	0.25	0.25	1938

Figure 34: Test Data on frequency features

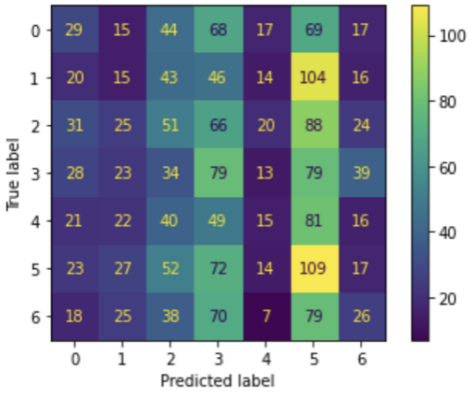


Figure 35: Confusion matrix for frequency domain of KNN without normalization

In this case, we have a lot of variance for 2 reasons. The model is almost complex, and it is not possible to separate the classes linearly. As a result the model has a

lot of bias. Also, due to the non-normality of the data, there is strong dispersion in each set of data, resulting in high model variance.
As mentioned earlier, due to the complexity of the model and the excessive complexity of the data and the lack of normalization of the data, the accuracy of the model on the test data is not satisfactory.

Figure 36 shows the ROC curve for each class. As mentioned above, this model with an area of 0.61 seems to perform well in predicting class 5, and the rest of the classes perform worst.

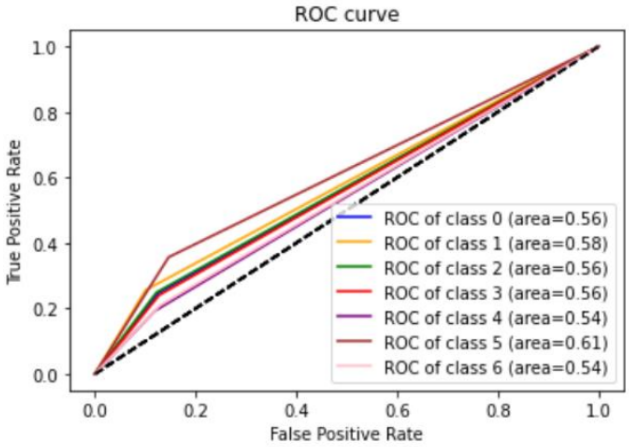


Figure 36: ROC curve of frequency domain KNN without normalization

The learning curve of the model can be seen in figure 37. It has been shown from the learning curve that with the increase in the number of samples in the training set, the accuracy of the test data increases with a slight slope and the accuracy of the training data remains constant. This means that with an increase in the number of samples in the training data, overfitting occurs. This is shown by the constant accuracy of the training and the low increase in the accuracy of the test data. The bias value is close to zero, but the model has a lot of variance and therefore has been overfitted.
The best accuracy for training data is around 100% while the best accuracy for test data is close to 30%.

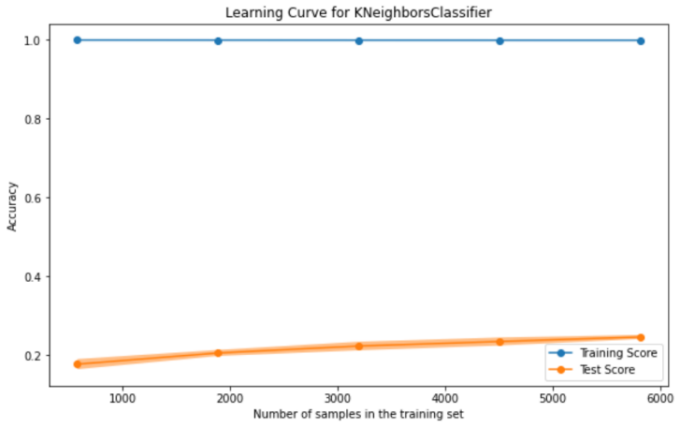


Figure 37: Learning curve for frequency domain KNN model without normalization

3. In the third stage, the time domain form with data standardization is examined. The time taken to train the model is approximately 45.07 milliseconds.
   Looking at the results, it seems like all class have the best value for precision, recall and f1 score on training data. But the accuracy of the test data has dropped drastically, and the best accuracy of the test data is related to the Mahur and Nava respectively with 45% and 39% accuracy, and the lowest accuracy is related to the Shur and Rast panjgah with 32% accuracy. The overall accuracy is 100% on training and 36% on test set.

Train Data on time features

	precision	recall	f1-score	support
0	1.00	1.00	1.00	777
1	1.00	1.00	1.00	775
2	1.00	1.00	1.00	914
3	1.00	1.00	1.00	883
4	1.00	1.00	1.00	733
5	1.00	1.00	1.00	942
6	1.00	1.00	1.00	790
accuracy			1.00	5814
macro avg	1.00	1.00	1.00	5814
weighted avg	1.00	1.00	1.00	5814

Figure 38: Train Data on time features

Test Data on time features

	precision	recall	f1-score	support
0	0.32	0.31	0.32	259
1	0.37	0.39	0.38	258
2	0.39	0.33	0.36	305
3	0.33	0.30	0.31	295
4	0.32	0.32	0.32	244
5	0.45	0.47	0.46	314
6	0.33	0.41	0.37	263
accuracy			0.36	1938
macro avg	0.36	0.36	0.36	1938
weighted avg	0.36	0.36	0.36	1938

Figure 39: Test Data on time features

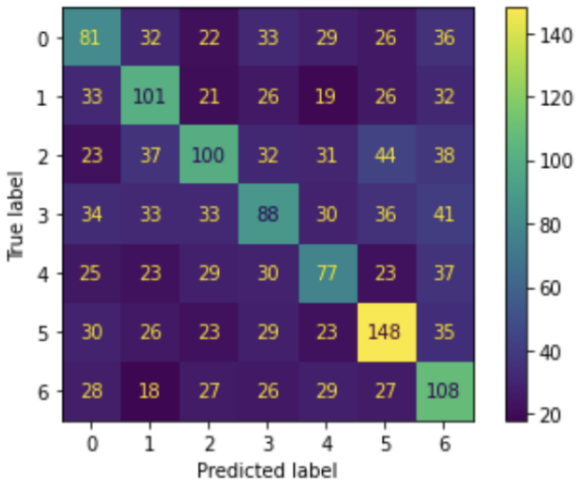


Figure 40: Confusion matrix of time domain KNN with normalization

Comparing the accuracy of test and train data provides evidence of model overfitting. Model accuracy on train data is 100% while it is 30% on test data. The model is not simple so bias has not occurred but with the large difference between test and train accuracies, variance error has taken place. This might be due to the fact that the number of features in the time domain dataset is large and KNN works best in lower dimensions.

Figure 41 shows the ROC curve for each class. Just as mentioned above, it seems that this model has the best performance on predicting class 5 with 0.68 AUC.

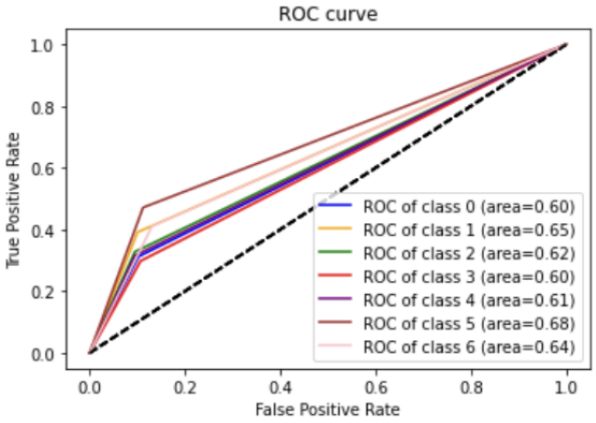


Figure 41: ROC curve of time domain KNN with normalization

Learning curve of the model can be seen in figure 42. It is demonstrated from the learning curve that the accuracy of the test data increases with a slight slope and the accuracy of the training data remains constant. This means that with the increase in the number of samples in the training data, overfitting occurs and this is shown by the constant accuracy of the training and the low increase in the accuracy of the test data. The bias value is close to zero, but the model has a lot of variance error. As mentioned earlier, due to the complexity of the model and the excessive complexity of the data, the accuracy of the model on the test data is not satisfactory.

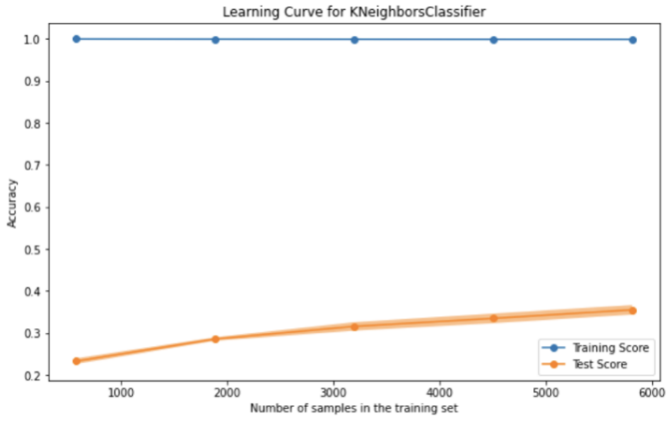


Figure 42: Learning curve for time domain KNN model with normalization

4. In the fourth stage, the time domain form without data standardization is examined.
   The time taken to train the model is approximately 45.73 milliseconds.

Looking at the results, it seems like all class have the best value for precision, recall and f1 score on training data. But the accuracy of the test data has dropped drastically, and the best accuracy of the test data is related to the Nava with 33% accuracy, and the lowest accuracy is related to the Shur with 16% accuracy. The overall accuracy is 100% on training and 20% on test set.

Train Data on time features

	precision	recall	f1-score	support
0	1.00	1.00	1.00	777
1	1.00	1.00	1.00	775
2	1.00	1.00	1.00	914
3	1.00	1.00	1.00	883
4	1.00	1.00	1.00	733
5	1.00	1.00	1.00	942
6	1.00	1.00	1.00	790
accuracy			1.00	5814
macro avg	1.00	1.00	1.00	5814
weighted avg	1.00	1.00	1.00	5814

Figure 43: Train Data on time features

Test Data on time features

	precision	recall	f1-score	support
0	0.16	0.16	0.16	259
1	0.20	0.21	0.20	258
2	0.23	0.22	0.22	305
3	0.18	0.17	0.17	295
4	0.17	0.15	0.16	244
5	0.33	0.32	0.33	314
6	0.17	0.20	0.18	263
accuracy			0.21	1938
macro avg	0.20	0.20	0.20	1938
weighted avg	0.21	0.21	0.21	1938

Figure 44: Test Data on time features

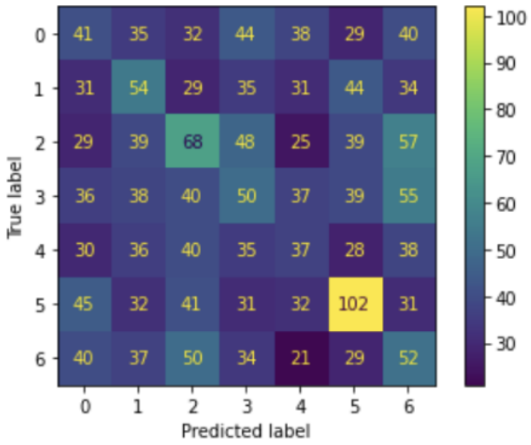


Figure 45: Confusion matrix of time domain KNN without normalization

The results show that although accuracy is a little bit better than random assignment, the model has enough complexity for the classification task. In addition, the classes are not linearly separable. This has resulted in a lot of variance in the model. Comparing train and test accuracy shows that overfitting has occurred. The accuracy of test and train data is far apart, so the model has a large variance error. If there is variance in a model, there is no bias error.

Figure 46 shows the ROC curve for each class. Just as mentioned above, it seems that this model has the best performance on predicting class 5 with 0.6 area and the rest of the classes have the worst efficiency by 6%.

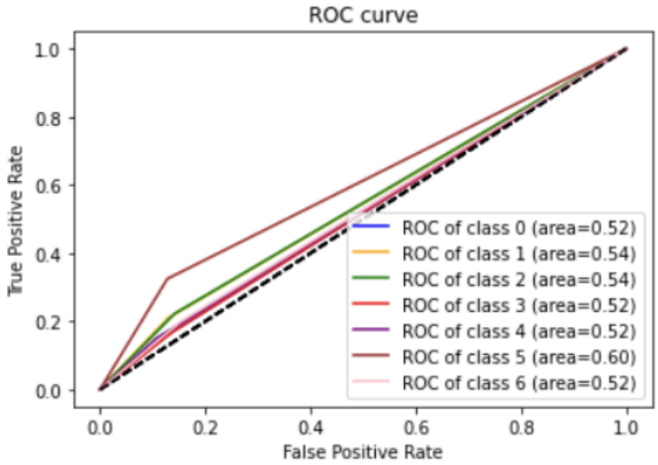


Figure 46: ROC curve of time domain KNN with normalization

Learning curve of the model can be seen in figure 47. It is demonstrated from

the learning curve that the accuracy of the test data increases with a linear slight slope and the accuracy of the training data remains constant. This means that with the increase in the number of samples in the training data, overfitting occurs and this is shown by the constant accuracy of the training and the low increase in the accuracy of the test data. The bias value is close to zero, but the model has a lot of variance.
Finally, the accuracy of the training and test data are not close to each other and there is a difference of 0.8 between them.
The best accuracy for training data is about 1 while the best one for test data is near to 0.2.

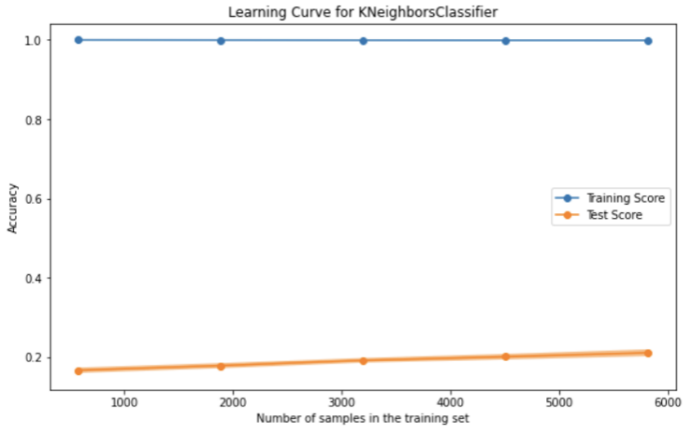


Figure 47: Learning curve for time domain KNN model without normalization

	Data standardization	Train Accuracy %	Test Accuracy %	Fit Time milliseconds
Frequency domain features	Standard	100	94	34.37
Frequency domain features	Not standard	100	25	45.41
Time domain features	Standard	100	36	45.07
Time domain features	Not standard	100	21	45.73

Based on the results provided in the table, the model with the best performance is the one using frequency features along with data standardization. This model has lower dimensions and the scaled features. As mentioned earlier, KNN works best with scaled and low-dimensional data. The opposite is also noticeable in the accuracy of 21% with time domain features that are not normalized. An accuracy of 100% is also expected that for train data, as KNN does not learn parameters but just compares the input with ones in the dataset. As KNN does not learn anything and just stores the training data, the fitting time is very low compared to other algorithms. However, it does take up much more space.

5.3 XGBoost

Extreme gradient boosted trees is a supervised learning classification, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models. This classifier is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems. The weakness of XGBoosting is that it does not perform so well on sparse and unstructured data. XGBoost is one of the highest Flexible algorithms. It uses the power of parallel processing. It is faster than gradient boosting and supports regularization.

1. First of all, the frequency domain form is examined along with data standardization.
   For this classifier, the time taken to train the model is approximately 20884.602 milliseconds. Evaluation metrics such as precision, recall and F1 score for each class and the overall model have been calculated. The metrics were calculated on both the training and test sets to evaluate model bias and variance errors.
   Looking at the results, it seems like class 1, 6 which is respectively, the Segah and Chargah dastgāh, has the highest value for precision, recall and F1 score. On the other hand, Mahur has been misclassified with 76% accuracy. The overall accuracy is 82% on training and 66% on test set.

Train Data on frequency features

	precision	recall	f1-score	support
0	0.85	0.72	0.78	777
1	0.87	0.81	0.84	775
2	0.76	0.84	0.80	914
3	0.77	0.82	0.80	883
4	0.85	0.80	0.82	733
5	0.79	0.85	0.82	942
6	0.87	0.86	0.86	790
accuracy			0.82	5814
macro avg	0.82	0.81	0.82	5814
weighted avg	0.82	0.82	0.82	5814

Figure 48: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.68	0.60	0.64	259
1	0.74	0.63	0.68	258
2	0.58	0.72	0.64	305
3	0.58	0.55	0.56	295
4	0.70	0.64	0.67	244
5	0.61	0.70	0.65	314
6	0.75	0.72	0.74	263
accuracy			0.65	1938
macro avg	0.66	0.65	0.65	1938
weighted avg	0.66	0.65	0.65	1938

Figure 49: Test Data on frequency features

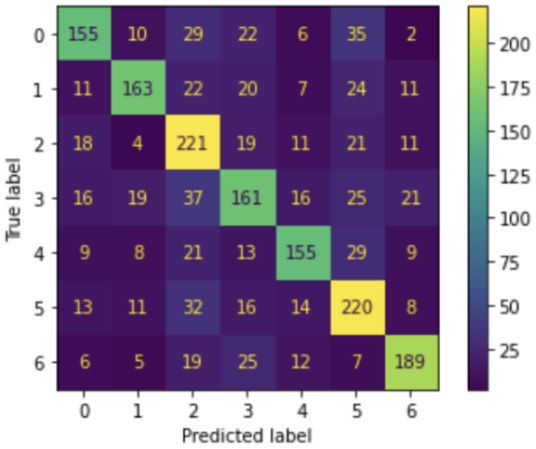


Figure 50: Confusion matrix of frequency domain XGBoost with normalization

The results show that the model has done a good task of classification and it discriminates well the classes that are distributed non-linearly. Because the accuracy of the train data is 82%, there is a little bias error, and variance error is unavoidable. The comparison of train and test accuracy shows that the model is Fair trained on the train data, but it still cannot completely separate the train and test data, so we have a little variance error.

Figure 51 shows the ROC curve for each class. Just as mentioned above, it seems that this model has the best performance on predicting class 6 with 0.84 area and has the worst performance on predicting class 3 with 0.74 area in comparison to other classes.

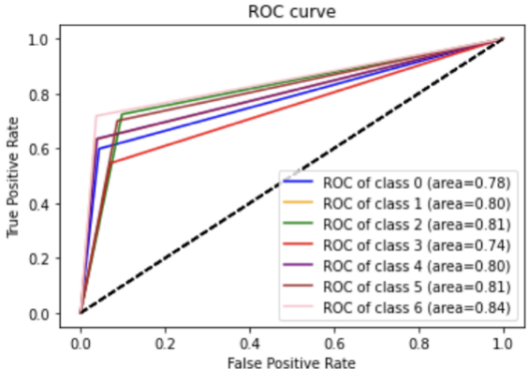


Figure 51: ROC curve of frequency domain XGBoost with normalization

Learning curve of the model can be seen in figure 52. It has been shown from the learning curve that with the increase in the number of samples in the training set, the accuracy of the test data with a suitable slope increases up to 70%, but the accuracy of the training data decreases to nearly 80%.
Here, overfitting did not occur, because during the training process, the accuracy on the test and training data converges. However, as we have a relatively small variance error, we have a bias error of about 20% on the train data.
The best accuracy for training data is about 0.98 while the best one for test data is near to 0.65.

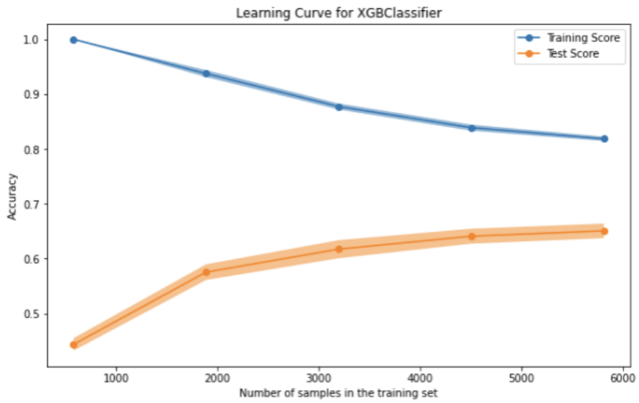


Figure 52: Learning curve for frequency domain XGBoost model with normalization

2. In the second stage, the frequency form without data standardization is examined.
The time taken to train the model is approximately 49299.03 milliseconds.
As you can see, the code execution time has increased nearly 2 times compared to when we normalize. However, we obtained the same accuracy, precision, recall

and F1 score.
Looking at the results, it seems like class 6, 1 which corresponds to Chargah and Segah dastgāh, has the highest value for precision, recall and F1 score. On the other hand, Mahur has been misclassified with 76% accuracy. The overall accuracy is 82% on training and 66% on test set.

Train Data on frequency features

	precision	recall	f1-score	support
0	0.85	0.72	0.78	777
1	0.87	0.81	0.84	775
2	0.76	0.84	0.80	914
3	0.77	0.82	0.80	883
4	0.85	0.80	0.82	733
5	0.79	0.85	0.82	942
6	0.87	0.86	0.86	790
accuracy			0.82	5814
macro avg	0.82	0.81	0.82	5814
weighted avg	0.82	0.82	0.82	5814

Figure 53: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.68	0.60	0.64	259
1	0.74	0.63	0.68	258
2	0.58	0.72	0.64	305
3	0.58	0.55	0.56	295
4	0.70	0.64	0.67	244
5	0.61	0.70	0.65	314
6	0.75	0.72	0.74	263
accuracy			0.65	1938
macro avg	0.66	0.65	0.65	1938
weighted avg	0.66	0.65	0.65	1938

Figure 54: Test Data on frequency features

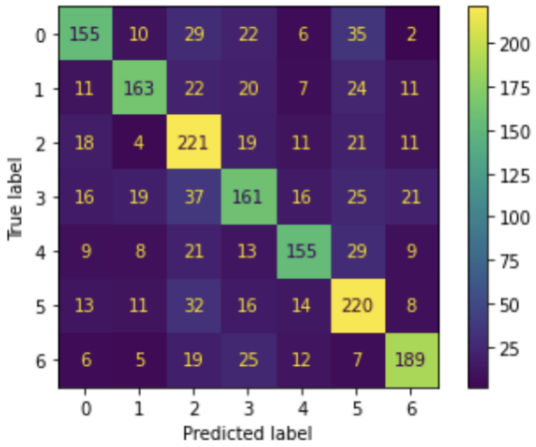


Figure 55: Confusion matrix of frequency domain XGBoost without normalization

Although the data is not standardized, the result is the same as the first part. But the area under the ROC curve of standardization and without standardization are different from each other.
Figure 56 shows the ROC curve for each class. Just as mentioned above, it seems that this model has the best performance on predicting class 6 with 0.84 area and the rest of the classes have performed worse with a difference of 6 percent.

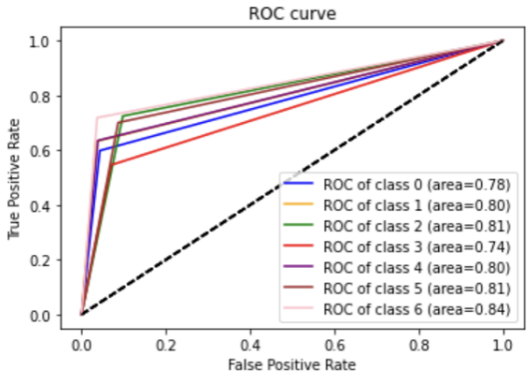


Figure 56: ROC curve of frequency domain XGBoost without normalization

Learning curve of the model can be seen in figure **??**. It has been shown from the learning curve that with an increase in the number of samples in the training set, the accuracy of the test data with a suitable slope increases up to 70%, but the accuracy of the training data decreases to nearly 80%.
Although the data is not standardized, the learning curve in this mode is the same as in the first part.
The best accuracy for training data is about 0.98 while the best one for test data is near 0.65.

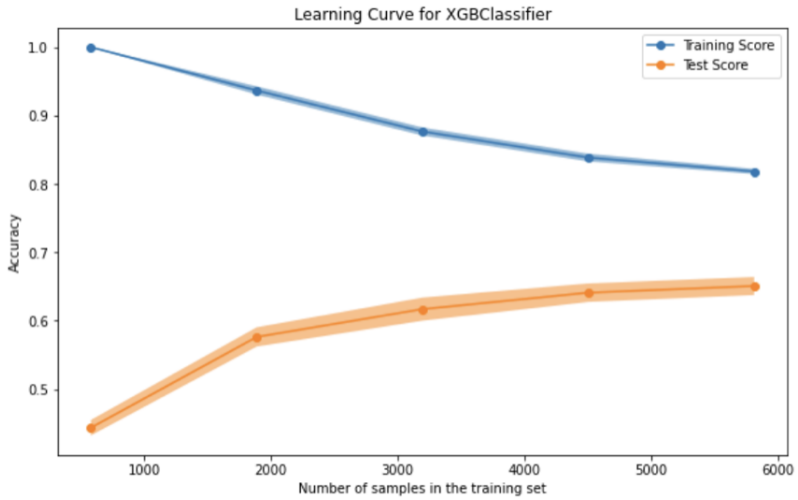


Figure 57: Learning curve for frequency domain XGBoost model without normalization

3. In the third stage, the time domain form with data standardization is examined. The time taken to train the model is approximately 21317.03 milliseconds. Looking at the results, it seems like class 4, the Homayun dastgāh, has the lowest values for precision, recall and F1 score. On the other hand, Rast-Panjgah has been classified better compared to the rest. The overall accuracy is 58% on training and 31% on the test set.

Train Data on time features

	precision	recall	f1-score	support
0	0.58	0.42	0.49	777
1	0.67	0.52	0.59	775
2	0.56	0.58	0.57	914
3	0.47	0.65	0.54	883
4	0.74	0.43	0.54	733
5	0.53	0.68	0.60	942
6	0.50	0.53	0.51	790
accuracy			0.55	5814
macro avg	0.58	0.54	0.55	5814
weighted avg	0.57	0.55	0.55	5814

Figure 58: Train Data on time features

	precision	recall	f1-score	support
0	0.25	0.14	0.18	259
1	0.41	0.26	0.32	258
2	0.29	0.33	0.31	305
3	0.24	0.32	0.27	295
4	0.36	0.23	0.28	244
5	0.34	0.52	0.41	314
6	0.28	0.28	0.28	263
accuracy			0.30	1938
macro avg	0.31	0.30	0.29	1938
weighted avg	0.31	0.30	0.30	1938

Figure 59: Test Data on time features

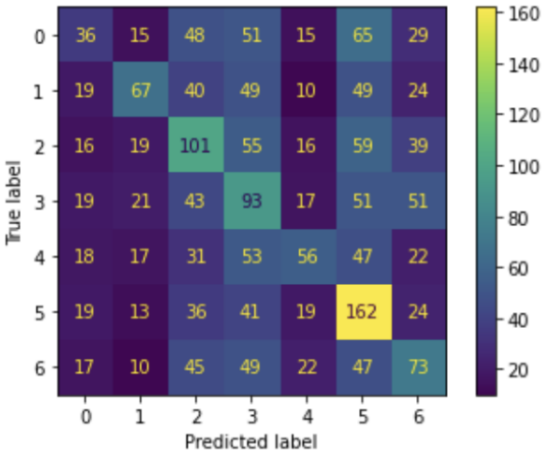


Figure 60: Confusion matrix of time domain XGBoost with normalization

The results show that accuracy on the training data is 60%, and on the test data is 30%. The model performs relatively well on the training data, but has not yet been able to separate the test data well. Comparing train and test accuracy shows that overfitting has occurred. The accuracy of test and train data is not close, so we have variance errors.
Considering that the model's accuracy on the training data is 60%, bias error is unavoidable.

Figure 61 shows the ROC curve for each class. As mentioned above, this model performs best in predicting class 5, with an area of 0.66, and class 0, with an area of 0.54, performs worst.

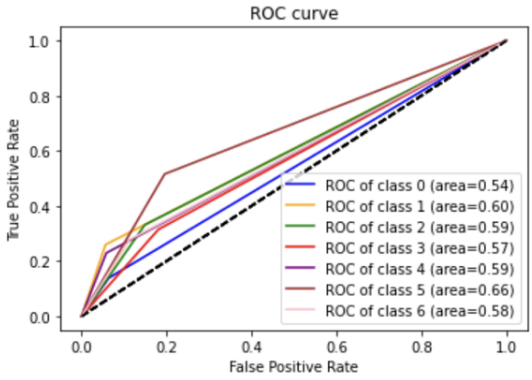


Figure 61: ROC curve of time domain XGBoost with normalization

The learning curve of the model can be seen in figure 62. It has been shown from the learning curve that with an increase in the number of samples in the training set, the accuracy of the test data increases with a slight slope, up to 30%, and the accuracy of the training data decreases sharply, around 40%. Finally, the accuracy of the training and test data almost converges. As mentioned earlier, the model's accuracy on the test data could be more satisfactory due to the model's simplicity and the excessive complexity of the data.
The highest accuracy for training data is about 0.98, while the most accurate for test data is near 0.3.

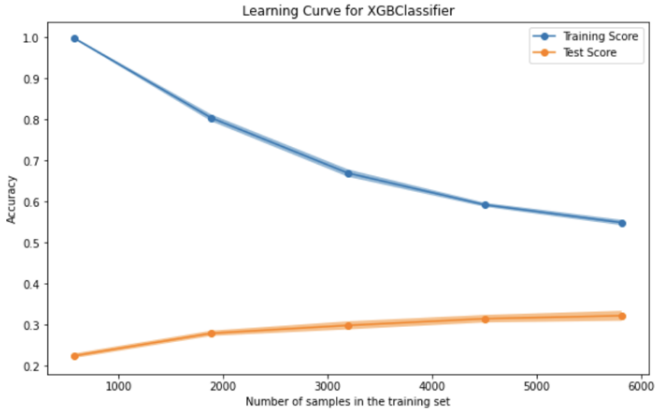


Figure 62: Learning curve for time domain XGBoost model with normalization

4. The time domain form without data standardization is examined in the fourth stage.
   The time taken to train the model is approximately 26832.26 milliseconds.

   Looking at the results, it seems like class 3, the Homayun dastgāh, has the

lowest value for precision, recall and f1 score. On the other hand, Rast panjgah dastgāh has been classified better compared to the rest. The overall accuracy is 58% on training and 31% on the test set.

Train Data on time features

	precision	recall	f1-score	support
0	0.58	0.42	0.49	777
1	0.67	0.52	0.59	775
2	0.56	0.58	0.57	914
3	0.47	0.65	0.54	883
4	0.74	0.43	0.54	733
5	0.53	0.68	0.60	942
6	0.50	0.53	0.51	790
accuracy			0.55	5814
macro avg	0.58	0.54	0.55	5814
weighted avg	0.57	0.55	0.55	5814

Figure 63: Train Data on time features

Test Data on time features

	precision	recall	f1-score	support
0	0.25	0.14	0.18	259
1	0.41	0.26	0.32	258
2	0.29	0.33	0.31	305
3	0.24	0.32	0.27	295
4	0.36	0.23	0.28	244
5	0.34	0.52	0.41	314
6	0.28	0.28	0.28	263
accuracy			0.30	1938
macro avg	0.31	0.30	0.29	1938
weighted avg	0.31	0.30	0.30	1938

Figure 64: Test Data on time features

Compared to the normalized mode, it has not changed the results. Therefore, data normalization does not affect the accuracy, precision, recall and F1 score of test and training data.
Whether normalization is done, all results and graphs are the same.

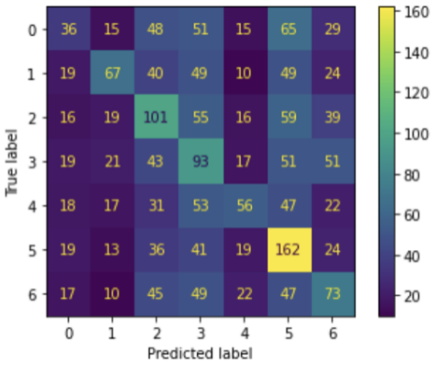


Figure 65: Confusion matrix of time domain XGBoost without normalization

The results show that the accuracy of test and train data is not close, so we have variance errors. Considering that the model's accuracy on the training data is 60%, the bias error is unavoidable.
Figure 66 shows the ROC curve for each class. As mentioned above, this model performs best in predicting class 5, with an area of 0.66, and class 0, with an area of 0.54, performs the worst.

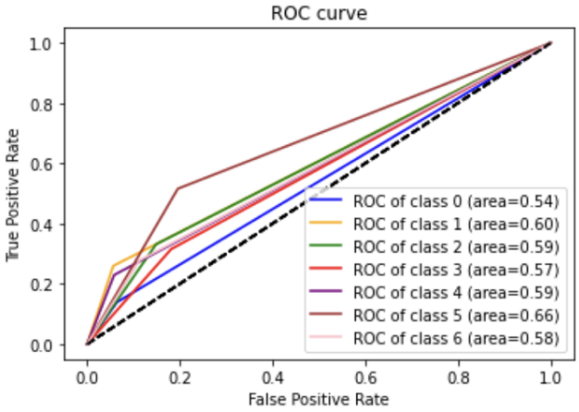


Figure 66: ROC curve of time domain XGBoost with normalization

The learning curve of the model can be seen in figure 67. It has been shown from the learning curve that with the increase in the number of samples in the training set, the accuracy of the test data increases with a slight slope, up to 30%, and the accuracy of the training data decreases sharply, around 40%. Finally, the accuracy of the training and test data almost converges. As mentioned earlier, due to the model's simplicity and the excessive complexity of the data, the model's accuracy on the test data is not satisfactory.
The best accuracy for training data is about 0.98, while the best for test data is near 0.3.

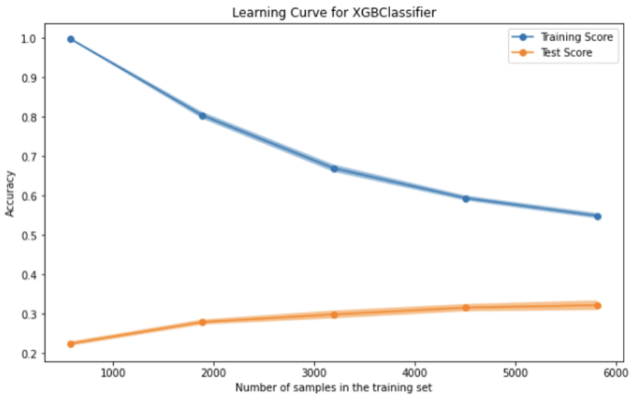


Figure 67: Learning curve for time domain XGBoost model without normalization

	Data standardization	Train Accuracy %	Test Accuracy %	Fit Time milliseconds
Frequency domain features	Standard	82	65	20884.6
Frequency domain features	Not standard	86	65	49299.0
Time domain features	Standard	55	30	21317.0
Time domain features	Not standard	55	30	26832.2

The results in the table show that XGBoost is not dependent on data normalization, and standardizing the data only helps with model fitting time. Like the other two classifiers, the model has higher accuracy based on frequency features. The difference between train and test accuracy demonstrates that the model has a slight variance error and therefore has overfitted. In the case of frequency features, bias error is less, while using time-domain features will result in a model with higher bias and variance.

	Time domain accuracy		Frequency domain accuracy	
	Accuracy	Time	Accuracy	Time
Logistic Regression	17%	501.64	44%	1584.17
KNN	36%	45.07	94%	34.37
XGBoost	30%	21317.0	65%	20884.6

The table above provides the accuracy score and fitting time taken to train the classifiers on normalized data. It can be seen that on all classifiers, time domain features work better in distinguishing dastgāhs from each other. KNN has the highest accuracy as features are normalized, and data is low dimensional. KNN also has no bias and a very small variance error. Fitting time is low as there are no parameters to fit. Testing, however, will take longer in KNN as there are a lot of comparisons that must be done. KNN also has a high spatial order.
In the second stage, XGBoost has a test accuracy of 65% but takes the longest to fit.

This model consists of bias and variance error which is a downside of using it. Last but not least is the model based on logistic regression, which is reasonable at training time, but the accuracy could be higher as classes are not linearly separatable. It might help if the features were transformed into a higher-dimensional space. The general result obtained from this part is that if there is no storage limit and no need for real-time prediction, KNN results in maximum accuracy. However, if there are limitations in both storage and prediction time, XGBoost is a better choice despite the bias and variance error. Another significant result is that frequency domain features better describe music signals' characteristics, and their combination helps build a classifier that better detects each dastgāh of Persian traditional music.

5.4 Dimension Reduction

Dimensionality reduction is a technique to reduce the number of features or variables in a dataset. It is used to reduce the complexity of a model, reduce the time and storage space required to train a model, and improve its accuracy. Dimensionality reduction can also help identify critical features in a dataset that may have been overlooked. This project uses four dimensionality reduction methods: PCA, LDA, forward selection, and backward elimination.

5.4.1 PCA

Principal Component Analysis (PCA) is one method of data dimensionality reduction. It is a form of unsupervised learning that identifies patterns in data and uses those patterns to project the data into a lower-dimensional space. PCA is used to identify correlations and patterns in high-dimensional datasets, allowing for more straightforward interpretation and visualization of complex datasets. PCA can also be used for feature selection, reducing the number of features in a dataset while preserving as much information as possible. One thing to notice is that PCA constructs new features that best explain the data rather than selecting some features and taking away the others. First applied PCA with two components on our dataset to see it in two dimensions. Here is the result of that:

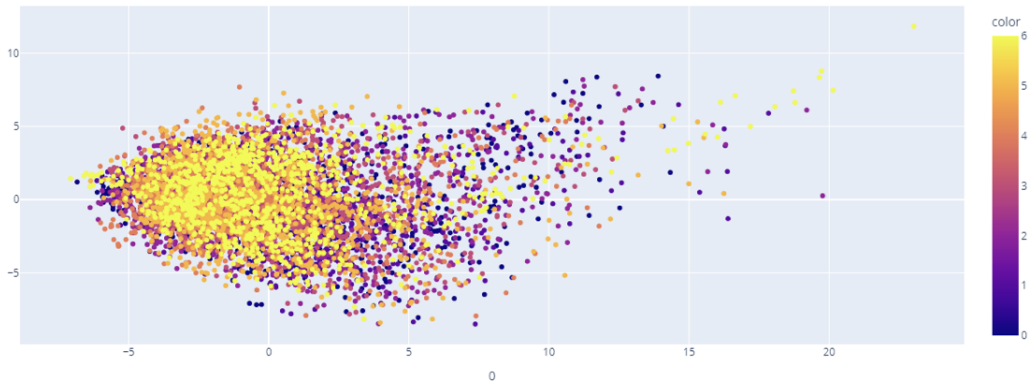


Figure 68: Scatter plot of PCA with two components

Different colours show different Dastgah. This plot shows that Dastgah 6 and 5 seem more concentrated than the others. In general, it is hard to distinguish between different Dastgahs.
To use a good number of components in PCA, he explained variance for each principal component has been found. First, to calculate explained variance, scale the data with the StandardScalar and then fit a PCA model to the scaled data.
After calculating cumulative variance, it has been plotted to show how it grows. Additionally, a line representing 95% of the explained variance has been plotted.

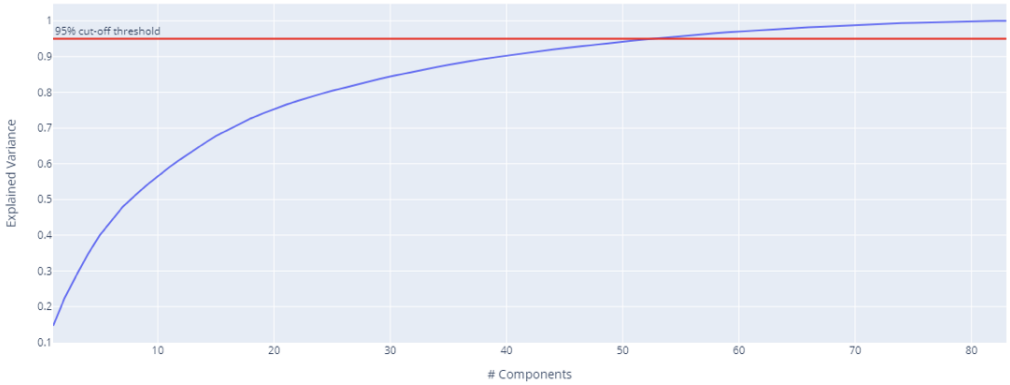


Figure 69: Plot of cumulative variance by number of components

After plotting the cumulative explained variance, we can see how the curve flattens slightly around 40 components. And where the line is drawn for 95%, the total explained variance is at approximately 52 components. So, 52 is selected as the best number of components for PCA. Finally, PCA with 52 components was applied to the dataset for dimension reduction.

5.4.2 LDA

Linear Discriminant Analysis (LDA) is a supervised machine learning technique for dimensionality reduction and classification. It is a method of finding a linear combination of features that best separates two or more classes of objects or events. It reduces the number of features in a dataset while preserving class separability. LDA can be used for both binary and multi-class classification problems.
LinearDiscriminantAnalysis, a class from the scikit library, was used to apply LDA to testing and training data.

5.4.3 Forward Selection

Forward selection is a method of dimension reduction by feature selection. This method works in such a way that it first selects the best single feature. Then, pairs of features are formed using one of the remaining features and this best feature, and the best pair is selected. Next, triplets of features are formed using one of the remaining features, and the best triplet is selected. This procedure continues until all or a predefined number of features are selected. To implement it, SequentialFeatureSelector from the mlxtend library was used. As such, a function was defined that uses this class, fits it on data and then returns the names of the selected features.

5.4.4 Backward Elimination

Backward elimination is another method of dimension reduction by feature selection. This method works so that, first, the criterion function is computed for all d features. Then, each feature is deleted one at a time, the criterion function is computed for all subsets with $d1$ features, and the worst feature is discarded. Next, each feature among the remaining d 1 is deleted one at a time, and the worst feature is discarded to form a subset with $d2$ features. This procedure continues until one feature or a predefined number of features are left.
To implement it, like forward selection, SequentialFeatureSelector from mlxtend library was used with this difference that the parameter forward is False, and the scoring method is a negative mean squared error. These parameters were True and accurate, respectively, in forward selection. The function's structure defined for this work is like forward selection; as such first fits this class on data and then returns the names of the selected features.

5.4.5 Applying Dimension Reduction Data on KNN

- PCA Dimension Reduction
  First apply PCA with 52 components on both training and testing data. The results of evaluation have got looks like below. As you see, all of the parameters are perfect for training data, but for testing data the performance is worse than the before. The accuracy is 0.21 which is not satisfactory in comparison to accuracy with this model on just normalized data. The metrics belongs to Rast Panjgah dastgah is the lowest of all for testing data.

Train Data on frequency features

	precision	recall	f1-score	support
0	1.00	1.00	1.00	777
1	1.00	1.00	1.00	775
2	1.00	1.00	1.00	914
3	1.00	1.00	1.00	883
4	1.00	1.00	1.00	733
5	1.00	1.00	1.00	942
6	1.00	1.00	1.00	790
accuracy			1.00	5814
macro avg	1.00	1.00	1.00	5814
weighted avg	1.00	1.00	1.00	5814

Figure 70: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.16	0.15	0.16	259
1	0.30	0.31	0.30	258
2	0.17	0.16	0.16	305
3	0.20	0.21	0.21	295
4	0.10	0.10	0.10	244
5	0.32	0.33	0.33	314
6	0.36	0.35	0.36	263
accuracy			0.23	1938
macro avg	0.23	0.23	0.23	1938
weighted avg	0.23	0.23	0.23	1938

Figure 71: Test Data on frequency features

ROC curve of this model on this data is as Figure 72. The performance for each class is almost the same and about 0.5.

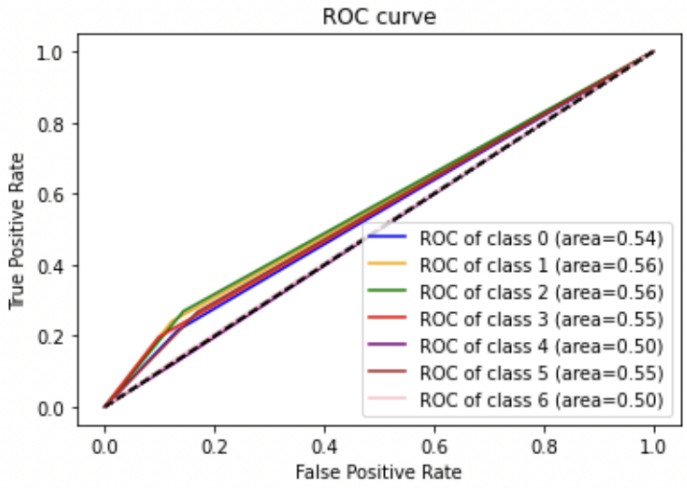


Figure 72: ROC curve of frequency domain KNN with PCA data

- LDA Dimension Reduction
  First apply LDA with maximum components on both training and testing data. The performance is a little better than data with PCA. But it is almost as worse as it in comparison to normalized data. Because what is important is performance on unseen data which is testing data and here the performance of model got worse when apply dimensionally reduced data by LDA. The accuracy is 0.23. The best performance belongs to Chahargah and the worst one belongs to Rast Panjgah dastgah.

Train Data on frequency features

	precision	recall	f1-score	support
0	1.00	1.00	1.00	777
1	1.00	1.00	1.00	775
2	1.00	1.00	1.00	914
3	1.00	1.00	1.00	883
4	1.00	1.00	1.00	733
5	1.00	1.00	1.00	942
6	1.00	1.00	1.00	790
accuracy			1.00	5814
macro avg	1.00	1.00	1.00	5814
weighted avg	1.00	1.00	1.00	5814

Figure 73: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.16	0.15	0.16	259
1	0.30	0.31	0.30	258
2	0.17	0.16	0.16	305
3	0.20	0.21	0.21	295
4	0.10	0.10	0.10	244
5	0.32	0.33	0.33	314
6	0.36	0.35	0.36	263
accuracy			0.23	1938
macro avg	0.23	0.23	0.23	1938
weighted avg	0.23	0.23	0.23	1938

Figure 74: Test Data on frequency features

Figure 75 is ROC curve of the model. As you see the best performance is for class 5 with 0.72 area.

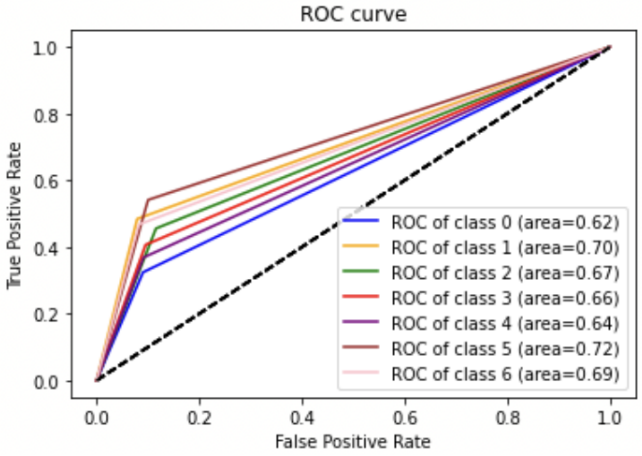


Figure 75: ROC curve of frequency domain KNN with LDA data

- Forward Selection
  Here, it is set to select 20 features. Again, applied KNN on this data with these selected features. The result is as below. The performance on training data is perfect. The accuracy is lower than the accuracy on data without feature selection but it is so good in comparison to what we had in PCA and LDA data. The model has the highest metrics for Mahoor dastgah with 0.9 precision, recall and f1-score

Train Data on frequency features

	precision	recall	f1-score	support
0	1.00	1.00	1.00	777
1	1.00	1.00	1.00	775
2	1.00	1.00	1.00	914
3	1.00	1.00	1.00	883
4	1.00	1.00	1.00	733
5	1.00	1.00	1.00	942
6	1.00	1.00	1.00	790
accuracy			1.00	5814
macro avg	1.00	1.00	1.00	5814
weighted avg	1.00	1.00	1.00	5814

Figure 76: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.85	0.90	0.88	259
1	0.88	0.88	0.88	258
2	0.90	0.90	0.90	305
3	0.90	0.82	0.86	295
4	0.86	0.89	0.88	244
5	0.88	0.87	0.87	314
6	0.86	0.88	0.87	263
accuracy			0.88	1938
macro avg	0.88	0.88	0.88	1938
weighted avg	0.88	0.88	0.88	1938

Figure 77: Test Data on frequency features

Figure 78 shows ROC curve of this model. As you see, all of the area is high and good and all of them is around 0.93.

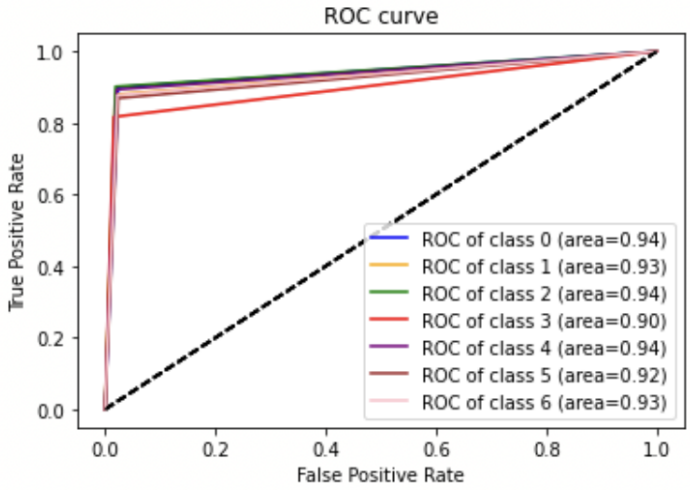


Figure 78: ROC curve of frequency domain KNN with data with forward feature selection

- Backward Elimination For this method of feature selection, it is set to select 72 features. This algorithm of feature selection takes a long time in comparison to forward selection and it takes longer time to select fewer number of features. Finally, the model was fitted on data with selected features and the results of evaluation of model got as below. As you see, the result is so close to what we see in applying KNN on normalized frequency data. The accuracy is 0.94 and average of other metrics is 0.94 for all of them which is equal to the metrics of KNN on normalized frequency data.

Train Data on frequency features

	precision	recall	f1-score	support
0	1.00	1.00	1.00	777
1	1.00	1.00	1.00	775
2	1.00	1.00	1.00	914
3	1.00	1.00	1.00	883
4	1.00	1.00	1.00	733
5	1.00	1.00	1.00	942
6	1.00	1.00	1.00	790
accuracy			1.00	5814
macro avg	1.00	1.00	1.00	5814
weighted avg	1.00	1.00	1.00	5814

Figure 79: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.94	0.91	0.92	259
1	0.97	0.95	0.96	258
2	0.94	0.95	0.95	305
3	0.95	0.93	0.94	295
4	0.91	0.95	0.93	244
5	0.94	0.94	0.94	314
6	0.94	0.95	0.95	263
accuracy			0.94	1938
macro avg	0.94	0.94	0.94	1938
weighted avg	0.94	0.94	0.94	1938

Figure 80: Test Data on frequency features

Figure 81 shows ROC curve of the model. Most of the has 0.97 area and the others has the area which is almost as high as this value. This area means the performance of the model is so high and awesome.

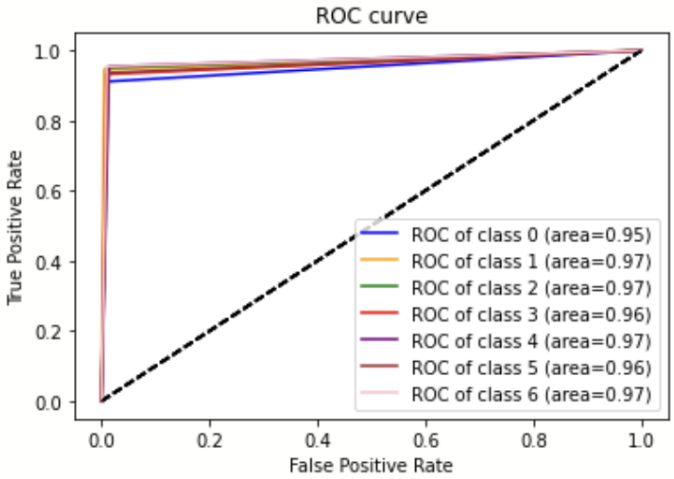


Figure 81: ROC curve of frequency domain KNN with data with backward feature elimination

5.5 Ensemble Learning

Ensemble learning tries to provide better prediction performance compared to a single model. The main idea of ensemble learning is to learn a set of classifiers, which may perform less on their own, and let them vote.
There are three main categories of ensemble learning methods: bagging, stacking, and boosting. Ensemble learning methods are more accurate when predicting the output than individual models. There are very useful when there is both linear and non-linear type in the dataset; different models can be combined to handle this data type. With ensemble methods, the bias/variance can be reduced, and most of the time, the model is not underfitting/overfitting. Therefore, the model shows an acceptable performance. There are always less noisy and are more stable.
On the other, Ensemble Learning methods are less interpretable, and the output of the ensemble model is hard to predict and explain. These methods are expensive in terms of both time and space.

1. First, the frequency domain form is examined along with data standardization. For this classifier, the time taken to train the model is approximately 19519.39 milliseconds, and evaluation metrics such as precision, recall and F1 score for each class and the overall model have been calculated. The metrics were calculated on training and test sets to evaluate model bias and variance errors.
   Looking at the results, it seems like class 1 and 6, which, respectively, the Rast Panjgah and Chargah dastgāh, has the highest value for precision, recall and f1 score. On the other hand, Mahur has been misclassified with 81% accuracy. The overall accuracy is 88% on training and 76% on the test set.

Train Data on frequency features

	precision	recall	f1-score	support
0	0.84	0.87	0.85	777
1	0.88	0.92	0.90	775
2	0.81	0.91	0.86	914
3	0.87	0.86	0.87	883
4	0.94	0.83	0.88	733
5	0.89	0.87	0.88	942
6	0.95	0.88	0.91	790
accuracy			0.88	5814
macro avg	0.88	0.88	0.88	5814
weighted avg	0.88	0.88	0.88	5814

Figure 82: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.67	0.81	0.73	259
1	0.74	0.84	0.79	258
2	0.67	0.82	0.74	305
3	0.78	0.65	0.71	295
4	0.82	0.66	0.73	244
5	0.81	0.74	0.77	314
6	0.87	0.75	0.80	263
accuracy			0.75	1938
macro avg	0.76	0.75	0.75	1938
weighted avg	0.76	0.75	0.75	1938

Figure 83: Test Data on frequency features

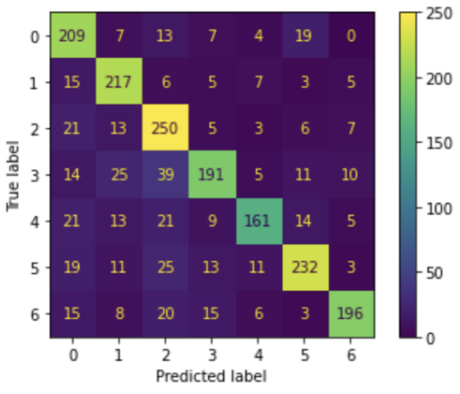


Figure 84: Confusion matrix of frequency domain Ensemble Learning with normalization

The results show that the model has done an excellent classification task, and

it discriminates well the distributed classes non-linearly. Because the accuracy of train data is 88%, there is little bias error, and variance error is unavoidable. The comparison of train and test accuracy shows that the model is Fair trained on the train data, but it still cannot completely separate the train and test data, so we have a little variance error.
Figure 85 shows the ROC curve for each class. As mentioned above, it seems that this model has the best performance on predicting class 1 with a 0.90 area and has the worst performance on predicting class 3 with a 0.81 area in comparison to other classes.

Figure 46 shows the ROC curve for each class. Just as mentioned above, it seems that this model has the best performance on predicting class 5 with 0.6 area and the rest of the classes have the worst efficiency by 6%.

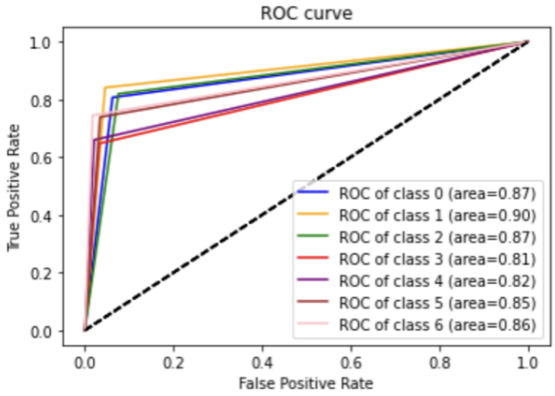


Figure 85: ROC curve of frequency domain Ensemble Learning with normalization

The learning curve of the model can be seen in figure 86. It has been shown from the learning curve that with the increase in the number of samples in the training set, the accuracy of the test data increases drastically up to 70%, and the accuracy of the training data decreases slightly by nearly 90%.
Here, overfitting did not occur because, during the training process, the accuracy of the test and training data converged. However, as we have a relatively small variance error, we have a bias error of about 10% on the train data.
The best accuracy for training data is about 0.97, while the best for test data is near 0.77.

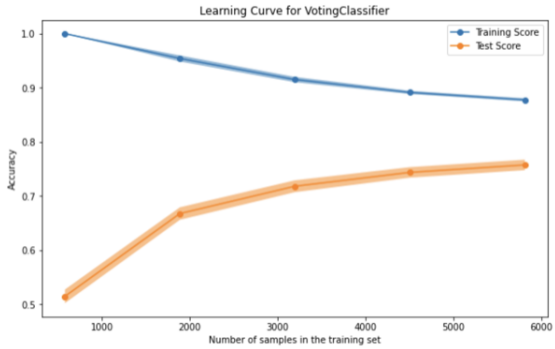


Figure 86: Learning curve for frequency domain Ensemble Learning model with normalization

2. In the second stage, the frequency form without data standardization is examined.

Train Data on frequency features

	precision	recall	f1-score	support
0	0.87	0.92	0.89	777
1	0.94	0.91	0.93	775
2	0.73	0.94	0.82	914
3	0.89	0.87	0.88	883
4	0.99	0.80	0.89	733
5	0.91	0.89	0.90	942
6	1.00	0.86	0.92	790
accuracy			0.89	5814
macro avg	0.90	0.89	0.89	5814
weighted avg	0.90	0.89	0.89	5814

Figure 87: Train Data on frequency features

Test Data on frequency features

	precision	recall	f1-score	support
0	0.39	0.57	0.46	259
1	0.48	0.48	0.48	258
2	0.30	0.63	0.40	305
3	0.44	0.33	0.37	295
4	0.59	0.15	0.24	244
5	0.52	0.52	0.52	314
6	0.76	0.16	0.26	263
accuracy			0.41	1938
macro avg	0.50	0.41	0.39	1938
weighted avg	0.49	0.41	0.40	1938

Figure 88: Test Data on frequency features

The non-standardization of data compared to the previous state has caused relatively large overfitting.

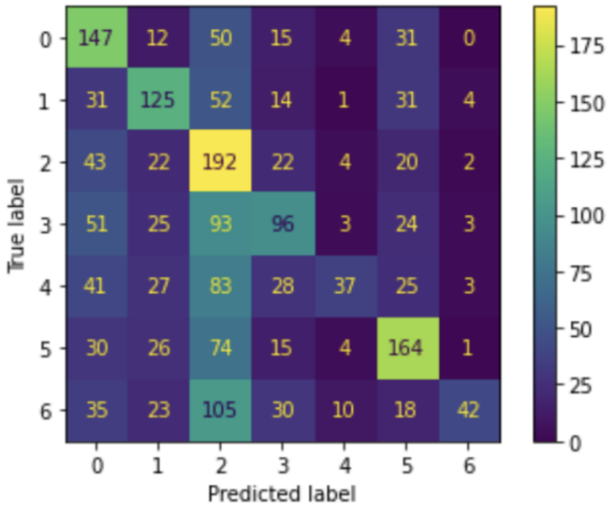


Figure 89: Confusion matrix for frequency domain of Ensemble Learning model without normalization

In this case, we have a lot of variances. The model is almost complicated, and the classes cannot be separated linearly. As a result, the model has a lot of bias. Also, due to the non-normality of the data, there is strong dispersion in each data set, resulting in high model variance.
As mentioned earlier, due to the complexity of the model and the excessive complexity of the data and the lack of normalization of the data, the model's accuracy on the test data could be more satisfactory.
Figure 90 shows the ROC curve for each class. As mentioned above, this model with an area of 0.72 performs best in predicting class 5; in classes 0 and 1, the model has performed well with a difference of one or two per cent, and the rest of the classes performed worst.

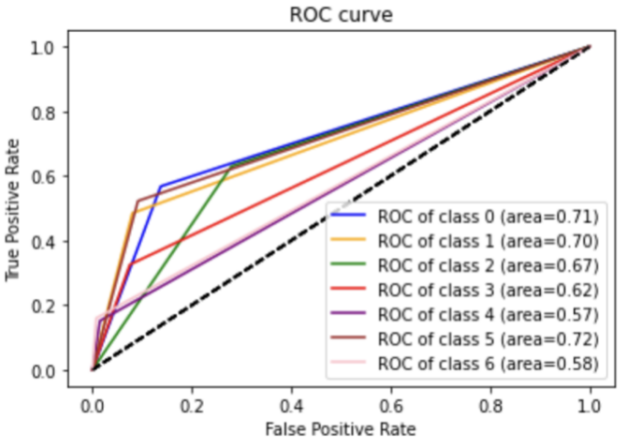


Figure 90: ROC curve of frequency domain Ensemble Learning model without normalization

The learning curve of the model can be seen in figure 91. It is shown from the learning curve that the accuracy of the test and training data increases with a slight slope as the number of samples in the training set increases. This means that overfitting occurs with the increase in the number of samples in the training data. The almost constant accuracy shows this in training and a slight increase in the accuracy of the test data. The model has a bias error of about 10%, but the model has too much variance and is therefore overfitted. During the training process, the test and training data accuracies do not converge and maintain their 50% gap.
The best accuracy for training data is around 100%, while the best accuracy for test data is close to 20%.

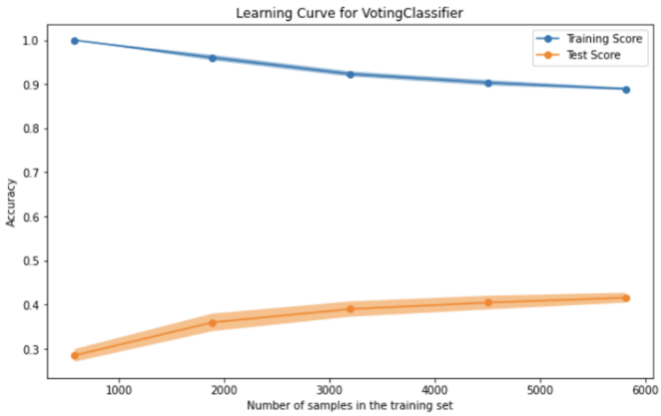


Figure 91: Learning curve for frequency domain Ensemble Learning model without normalization

3. The time domain form with data standardization is examined in the third stage. The time taken to train the model is approximately 21740.23 milliseconds.
   Looking at the results, all the classes have average value performance for precision, recall and f1 score on the training data. But the accuracy of the test data has been dramatically reduced, and the best test data is related to Rast Panjgah and Chargah with 96% and 88% accuracy. The lowest accuracy is related to Shur and Homayun, with 65%. The overall accuracy is 76% in training and 35% in the test set.
   Regarding the frequency characteristics, here we have a drop of about 30% accuracy on the test and training data. Therefore, the Ensemble Learning model provides a better result on the extracted frequency data.

Train Data on time features

	precision	recall	f1-score	support
0	0.65	0.86	0.74	777
1	0.73	0.81	0.77	775
2	0.66	0.77	0.71	914
3	0.65	0.78	0.71	883
4	0.96	0.50	0.66	733
5	0.79	0.79	0.79	942
6	0.88	0.55	0.67	790
accuracy			0.73	5814
macro avg	0.76	0.72	0.72	5814
weighted avg	0.76	0.73	0.72	5814

Figure 92: Train Data on time features

Test Data on time features

	precision	recall	f1-score	support
0	0.26	0.37	0.30	259
1	0.29	0.34	0.31	258
2	0.29	0.35	0.32	305
3	0.26	0.31	0.28	295
4	0.51	0.16	0.25	244
5	0.42	0.45	0.43	314
6	0.42	0.22	0.29	263
accuracy			0.32	1938
macro avg	0.35	0.31	0.31	1938
weighted avg	0.35	0.32	0.32	1938

Figure 93: Test Data on time features

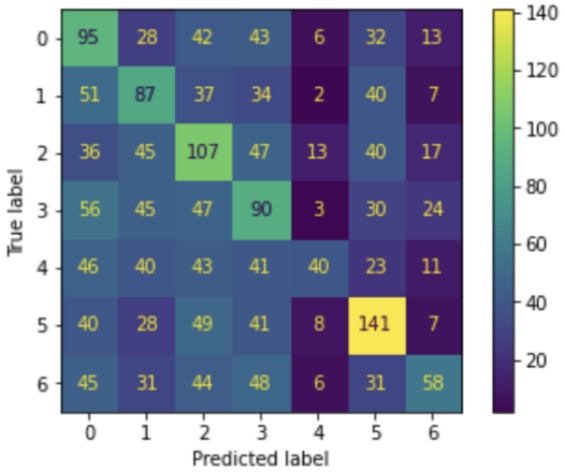


Figure 94: Confusion matrix of time domain Ensemble Learning model with normalization

The results show that the accuracy on the training data is 76%, and on the test data is 35%. The model performs relatively well on the training data, but the model has yet to be able to separate the test data well. Comparing train and test accuracy shows that overfitting has occurred. The accuracy of test and train data is not close, so we have variance errors.
Considering that the model's accuracy on the training data is 76%, the bias error is unavoidable.
Figure 95 shows the ROC curve for each class. As mentioned above, this model best predicts class 5 with 0.66 AUC.

Figure 46 shows the ROC curve for each class. Just as mentioned above, it seems that this model has the best performance on predicting class 5 with 0.6 area and the rest of the classes have the worst efficiency by 6%.

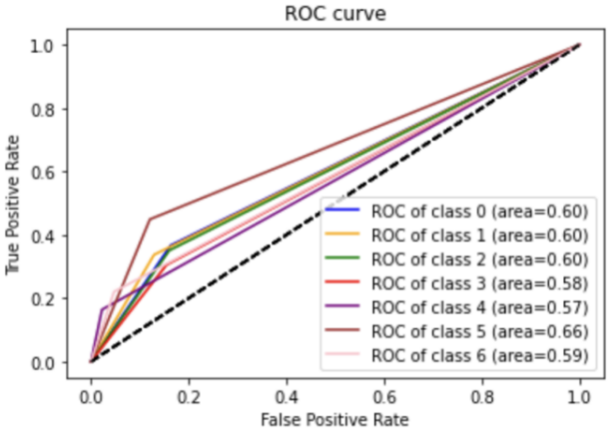


Figure 95: ROC curve of time domain Ensemble Learning model with normalization

The learning curve of the model can be seen in figure 96. It is shown that with the increase in the number of samples in the training set, the accuracy of the test data increases with a slight slope up to 30%, and the accuracy of the training data drops drastically, around 30%.
Finally, the distance between the accuracy of the training and test data is maintained at about 40%. As mentioned before, due to the model's simplicity and the excessive complexity of the data, the model's accuracy on the test data could be more satisfactory. The best accuracy for the training data is around 0.98, while the best for the test data is close to 0.3.

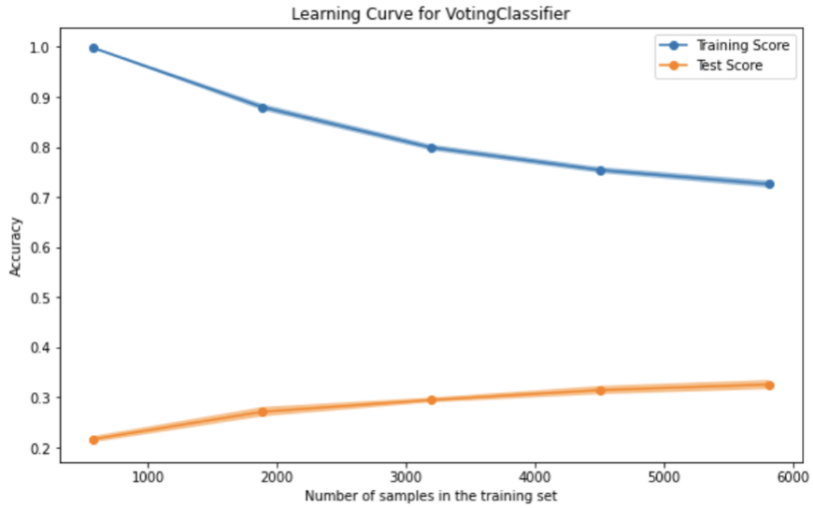


Figure 96: Learning curve for time domain Ensemble Learning model with normalization

4. The time domain form without data standardization is examined in the fourth stage.
The time taken to train the model is approximately 23663.02 milliseconds. Looking at the results, most classes have the best value for precision, recall and f1 score on training data. But the accuracy of the test data has dropped drastically; the best accuracy of the test data is related to the Nava with 60% accuracy, and the lowest accuracy is related to the Homayun with 19% accuracy. The overall accuracy is 83% on training and 33% on the test set.

Train Data on time features

	precision	recall	f1-score	support
0	0.77	0.86	0.81	777
1	0.84	0.84	0.84	775
2	0.77	0.76	0.76	914
3	0.46	1.00	0.63	883
4	1.00	0.43	0.60	733
5	1.00	0.68	0.81	942
6	1.00	0.53	0.69	790
accuracy			0.73	5814
macro avg	0.84	0.73	0.74	5814
weighted avg	0.83	0.73	0.74	5814

Figure 97: Train Data on time features

Test Data on time features

	precision	recall	f1-score	support
0	0.20	0.23	0.21	259
1	0.28	0.31	0.29	258
2	0.24	0.26	0.25	305
3	0.19	0.53	0.28	295
4	0.52	0.05	0.10	244
5	0.60	0.23	0.34	314
6	0.29	0.07	0.12	263
accuracy			0.25	1938
macro avg	0.33	0.24	0.23	1938
weighted avg	0.33	0.25	0.23	1938

Figure 98: Test Data on time features

The accuracy of the training data has increased compared to the previous mode we have standardized. But the accuracy of the test data has remained constant, so it can be concluded that the standardization of the temporal features, as opposed to the frequency features in the Ensemble Learning model, reduces the model's efficiency.

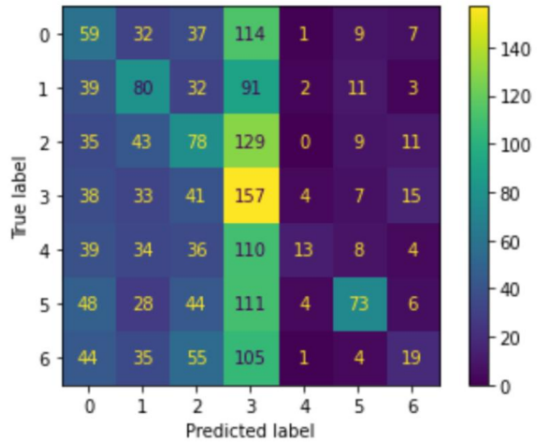


Figure 99: Confusion matrix of time domain Ensemble Learning model without normalization

The results show that the accuracy on the training data is 84%, and on the test data is 33%. The model performs relatively well on the training data, but the model has yet to be able to separate the test data well. Comparing train and test accuracy shows that overfitting has occurred. The accuracy of the test and train data are far apart, so the model has a significant variance error. Considering that the model's accuracy on the training data is 84%, the bias error is unavoidable. Figure 100 shows the ROC curve for each class. As mentioned above, this model has the best performance on predicting class 5 with 0.6 area, and the rest of the classes have the worst efficiency by 6%.

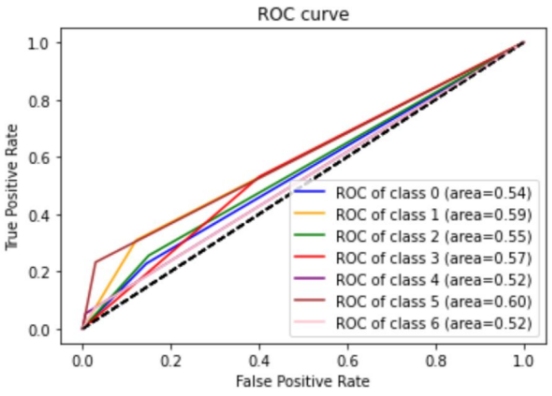


Figure 100: ROC curve of time domain Ensemble Learning model without normalization

The learning curve of the model can be seen in figure 101. It is shown that as the number of samples in the training set increases, the accuracy of the test data increases with an almost linear slope of up to 30%, and the accuracy of the

training data decreases sharply, around 30%. Overfitting occurs in the ensemble learning model, which is shown by the low decrease accuracy of the training and the low increase in the accuracy of the test data. The bias value is around 30%, but the model has a lot of variances.
Finally, the gap between training and test data accuracy is kept at about 60%. As mentioned earlier, the model's accuracy on the test data could be more satisfactory due to the excessive complexity of the data. The absolute accuracy for training data is about 0.8, while the final for test data is near 0.24.

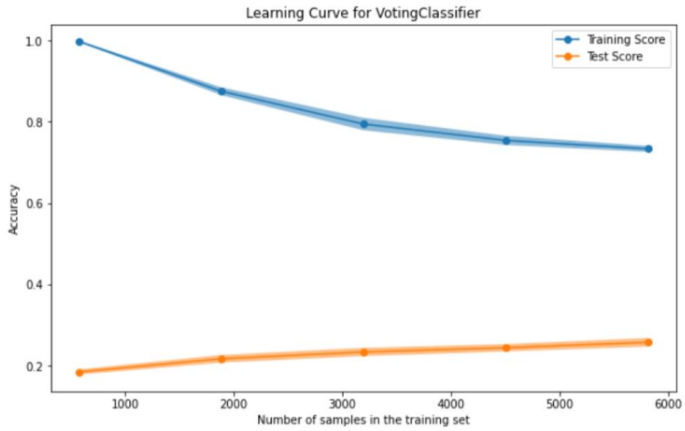


Figure 101: Learning curve for time domain Ensemble Learning model without normalization

	Data standardization	Train Accuracy %	Test Accuracy %	Fit Time milliseconds
Frequency domain features	Standard	88	76	19519.39
Frequency domain features	Not standard	90	50	25117.25
Time domain features	Standard	76	35	21740.23
Time domain features	Not standard	100	33	23663.02

Table 2:

The results in the table show that the model with the best performance uses frequency features along with data standardization, just like the KNN model.
It shows 100% accuracy on non-standardized data with time domain features, and the model can train well on extensive data with temporal features. But it needs to indicate reasonable accuracy on test data with few numbers.
That's why we use frequency features to have satisfactory test and training data accuracy. As we can see, the standardization of the frequency characteristics has had a favourable effect. It has improved the results, while the standardization of the time extracted data has worsened the output results.
Therefore, we can only sometimes conclude that standardization improves model performance, as we observed this exception in this model.

5.6 Deep learning

One of the best ways to classify sequential data, such as sound signals, is to use a neural network or a deep learning network. Deep learning methods can be supervised, semi-supervised or unsupervised. The architecture of a deep neural network can consist of multiple layers with multiple activation functions. Each layer added to the mode can have different functionalities. For instance, there are convolutional layers that tend to extract features from the input. Convolution is the simple application of a filter to an input that results in activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of detected features in an input. There are also dropout layers in which the result of some neurons is ignored from the last layer, and only a percentage of the outcome from the previous layer is passed to the next.
This is helpful for overfitting prevention. The output of each layer is a set of parameters, and if the number of these parameters extends without any limitations, the model becomes complex, and therefore overfitting occurs. After the features and parameters are extracted, there must be layers in the neural networks to perform the classification task.
At the end of deep neural networks, there are usually layers called dense layers which take all the generated parameters from the previous layer and it is fully connected to the preceding layer. It performs matrix multiplication and outputs a matrix of the required size. Dense layers can be put one after another to generate better classification. The output of the last layer must have the dimension of the true class labels in order to be trained using supervised methods and also to later predict class labels. The dense layers must have an activation function to decide which class the input belongs to. Last but not least, deep neural networks must have an optimizer and a metric to perform the learning process based on. Optimizers are procedures that change the parameters and weights of the model in order to reduce the losses. Optimizers can also change attributes such as learning rates. The metric chosen is also important as, based on the type and variability of data and also the meaning of the problem, the model should optimize a certain metric. Therefore, the best-suited metric is chosen and used in the learning process.
For a task in which a simpler classifier might give out good results, there is no need for the use of deep neural networks as these networks mostly take longer to train, and because of the high number of parameters generated, they are at high risk of being overfitted. The simpler the model, the less chance it has to be overfitted. Another fact about neural networks is that they require a huge amount of data. If the dataset is not large enough, the model might get biased toward the input data and perform badly on unseen data. In order to avoid overfitting and also increase the generalization and interpretability of the model, a simple model is built to train the data. This model consists of three one-dimensional convolutional layers, each followed by a dropout layer. To perform the classification, two dense layers are also added to the model. Dense layers only take one-dimensional data, and therefore a flattening layer is added between the two parts to flatten the output of convolutional layers and pass it to fully connected parts. The model is built using the Keras library, and the architecture can be seen in figure 102.

```
model1 = Sequential()
model1.add(Conv1D(32, kernel_size=3, activation='relu', input_shape=fxTrain.shape[1:]))
model1.add(Dropout(0.2))
model1.add(Conv1D(64, kernel_size=3, activation='relu'))
model1.add(Dropout(0.2))
model1.add(Conv1D(128, kernel_size=3, activation='relu'))
model1.add(Dropout(0.2))
model1.add(Flatten())
model1.add(Dense(128, activation='relu'))
model1.add(Dense(7, activation='softmax'))

model1.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Figure 102: CNN model architecture

As can be seen in figure 102, convolutional layers have a kernel size of 3 and use the activation function of ReLU. The ReLU activation function is computed efficiently and helps with better propagation. As mentioned earlier, a large number of model parameters results in overfitting, and to help build a better model, a dropout layer is added to ignore 20% of the parameter in the next layer.
The last layer is a fully connected layer with shape 7, indicating the seven classes and an activation function of softmax. Softmax, or normalized exponential function, converts a vector of n real numbers to the probability of belonging to n classes. The model is compiled using adam optimizer as adam generally works better than any other optimizer, and it is faster in computation and requires fewer parameters to tune. Also, binary cross entropy is used as the loss function, and model accuracy is monitored. This model is to be tested on the two-dimensional datasets of frequency and time domain features. Features are normalized and splatted into train, validation and test sets. The model is to be trained for 50 epochs with an early stop monitoring train loss.

```
es = EarlyStopping(monitor='loss', mode='min', verbose=1, patience=3)
# Fit data to model
fhistory = model1.fit(fxTrain, fyTrain, validation_data=(fxValid, fyValid), epochs=50, callbacks=[es])
```

Figure 103: raining the CNN model

The training process stops after 27 epochs for frequency domain features and 31 epochs for time domain features. The training and validation loss and accuracy values are plotted in Figures 104 and 105 for both feature sets. As seen in the previous models, frequency domain features result in a better classifier with higher accuracy. Based on the plots, it can be seen that both models have overfitted on train data, but the difference between validation and train accuracy for the model on frequency features is much less than the model with time-domain features. The time domain model results from a training accuracy of 99% while maintaining a validation accuracy of approximately 30%. The same results can also be seen when the model is tested using a test dataset. The test dataset consists of 776 unseen music records of a length of 20 seconds, and the prediction on frequency domain features is 93% accurate, while the prediction of time domain features is only 29% accurate.

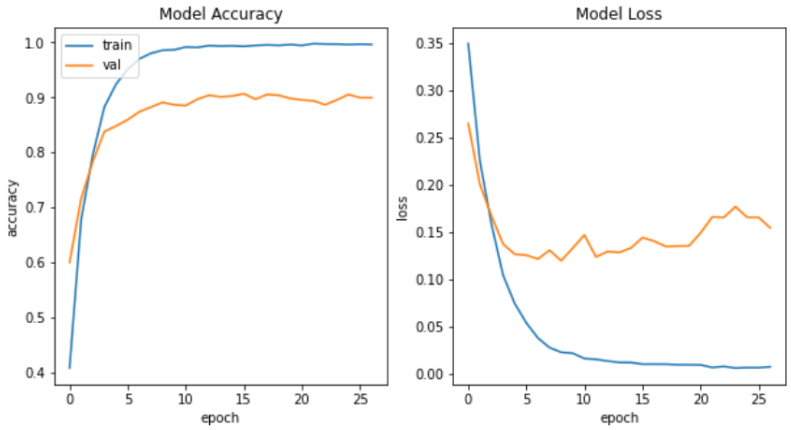


Figure 104: CNN model metrics for frequency domain features

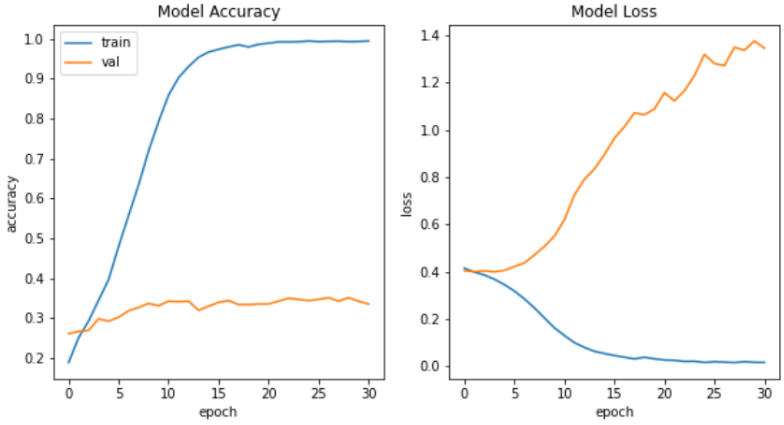


Figure 105: CNN model metrics for time domain features

One of the main advantages of deep neural networks is that there is no need to have multiple features, and there is no need for complex preprocessing of data before passing it to the classifier. Therefore, one other approach that was taken in this section was to generate new features from the dataset and, without preprocessing the signal or combining it with other features, train a deep neural network on it. The extracted feature is the MFCC explained in the first part. This feature is calculated for music with a length of 20 seconds, resulting in 216 frames for each piece and ten components for each frame. The dataset is also normalized and given to a two-dimensional convolutional network. The network is exactly the same as the one-dimensional network explained in the previous section. The reason that it is two-dimensional is that features are 216 * 10 for each piece of music. The network is also trained with the exact same conditions to ease the comparison process.

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=xTrain.shape[1:]))
model.add(Dropout(0.2))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(Dropout(0.2))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(7, activation='softmax'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Figure 106: CNN model architecture on new set of features

The training process stopped after 17 epochs with a training accuracy of 98.96% and a validation accuracy of 62.9%. Train and validation loss and accuracy are plotted in figure 107. Testing the result on the test set also results in 62% accuracy, which is reasonable considering only one feature. It was seen earlier that using a handful set of features results in 94% accuracy with KNN and 65% with XGBoost. Considering that the model is simple, does not take much space or time to train, and converges in early epochs, it could be a better choice than XGBoost. The only problem with the model is that according to the plots, it has been overfitted considerably as the dataset is small and the model has 13,856,263 parameters which is a considerable number, which results after dropping 20% of all parameters generated in convolutional layers. One way it can be solved is to add max pool layers, but because of the low dimension of the 3rd dimension of the dataset, this was impossible and did not work out when it was tried to implement the model that way.

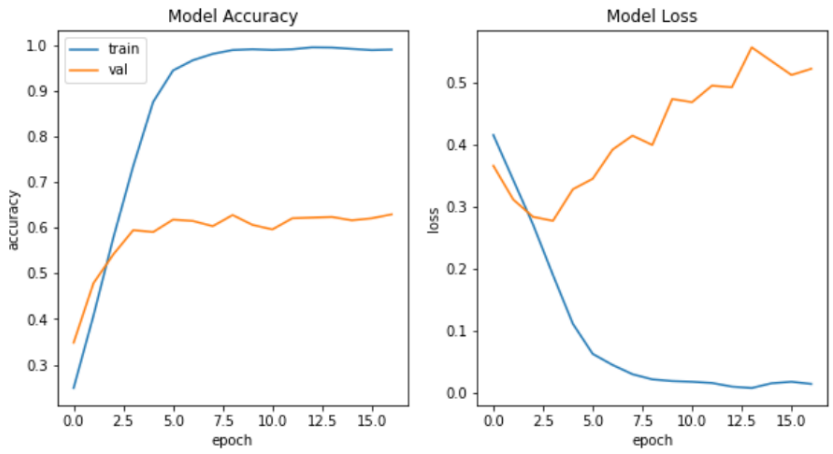


Figure 107: CNN model metrics for MFCC Features

5.6.1 GRU(Gated Recurrent Unite)

GRUs are a mechanism that controls input as a gated unit with a recurrent neural network. GRU is functionally similar to a long short-term memory (LSTM) with a forgetting gate, except that it does not have an output gate, which makes it have fewer parameters than LSTM.
One of the goals of GRU was to solve the vanishing gradient problem, which is caused by ignoring and reducing the loss effect of inputs given to the network long ago.
In the following, we will examine the difference between CNN and RNN networks. The main difference between RNN and CNN is that RNN uses memory to retrieve any adequate information from previous inputs that, combined with the current input, affects the current output. In CNN, we assume that the outputs and inputs are independent of each other, while in RNNs and especially GRUs, the output of the current time step is dependent on the input of the previous time step. RNNs have the same weights in each layer of the network, while in CNNs, the weights differ from each node.
GRU models are at high risk of being overfitted because of the high number of parameters generated. The simpler the model, the less chance it has to be overfitted. This model consists of three Bidirectional, which involve GRU layers. Four dense layers are added to the model to perform the classification. Dense layers only take one-dimensional data; therefore, a flattening layer is added between the two parts to flatten the output of GRU layers and pass it to fully connected layers. The model is built using the Keras library, and the architecture can be seen in figure 108.

```
myinput = layers.Input((216, 10)) #6278,

#x = layers.GaussianNoise(10)(x)
x = layers.Bidirectional(layers.GRU(100,return_sequences=True))(myinput)
x = layers.Bidirectional(layers.GRU(100,return_sequences=True))(x)
x = layers.Bidirectional(layers.GRU(100,return_sequences=True))(x)
x = layers.Flatten()(x)
x = layers.Dense(128, 'tanh')(x)
x = layers.Dense(128, 'tanh')(x)
x = layers.Dense(128, 'tanh')(x)
x = layers.Dense(7, activation='sigmoid')(x)
model = tf.keras.models.Model(myinput, x)
```

Figure 108: GRU model architecture

As seen in figure 108, GRU layers use the activation function of ReLU. The ReLU activation function is computed efficiently and helps with better propagation. In the GRU layer, the previous layer's output is connected to the input of the next layer. The last layer is a fully connected layer with shape 7 representing seven classes and a sigmoid activation function. The model is compiled using the adam optimizer because adam usually works better than any other optimizer, is faster in computation, and requires fewer parameters to tune.
Also, binary cross-entropy is monitored as a loss function and model accuracy. This model will be tested on datasets of two-dimensional frequency and time domain features. The features are normalized and put into train, validation and test sets.

```
history=model.fit(xTrain,yTrain, batch_size= num_batch_size, epochs = num_epoch,validation_split=0.2,callbacks=[clrs])
loss, accuracy, f1_score, precision, recall = model.evaluate(xTest, yTest, verbose=0)
model.save('/content/drive/MyDrive/ML_Project/sina_MFCC_feature.h5')
```

Figure 109: training the GRU model

The training process stops after 110 epochs for frequency domain features and 100 epochs for time domain features. The training and validation loss and accuracy values are plotted in Figures 110 and 111 for both feature sets. As seen in the previous models, frequency domain features result in a better classifier with higher accuracy. Based on the plots, it can be seen that both models have overfitted on train data. Still, the difference between validation and train accuracy for the model, which is trained on frequency features, is much less than the model with time-domain features. The time domain model results in a training accuracy of 99% while maintaining a validation accuracy of approximately 30%. The same results can be seen when the model is tested using a test dataset. The test dataset consists of 776 unseen music records of 20 seconds. The prediction of frequency domain features is 89% accurate, while the prediction of time domain features is only 28% accurate.

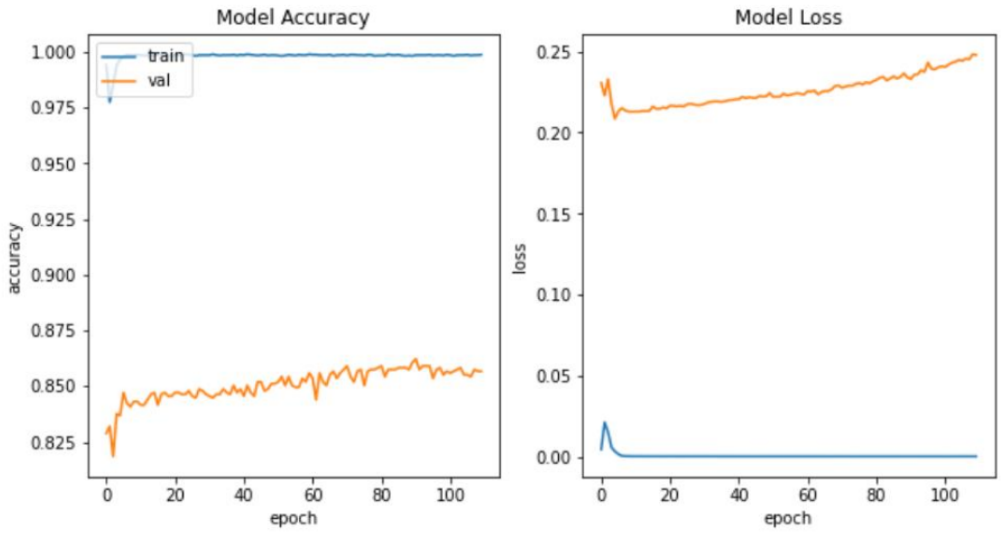


Figure 110: GRU model metrics for frequency domain features

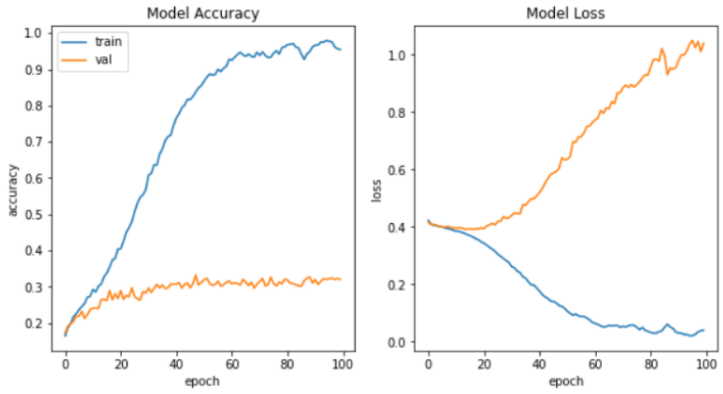


Figure 111: GRU model metrics for time domain features

The extracted feature is the MFCC explained in the CNN section. This feature is calculated for music with a length of 20 seconds, resulting in 216 frames for each piece and ten components for each frame. The dataset is also normalized and given to a Bidirectional GRU model, which is explained in figure 108. The network is also trained with the same conditions to ease the comparison process.

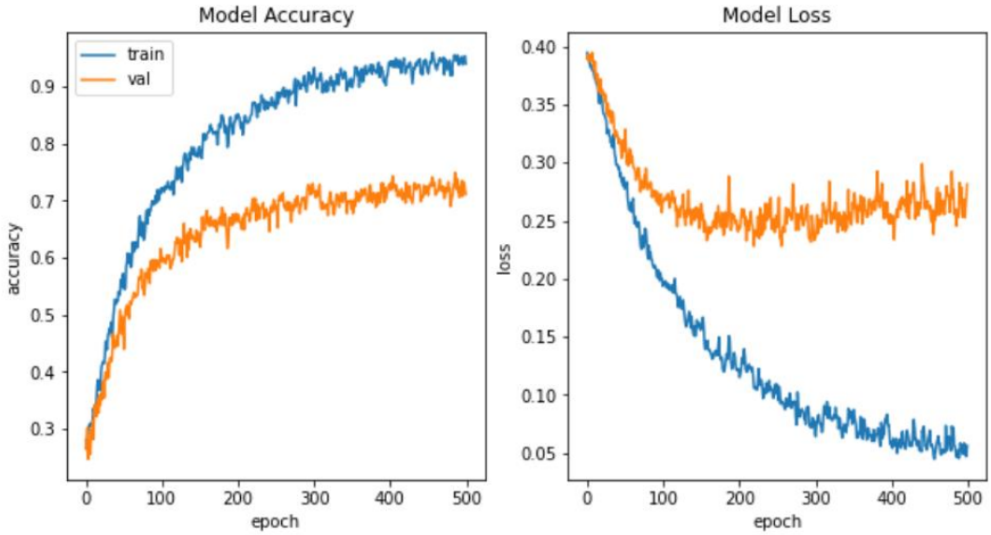


Figure 112: GRU model metrics for MFCC Features

The number of epochs is considered to be 500, the accuracy of the model on the training data is about 95.16%, and the validation accuracy is 73.17%.
Train and validation loss and accuracy are plotted in figure 112. Testing the result on the test set also results in 70.48% accuracy, which is reasonable considering only one feature.
The only problem with the model is that according to the graphs, it could be more balanced because the data set is small, and the model has 5,993,255 parameters.
Compared to the CNN network, the model has obtained about 10% higher accuracy

on the test and validation data, and the overfitting is significantly lower than the CNN model because the number of parameters of the model has been reduced to one-third compared to CNN.
Finally, a table is provided to compare these two models.

	CNN		Bidirectional-GRU	
	Train Accuracy	Test Accuracy	Train Accuracy	Test Accuracy
Time domain	99.41%	29.25%	95.48%	28.99%
Frequency domain	99.63%	93.04%	99.88%	89.04%
MFCC feature	98.96%	61.98%	95.16%	70.48%

6 Clustering

۶ خوشه‌بندی

باید در نظر داشت که برای خوشه‌بندی باید ویژگی‌ها از منظر فواصل اقلیدسی تفاوت معنا داری باهم داشته باشند تا بتوان به خوشه‌بندی درستی رسید و به همین علت در هر دو دسته ویژگی چه دامنه‌ای و چه فرکانسی جداکننده‌های خوبی برای ما نیستند در نتیجه ما در اکثر خوشه‌ها توزیعی نزدیک به میانگین در هر خوشه یعنی نزدیک به ۱۴ درصد (۱۰۰/۷) را شاهد هستیم. به طور کلی خوشه‌های ما به سه دسته تقسیم می‌شوند: ۱. خوشه هایی که همه دستگاه‌ها به طور یکسان توزیع شده‌اند. در رابطه با این خوشه‌ها هیچ تحلیلی نمی‌توان کرد. ۲. خوشه‌هایی که توزیع اکثریت دستگاه‌ها نزدیک به هم و یک یا دو دستگاه به طور چشم‌گیری کمتر و یا بیشتر از مابقی دستگاه‌ها حضور دارند. در این نوع از خوشه‌ها می‌توان تحلیل‌هایی مبنی بر متفاوت بودن این دستگاه‌ها در آن ویژگی خاص نسبت به سایر دستگاه‌ها را نتیجه‌گیری کرد. ۳. خوشه‌هایی که شامل تعداد محدودی دستگاه به طور مجزا هستند. در این نوع از خوشه نتیجه‌گیری‌هایی از قبیل شبیه بودن آن دستگاه‌ها و یا متفاوت بودن یک دستگاه با سایر دستگاه‌ها را می‌توان نتیجه‌گیری نمود.

نخست نگاهی بیندازیم به فراوانی دسته‌ها در حالت کلی:

۰	شور	۱۰۳۶
۱	سه‌گاه	۱۰۳۳
۲	ماهور	۱۲۱۹
۳	همایون	۱۱۷۸
۴	راست‌پنج‌گاه	۹۷۷
۵	نوا	۱۲۵۶
۶	چهارپنج‌گاه	۱۰۵۳

باتوجه به فراوانی دستگاه‌ها می‌توان گفت توزیع‌ها نزدیک به هم و به اصطلاح بالانس هستند.

خوشه بندی چیست؟

گروه‌بندی داده‌ها بر اساس یک معیاری از شباهت را گوییم که در این پروژه ما سه الگوریتم متفاوت را برای خوشه‌بندی پیاده‌سازی کرده و نتایج هر یک را به تفضیل بیان خواهیم کرد.

- Kmeans
- Clustering Agglomerative
- Mixture Gaussian

باید به این نکته توجه داشت که فاصله در ابعاد بالا مفهوم متفاوتی در عمل خواهد داشت پس بهتر است قبل از اجرای الگوریتم‌ها ابتدا کاهش بعد داشته باشیم که در ادامه هر دو مورد بررسی خواهد شد.

Kmeans ۱.۶

یکی از الگوریتم های ساده و بدون نظارت اما در عین حال پرکاربرد خوشه بندی، الگوریتم k-means میباشد. این الگوریتم بر مبنای فاصله درون گروهی و برون گروهی کار کرده و برای مسائلی مناسب است که فاصله فیزیکی در آنها معنا داشته باشد. یکی از نقاط ضعف این الگوریتم این است که در ابتدای کار، یعنی قبل از انجام خوشه بندی، تعداد خوشه ها باید مشخص باشد. این امر برای مسائلی که هدف آن مشخص کردن توزیع های مختلف است، یک عیب بزرگ تلقی شده و نیاز به آزمون های متعدد دارد. این الگوریتم بر پایه ی تکراری است که سعی میکند مجموعه داده ها را به زیرگروههای متمایز بدون همپوشانی تعریف کند. روش کار این الگوریتم بدین صورت است که به صورت تصادفی K نشان دسته انتخاب کرده (معمولا میانگین) و سپس فاصله همه نقاط را با همه K نشان دسته حساب میکند. سپس هر نقطه را به دسته ایی که کمترین فاصله را با نشان دسته آن داشته اختصاص می دهد. در مراحل بعد که به صورت تکراری طی شده که یا تا به تعداد تکرار مورد نظر و یا تا ثابت شدن فواصل به طول می انجامد، نشان دسته ها با توجه به نقاط هر دسته تغییر کرده و مجدد فواصل نقاط از نشان دسته ها محاسبه شده و در صورت لزوم، تعلق نقاط نیز بروزرسانی میشوند. در ادامه این الگوریتم بر روی دادگان، با تعداد دسته های متغیر، اعمال شده و نتایج حاصل بررسی میشوند.

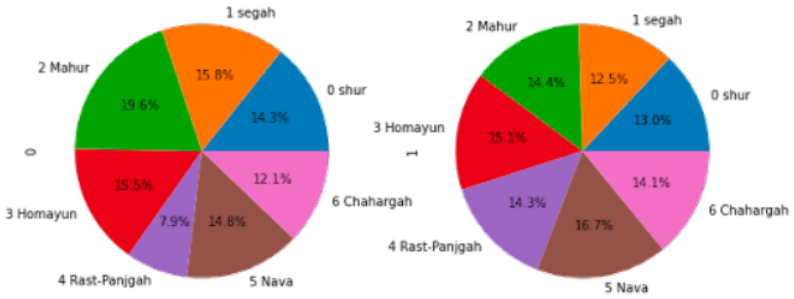
Inertia میزان خوشه‌بندی یک مجموعه داده توسط K-Means را اندازه‌گیری می‌کند. با اندازه‌گیری فاصله بین هر نقطه داده و مرکز آن، مجذور کردن این فاصله و جمع کردن این مربع‌ها در یک خوشه محاسبه می‌شود. یک مدل خوب مدلی با اینرسی کم و تعداد خوشه کم (K) است.
دیگر معیاری که در این تحلیل بررسی شده است score silhouette است که به نوعی بیان می‌دارد که یک ابجکت به چه میزان به یک دسته تعلق دارد یا از آن دسته دور است.

frequency features - Two clusters - Without PCA ۱.۱.۶

inertia: 583896
silhouette score: 16.2%

inertia:	583896.3741192305	
predict	actual	
0	0	288
	1	318
	2	395
	3	312
	4	159
	5	297
	6	244
1	0	748
	1	715
	2	824
	3	866
	4	818
	5	959
	6	809

شکل ۱: توزیع دستگاه‌های مختلف در دو خوشه



شکل ۲: Distribution of frequency features - Two clusters - Without PCA

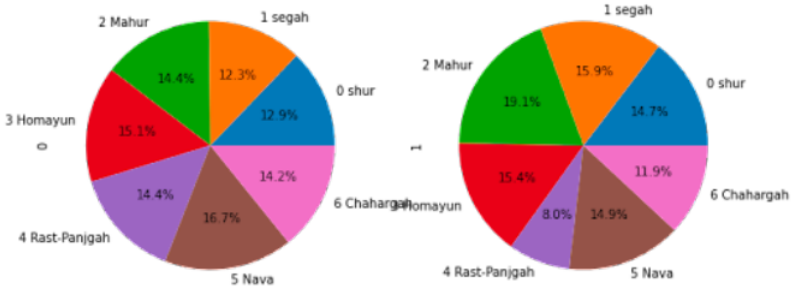
تقریبا دستگاه‌ها در هر خوشه به نسبت مساوی تقسیم شده اند یک حدس اولیه آن است که دستگاه راست‌پنج‌گاه و ماهور از منظر فرکانسی تفاوت‌هایی باهم دارند چرا که در خوشه اول به میزان ۶.۱۹ درصد دستگاه ماهور است درصورتی که دستگاه راست‌پنج‌گاه تنها نزدیک به ۸ درصد از کل را شامل می‌شود اما در خوشه دوم که قاعدتا باید از منظر فرکانسی تفاوت معناداری با خوشه اول داشته باشد دستگاه راست‌پنج‌گاه تقریبا دو برابر خوشه قبل یعنی ۳.۱۴ درصد از این خوشه را به خود اختصاص داده است و این درحالی است که ماهور یک کاهش نزدیک به ۵ درصدی را رقم زده است.
مابقی دستگاه‌ها تقریبا در هر دو خوشه ثابت بوده‌اند و نمی‌توان درمورد شباهت و عدم شباهت آن‌ها نظری داد.

frequency features - Two clusters - With PCA ۲.۱.۶

inertia: 85706
silhouette score: 43.4%

```
inertia:  85706.538579952
predict  actual
0        0         720
         1         690
         2         808
         3         846
         4         804
         5         935
         6         796
1        0         316
         1         343
         2         411
         3         332
         4         173
         5         321
         6         257
```

شکل ۳: توزیع دستگاه‌های مختلف در دو خوشه



شکل ۴: Distribution of frequency features - Two clusters - With PCA

در این‌جا نیز ما فضای ویژگی‌ها را به دو بعد کاهش دادیم و همان‌طور که مشاهده می‌شود نتایج با تقریب خوبی بسیار شبیه به قسمت قبل است ماهور و راست‌پنج‌گاه در دو طیف فرکانسی متفاوت قرار دارند در اینجا به نظر می‌رسد چهارگاه هم ویژگی فرکانسی مشابهی با دستگاه راست‌پنج‌گاه دارد به طوری که در خوشه اول میزان هر دو به نسبت خوشه دوم زیادتر است.
مابقی دستگاه‌ها هم تحلیل قابل بیانی را نمی‌توان گفت.

۳.۱.۶ frequency features - Seven clusters - Without PCA

inertia: 492404
silhouette score: 5.1%

شکل ۵: Distribution of frequency features - Seven clusters - Without PCA

همان‌طور که مشاهده می‌شود در خوشه‌های مختلف رفتار دستگاه نوا و ماهور کاملا عکس هم عمل می‌کنند بدین صورت که با زیاد شدن یکی در یک خوشه دیگری کاهش می‌ابد و برعکس بدین منظور شاید بتوان نتیجه گیری کرد که احتمالا در دو طیف فرکانسی متفاوت قرار دارند این دو دستگاه.
همایون و شور تقریبا در همه خوشه‌ها توزیع نزدیک به میانگین حوالی ۱۴ درصد را دارند.
دستگاه سه‌گاه و چهارگاه هم در هر خوشه یا درصد مشابه و نزدیک به هم را دارند و یا در جهت عکس یک دیگر با فاصله گرفتن یکی از ۱۴ درصد دیگری نسبت به میانگین کاهش می‌یابد در خوشه‌های مختلف و به نوعی می‌توان گفت این دو دستگاه از منظر فرکانسی متفاوت عمل می‌کنند.

۴.۱.۶ frequency features - Seven clusters - With PCA

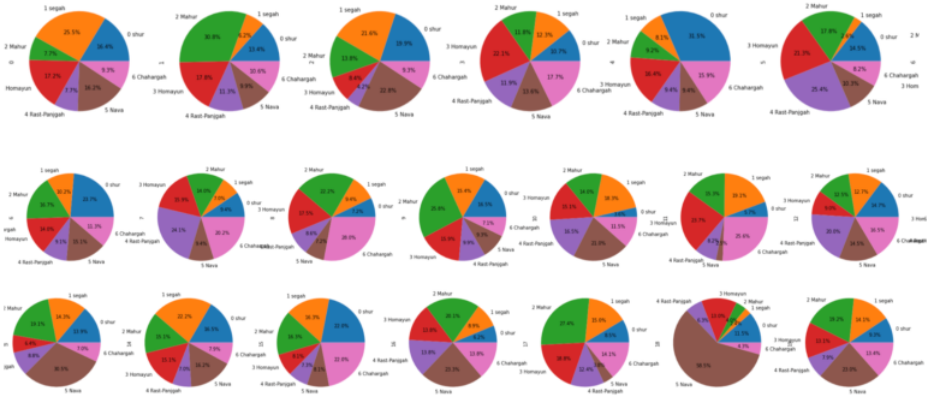
inertia: 27455
silhouette score: 33.1%

شکل ۶: Distribution of frequency features - Seven clusters - With PCA

در نمونه کاهش یافته این ویژگی‌ها آن‌چه درمورد دستگاه ماهور و نوا گفتیم در اینجا فقط در برخی خوشه‌ها صادق است (در خوشه‌هایی که توزیع آن‌ها به طرز چشم‌گیری کم است مثلا در خوشه ۲ نوا ۱.۵ درصد است در حالی که ماهور سهم زیادی از این خوشه نزدیک به ۲۲ درصد را در برگرفته است اما در خوشه ۶ توزیع هر دو این دستگاه‌ها زیاد است.) در نتیجه نمی‌توان استباطی را بیان کرد.
راست‌پنج‌گاه و سه‌گاه هم در ویژگی‌های فرکانسی متفاوت عمل می‌کنند.
همایون در همه خوشه‌ها توزیع نزدیک به میانگین دارد.

frequency features - Twenty clusters - Without PCA ۵.۱.۶

inertia: 424834
silhouette score: 5.2%

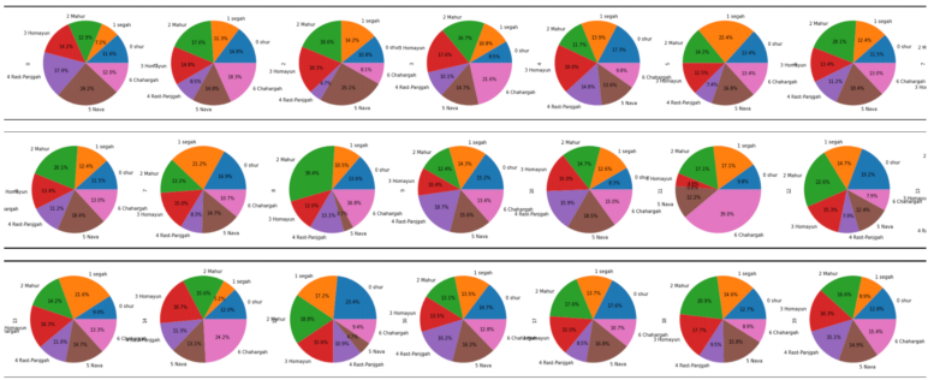


شکل ۷: Distribution of frequency features - Twenty clusters - Without PCA

همان‌طور که مشاهده می‌شود دستگاه ماهور و سه‌گاه در دو طیف فرکانسی متفاوت قرار دارند با فاصله گرفتن یکی از میانگین دیگری در جهت عکس آن عمل می‌کند.
نوا نیز در اکثر خوشه‌ها متفاوت از سایرین عمل کرده است و به نظر می رسد در طیف فرکانسی متفاوتی از سایر دستگاه‌ها قرار دارد.
درمورد مابقی دستگاه‌ها نمی‌توان نتیجه‌گیری کرد.

frequency features - Twenty clusters - With PCA ۶.۱.۶

inertia: 10139
silhouette score: 33.2%



شکل ۸: Distribution of frequency features - Twenty clusters - With PCA

با کاهش بعد در این مرحله دستگاه نوا در اکثریت دسته‌ها نزدیک به میانگین می‌شود.
اما دستگاه ماهور و سه‌گاه همچنان می‌توان استباط کرد که در دو طیف فرکانسی مختلف قرار دارند.
درمورد مابقی دستگاه هابقی ویژگی‌ها نمی‌توان نتیجه‌گیری کرد.

نتیجه‌گیری نهایی:

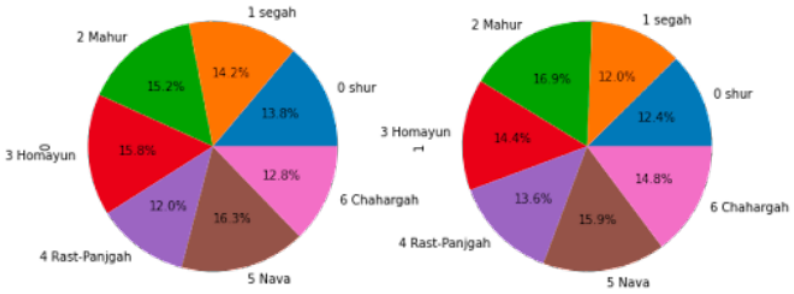
بر روی ویژگی‌های فرکانسی و با استفاده از الگوریتم K-Means با وجود این‌که در اکثر خوشه‌های تشکیل شده توزیع دستگاه‌ها نزدیک به میانگین یعنی حوالی ۱۴ درصد است و این قضیه نشان دهنده‌ی آن است که طیف گسترده‌ای از آهنگ‌ها جدای از نوع دستگاه در طیف فرکانسی یکسانی قرار دارند.
اما با استنباط و تحلیل کلی از هر ۳ مدل خوشه‌بندی می‌توان گفت دستگاه‌های راست‌پنج‌گاه و چهارگاه در برخی از آهنگها در فرکانس شبیه به هم و در مقابل سه‌گاه و ماهور در فاصله دورتری نسبت به این دو قرار دارد و متفاوت عمل می‌کند.
نوا نیز به طور جداگانه‌ای متفاوت از مابقی دستگاه‌ها عمل کرده است.

۷.۱.۶ domain time features - Two clusters - Without PCA

inertia: 2328509
silhouette score: 25.3%

predict	actual	
0	0	652
	1	670
	2	714
	3	742
	4	564
	5	767
	6	601
1	0	368
	1	356
	2	501
	3	428
	4	404
	5	473
	6	439

شکل ۹: توزیع دستگاه‌های مختلف در دو خوشه



شکل ۱۰: Distribution of domain time features - Two clusters - Without PCA

آن‌چه در قسمت قبل مشاهده شد در اینجا نیز پس از کاهش بعد صادق است و دستگاه سه‌گاه و چهارگاه در دو خوشه متفاوت عکس هم عمل کرده‌اند.
از دیدگاه دیگر می‌توان دستگاه‌ها را به دسته تقسیم‌بندی کرد دستگاه‌های همایون، سه‌گاه، شور و نوا در خوشه اول به میزان زیادتری نسبت به خوشه دوم قرار دارند در نتیجه شاید از منظر دامنه‌ای بیشتر به هم شبیه باشند همین موضوع درمورد راست‌پنج‌گاه، ماهور، چهارگاه نیز صادق خواهد بود.

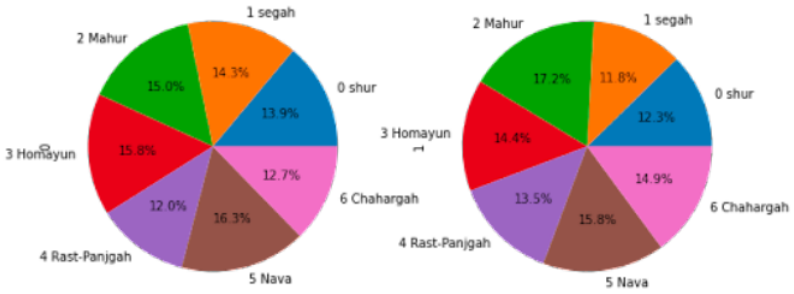
در هر دو خوشه تمام دستگاه‌ها توزیع نزدیک به میانگین حدودا ۱۴ درصد را دارند.
تنها می‌توان گفت سه‌گاه و چهارگاه متفاوت عمل کرده و شاید بتوان گفت از منظر دامنه‌ای این دو دستگاه متفاوتند.

۸.۱.۶ domain time features - Two clusters - With PCA

inertia: 529808
silhouette score: 25.3%

```
inertia:  529808.7265738118
predict  actual
0        0          653
         1          673
         2          703
         3          740
         4          565
         5          768
         6          596
1        0          367
         1          353
         2          512
         3          430
         4          403
         5          472
         6          444
```

شکل ۱۱: توزیع دستگاه‌های مختلف در دو خوشه



شکل ۱۲: Distribution of domain time features - Two clusters - With PCA

آن‌چه در قسمت قبل مشاهده شد در اینجا نیز پس از کاهش بعد صادق است و دستگاه سه‌گاه و چهارگاه در دو خوشه متفاوت عکس هم عمل کرده‌اند.
از دیدگاه دیگر می‌توان دستگاه‌ها را به دسته تقسیم‌بندی کرد دستگاه‌های همایون، سه‌گاه، شور و نوا در خوشه اول به میزان زیادتری نسبت به خوشه دوم قرار دارند در نتیجه شاید از منظر دامنه‌ای بیشتر به هم شبیه باشند همین موضوع درمورد راست‌پنج‌گاه، ماهور، چهارگاه نیز صادق خواهد بود.

۹.۱.۶ domain time features - Seven clusters - Without PCA

inertia: 1906599
silhouette score: 8.4%

شکل ۱۳: Distribution of domain time features features - Seven clusters - Without PCA

در خوشه نخست و آخر تمامی دستگاه‌ها توزیعی نزدیک به نرمال دارند.

با توجه به خوشه دوم می‌توان نتیجه گرفت که دستگاه ماهور، شور و راست‌پنج‌گاه از منظر دامنه‌ای شبیه به هم هستند برای تصدیق این قضیه می‌توان دید در مابقی خوشه‌ها هم این سه دستگاه به میزان تقریبا یکسانی قرار دارند.
باتوجه به خوشه ۲ و ۳ به نظر می‌رسد نوا ویژگی‌های دامنه‌ای متفاوتی نسبت به بقیه دستگاه‌ها دارد هرچند اشتراکاتی هم با همه دستگاه‌ها دارد اما در تعدادی از آهنگ‌ها متفاوت عمل کرده است.

۱۰.۱.۶ domain time features - Seven clusters - With PCA

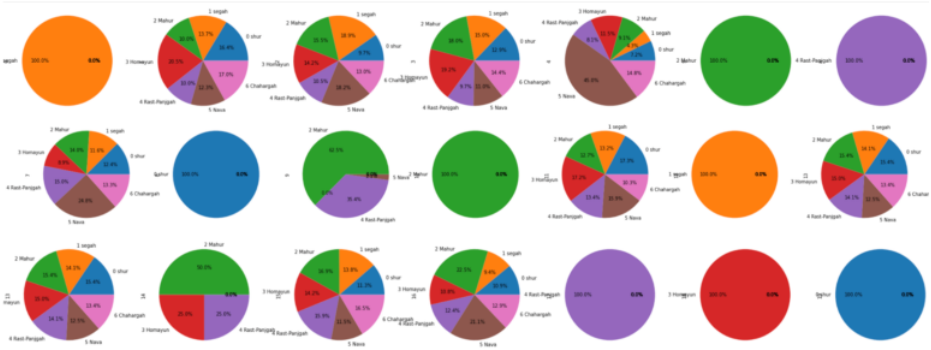
inertia: 140575
silhouette score: 47.9%

شکل ۱۴: Distribution of frequency features - Seven clusters - With PCA

با توجه به خوشه ۱ و ۵ به این نتجه می توان رسید همانند قسمت قبل یک شباهت قوی بین ویژگی های دامنه‌ای دستگاه راست‌پنج‌گاه و ماهور قرار دارند شور را هم می‌توان در این دسته‌بندی قرار داد.
در تایید بخش قبل دسته‌ای از آهنگ‌هایی که در نوا نواخته شده است همانطور که در خوشه ۶ می‌توان دید عملکرد متفاوتی نسبت به مابقی دستگاه‌ها پس از منظر دامنه‌ای متفاوت عمل می‌کنند.
درمورد مابقی خوشه‌ها و دستگاه‌ها نمی‌توان به نتیجه خاصی گرفت.

۱۱.۱.۶ domain time features - Twenty clusters - Without PCA

inertia: 1624832
silhouette score: 5.1%

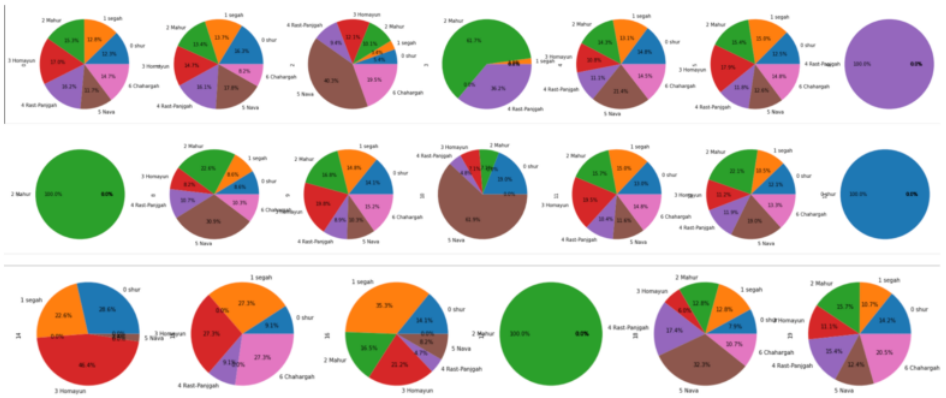


شکل ۱۵: Distribution of domain time features - Twenty clusters - Without PCA

دستگاه سه‌گاه در خوشه های ۰ و ۱۲ به طور کامل از سایر دستگاه‌ها جدا شده‌اند حضور این دستگاه درسایر خوشه‌ها یا بدین صورت است که اصلا حضور ندارند یا نزدیک به توزیع میانگین است در نتیجه می‌توان گفت سه‌گاه از منظر دامنه‌ای در طیف جداگانه‌ای نسبت به سایر دستگاه‌ها هستند.
همانند سه‌گاه، شور نیز در چند خوشه کاملا مجزا قرار دارد.
راست‌پنج‌گاه و ماهور هم تفکیک خوبی به نسبت سایر دستگاه‌ها دارند در این بین باتوجه به خوشه۹ و ۱۴ می‌توان شباهت‌هایی را بین این دو دستگاه بیان کرد.
همایون نیز در خوشه ۱۸ کاملا تفکیک شده و باتوجه به خوشه ۱۴ می‌توان گفت شباهت‌هایی به راست‌پنج‌گاه و ماهور دارد.
نوا تقریبا در همه خوشه‌ها حضور دارد و از منظر دامنه‌ای نمی‌توان تفکیک خاصی را برای این دستگاه قائل بود.

۱۲.۱.۶ domain time features - Twenty clusters - With PCA

inertia: 29969
silhouette score: 44.4%



شکل ۱۶: Distribution of domain time features - Twenty clusters - With PCA

زمانی که ویژگی‌ها را به دو بعد کاهش می‌دهیم، تا حد زیادی نتایج بسیار شبیه به قسمت قبل به دست می‌آید.
ماهور به طور مشخص از منظر دامنه‌ای تفاوت قابل توجهی با سایر دستگاه‌ها دارد. همین موضوع برای راست‌پنج‌گاه نیز صادق است همانطورکه در خوشه ۳ می‌بینید این دو دستگاه در دسته‌ای از آهنگ‌ها از منظر دامنه‌ای نزدیک به یک دیگر قرار گرفته‌اند.
شور نیز در خوشه ۱۳ به طور مستقلی جدا شده و به نظر می‌رسد تفکیک پذیری خوبی در این دسته از ویژگی‌ها در برخی از آهنگ‌ها دارد.
باتوجه به دسته ۱۴ همایون تا حدی به شور و با در نظر گرفتن خوشه ۱۴ نزدیک به دستگاه سه‌گاه است.

نتیجه‌گیری نهایی:

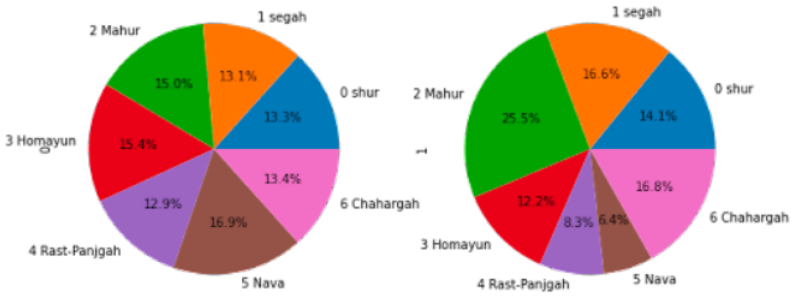
بر روی ویژگی‌های دامنه‌ای و با استفاده از الگوریتم K-Means با وجود این‌که در اکثر خوشه‌های تشکیل شده توزیع دستگاه‌ها نزدیک به میانگین یعنی حوالی ۱۴ درصد است و این قضیه نشان دهنده‌ی آن است که طیف گسترده‌ای از آهنگ‌ها جدای از نوع دستگاه از منظر ویژگی‌های دامنه‌ای تاحدی یکسان هستند.
اما با استنباط و تحلیل کلی از هر ۳ مدل خوشه‌بندی می‌توان گفت دستگاه‌های نوا و چهارگاه با تمام دستگاه‌ها از منظر دامنه‌ای اشتراکات زیادی با دیگر دستگاه‌ها دارند.
سه‌گاه تا حد خوبی از مابقی دستگاه‌ها جدا شده است و همین‌طور دستگاه شور در تعدادی از آهنگ‌ها کاملا متفاوت از مابقی دستگاه‌ها عمل می‌کند.
دستگاه ماهور و همایون هر کدام به طور جداگانه مشابهتی در این نوع ویژگی با راست‌پنج‌گاه دارند.

۲.۶ Agglomerative Clustering

از دیگر الگوریتم های خوشه بندی که به صورت سلسله مراتبی بوده و به آن روش پایین به بالا نیز گفته می‌شود، خوشه بندی agglomerative است. در این روش هر نمونه ابتدا خود یک خوشه است و در هر مرحله نمونه‌ها به هم دیگر می‌چسبند تا خوشه‌های بزرگ‌تر را تولید کنند و در نهایت همه‌ی نمونه‌ها با هم یک خوشه‌ی بزرگ را تشکیل می‌دهند. این الگوریتم نیز بر مبنای فاصله بوده و برای مسائلی که فاصله در آنها معنای ثابت خود را ندارد، به درستی عمل نخواهد کرد. در این الگوریتم شرط پایان می‌تواند این باشد که الگوریتم به یک تعداد مشخص خوشه برسد و یا اینکه فواصل بین خوشه ها به حد مشخصی برسند. از جمله مزایای این روش می توان به عدم نیاز به دانستن تعداد خوشه ها از ابتدا اشاره نمود. در ادامه این الگوریتم بر روی دادگان، با تعداد دسته های متغیر، اعمال شده و نتایج حاصل بررسی میشوند.

۱.۲.۶ frequency features - Two clusters - Without PCA

silhouette score: 27.5%

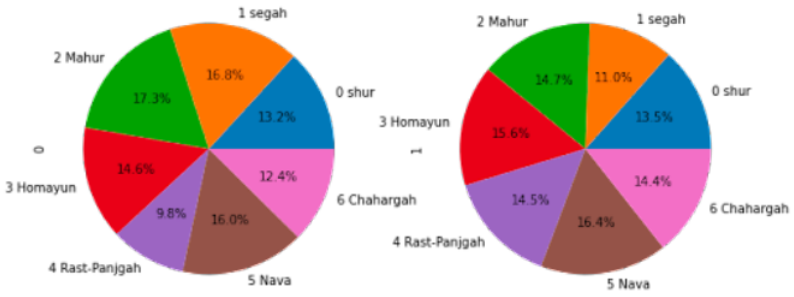


شکل ۱۷: Distribution of frequency features - Two clusters - Without PCA

برخلاف آن‌چه در الگوریتم Kmeans دیدیم در این‌جا می‌توان گفت ماهور در مقایسه با راست‌پنج‌گاه و نوا از منظر فرکانسی متفاوت هستند.
سه‌گاه و چهارگاه در مقایسه با همایون نیز در ویژگی‌های فرکانسی تفاوت معنا داری باهم دارند.

۲.۲.۶ frequency features - Two clusters - With PCA

silhouette score: 43.4%



شکل ۱۸: Distribution of frequency features - Two clusters - With PCA

اما زمانی که ویژگی‌ها را به دو بعد کاهش می‌دهیم توزیع هر دستگاه در دو خوشه نزدیک به میانگین است و در نگاه دقیق‌تر همان نتجه‌گیری‌های قسمت قبل را می‌توان انجام داد با این تفاوت که سه‌گاه و چهارگاه که در قسمت قبل در ویژگی‌های فرکانسی شبیه به هم عمل می‌کردند در این‌جا عکس هم عمل کرده‌اند. بدین معنا که چهارگاه و راست‌پنج‌گاه ویژگی‌های فرکانسی شبیه به هم و متفاوت از سه‌گاه دارند.

frequency features - Seven clusters - Without PCA ۳.۲.۶

silhouette score: 0.9%

شکل ۱۹: Distribution of frequency features - Seven clusters - Without PCA

با توجه به خوشه ۰ و ۲ هم‌چنین ۳ و ۴ می توان دید دستگاه نوا از منظر ویژگی‌های فرکانسی متفاوت از مابقی دستگاه‌ها عمل کرده است و همین قضیه برا سه‌گاه نیز صادق است به‌طوری که حتی می‌توان گفت سه‌گاه و نوا در اکثر دسته‌ها توزیعی شبیه به هم دارند.
همایون و سه گاه هم در فرکانس متفاوت از هم هستند با فاصله گرفتن یکی از میانگین دیگری در جهت عکس از آن فاصله می‌گیرد.

frequency features - Seven clusters - With PCA ۴.۲.۶

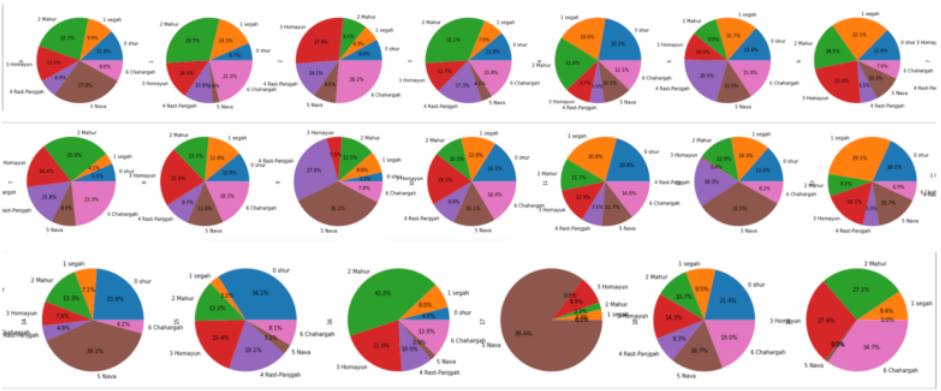
silhouette score: 24.5%

شکل ۲۰: Distribution of frequency features - Seven clusters - With PCA

ماهور و راست‌پنج‌گاه در دو طیف فرکانسی مختلف قرار دارند با فاصله گرفتن یکی از میانگین دیگری در جهت عکس حرکت می‌کند.
سه‌گاه و همایون یا نزدیک به توزیع میانگین هستند و درصورتی که کمی از میانگین فاصله بگیرند یه جز یک خوشه در جهت عکس هم عمل می‌کنند.
درمورد توزیع دیگر دستگاه‌ها نمی‌توان استدلالی کرد.

frequency features - Twenty clusters - Without PCA ۵.۲.۶

silhouette score: 2.6%

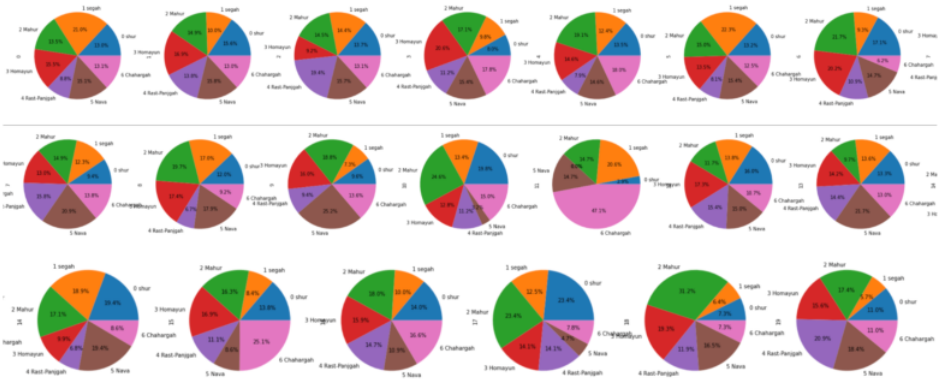


شکل ۲۱: Distribution of frequency features - Twenty clusters - Without PCA

همان‌طور که مشاهده می‌شود دستگاه ماهور و سه‌گاه در دو طیف فرکانسی متفاوت قرار دارند با فاصله گرفتن یکی از میانگین دیگری در جهت عکس آن عمل می‌کند.
نوا نیز در اکثر خوشه‌ها متفاوت از سایرین عمل کرده است و به نظر می رسد در طیف فرکانسی متفاوتی از سایر دستگاه‌ها قرار دارد.
درمورد مابقی دستگاه‌ها نمی‌توان نتیجه‌گیری کرد.

۶.۲.۶ frequency features - Twenty clusters - With PCA

inertia: 10139
silhouette score: 25.1%



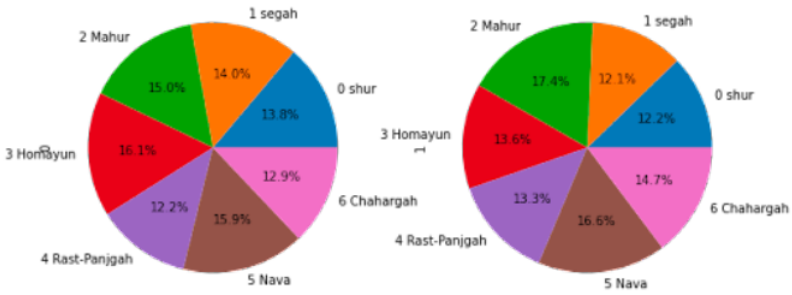
شکل ۲۲: Distribution of frequency features - Twenty clusters - With PCA

دستگاه نوا به طور خاص در خوشه‌های ۱۷، ۱۴،۹ و ۱۲ تا حد خوبی تفکیک شده و اشتراکاتی با تعداد کمی از آهنگ‌های مابقی دستگاه‌ها دارد.
چهارگاه، همایون ماهور برطبق آنچه در دسته‌های ۱۶،۱۹ و ۲ شباهت‌هایی در ویژگی‌های فرکانسی دارند.
درمورد مابقی دستگاه‌ها نتیجه‌گیری نمی‌توان کرد.

نتیجه‌گیری نهایی:
دستگاه‌های ماهور، شور و راست‌پنجگاه تقریبا می‌توان گفت از منظر دامنه‌ای تفکیک شده‌اند.
میان دستگاه‌های چهارگاه و شور، چهارگاه و همایون شباهت وجود دارد.
نوا با اکثر دستگاه‌ها اشتراکات زیادی دارند.

۷.۲.۶ domain time features - Two clusters - Without PCA

silhouette score: 24.9%

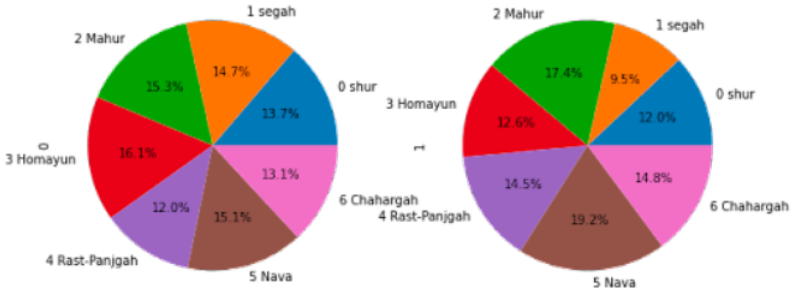


شکل ۲۳: Distribution of domain time features - Two clusters - Without PCA

این‌جا نیز توزیع‌ها نزدیک به میانگین است اما می‌توان گفت سه‌گاه، شور،همایون را شبیه در ویژگی‌های دامنه‌ای تصور نمود باتوجه به میزان کاهش این دستگاه‌ها با حرکت از خوشه نخست به خوشه دوم.

۸.۲.۶ domain time features - Two clusters - With PCA

silhouette score: 52.8%



شکل ۲۴: Distribution of domain time features - Two clusters - With PCA

سه‌گاه، همایون درمقابل نوا و ماهور متفاوت عمل کرده اند و در ویژگی‌های دامنه‌ای باهم تفاوت دارند.

۹.۲.۶ domain time features - Seven clusters - Without PCA

silhouette score: 11.8%

شکل ۲۵: Distribution of domain time features - Seven clusters - Without PCA

همان‌طور که مشاهده می‌شود سه‌گاه به خوبی از سایر دسته‌ها تفکیک شده است.
با توجه به خوشه ۶ و ۴ شباهت بسیار زیادی در دستگاه راست‌پنج‌گاه و ماهور دیده می‌شود شور نیز تا حدی به این دو نزدیک است. در دسته‌ای از آهنگ ها بخش عمده‌ای از آهنگ‌های نوا شباهت زیادی با دیگر دسته‌ها دارند.

۱۰.۲.۶ domain time features - Seven clusters - With PCA

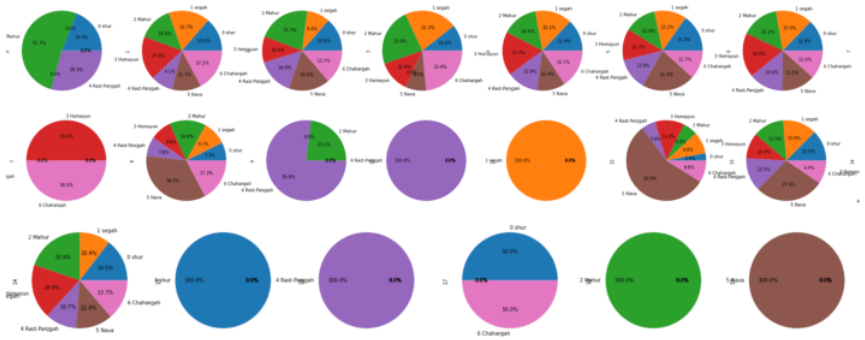
silhouette score: 43.1%

شکل ۲۶: Distribution of domain time features - Seven clusters - With PCA

مطابق با قسمت قبل راست‌پنج‌گاه و ماهور بسیار در ویژگی‌های دامنه‌ای شبیه به هم عمل می‌کنند و شور هم تا حدی به این دو نزدیک است.
مابقی دستگاه‌ها نیز مانند بخش قبل می‌توان تحلیل نمود.

۱۱.۲.۶ frequency features - Twenty clusters - Without PCA

silhouette score: 1.9%

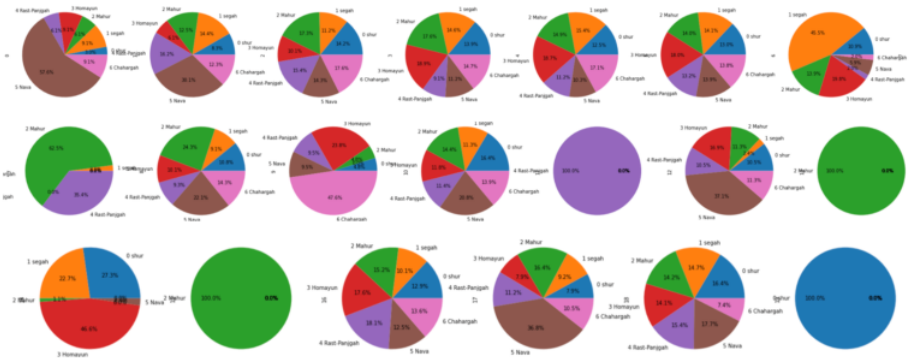


شکل ۲۷: Distribution of domain time features - Twenty clusters - Without PCA

هر یک از دستگاه‌های نوا، ماهور، شور، سه‌گاه و راست‌پنج‌گاه به خوبی توانسته‌اند از سایرین در دسته‌ای از آهنگ‌ها به صورت مستقل جدا شوند و از منظر ویژگی‌های دامنه‌ای در طیف‌های گسترده‌ای قرار گیرند.
میان دستگاه چهارگاه و شور، چهارگاه و همایون شباهت وجود دارد از طرفی راست‌وپنج‌گاه و ماهور و شور با توجه به خوشه نخست در ویژگی‌های دامنه‌ای نزدیک به هم هستند.

domain time features - Twenty clusters - With PCA ۱۲.۲.۶

inertia: 10139
silhouette score: 42.1%



شکل ۲۸: Distribution of domain time features - Twenty clusters - With PCA

هر یک از دستگاه‌های ماهور، شور و راست‌پنج‌گاه به خوبی توانسته‌اند از سایرین در دسته‌ای از آهنگ‌ها به صورت مستقل جدا شوند و از منظر ویژگی‌های دامنه‌ای در طیف‌های گسترده‌ای قرار گیرند.
نوا نیز هرچند با مابقی دستگاه‌ها در اکثر خوشه‌ها اشتراکاتی دارد اما توانسته در تعدادی از خوشه‌ها سهم بیشتری از آنها را در بر بگیرد.
خوشه ۱۴ بیانگر وجود شباهت در سه دستگاه همایون، شور سه‌گاه است.
ماهور و راست‌پنج‌گاه هم در دسته‌ای از آهنگ‌ها از منظر ویژگی‌های دامنه‌ای شبیه به هم‌اند.

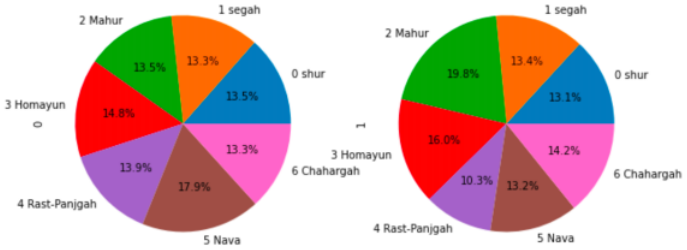
نتیجه‌گیری نهایی:
دستگاه نوا از منظر فرکانسی در اکثر تحلیل‌های بالا از دیگر دستگاه‌ها جدا شده و متفاوت عمل می‌کند.
سه‌گاه و همایون هم در دو طیف فرکانسی متفاوت قرار دارند.
ماهور و راست‌پنج‌گاه هم متفاوت از هم عمل کرده‌اند.

۳.۶ gaussian mixture model

از دیگر الگوریتم های خوشه بندی با نظارت می توان به الگوریتم GMM اشاره نمود. به طور کلی این الگوریتم فرض می کند که نمونه ها از k توزیع گوسی آمده که هر کدام از این توزیع ها بیانگر یک خوشه هستند. بنابراین GMM تلاش می کند تا نمونه های هر توزیع را بدست آورده و آنها را دسته بندی کند. هر یک از این توزیع ها دارای یک میانگین و یک واریانس مشخص اند، می توان با استفاده از maximum likelihood روی هر دسته، مقادیر میانگین و واریانس را برای هر دسته بدست آورد. با توجه به اینکه هدف بدست آوردن نمونه های هر توزیع است، نمی توان به صورت مستقیم از likelihood maximum استفاده نمود. بنابراین از روش expectation maximization استفاده میشود که با مقادیر اولیه برای هریک از پارامتر ها شروع نموده و با تکرار د. گام مقدار دهی و سپس ،maximization مقادیر مناسبی را برای پارامتر های مدل های گوسی و همچنین میزان تعلق هر نمونه به هر توزیع بدست می آورد. در ادامه این الگوریتم بر روی دادگان، با تعداد دسته های متغیر، اعمال شده و نتایج حاصل بررسی میشوند.

۱.۳.۶ frequency features - Two clusters - Without PCA

silhouette score: 11.8%

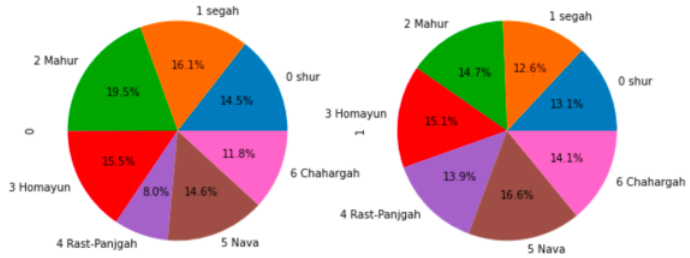


شکل ۲۹: Distribution of frequency features - Two clusters - Without PCA

تقریبا دستگاه ها در هر خوشه به نسبت مساوی تقسیم شده اند. فقط نوا درخوشه صفر و ماهور در خوشه یک اندکی فراوانی بیشتری نسبت به دستگاه های دیگر دارند. همچنین در خوشه یک فراوانی راست پنج گاه از بقیه دستگاه ها کمتر است. با مقایسه ی دو خوشه میبینیم که از خوشه صفر به خوشه یک، درصد فراوانی همایون و چهارگاه تقریبا به یک اندازه بیشتر شده و احتمالا به دلیل وجوه مشترک بین این دو دستگاه در ویژگی های فرکانسی باشد.

۲.۳.۶ frequency features - Two clusters - With PCA

silhouette score: 46.4%



شکل ۳۰: Distribution of frequency features - Two clusters - With PCA

در خوشه صفر بیشترین درصد فراوانی متعلق به ماهور با ۵.۱۹ می باشد. در این خوشه کمترین فراوانی متعلق به راست پنج گاه با ۸ می باشد. به نظر می رسد دستگاه ماهور با سه گاه و شور مشابه و دستگاه نوا با چهارگاه از لحاظ ویژگی های فرکانسی شبیه هم باشند.

frequency features - Seven clusters - Without PCA ۳.۳.۶

silhouette score: 2.8%

شکل ۳۱: Distribution of frequency features - Seven clusters - Without PCA

همان طور که مشاهده می شود همانند روش Kmeans در خوشه های مختلف رفتار دستگاه نوا و ماهور کاملا عکس هم عمل می کنند و میتوان نتیجه مشابه گرفت که این دو دستگاه احتمالا در دو طیف فرکانسی متفاوت قرار دارند. همین اتفاق برای دستگاه های راست پنج گاه با سه گاه و همایون با شور نیز افتاده است.
در بهترین حالت ماهور با ۶.۲۴ بیشترین درصد فراوانی را در خوشه ۶ دارد.

frequency features - Seven clusters - With PCA ۴.۳.۶

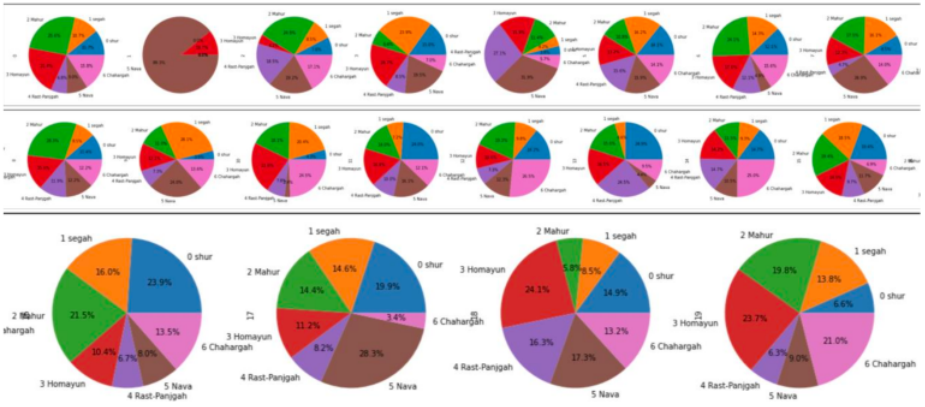
silhouette score: 32.2%

شکل ۳۲: Distribution of frequency features - Seven clusters - With PCA

در نمونه کاهش یافته این ویژگی ها، رفتار مشابهی نسبت به حالت بدون PCA را برای دستگاه های راست پنج گاه با سه گاه و همایون با شور مشاهده میکنیم. یعنی می توان به طور تقریبی گفت که این دستگاه ها نظیر به نظیر عکس هم عمل کرده و از نظر ویژگی های فرکانسی متفاوت هم می باشند. در مورد نوا و ماهور در خوشه های صفر و سه درصد فراوانی نزدیک به هم دارند پس نمی توان در مورد رفتار این دو دستگاه نظر داد.

frequency features - Twenty clusters - Without PCA ۵.۳.۶

silhouette score: 3.5%

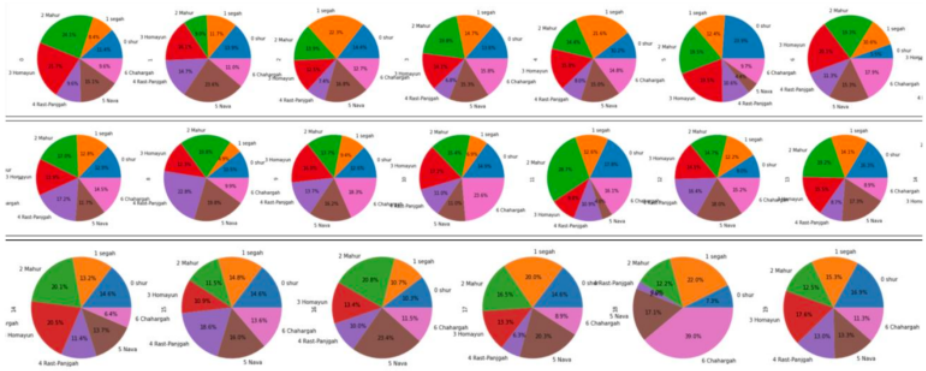


شکل ۳۳: Distribution of frequency features - Twenty clusters - Without PCA

دستگاه نوا به طرز نسبتا خوبی در خوشه ۱ نسبت به بقیه دستگاه ها تفکیک شده و به نظر میرسد ویژگی های فرکانسی مشترکی با دستگاه همایون دارد. به نظر می رسد همایون از لحاظ ویژگی های فرکانسی به بقیه دستگاه ها مثل شور و سه گاه تشابه بیشتری دارد.
در مورد بقیه دستگاه ها نمی توان نظری داد.

۶.۳.۶ frequency features - Twenty clusters - With PCA

inertia: 10139
silhouette score: 30.9%



شکل ۳۴: Distribution of frequency features - Twenty clusters - With PCA

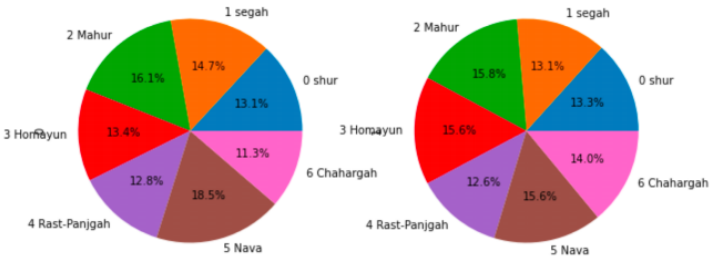
در نمونه کاهش یافته این ویژگی ها، در خوشه هجدهم چهارگاه درصد خوبی از فراوانی را به خود اختصاص داده است. در همین خوشه راست پنج گاه کمترین فراوانی را دارد و به نظر می رسد در بعضی از ویژگی های فرکانسی متفاوت از چهارگاه می باشد.
نوا در دو خوشه و سه گاه در سه خوشه کمترین درصد فراوانی را در میان سایرین دارند که میتوان گفت این دستگاه ها در برخی ویژگی های فرکانسی متفاوت از بقیه دستگاه ها هستند.

نتیجه‌گیری نهایی:

دستگاه های همایون و شور در برخی از روش های خوشه بندی مشابه هم عمل کردند و در برخی متفاوت. پس نمی توان یک نتیجه گیری کلی برای این دو دستگاه انجام داد. برای نوا و ماهور هم همین طور. به نظر می رسد دستگاه نوا در کل رفتار های متفاوتی نسبت به بقیه دستگاه ها در ویژگی های فرکانسی دارد.

۷.۳.۶ domain time features - Two clusters - Without PCA

silhouette score: 22.3%

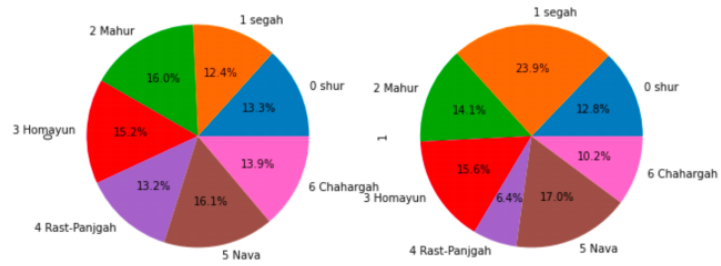


شکل ۳۵: Distribution of domain time features - Two clusters - Without PCA

در هر دو خوشه و به خصوص خوشه یک دستگاه ها به طور تقریبا یکنواختی در خوشه فراوانی دارند. فقط فراوانی دستگاه نوا اندکی بیشتر از بقیه دستگاه ها در خوشه صفر است.

۸.۳.۶ domain time features - Two clusters - With PCA

silhouette score: 36.1%

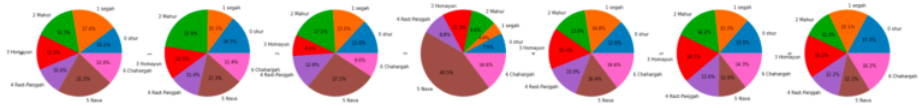


شکل ۳۶: Distribution of domain time features - Two clusters - With PCA

در نمونه کاهش یافته، در خوشه یک فراوانی سه گاه از بقیه دستگاه ها بیشتر است. فراوانی دستگاه های راست پنج گاه و چهارگاه نسبت به بقیه دستگاه ها کاهش بیشتری از خوشه صفر به خوشه یک داشته است. شاید بتوان گفت این دو دستگاه در ویژگی های دامنه ای احتمالا تفاوت هایی با دستگاه سه گاه دارند.

۹.۳.۶ domain time features - Seven clusters - Without PCA

silhouette score: -0.37%



شکل ۳۷: Distribution of domain time features - Seven clusters - Without PCA

در اولین نگاه، خوشه سه به چشم می آید که دستگاه نوا بیشترین فراوانی را با ۵.۴۳ به خود اختصاص داده است. در این خوشه سه گاه کمترین فراوانی را با ۶.۴ دارد. با مقایسه بقیه خوشه ها می توان فهمید که در ویژگی های دامنه ای احتمالا متفاوت با دستگاه نوا می باشد. همین اتفاق برای راست پنج گاه و همایون نیز افتاده است و اختلاف آن ها با نوا در اکثر خوشه ها میتواند نشان از تفاوت های دامنه ای این دو دستگاه با نوا باشد.

۱۰.۳.۶ domain time features - Seven clusters - With PCA

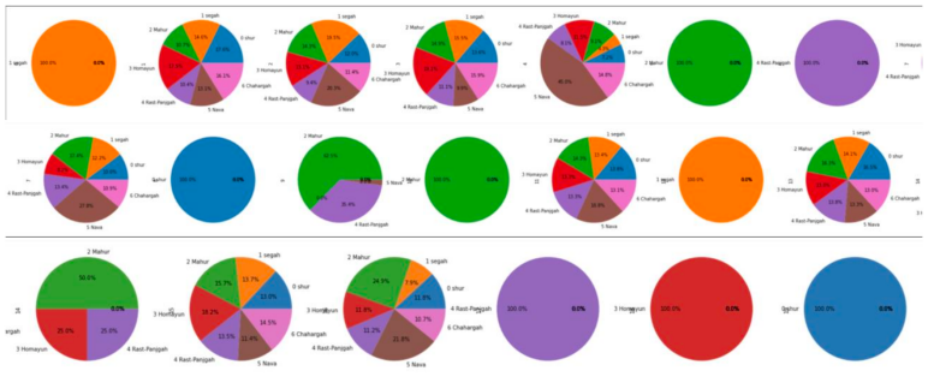
silhouette score: 7.2%

شکل ۳۸: Distribution of domain time features - Seven clusters - With PCA

دستگاه ماهور به خوبی توانسته است از منظر ویژگی های دامنه ای از بقیه دستگاه در خوشه شش جدا شود. دستگاه شور با ماهور در خوشه صفر، چهارگاه با سه گاه در خوشه سه و راست پنج گاه با نوا در خوشه چهار، بیشترین اختلاف ها را در خوشه های مرتبط دارند و این نشان می دهد که این جفت دستگاه در برخی از ویژگی های دامنه ای با هم تفاوت دارند.

۱۱.۳.۶ frequency features - Twenty clusters - Without PCA

silhouette score: 2.8%



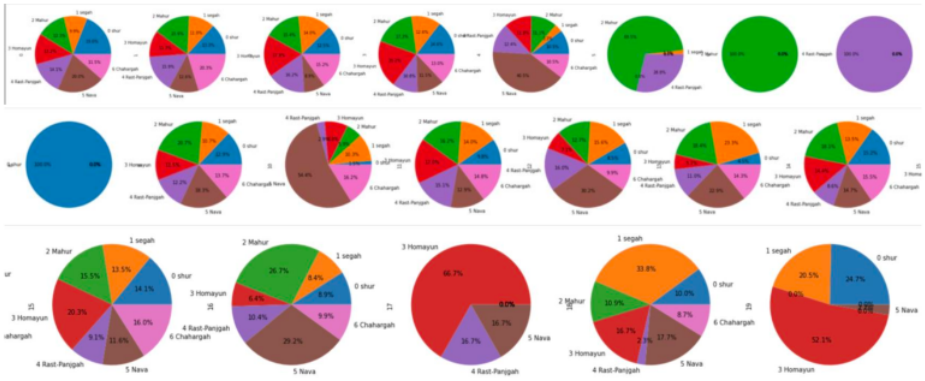
شکل ۳۹: Distribution of domain time features - Twenty clusters - Without PCA

هر یک از دستگاه های سه گاه، ماهور، راست پنج گاه، شور و همایون به خوبی توانسته اند از سایرین در دسته ای از آهنگ ها به صورت مستقل جدا شوند و از منظر ویژگی های دامنه ای در طیف های گسترده ای قرار گیرند.
دستگاه نوا نیز در خوشه دو، چهار و هفت بیشترین فراوانی را به خود اختصاص داده است.
خوشه نه بیان گر تشابه دامنه ای راست پنج گاه و ماهور می باشد.

همچنین خوشه چهارده بیان گر تشابه دامنه ای بین سه دستگاه ماهور، همایون و راست پنج گاه می باشد.

domain time features - Twenty clusters - With PCA ۱۲.۳.۶

inertia: 10139
silhouette score: 22.7%



شکل ۴۰: Distribution of domain time features - Twenty clusters - With PCA

در نمونه کاهش یافته، فقط سه دستگاه ماهور، راست پنج گاه و شور به خوبی و به صورت مستقل از بقیه دستگاه ها جدا شده اند.
در خوشه هفده و نوزده دستگاه همایون و در خوشه ده نیز نوا بیشتر از نصف فراوانی را به خود اختصاص داده اند.
خوشه پنج بیانگر تشابه دامنه ای دستگاه های ماهور و راست پنج گاه می باشد. خوشه هفده بیانگر تشابه دامنه ای همایون، راست پنج گاه و نوا و خوشه نوزده بیانگر تشابه دامنه ای شور، سه گاه و همایون می باشد.

نتیجه‌گیری نهایی:

در خوشه بندی با تعداد هفت به نظر می رسد نوا و راست پنج گاه تفاوت های بیشتری در زمینه ی دامنه ای نسبت به بقیه دارند. در خوشه بندی با تعداد ۲۰ با هر دو روش به تشابه دامنه ای دستگاه های راست پنج گاه و ماهور می رسیم. احتمالا همایون و راست پنج گاه هم در برخی ویژگی های دامنه ای مشابه هم هستند.

7 Extra Part

Extra part Cutting music to shorter length and extracting features from them, then splitting it to train and test set, will result different from splitting train and test, then cutting music in each set to shorter length and extracting features. Up until now, the report was based on the first method and in this extra part, second method is tested. To demonstrate accuracy of models more sensibly and real-world conditions, the music must first be splat to train and test and then be broken down to 20 seconds to extract features for. This way it can be made sure that the similarity between train and test sets are as low as possible and returned models are more usable. In order to perform the mentioned step, a function is written to split music to train and test by using a Bernoulli distribution with p=0.75, which is the rate of train set size. After it is decided which set a music belongs to, it can be broken down to shorter and fixed length audio and features can be extracted. Then the features for a whole piece of music are added to either the training or the test set. If the features of 20 seconds audio files are extracted and then splat to train and test sets, there might be some parts of the same music in both sets that will affect the value of test accuracy specially in classifiers such as KNN, which classify based on similarity.
The function written to split the music in to 20 seconds pieces, ignores the part left at the end of music (part shorter than 20 seconds). These 20 seconds pieces are saved one by one (with replacement to avoid extra memory consumption) in .wav format and then read with librosa and passed to the feature extraction function. All time and frequency features are extracted and returned in separate vectors to be stored in the dataset. It is also worthy of saying that features were extracted separately for each dastgāh and then the datasets were concatenated to one csv file for frequency features and one csv file for time domain features. These datasets will be used for model fitting and prediction on the upcoming parts.
Using the new generated dataset, analysis of the learning curve and ROC curve will be similar to the last part and only model evaluation metrics are compared to each other and to the first dataset generation method. Train and test accuracies and also training times can be seen in the tables below.

Frequency features		Train accuracy	Test accuracy	Train time (ms)
Normalized data	Logistic regression	51	25	1139.36
	KNN	99	42	49.01
	XGBoost	82	32	70594.26
	Ensembled	88	34	31728.27
XGBoost	Logistic regression	19	15	1055.77
	KNN	99	18	55.73
	XGBoost	82	32	48346.58
	Ensembled	91	24	29861.27

Frequency features		Train accuracy	Test accuracy	Train time (ms)
Normalized data	Logistic regression	34	18	1715.00
	KNN	99	23	259.66
	XGBoost	61	23	142202.07
	Ensembled	74	21	126421.72
XGBoost	Logistic regression	11	14	420.83
	KNN	99	21	239.62
	XGBoost	61	23	146006.47
	Ensembled	67	17	127341.61

As can be seen from the results, normalized frequency features output a more accurate classifier and just like the previous section, KNN is ahead of all classifiers with a noticeable difference. This is due to the fact that music from the same dastgāh tend to have similar features sets and distance is actually meaningful in this concept. Logistic regression has the lowest accuracy and this is reasonable as it is a very basic linear classifier. Ensembled model also has a better output and this is probably due to the fact that KNN is used in the model. Another interesting result is that all classifiers with all types of data, except for logistic regression, have been overfitted and this is because there is not enough data and these classifiers are complicated. With a complicated classifier, there must be good amount of training data available. Otherwise, overfitting is expected to happen. This is what has happened in this case.
If the training times are to be compared, it can be seen that KNN also has the lowest training time but as know, KNN does not perform much of a training task and prediction takes longer as a whole lot of comparisons is needed. KNN also has the downside of memory. Ensembled model also takes the longest as it requires the training of all other three classifiers along with the time for voting. Also, as it uses KNN, it must keep all data and inefficient in memory. Ensembled model might have a better performance compared to other three, but it takes both memory and time. XGBoost also takes long to train as it uses a decision-tree-like approach. As the dataset can not be explained by each independent feature, it takes longer for XGBoost.
As KNN has the best performance on normalized frequency features, dimensionality reduction methods are applied on this classifier to see if the results improve.

	Train accuracy %	Test accuracy %
PCA	100	19
LDA	100	40
Forward selection	100	56
Backward elimination	100	49

The overfitting problem is still present and the difference between train and test accuracy scores is very high. There is no good way to solve this problem except increase

training data and the variety of it. Applying PCA and LDA decreases model accuracy but backward elimination and especially forward selection, increase test accuracy. This is due to the fact that KNN works better with less features and in lower dimensions. Therefore, it can be concluded that with the generated feature set, it is best to use normalized frequency features along with KNN and forward selection dimensionality reduction algorithm. This will result a test accuracy of 56%. The models are yet to be improved and this can be done better by increasing the number of data points in the dataset.

One more comparison that is worthy of mentioning, is that it was seen earlier that KNN performed with 94% accuracy on the data that was first broken into short parts and then splat to train and test. The reason to this is mainly because parts from the same music that are very alike appear in both training and test sets and therefore making it easier for KNN to predict the class of a music in test set. This improvement is also seen in other classifiers which might be due to the same reason but it should also be considered that splitting music this way will increase the variety of data I training set and therefore result better accuracy. Another benefit of the first splitting method is that with a batter variability, less overfitting occurs that can be seen slightly in the results.