به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

یادگیری ماشین

گزارش پروژه نهایی

علی اخگری ۸۱۰۱۹۸۳۴۱

امید پناکاری ۸۱۰۱۹۸۵۲۸

پریا خوش‌تاب ۸۱۰۱۹۸۳۸۷

صبا شهسواری ۸۱۰۱۹۸۴۲۰

بهمن‌ماه ۱۴۰۱

فهرست

۱ تمیز کردن داده

۱-۱ حذف داده‌های با فرمت نامناسب

۱-۲ تبدیل فرمت داده‌ها به wav

۱-۳ شکستن فایل‌های صوتی به بخش‌هایی با طول یکسان

۱-۴ مصورسازی شکل موج

۲ استخراج ویژگی

۲ـ۱ ویژگی‌های مختلف یک سیگنال صوتی

۲ـ۲ ویژگی‌های استخراج شده

۲-۲-۱ ویژگی‌های طیفی

۲-۲-۱-۱ Mel Frequency Cepstral Coefficients(MFCC)

۲-۲-۱-۲ Spectral Roll off

۲-۲-۱-۳ Spectral Centroid

۲-۲-۱-۴ Spectral Flux

۲-۲-۱-۵ Spectral Bandwidth

۲-۲-۱-۶ Spectral Contrast

۲-۲-۲ ویژگی‌های هارمونیک

۲-۲-۲-۱ Chromagram

۲-۲-۲-۲ Tonnetz

۲-۲-۳ ویژگی‌های زمانی

۲-۲-۳-۱ Zero Crossing Rate

۲-۲-۳-۲ Root Mean Square Energy

۳ طبقه‌بندی

۳-۱ تقسیم داده به دو بخش آموزش و آزمون

۳-۲ استانداردسازی داده

۳-۳طبقه‌بندی براساس الگوریتم‌های یادگیری ماشین
۳-۳-۱ K-Nearest Neighbors
۳-۳-۲ Multilayer Perceptron(MLP)
۳-۳-۳ Logistic Regression
۳-۳-۴ Long Short-Term Memory
۳-۴کاهش ابعاد بردار ویژگی
۴ خوشه‌بندی
۴.۱.۱ الگوریتم k-means
۴.۲ خوشه بندی سلسه‌مراتبی (Hierarchical Clustering)
۴.۲.۱ الگوریتم Agglomerative
۵ جمع‌بندی

۱ تمیز کردن داده

تمیز کردن داده‌های صوتی قبل از طبقه‌بندی، یک مرحله مهم در فرآیند یادگیری ماشین است، زیرا می‌تواند بر عملکرد طبقه‌بند تاثیر زیادی بگذارد.

۱-۱ حذف داده‌های با فرمت نامناسب

در دیتاست اصلی داده شده، برخی از فایل‌های صوتی دارای فرمت نامناسب می‌باشند که این امر باعث می‌شود این فایل‌های صوتی در هنگام لود شدن توسط توابع موجود در زبان پایتون، با مشکل مواجه شوند؛ بنابراین در این مرحله ما این داده‌ها را حذف می‌کنیم.

برای حذف داده‌هایی که فرمت مناسبی ندارند، از کتابخانه fleep استفاده کرده‌ایم. این کتابخانه با خواندن بایت‌های اولیه‌ی هر فایل صوتی فرمت اصلی آن‌ها را تشخیص می‌دهد و در صورتی که فایل صوتی فرمت نامناسبی داشته باشد این کتابخانه نمی‌تواند آن‌ها را بخواند. همچنین استفاده از این کتابخانه به ما کمک می‌کند که فرمت واقعی فایل صوتی را پیدا کنیم زیرا ممکن است یک فایل صوتی با آن که اکستنشن mp3 دارد، دارای فرمت دیگری مانند mp4 باشد.

۱-۲ تبدیل فرمت داده‌ها به wav

می‌دانیم فرمت بیشتر داده‌های درون دیتاست پروژه mp3 می‌باشد. فایل‌های mp3 کوچک، فشرده‌شده و lossy هستند. بنابراین ابتدا تمام داده‌های پروژه را به فرمت wav تبدیل می‌کنیم، زیرا داده‌های با این فرمت، lossless هستند، فشرده نشده‌اند و کیفیت صدای بالاتری دارند. از طرف دیگر ما برای لود کردن فایل‌های صوتی از متد load از کتابخانه librosa استفاده می‌کنیم، اما لود کردن داده‌های با فرمت mp3 برخلاف wav، با استفاده از این متد زمان زیادی طول می‌کشد.

برای تبدیل فرمت فایل‌های صوتی به wav از کتابخانه pydub استفاده کرده‌ایم؛ این کتابخانه یک رابط کاربری ساده و آسان برای کار با فایل های صوتی ارائه می دهد. یکی از کلاس های اصلی آن AudioSegment است که به شما امکان می دهد فایل های صوتی را به روش های مختلف دستکاری کنید. از ویژگی های کلیدی AudioSegment، می‌توان به export کردن فایل‌های صوتی به فرمت های مختلف و دستکاری کردن آن‌ها از طریق عملیاتی مانند الحاق کردن فایل‌های صوتی یا برش دادن آن‌ها اشاره کرد.

۱-۳ شکستن فایل‌های صوتی به بخش‌هایی با طول یکسان

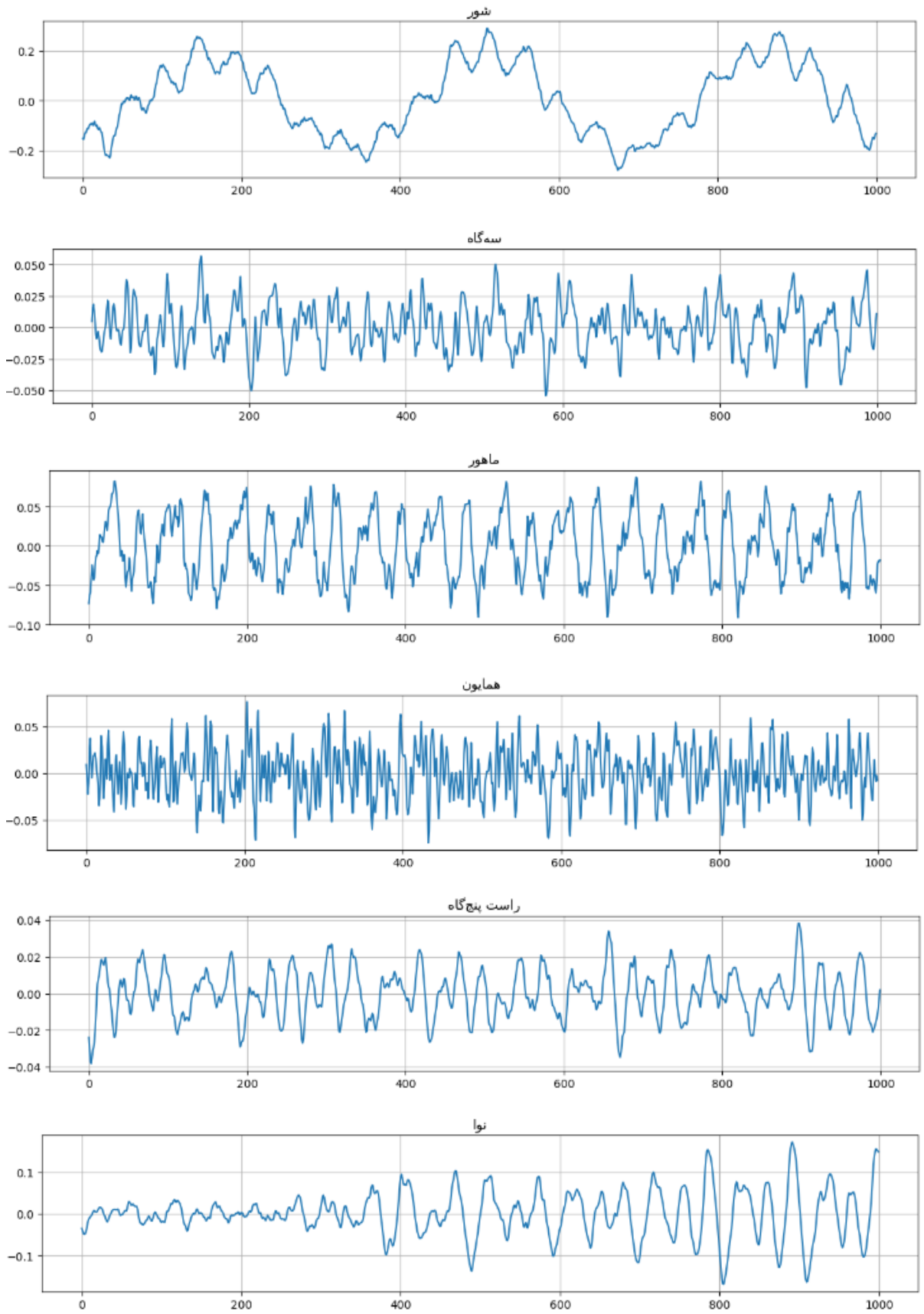
تقسیم فایل های صوتی به بخش های کوچکتر قبل از استخراج ویژگی ها به چند دلیل ضروری است:

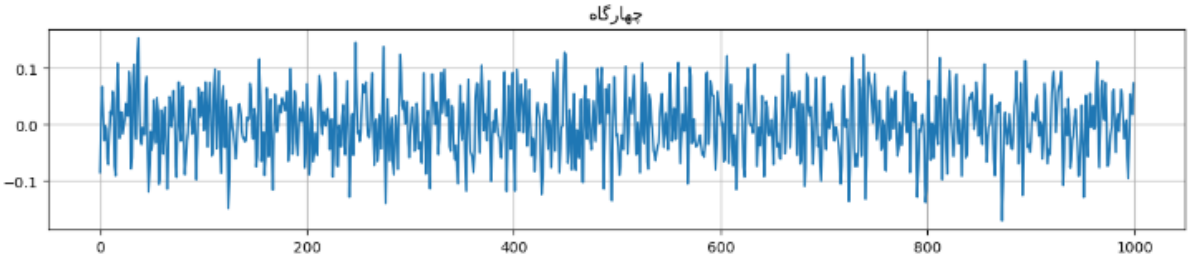
- **بهبود فرایند استخراج ویژگی:** با تقسیم فایل‌های صوتی به بخش‌های کوچک‌تر، می‌توان ویژگی‌هایی را استخراج کرد که نشان‌دهنده سیگنال اصلی هستند. به عنوان مثال، اگر یک فایل صوتی طولانی را به بخش های چند ثانیه ای تقسیم کنید، می توانید ویژگی هایی را استخراج کنید که ویژگی های محلی سیگنال، مانند محتوای فرکانس، را بهتر منعکس کنند.
- **کاهش نویز :** تقسیم فایل‌های صوتی به بخش‌های کوچک‌تر می‌تواند به کاهش تاثیر نویز و سایر مصنوعات موجود در سیگنال کمک کند. به عنوان مثال، اگر تغییر ناگهانی در سطح نویز پس‌زمینه ایجاد شود، این تنها بخش کوچکی از سیگنال را تحت تاثیر قرار می‌دهد و تاثیر آن بر استخراج کلی ویژگی کاهش می‌یابد.
- **پردازش بهینه‌تر:** تقسیم فایل‌های صوتی به بخش‌های کوچک‌تر نیز می‌تواند کارایی فرآیند استخراج ویژگی را بهبود بخشد. با پردازش قطعات کوچکتر، امکان استخراج ویژگی به صورت موازی وجود دارد که می تواند این روند را تسریع کند و نیاز به حافظه را کاهش دهد.
- **کاهش ابعاد بردار ویژگی:** فایل‌های صوتی می‌توانند خیلی بزرگ باشند. بنابراین با تقسیم این فایل‌ها به بخش‌های کوچکتر، می‌توانیم ابعاد بردار ویژگی را کاهش دهیم.
- **افزایش تعداد نمونه‌ها:** شکستن فایل‌های صوتی به بخش‌های با اندازه مساوی و کوچکتر سبب می‌شود که نمونه‌های بیشتری جهت استفاده در بخش یادگیری داشته باشیم.
- **شرایط یکسان نمونه‌ها:** فایل‌های صوتی در حالت عادی مدت زمان‌های متفاوتی دارند، اما تقسیم آن‌ها به بخش‌های کوچکتر با طول یکسان سبب می‌شود که نمونه‌ها دارای شرایط یکسان باشند که برای یادگیری مناسب‌تر است.
- **بهبود عملکرد طبقه بندی:** طبق مواردی که در بالا اشاره شد، قاعدتا تقسیم فایل‌های صوتی به بخش‌های کوچکتر باعث می‌شود عملکرد طبقه بندی بهبود یابد.

برای شکستن فایل‌های صوتی به بخش‌هایی با طول یکسان، از خروجی قسمت قبل که با استفاده از کلاس AudioSegment به دست آوردیم استفاده می‌کنیم. به این صورت که ابتدا طول کلی یک فایل صوتی را در نظر می‌گیریم و سپس با تقسیم این طول به یک عدد دلخواه(در اینجا ما ۳۰ ثانیه را برای این عدد در نظر گرفته‌ایم)، تعداد بخش‌هایی که این فایل صوتی دارد را محاسبه می‌کنیم. همچنین بخش‌های اضافی که پس از تقسیم طول صدا به قسمت های مساوی باقی می مانند را به طور مساوی از ابتدا و انتهای فایل دور می‌ریزیم، زیرا به طور معمول در ابتدا و انتهای فایل‌ها مقداری سکوت وجود دارد.
در نهایت این قطعات صوتی را به ازای دستگاه‌های مختلفی که به آن تعلق دارند، در یک پوشه جدید ذخیره می‌کنیم.

۴-۱ مصورسازی شکل موج

در این بخش شکل موج مربوط به یک نمونه‌ی تمیزشده از هر دستگاه را رسم کرده‌ایم.


شور
سه‌گاه
ماهور
همایون
راست پنج‌گاه
نوا



۲ استخراج ویژگی

استخراج ویژگی در طبقه بندی فایل‌های صوتی اهمیت زیادی دارد، زیرا به ضبط اطلاعات مربوطه در سیگنال صوتی که برای طبقه بندی صدا به دسته های مختلف ضروری است، کمک می کند. سیگنال صوتی خام، اغلب پیچیده است و حاوی مقدار زیادی اطلاعات است که بسیاری از آنها ممکن است برای کار طبقه بندی بی ربط یا اضافی باشند. استخراج ویژگی به شما این امکان را می‌دهد تا ویژگی های معنادار و معرف را از سیگنال صوتی استخراج کنید و به عنوان ورودی به طبقه‌بند دهید.

۱.۲ ویژگی‌های مختلف یک سیگنال صوتی

ویژگی‌های مختلف زیادی از فایل‌های صوتی می‌توان استخراج کرد تا در فرآیند طبقه‌بندی از آن‌ها استفاده کرد. از رایج‌ترین ویژگی‌ها، می‌توان ویژگی‌های زیر را برشمرد:

- **ویژگی‌های طیفی(Spectral Features):** این ویژگی‌ها، محتوای طیفی سیگنال صوتی را توصیف می‌کنند، مانند چگالی طیف توان (PSD)، ضرایب کپسترال فرکانس مل (MFCCs)، یا ویژگی‌های رنگی(chroma).
- **ویژگی‌های زمانی(Temporal Features):** این ویژگی‌ها، ساختار زمانی سیگنال صوتی را توصیف می‌کنند، مانند zero-crossing rate ، انرژی یا همبستگی خودکار.
- **ویژگی‌های آماری(Statistical Features):** این ویژگی‌ها، ویژگی‌های آماری سیگنال صوتی مانند میانگین، انحراف معیار، چولگی یا کشیدگی را توصیف می‌کنند.
- **ویژگی‌های ریتمیک(Rhythmic Features):** این ویژگی‌ها، محتوای ریتمیک یک سیگنال صوتی را توصیف می‌کنند، مانند هیستوگرام ضربان، عملکرد تشخیص شروع، یا سرعت.
- **ویژگی های هارمونیک(Harmonic Features):** این ویژگی‌ها، محتوای هارمونیک سیگنال صوتی را توصیف می کنند، مانند گام، نسبت هارمونیک یا تونالیته.

- **ویژگی‌های رویداد صوتی(Audio Event Features):** این ویژگی‌ها، رویدادها یا الگوهای صوتی خاصی را در یک سیگنال صوتی توصیف می‌کنند، مانند حضور یک رویداد صوتی خاص، موقعیت یک رویداد در زمان، یا مدت یک رویداد.

انتخاب ویژگی‌ها، به وظیفه خاص و نوع سیگنال صوتی که طبقه‌بندی می‌شود، بستگی دارد. برخی از تکنیک‌های استخراج ویژگی ممکن است برای سیگنال‌های گفتاری مناسب‌تر باشند، در حالی که برخی دیگر ممکن است برای سیگنال‌های موسیقی مناسب‌تر باشند.

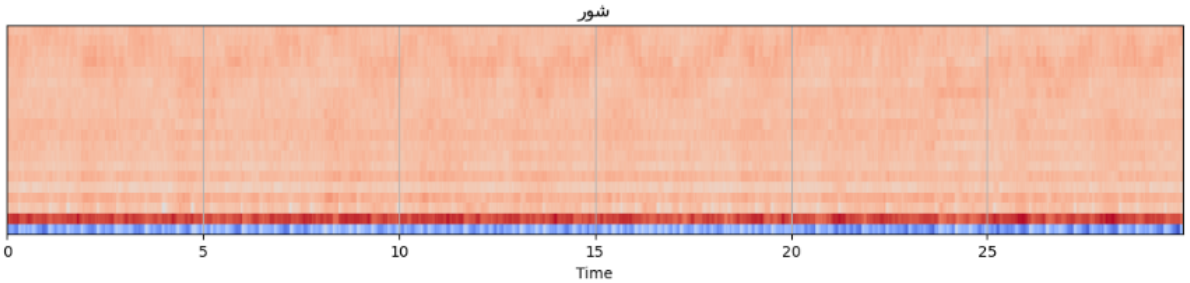
۲.۲ ویژگی‌های استخراج شده

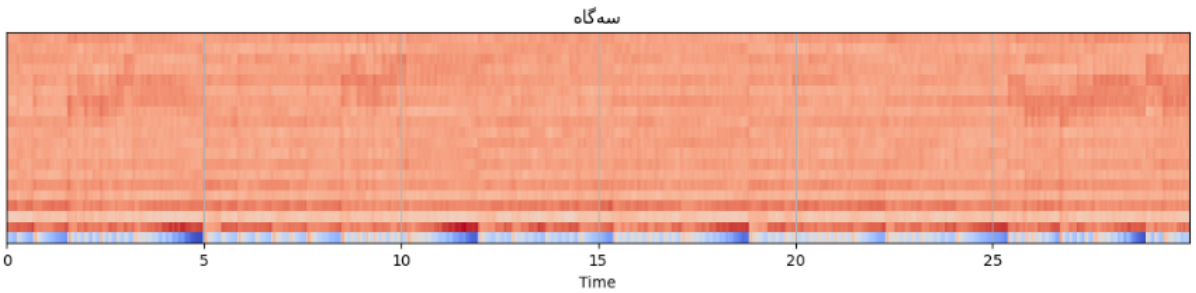
در این پروژه ما ویژگی‌ها را از سه دسته ویژگی‌های زمانی، ویژگی‌های طیفی و ویژگی‌های هارمونیک انتخاب کرده‌ایم. دو روش برای استخراج ویژگی‌های گفته شده استفاده شده است؛ در روش اول (Feature Set 1) از هر سه نوع از ویژگی‌ها استفاده شده است و در روش دوم (Feature Set 2) تنها ویژگی‌های طیفی و زمانی استفاده شده‌اند. در ادامه به بررسی ویژگی‌هایی که از هر دسته استخراج شده‌اند می‌پردازیم.

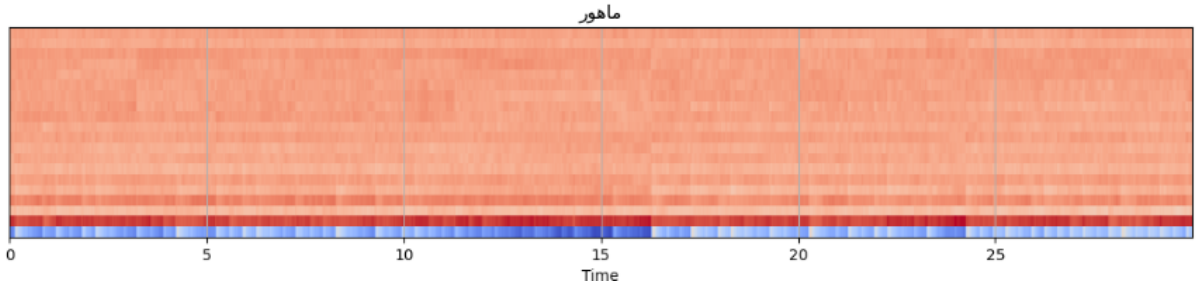
۲-۲-۱ ویژگی‌های طیفی

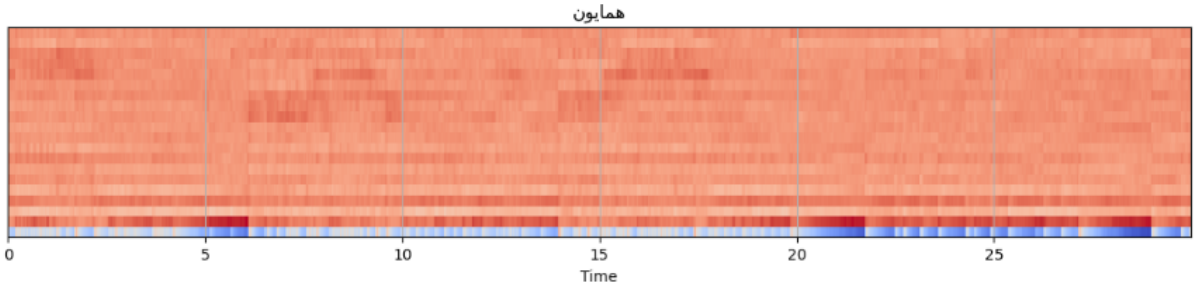
۲-۲-۱-۱ Mel Frequency Cepstral Coefficients(MFCC)

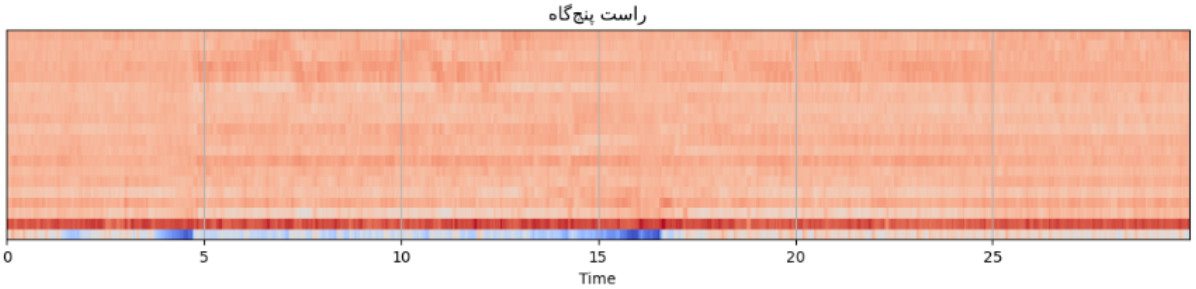
MFCCs از مهمترین ویژگی‌های قابل استخراج از سیگنال‌های صوتی هستند که در مسائل طبقه‌بندی سیگنال‌های صوتی و speech recognition بسیار خوب عمل می‌کنند. این ویژگی‌ها شامل تعدادی ضریب هستند که سیگنال را به خوبی توصیف می‌کنند.

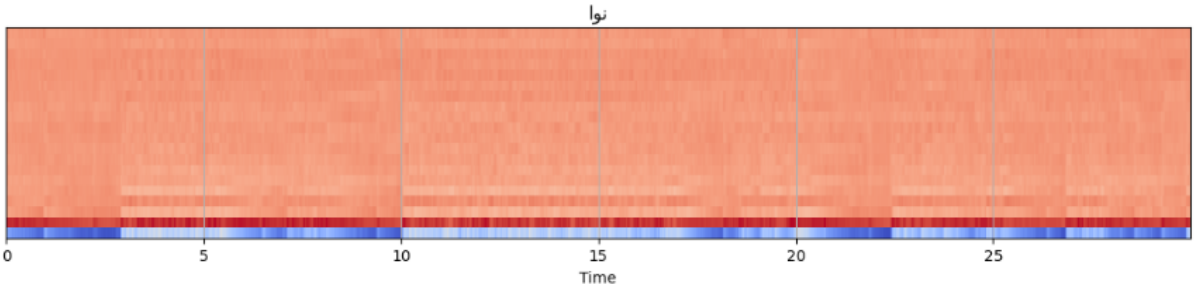


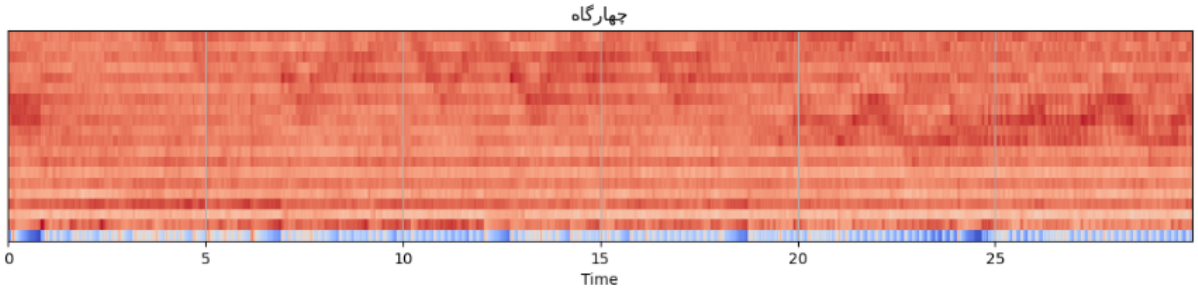

سه‌گاه
0
5
10
15
20
25
Time


ماهور
0
5
10
15
20
25
Time


همایون
0
5
10
15
20
25
Time

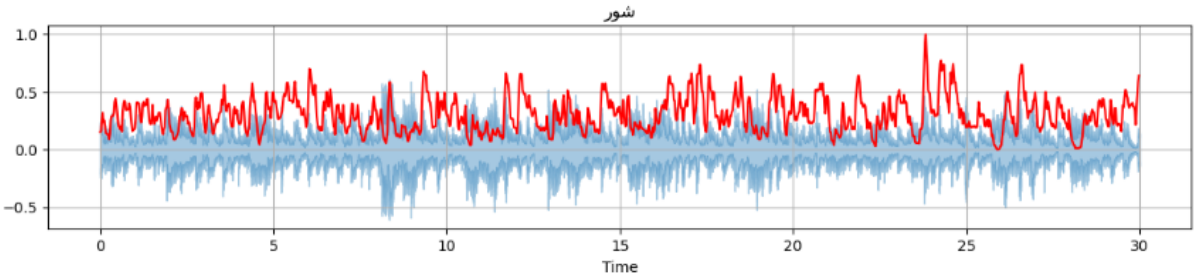

راست پنج‌گاه
0
5
10
15
20
25
Time

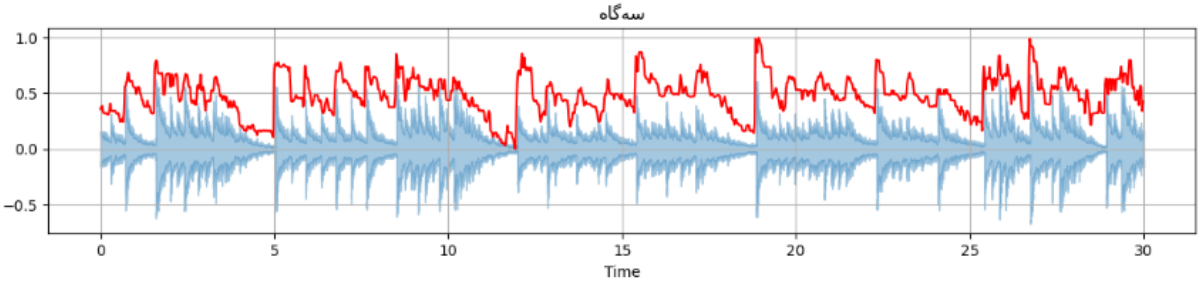

نوا
0
5
10
15
20
25
Time

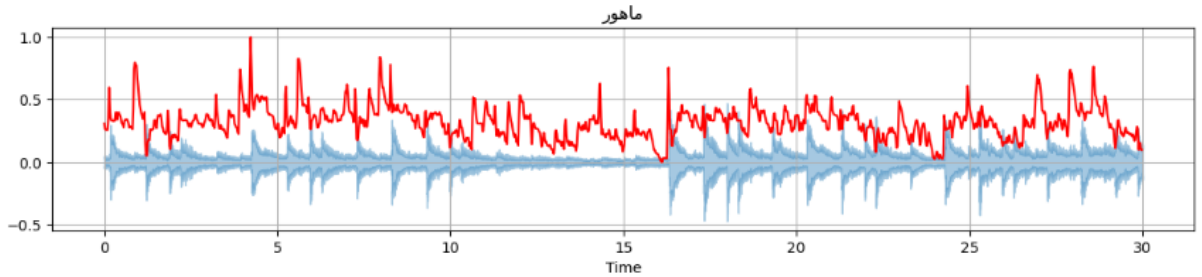


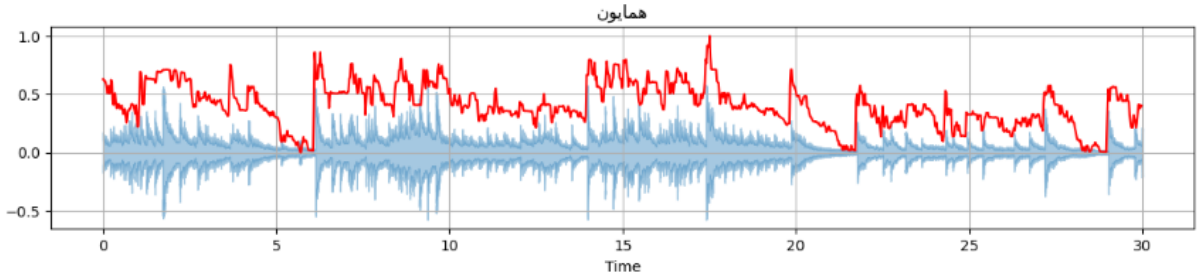
Spectral Roll off ۲-۱-۲-۲

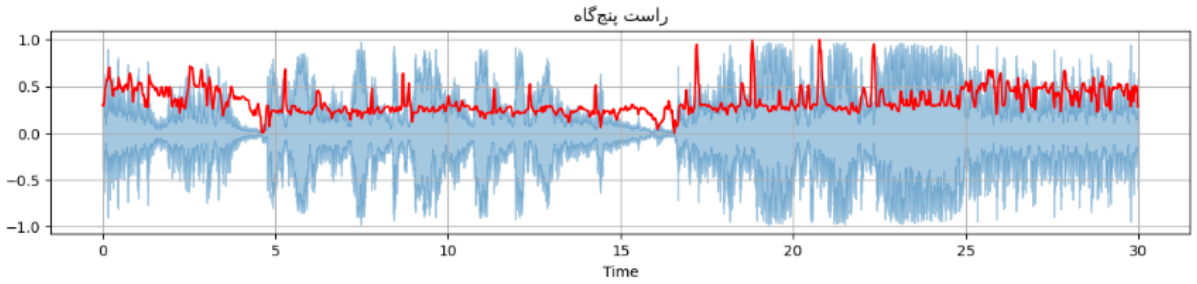
این ویژگی فرکانسی را نشان می‌دهد که درصد مشخصی از انرژی سیگنال مانند ۸۵٪ در زیر آن قرار دارد.

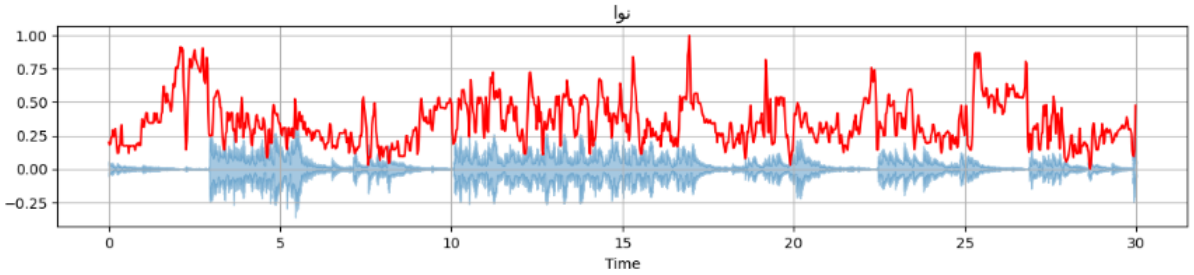


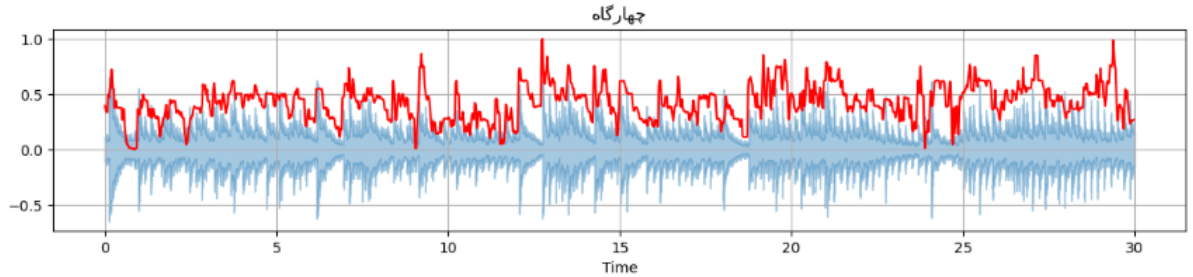












Spectral Centroid ۳-۱-۲-۲

یکی دیگر از ویژگی‌های طیفی سیگنال‌های صوتی است در طبقه‌بندی موسیقی‌ها کاربرد دارد. این ویژگی نشان دهنده مرکز جرم طیف سیگنال می‌باشد و از خیلی لحاظ با ویژگی قبلی شباهت دارد.

Spectral Flux ۴-۱-۲-۲

این ویژگی از حاصل جمع مربع اختلاف فریم به فریم اندازه طیف سیگنال بر روی فرکانس به دست می‌آید. اندازه‌ی بالای این ویژگی نشان‌دهنده‌ی تغییرات ناگهانی بر روی طیف سیگنال می‌باشد.

$$F_r = \sum_{k=1}^{\frac{N}{2}} \left(|x_r[k]| - |x_{r-1}[k]|\right)^2$$

۲-۲-۱-۵ Spectral Bandwidth

نشانگر بازه‌ای حول مرکز جرم می‌باشد که مقدار زیادی از انرژی طیف در آن قرار دارد. Spectral bandwidth مرتبه pام به صورت زیر محاسبه می‌شود:

$$\left(\sum_k S(k)\left(f(k) - f_c\right)^p\right)^{\frac{1}{p}}$$

که در آن $S(k)$ اندازه طیف و $f(k)$ مقدار فرکانس و f_c مقدار فرکانس مرکز جرم در برش kام سیگنال هستند. به ازای مقدار p=2 این ویژگی نشان‌دهنده واریانس وزن دار حول مرکز جرم طیف می‌باشد.

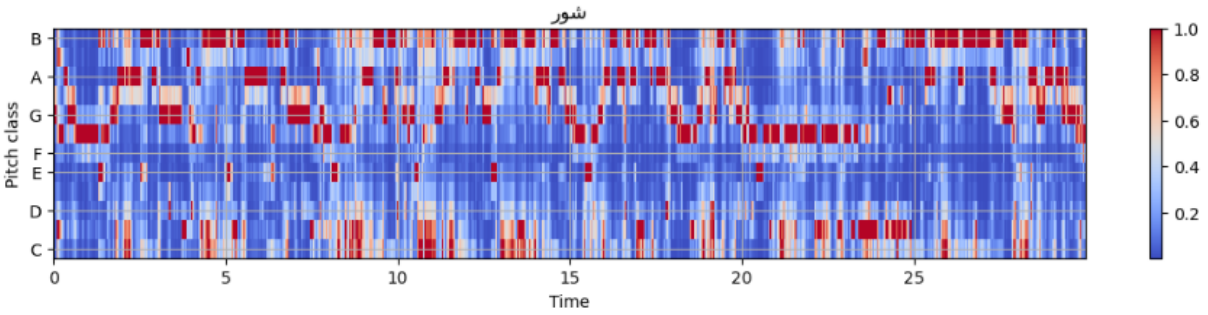
۲-۲-۱-۶ Spectral Contrast

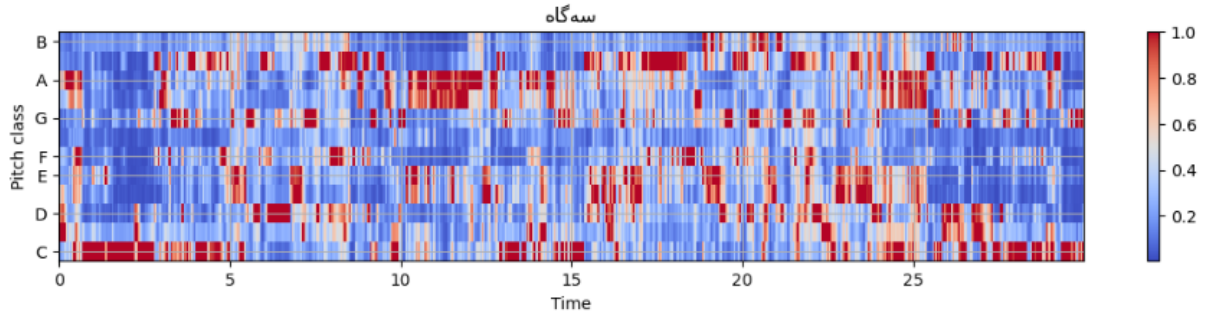
این ویژگی نشان‌دهنده اختلاف سقف و کف فرکانس در هر زیر برش فرکانسی از سیگنال می‌باشد.

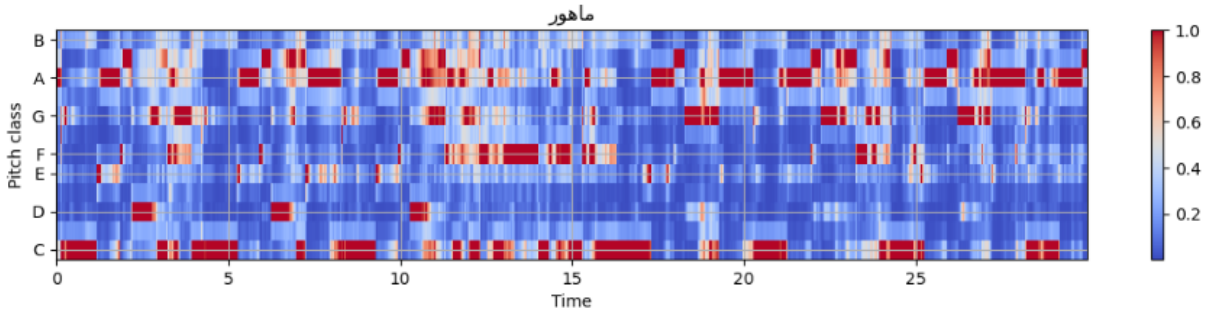
۲-۲-۲ ویژگی‌های هارمونیک

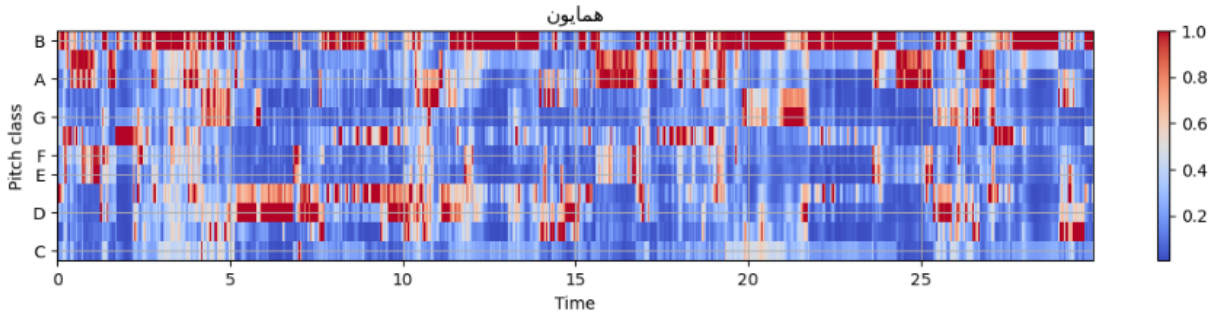
۲-۲-۲-۱ Chromagram

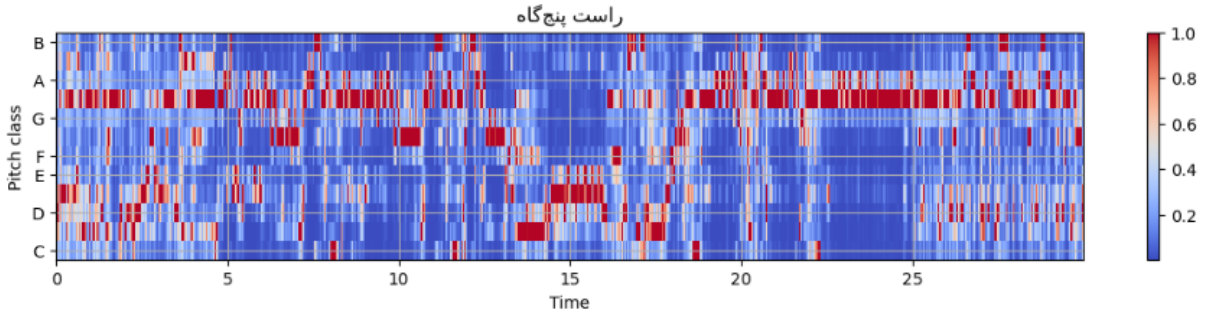
ویژگی‌های chroma ویژگی‌های قدرتمندی برای نمایش سیگنال‌های صوتی هستند که رابطه‌ی نزدیکی با ۱۲ کلاس متفاوت pitch(گام صدا) دارند. این ویژگی‌ها به خصوص در زمانی که گام‌ها نقش مهمی در طبقه‌بندی داشته باشند، می‌توانند ابزار بسیار خوبی برای تحلیل موسیقی باشند و حتی می‌توانند شباهت‌هایی را آشکارکنند که در طیف اصلی به سادگی قابل مشاهده نباشند.

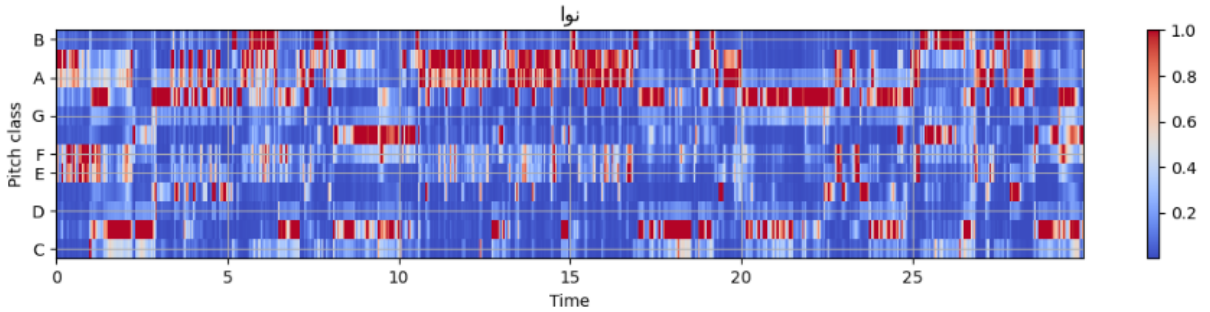


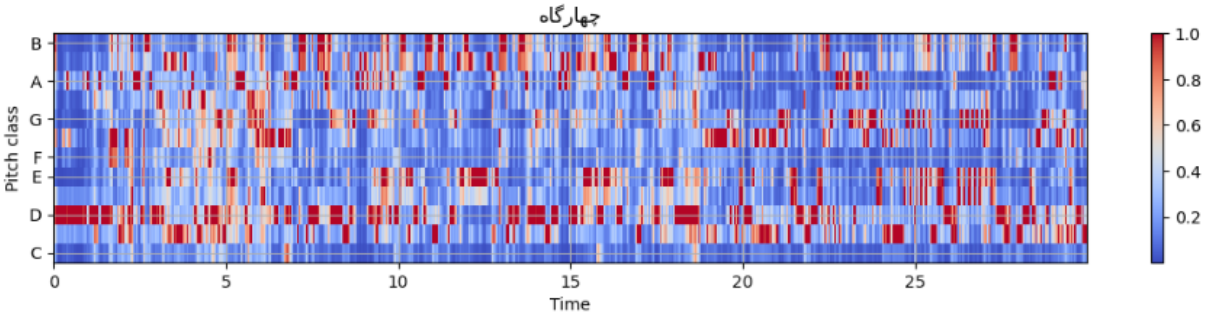

سه‌گاه
Pitch class
B
A
G
F
E
D
C
0
5
10
15
20
25
Time
1.0
0.8
0.6
0.4
0.2


ماهور
Pitch class
B
A
G
F
E
D
C
0
5
10
15
20
25
Time
1.0
0.8
0.6
0.4
0.2


همایون
Pitch class
B
A
G
F
E
D
C
0
5
10
15
20
25
Time
1.0
0.8
0.6
0.4
0.2


راست پنج‌گاه
Pitch class
B
A
G
F
E
D
C
0
5
10
15
20
25
Time
1.0
0.8
0.6
0.4
0.2


نوا
Pitch class
B
A
G
F
E
D
C
0
5
10
15
20
25
Time
1.0
0.8
0.6
0.4
0.2



۲-۲-۲-۲ Tonnetz

Tonnetz یک نمایش هندسی از هارمونی موسیقی است که در تئوری موسیقی برای تجسم روابط بین آکوردها و کلیدهای مختلف استفاده می شود. در زمینه فایل های صوتی، ویژگی Tonnetz به نمایش ریاضی ساختار هارمونیک یک قطعه موسیقی، بر اساس مفهوم Tonnetz اشاره دارد. در این ویژگی از یک قطعه موسیقی به عنوان مجموعه ای از نقاط در یک فضای چند بعدی، که در آن هر بعد مربوط به یک رابطه موسیقایی مانند فاصله بین آکوردها بر حسب نیم صدا است استفاده می‌شود.

۲-۲-۳ ویژگی‌های زمانی

۲-۲-۳-۱ Zero Crossing Rate

این ویژگی نرخ تغییرات علامت در یک سیگنال را نشان می‌دهد؛ یعنی تعداد دفعاتی که علامت سیگنال از مثبت به منفی و یا به صورت برعکس تغییر می‌کند. این ویژگی به طور گسترده در بحث speech recognition و بازیابی اطلاعات موسیقی مورد استفاده قرار گرفته است.

۲-۲-۳-۲ Root Mean Square Energy

انرژی به معنای اندازه کلی سیگنال در یک بازه می‌باشد. در سیگنال‌های صوتی این ویژگی نشان‌دهندی میزان بلندی صدای موسیقی می‌باشد. نحوه‌ی محاسبه RMSE یا انرژی rms به صورت زیر می‌باشد.

$$\sqrt{\frac{1}{N}\sum |x(n)|^2}$$

۳ طبقه‌بندی

۳-۱ تقسیم داده به دو بخش آموزش و آزمون

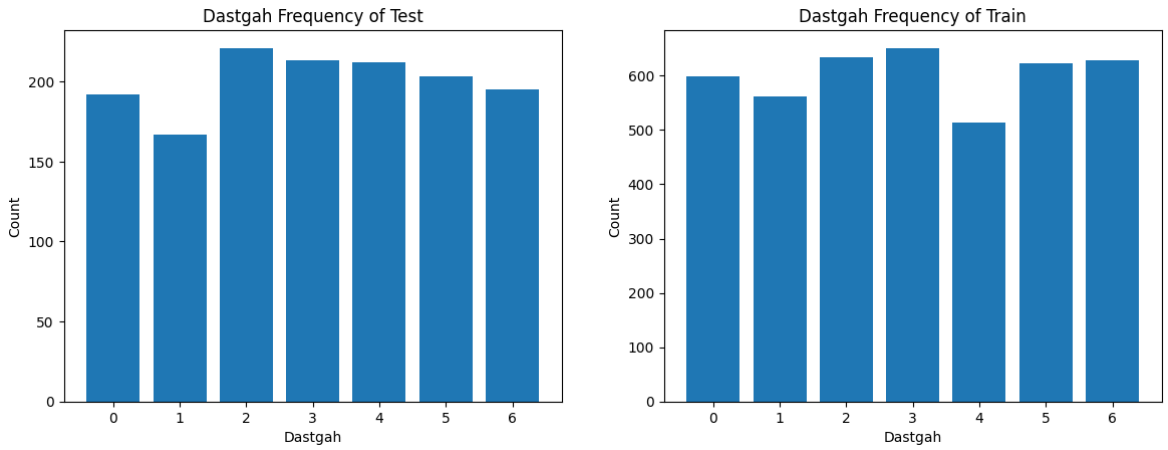
تقسیم داده به دو بخش آموزش و آزمون، یک مرحله مهم در یادگیری ماشین است. این کار باعث می‌شود از برازش بیش از حد (overfitting) جلوگیری شود. با تقسیم داده ها به یک مجموعه آموزشی و یک مجموعه آزمایشی، می توانیم از مجموعه آموزشی برای آموزش مدل و مجموعه آزمایشی برای ارزیابی عملکرد آن استفاده کنیم. این به ما کمک می کند تا ایده ای از عملکرد مدل بر روی داده های جدید و نادیده به دست آوریم و تخمین واقعی تری از عملکرد کلی آن به ما می دهد.

شایان ذکر است که نسبت تقسیم باید با دقت انتخاب شود، زیرا یک مجموعه آزمایشی بزرگ تخمین مطمئن تری از عملکرد مدل ارائه می دهد، اما مجموعه آموزشی کوچکتر ممکن است منجر به عدم تناسب شود.

برای تقسیم داده‌هایی که در قسمت قبل به‌دست آوردیم، از متد **train-test-split** از کتابخانه **scikit-learn** استفاده می‌کنیم. با استفاده از این تابع می‌توانیم داده‌ها را با نسبت معین به دو بخش آموزش و آزمون تقسیم کنیم. در اینجا ۲۵ درصد از داده ها به مجموعه آزمایش و ۷۵ درصد از داده‌ها به مجموعه آموزش اختصاص داده شده‌اند.

نکته دیگری که باید حتما در نظر بگیریم، متعادل کردن داده‌ها برای تقسیم است. متعادل کردن داده‌ها برای تقسیم، به این معنی است که سعی شود توزیع کلاس‌ها به صورت برابر باشند. یک مجموعه داده نامتعادل، می‌تواند منجر به مدل‌های مغرضانه شود که در کلاس‌هایی که کمتر دیده شده‌اند، باعث ضعف در عملکرد می‌شود.

در نمودارهای زیر، مشاهده می‌کنیم که کلاس‌ها در داده‌های آموزش و آزمون تقریبا به طور متعادل پخش شده‌اند:



۳-۲ استانداردسازی داده

استانداردسازی داده‌ها، فرآیند تبدیل داده‌ها به گونه ای است که میانگین صفر و انحراف معیار یک داشته باشد. این کار به چند دلیل مهم است:

- **بهبود عملکرد مدل:** بسیاری از الگوریتم‌های یادگیری ماشین به مقیاس ویژگی‌های ورودی حساس هستند. استانداردسازی داده‌ها می تواند با تبدیل داده‌ها به مقیاسی که برای الگوریتم مورد استفاده مناسب تر است، به بهبود عملکرد مدل کمک کند.
- **مقایسه منصفانه ویژگی‌ها:** هنگام مقایسه اهمیت ویژگی‌های مختلف، مهم است که آنها را در مقیاس مشابه داشته باشید. استانداردسازی داده‌ها می‌تواند به اطمینان از مقایسه منصفانه ویژگی‌ها و اینکه اهمیت نسبی هر ویژگی به دقت منعکس می‌شود کمک کند.
- **کاهش حجم محاسبات:** به طور کلی، استانداردسازی داده‌ها گامی حیاتی در پیش پردازش داده‌ها برای یادگیری ماشینی است و کمک می کند تا اطمینان حاصل شود که مدل قادر به یادگیری الگوهای اساسی در داده‌ها است و به خوبی به داده های جدید و دیده نشده تعمیم می دهد.

۳-۳ طبقه‌بندی براساس الگوریتم‌های یادگیری ماشین

در این قسمت توضیح مختصری درباره‌ی طبقه‌بندهای استفاده شده و پارامترهای آن‌ها می‌دهیم، سپس به ازای هر مدل آموزش داده‌شده، نمودار ROC و معیارهای accuracy، precision، recall، F1-score و error برای داده آموزش و آزمون هر دو feature set را نمایش می‌دهیم.

اروری که برای طبقه‌بندی در این پروژه استفاده کرده‌ایم، cross-entropy می‌باشد. این ارور، یک تابع از دست دادن متداول برای آموزش مدل های یادگیری ماشین، به ویژه برای مشکلات طبقه بندی است.cross-entropy error تفاوت بین توزیع احتمال پیش‌بینی‌شده و توزیع احتمال واقعی را برای یک مجموعه معین از کلاس‌ها اندازه‌گیری می‌کند.

$$accuracy = \frac{TP+FP}{TP+TN+FP+FN}$$

$$precision = \frac{TP}{TP+FN}$$

$$recall = \frac{TP}{TP+FP}$$

$$F1 = 2 * \frac{precision * recall}{precision+recall}$$

۳-۳-۱ K-Nearest Neighbors

الگوریتم KNN یک روش یادگیری supervised و غیرپارامتری است. غیر پارامتری بودن این الگوریتم بسیار خوب است زیرا اغلب داده‌ها در دنیای واقعی از فرضیات نظری معمول، تبعیت نمی‌کنند، از این رو زمانی که دانش قبلی درباره توزیع داده‌ها نداریم، یکی از بهترین گزینه‌ها برای طبقه‌بندی، استفاده از الگوریتم KNN است. عملکرد این الگوریتم مبتنی بر شباهت ویژگی‌هاست. به عبارتی زمانی که یک داده‌ی آزمون در اختیار ما قرار می‌گیرد، با توجه به میزان شباهت آن به داده‌های آموزشی، لیبل کلاس آن را مشخص می‌کنیم.

مراحل این الگوریتم در مسئله طبقه‌بندی، به صورت زیر می‌باشد:

۱. مقدار K را تعیین می‌کنیم که همان تعداد نزدیک‌ترین همسایه‌ها است.

۲. برای هر نمونه داده جدید (آزمون)، فاصله‌ی بین داده جدید را از نمونه داده‌های موجود (آموزش) محاسبه می‌کنیم که این فاصله می‌تواند از نوع اقلیدسی، منهتن یا ... باشد.

۳. داده‌های آموزش را بر اساس فاصله از نمونه داده جدید، به صورت صعودی مرتب می‌کنیم.

۴. K نمونه اول از داده‌های مرتب‌شده را به عنوان K نزدیک‌ترین همسایه انتخاب می‌کنیم.

۵. لیبل نمونه جدید را پرتکرارترین لیبل در بین این K همسایه در نظر می‌گیریم.

ما در این قسمت از KNeighborsClassifier از کتابخانه sklearn استفاده می‌کنیم. تعدادی از پارامترهای تاثیرگذار در این مدل به شرح زیر است:

- **n_neighbors:** تعداد نزدیک‌ترین همسایه‌ها (K) می‌باشد؛ در این پروژه برای یافتن بهترین K از تابع GridSearchCV و 10 fold cross validation استفاده می‌کنیم و مدل را به ازای K در بازه ۲ تا ۲۰ بررسی می‌کنیم زیرا می‌دانیم طبقه‌بند با K=1 نسبت به نویز خیلی حساس است، دچار overfitting می‌شود و روی داده آموزش به خوبی جواب نمی‌دهد. بهترین مقدار K با استفاده از این روش ۲ می‌شود.
- **weights:** تابع وزن استفاده شده در پیش‌بینی می‌باشد؛ یکی از مهمترین معایب knn این است که از نظر همسایه‌ها به طور یکسان استفاده می کند، در حالی که باید هر همسایه‌ای بر اساس فاصله‌ای که به نمونه ی تست دارد، سهم متفاوتی در پروسه تصمیم گیری داشته باشد. همین موضوع باعث شده که الگوریتم knn به تعداد k بسیار حساس باشد. در این پروژه برای حل این مشکل، این پارامتر را به distance ست می‌کنیم و از الگوریتم Weighted KNN استفاده می‌کنیم. این الگوریتم رای‌گیری را به صورت وزن‌دار انجام می‌دهد و به داده‌ها متناسب با معکوس فاصله آن‌ها وزن می‌دهد، یعنی به داده‌های نزدیک‌تر وزن بیشتر و به داده‌های دورتر وزن کم‌تری اختصاص می‌دهد.

- **metric:** معیار محاسبه فاصله می‌باشد؛ در این پروژه از متریک منهتن استفاده می‌کنیم زیرا در این مسئله نتیجه بهتری از فاصله اقلیدسی دارد.

معیارهای accuracy، precision، recall، F1-score و error برای داده آموزش و آزمون مدل فیت‌شده به شرح زیر است:

- Feature Set 1

```
-------------------------------Train-------------------------------

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       598
           1       1.00      1.00      1.00       562
           2       1.00      1.00      1.00       633
           3       1.00      1.00      1.00       651
           4       0.99      1.00      0.99       513
           5       1.00      0.99      0.99       623
           6       1.00      1.00      1.00       629

    accuracy                           1.00      4209
   macro avg       1.00      1.00      1.00      4209
weighted avg       1.00      1.00      1.00      4209

Error: 0.0032936430532678803
```

```
-------------------------------Test-------------------------------

              precision    recall  f1-score   support

           0       0.93      0.95      0.94       192
           1       0.93      0.98      0.95       167
           2       0.92      0.94      0.93       221
           3       0.91      0.94      0.92       213
           4       0.95      0.84      0.89       212
           5       0.90      0.91      0.90       203
           6       0.95      0.95      0.95       195

    accuracy                           0.93      1403
   macro avg       0.93      0.93      0.93      1403
weighted avg       0.93      0.93      0.93      1403

Error: 2.0052435632255974
```

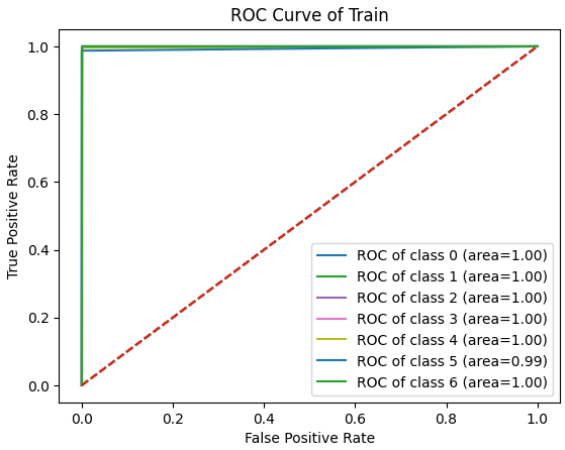
- Feature Set 2

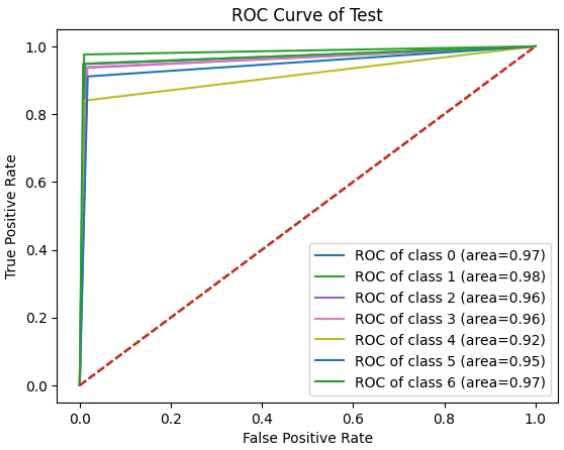
```
-------------------------------Train-------------------------------

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       598
           1       1.00      1.00      1.00       562
           2       1.00      1.00      1.00       633
           3       1.00      1.00      1.00       651
           4       0.99      1.00      0.99       513
           5       1.00      0.99      0.99       623
           6       1.00      1.00      1.00       629

    accuracy                           1.00      4209
   macro avg       1.00      1.00      1.00      4209
weighted avg       1.00      1.00      1.00      4209

Error: 0.0032936430532678803
```

```
-------------------------------Test-------------------------------

              precision    recall  f1-score   support

           0       0.83      0.85      0.84       192
           1       0.88      0.93      0.90       167
           2       0.88      0.88      0.88       221
           3       0.85      0.86      0.85       213
           4       0.89      0.82      0.85       212
           5       0.82      0.84      0.83       203
           6       0.87      0.85      0.86       195

    accuracy                           0.86      1403
   macro avg       0.86      0.86      0.86      1403
weighted avg       0.86      0.86      0.86      1403

Error: 3.397213325835309
```

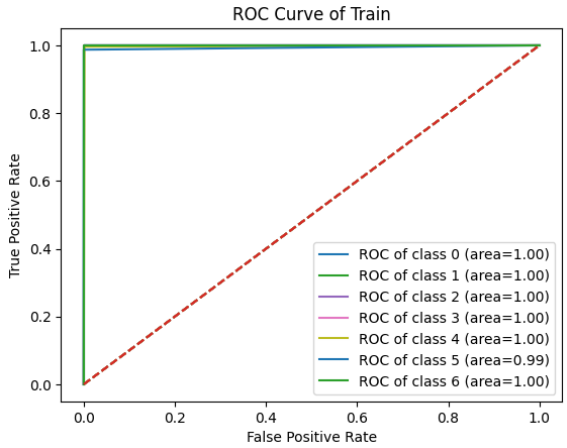
نمودارهای ROC رسم شده به صورت زیر هستند:

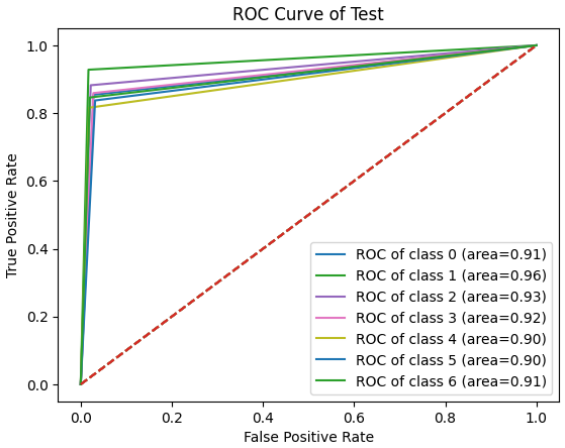
- Feature Set 1





- Feature Set 2





همان‌گونه که مشاهده می‌شود این روش طبقه‌بندی دقت بسیار خوبی را در مجموعه train و test به ما می‌دهد و در بین تمامی مدل‌ها دقت بالاتری دارد و خیلی کمتر دچار overfitting و بایاس می‌شود. همچنین به مانند سایر مدل‌ها این مدل نیز در مجموعه ویژگی‌های اولی عملکرد بهتری از خود نشان می‌دهد و با دقت یکسان در مجموعه train، دقت بالاتری در مجموعه test به ما می‌دهد.

۳-۳-۲ Support Vector Machine

الگوریتم SVM یک روش یادگیری supervised است که که نمونه‌ی داده‌هایی را که به صورت نقاطی در فضا نشان داده شده است، با استفاده از یک خط یا hyperplane، از هم جدا می‌کند. این جداسازی به‌گونه‌ای است که نقاط داده‌ای که در یک طرف خط هستند مشابه به هم و در یک گروه قرار می‌گیرند. نمونه‌داده‌های جدید هم بعد از اضافه‌شدن به همان فضا در یکی از دسته‌های موجود قرار خواهند گرفت.اگر داده‌ها به صورت خطی جداپذیر باشند، معیار انتخاب بهترین hyperplane این است که بیشترین حاشیه اطمینان (margin) بین کلاس‌ها را ایجاد کند، اما در صورتی که داده‌ها به صورت خطی جداپذیر نباشند، داده‌ها به کمک یک تابع ریاضی (Kernel function) به یک فضای دیگر با ابعاد بالاتر نگاشت داده می‌شوند، که در آن فضا داده‌ها به صورت خطی تفکیک‌پذیر هستند و می‌توان SVM آن‌ها را به راحتی تعیین کرد. تعیین درست این تابع نگاشت در عملکرد الگوریتم SVM بسیار موثر است. حل معادله پیدا کردن خط (hyperplane) بهینه برای داده‌ها به وسیله روش‌های برنامه سازی غیرخطی که روش‌های شناخته شده‌ای در حل مسائل محدودیت‌دار هستند صورت می‌گیرد.

ما در این قسمت از SVC از کتابخانه sklearn استفاده می‌کنیم. تعدادی از پارامترهای تاثیرگذار در این مدل به شرح زیر است:

- **kernel:** نوع تابع نگاشت می‌باشد؛ در این پروژه از کرنل rbf استفاده می‌کنیم زیرا در این مسئله نتیجه بهتری از سایر کرنل‌ها می‌دهد. همچنین این کرنل گوسی به صورت $K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2)$ می‌باشد و پرکاربردترین کرنل برای حل مسائل طبقه‌بندی مجموعه داده‌هایی است که به صورت خطی جداپذیر نیستند.

این کرنل با پارامترهای خاص، عملکرد خوبی دارد و نتایج ترین آن در مقایسه با سایر کرنل‌ها، خطای کمتری دارد زیرا ویژگی‌ها را به فضایی با بعد بی‌نهایت می‌برد.

- **Decision_function_shape:** این پارامتر One-vs-One(OvO) یا One-vs-Rest (OvR) بودن را مشخص می‌کند که روش‌هایی برای مسئله طبقه‌بندی چندکلاسه با استفاده از الگوریتم‌های طبقه‌بندی باینری می‌باشد. در این پروژه از OvR استفاده می‌کنیم. در این روش به ازای هر کلاس یک طبقه‌بند باینری فیت می‌کنیم یعنی هر کلاس در برابر سایر کلاس‌ها طبقه‌بندی می‌شود. از جمله مزایای این روش کارایی محاسباتی و تفسیرپذیری آن است؛ از آنجایی که هر کلاس فقط با یک طبقه‌بنده نشان داده می‌شود، می‌توان با بررسی طبقه‌بند مربوطه، اطلاعاتی در مورد کلاس کسب کرد. این متداول ترین استراتژی مورد استفاده برای طبقه‌بندی چند کلاسه است و یک انتخاب پیش فرض منصفانه است.
- **C:** پارامتر regularization می‌باشد و قدرت regularization با C نسبت معکوس دارد. در این پروژه برای یافتن بهترین C از تابع GridSearchCV و 10 fold cross validation استفاده می‌کنیم و مدل را به ازای C در مجموعه {0.1 , 1, 10} بررسی می‌کنیم. بهترین مقدار C با استفاده از این روش 10 می‌شود.
- **gamma:** پارامتری مربوط به کرنل‌های poly، rbf و sigmoid است. به عنوان مثال کرنل rbf که در این پروژه استفاده شده است، یک پارامتر $\gamma = \frac{1}{2\sigma^2}$ دارد که می توان آن را به عنوان معکوس شعاع تأثیر نمونه‌های انتخاب شده توسط مدل به عنوان SV مشاهده کرد. بنابراین هر چه γ بزرگ‌تر شود، مرز تصمیم‌گیری پیچیده‌تر می‌شود و تعداد SVها بیشتر می‌شود، به طوری که در γ خیلی بزرگ، هر داده در دیتاست یک SV است و مدل overfit می‌کند. دقت مدل با γ خیلی کوچک یا خیلی بزرگ، کم خواهد بود. در این پروژه برای یافتن بهترین γ از تابع GridSearchCV و 10 fold cross validation استفاده می‌کنیم و مدل را به ازای γ در مجموعه {0.1 , 1, 10} بررسی می‌کنیم. بهترین مقدار γ با استفاده از این روش 0.1 می‌شود.

معیارهای accuracy، precision، recall، F1-score و error برای داده آموزش و آزمون مدل فیت‌شده به شرح زیر است:

- Feature Set 1

```
-------------------------------Train-------------------------------

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       598
           1       1.00      1.00      1.00       562
           2       1.00      1.00      1.00       633
           3       1.00      1.00      1.00       651
           4       1.00      0.98      0.99       513
           5       0.99      1.00      0.99       623
           6       1.00      1.00      1.00       629

    accuracy                           1.00      4209
   macro avg       1.00      1.00      1.00      4209
weighted avg       1.00      1.00      1.00      4209

Error: 0.008576157620670107
```

```
-------------------------------Test-------------------------------

              precision    recall  f1-score   support

           0       0.93      0.89      0.91       192
           1       0.95      0.90      0.93       167
           2       0.83      0.95      0.88       221
           3       0.71      0.96      0.82       213
           4       0.97      0.71      0.82       212
           5       0.90      0.85      0.88       203
           6       0.96      0.85      0.90       195

    accuracy                           0.87      1403
   macro avg       0.89      0.87      0.88      1403
weighted avg       0.89      0.87      0.87      1403

Error: 0.4322361882123458
```

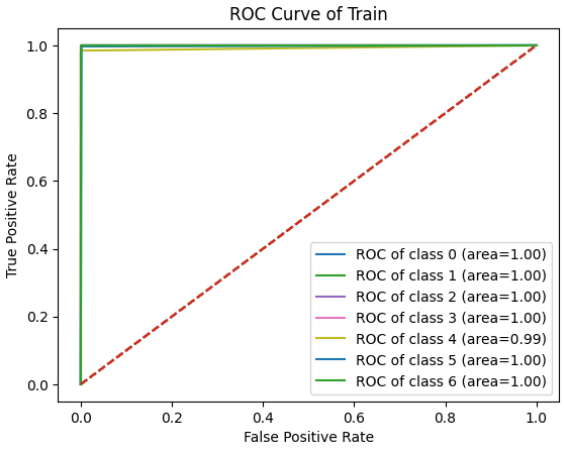
- **Feature Set 2**

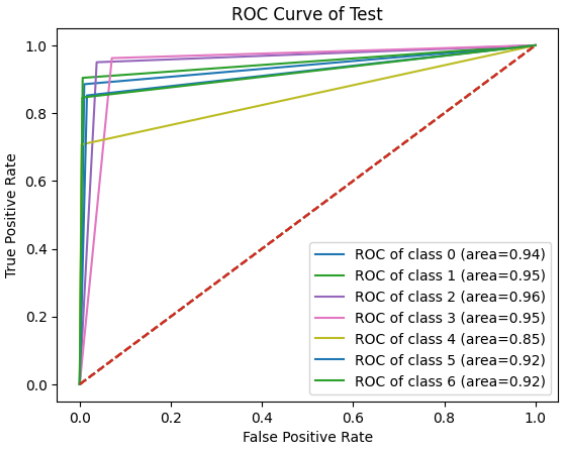
```
-----------------------------Train-----------------------------

              precision    recall  f1-score   support

           0       0.99      1.00      0.99       598
           1       1.00      0.99      1.00       562
           2       0.99      0.99      0.99       633
           3       0.99      0.99      0.99       651
           4       0.99      0.99      0.99       513
           5       0.99      0.99      0.99       623
           6       1.00      0.99      1.00       629

    accuracy                           0.99      4209
   macro avg       0.99      0.99      0.99      4209
weighted avg       0.99      0.99      0.99      4209

Error: 0.08663110788142857
```

```
-----------------------------Test-----------------------------

              precision    recall  f1-score   support

           0       0.74      0.84      0.79       192
           1       0.87      0.90      0.88       167
           2       0.84      0.83      0.84       221
           3       0.80      0.85      0.82       213
           4       0.83      0.75      0.79       212
           5       0.80      0.75      0.77       203
           6       0.85      0.81      0.83       195

    accuracy                           0.82      1403
   macro avg       0.82      0.82      0.82      1403
weighted avg       0.82      0.82      0.82      1403

Error: 0.5902545594060903
```

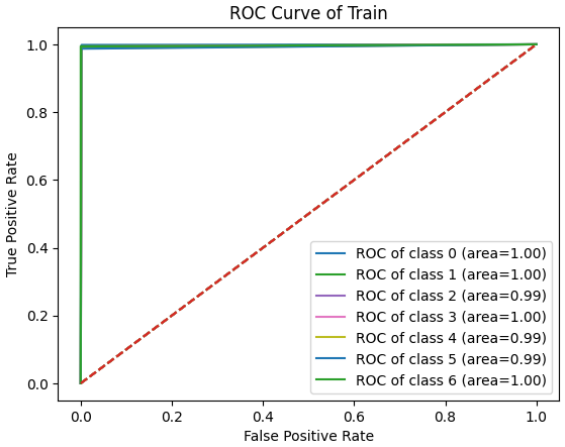
نمودارهای ROC رسم شده به صورت زیر هستند:

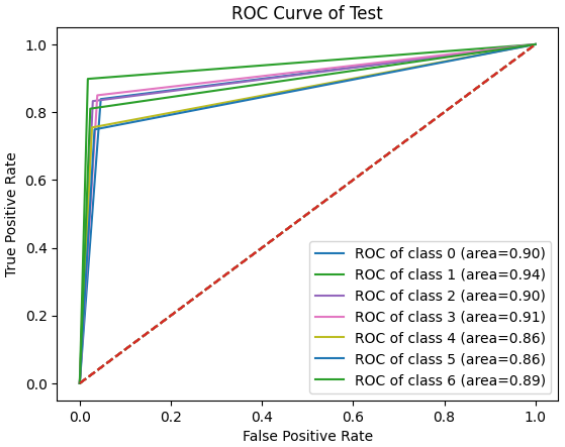
- **Feature Set 1**





- **Feature Set 2**





همان‌گونه که مشاهده می‌شود این روش طبقه‌بندی دقت بسیار خوبی را در مجموعه train و test به ما می‌دهد و پس از knn از بقیه مدل‌ها دقت بالاتری دارد و و خیلی کمتر دچار overfitting و بایاس می‌شود. همچنین به مانند سایر مدل‌ها این مدل نیز

در مجموعه ویژگی‌های اولی عملکرد بهتری از خود نشان می‌دهد و دقت بالاتری در مجموعه train و test به ما می‌دهد.

۳-۳-۳ Extreme Gradient Boosting(XGBoost)

روش boosting به معنای ترکیب چندین مدل ضعیف با یکدیگر است، به گونه‌ای که مدل به دست آمده از ترکیب آن‌ها یک مدل قدرتمند باشد. Gradient boosting افزونه‌ای بر روی boosting است که در آن مدل‌های ضعیف جدید با استفاده از الگوریتم gradient descent بر روی یک تابع هدف به دست می‌آیند. در این روش مدل بعدی با هدف کمینه کردن مقدار خطا ایجاد می‌شود.

XGBoost یک پیاده‌سازی بسیار دقیق و انعطاف‌پذیر از روش gradient boosting می‌باشد که بر اساس درخت‌های تصمیم‌گیری کار می‌کند. در این روش دررخت‌ها به صورت موازی ساخته می‌شوند و در هر مرحله با توجه مقدار به مقادیر گرادیان کیفیت انواع روش‌های شکستن دیتاست ترین بررسی و سپس تصمیم‌گیری می‌شود.

این الگوریتم توسط کتابخانه‌ی xgboost در پایتون پیاده‌سازی شده است. تعدادی از پارامترهای تاثیرگذار در این مدل به شرح زیر است:

- **n_estimators:** تعداد درخت‌های تصمیم‌گیری که در مدل استفاده می‌شوند. مقدار این پارامتر برابر با ۳۰۰ قرار گرفته است. طبق تست‌های انجام شده، دقت در داده‌های تست تقریبا در تمام حالات برابر ۱ می‌شود و بیشینه دقت داده تست نیز در این تعداد از درخت تصمیم‌گیری رخ می‌دهد و افزایش تعداد درخت‌ها سبب overfit شدن مدل و افزایش خطای generalization در مدل می‌شود.
- **max_depth:** نشان‌دهنده حداکثر عمق پیشروی درخت‌های تصمیم‌گیری می‌باشد. به صورت پیش‌فرض مقدار این پارامتر برابر ۶ می‌باشد که بهترین دقت را در خروجی به ما می‌دهد.
- **max_leaves:** نشان‌دهنده حداکثر تعداد برگ‌های ممکن در یک درخت تصمیم‌گیری می‌باشد. در مدل ما مقدار این پارامتر برابر ۰ در نظر گرفته شده است که به معنای تعداد نامحدود برگ‌ها است و به جای آن از پارامتر بالایی برای کنترل اندازه درخت‌ها استفاده شده است.
- **gamma:** مشخص می‌کند برای آن‌که یک برگ از درخت را بشکنیم، حداقل به چه مقدار باید تابع loss را کاهش دهد. در این مدل مقدار این پارامتر نیز ۰ در نظر گرفته کمترین مقدار ممکن است و بهترین دقت را در هر دو مجموعه داده به ما می‌دهد.

معیارهای accuracy، precision، recall، F1-score و error برای داده آموزش و آزمون مدل فیت‌شده به شرح زیر است:

- **Feature Set 1**

```
-------------------------------Train-------------------------------

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       598
           1       1.00      1.00      1.00       562
           2       1.00      1.00      1.00       633
           3       1.00      1.00      1.00       651
           4       0.99      0.99      0.99       513
           5       1.00      0.99      0.99       623
           6       1.00      1.00      1.00       629

    accuracy                           1.00      4209
   macro avg       1.00      1.00      1.00      4209
weighted avg       1.00      1.00      1.00      4209

Error: 0.005794619381878384
```

```
-------------------------------Test-------------------------------

              precision    recall  f1-score   support

           0       0.79      0.90      0.84       192
           1       0.88      0.89      0.88       167
           2       0.84      0.85      0.84       221
           3       0.80      0.86      0.83       213
           4       0.91      0.72      0.81       212
           5       0.83      0.81      0.82       203
           6       0.88      0.88      0.88       195

    accuracy                           0.84      1403
   macro avg       0.85      0.84      0.84      1403
weighted avg       0.85      0.84      0.84      1403

Error: 0.5196322246788799
```

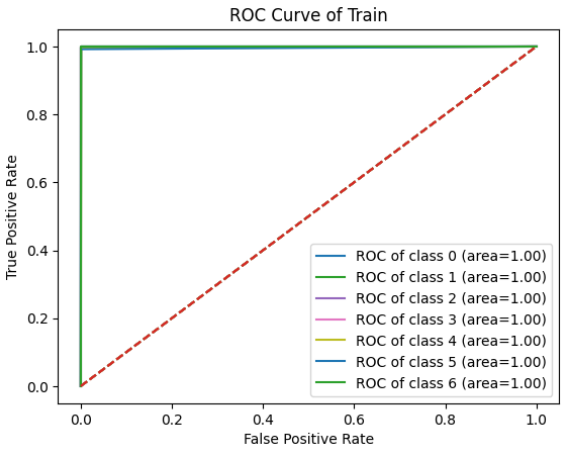
- **Feature Set 2**

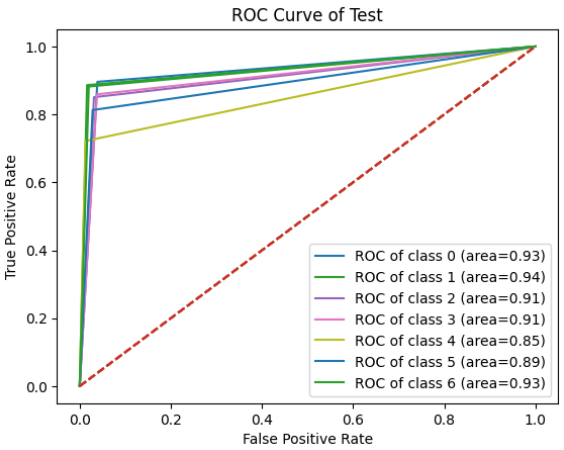
```
-------------------------------Train-------------------------------

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       598
           1       1.00      1.00      1.00       562
           2       1.00      1.00      1.00       633
           3       1.00      1.00      1.00       651
           4       0.99      0.99      0.99       513
           5       0.99      0.99      0.99       623
           6       1.00      1.00      1.00       629

    accuracy                           1.00      4209
   macro avg       1.00      1.00      1.00      4209
weighted avg       1.00      1.00      1.00      4209

Error: 0.008221420472188464
```

```
-------------------------------Test-------------------------------

              precision    recall  f1-score   support

           0       0.73      0.78      0.75       192
           1       0.80      0.84      0.82       167
           2       0.84      0.77      0.80       221
           3       0.74      0.81      0.77       213
           4       0.81      0.68      0.74       212
           5       0.73      0.78      0.76       203
           6       0.75      0.76      0.76       195

    accuracy                           0.77      1403
   macro avg       0.77      0.77      0.77      1403
weighted avg       0.77      0.77      0.77      1403

Error: 0.7622170174145306
```

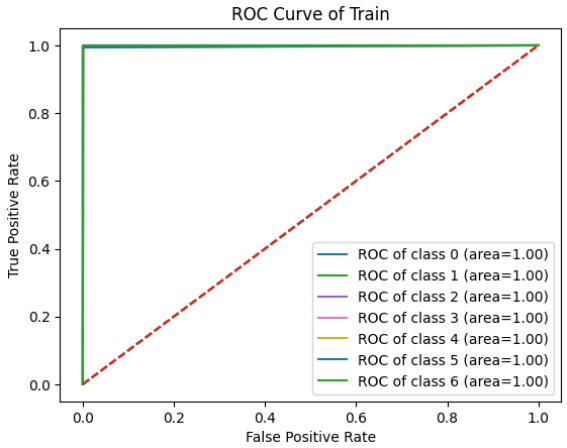
نمودارهای ROC رسم شده به صورت زیر هستند:

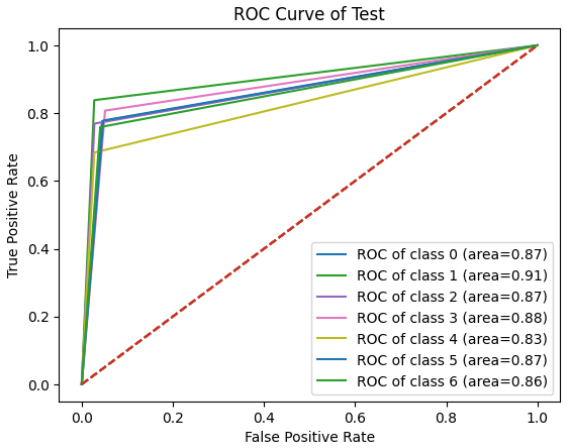
- **Feature Set 1**





- Feature Set 2





همان‌گونه که مشاهده می‌شود این روش طبقه‌بندی دقت بسیار خوبی را در مجموعه train و test به ما می‌دهد و پس از svm و knn بهترین دقت را در مجموعه تست به ما می‌دهد. همچنین به مانند سایر مدل‌ها این مدل نیز در مجموعه ویژگی‌های اولی عملکرد بهتری از خود نشان می‌دهد و با دقت یکسان در مجموعه train، دقت بالاتری در مجموعه test به ما می‌دهد.

۴-۳-۳ Multilayer Perceptron(MLP)

پرسپترون چند لایه یا MLP دسته‌ای از شبکه‌های عصبی FeedForward است که از پشت سر هم قرار گرفتن چندین پرسپترون تک لایه به دست می‌آید. وجود لایه نهان در MLP باعث می‌شود بتواند به خوبی همبستگی بین داده‌های ورودی و خروجی را مدل کند و حتی داده‌هایی که به صورتی خطی تفکیک پذیر نیستند را نیز تفکیک کند.

مرحله آموزش در MLP شامل تنظیم وزن یال‌ها و بایاس هر گره است به طوری‌که میزان خطا حداقل شود. این فرآیند یادگیری از یک حرکت رو به جلو و یک حرکت رو به عقب تشکیل شده است. در حرکت رو به جلو جریان سیگنال از لایه ورودی شروع می‌شود و بعد از عبور از لایه‌های میانی به لایه خروجی می‌رسد که به آن خروجی پیش‌بینی شده می‌گویند. با مقایسه این خروجی پیش بینی شده با خروجی مورد انتظار، مقدار loss شبکه محاسبه می‌شود سپس این مقدار خطا به صورت معکوس در شبکه منتشر می‌شود. برای پیدا کردن مجموعه وزن‌های جدید و کمینه کردن خطا از روش گرادیان کاهشی استفاده می‌شود. برای گره‌های لایه‌های میانی مقدار گرادیان از روش مشتق زنجیره‌ای و برای گره‌های لایه آخر از روش مشتق جزیی به دست می‌آید و با استفاده از این مقدار وزن هر گره در جهت کاهش خطا بروزرسانی شوند.

ما برای پیاده‌سازی این طبقه‌بند از MLPClassifier که در کتابخانه sklearn موجود است استفاده می‌کنیم. تعدادی از پارامترهای تاثیرگذار در این مدل به شرح زیر است:

- Hidden_layer_sizes: این پارامتر تعداد لایه های پنهان و تعداد نورون‌های هر لایه را مشخص می‌کند. در این پروژه ما چهار لایه پنهان با اندازه‌های 128 نورون، ۶۴ نورون، ۳۲ نورون و ۸ نورون تعریف کردیم. لایه ورودی و

خروجی نیز خود با توجه به مجموعه داده تنظیم می‌شوند به طوری که همیشه تعداد نورون‌های لایه ورودی برابر با تعداد ویژگی‌های داده‌ها است و تعداد نورون‌های لایه خروجی نیز برابر با تعداد کلاس‌ها است.

- Batch_size: این پارامتر تعداد نمونه‌های که در هر مرحله برای یادگیری به شبکه داده می‌شوند را مشخص می‌کند. هر چه batch size بزرگتر باشد مقدار گرادیان به دست آمده در هر دور پایدارتر خواهد بود و نوسانات کمتری دارد و شبکه به کندی به سمت نقطه بهینه همگرا می‌شود. هر چه batch size کوچک تر باشد فرآیند یادگیری سریع‌تر می‌شود زیرا وزن های شبکه به ازای هر دسته داده ورودی یک بار به روزرسانی می‌شوند. همچنین batch size کوچکتر به حافظه کمتری نیاز دارد اما کاهش batch size موجب ناپایداری گرادیان می‌شود و در صورتی که نرخ یادگیری به درستی تعیین نشده باشد می‌تواند باعث شود شبکه در بهینه محلی گیر کند. ما برای مدل خود batch size را ۳۲ قرار دادیم تا نه آنقدر کوچک باشد که فرآیند یادگیری را مختل کند و نه آن قدر بزرگ باشد که زمان و حافظه خیلی زیادی نیاز داشته باشد.
- Solver: این پارامتر روش بهینه سازی مورد استفاده در شبکه را مشخص می‌کند. این پارامتر می‌تواند یکی از مقادیر adam ، sgd و lbfgs باشد. همان‌طور که گفته شد در فاز back propagation برای آپدیت کردن وزن‌ها از روش گرادیان کاهش استفاده می‌شود. در این روش لازم است در هر دور گرادیان کل دسته داده‌های ورودی حساب شود که اگر تعداد داده‌ها زیاد باشد بسیار وقت‌گیر است و حتی ممکن است به دلیل محدودیت حافظه ناممکن باشد. در نتیجه اکثرا از روش stochastic gradient descent استفاده می‌شود. در این روش در هر دور یک داده به تصادف انتخاب شده و محاسبه گرادیان برای آن انجام می‌شود و وزن ها با توجه به آن مقدار آپدیت می‌شوند. البته ممکن است این روش باعث شود شبکه به نقطه بهینه سراسری نرسد و در اطراف آن نوسان کند.
- Momentum: استفاده از momentum مناسب می‌تواند به حل مشکل SGD کمک کند. در این روش به جای‌این که برای به‌روزرسانی وزن‌ها فقط از مقدار گرادیان مرحله فعلی استفاده شود، مقادیر گرادیان مرحله‌های قبلی نیز در نظر گرفته می‌شود. مقدار momentum ضریب تاثیر گرادیان‌های قبلی را مشخص می‌کند. اگر این ضریب خیلی کوچک باشد نمی‌تواند به SGD کمک کند از بهینه محلی خارج شود و اگر خیلی بزرگ باشد با آپدیت کردن غیر دقیق وزن ها باعث کاهش دقت مدل می‌شود. ما در این پروژه momentum را 0.85 در نظر گرفتیم.
- Learning_rate_init: نرخ یادگیری یک عدد اسکالر است که در گرادیان تابع خطا ضرب می‌شود و سرعت یادگیری مدل را کنترل می‌کند. نرخ یادگیری کوچک باعث می‌شود در هر مرحله به‌روز رسانی وزن‌ها دقیق تر باشد که موجب می‌شود فرآیند آموزش بسیار طولانی شود و یا حتی در یک بهینه محلی متوقف شود. اگر نرخ یادگیری خیلی بزرگ باشد به روزرسانی غیر دقیق وزن‌ها باعث می‌شود تغییرات تابع تا حدی تصادفی باشد و نوسانات زیادی داشته باشد. در این مدل مقدار این پارامتر را 0.006در نظر گرفتیم.

- Max_iter: این پارامتر تعداد بیشینه epoch ها را مشخص می‌کند. در هر epoch یک دوره کل داده‌ها برای آموزش به شبکه داده می‌شوند. هر چه تعداد epoch ها بیشتر باشد وزن ها تعداد دفعات بیشتری آپدیت می‌شوند و دقت شبکه بیشتر می‌شود اما اگر تعداد epoch ها بیش از حد زیاد باشد مدل overfit می‌شود و مهم است فرآیند آموزش قبل از این نقطه متوقف شود تا قدرت generalization مدل کاهش پیدا نکند. در این مدل مقدار این پارامتر را 400 در نظر گرفتیم.

معیارهای accuracy، precision، recall، F1-score و error برای داده آموزش و آزمون مدل فیت‌شده به شرح زیر است:

- Feature Set 1

```
-----------------------------Train-----------------------------

              precision    recall  f1-score   support

           0       1.00      0.99      1.00       598
           1       1.00      1.00      1.00       562
           2       1.00      1.00      1.00       633
           3       1.00      1.00      1.00       651
           4       1.00      0.98      0.99       513
           5       0.99      1.00      0.99       623
           6       1.00      1.00      1.00       629

    accuracy                           1.00      4209
   macro avg       1.00      1.00      1.00      4209
weighted avg       1.00      1.00      1.00      4209

Error: 0.004715097315903805
```

```
-----------------------------Test-----------------------------

              precision    recall  f1-score   support

           0       0.78      0.82      0.80       192
           1       0.83      0.84      0.84       167
           2       0.84      0.85      0.84       221
           3       0.81      0.84      0.82       213
           4       0.83      0.71      0.77       212
           5       0.79      0.82      0.80       203
           6       0.85      0.84      0.85       195

    accuracy                           0.82      1403
   macro avg       0.82      0.82      0.82      1403
weighted avg       0.82      0.82      0.82      1403

Error: 1.2389900679502492
```

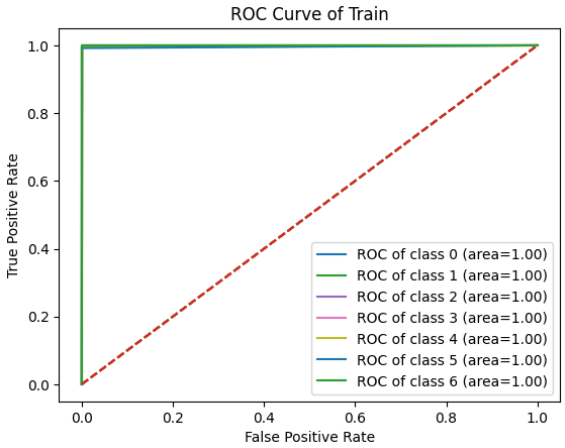
- Feature Set 2

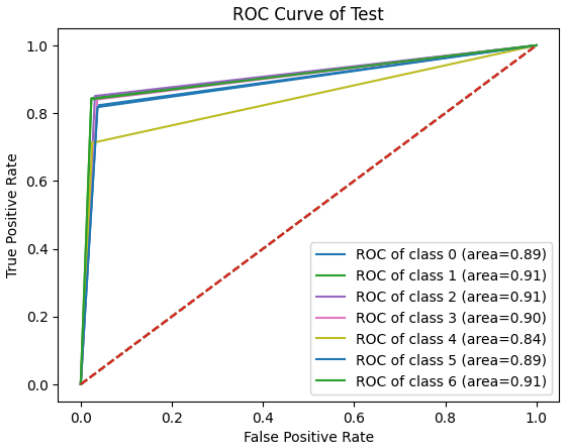
```
-----------------------------Train-----------------------------

              precision    recall  f1-score   support

           0       0.99      0.99      0.99       598
           1       1.00      1.00      1.00       562
           2       1.00      0.99      0.99       633
           3       0.99      1.00      0.99       651
           4       1.00      0.98      0.99       513
           5       0.98      1.00      0.99       623
           6       1.00      1.00      1.00       629

    accuracy                           0.99      4209
   macro avg       0.99      0.99      0.99      4209
weighted avg       0.99      0.99      0.99      4209

Error: 0.02525958885441375
```

```
-----------------------------Test-----------------------------

              precision    recall  f1-score   support

           0       0.72      0.81      0.76       192
           1       0.81      0.80      0.80       167
           2       0.78      0.74      0.76       221
           3       0.75      0.81      0.78       213
           4       0.77      0.67      0.72       212
           5       0.69      0.73      0.71       203
           6       0.77      0.74      0.75       195

    accuracy                           0.75      1403
   macro avg       0.76      0.76      0.76      1403
weighted avg       0.76      0.75      0.75      1403

Error: 1.646954450207116
```

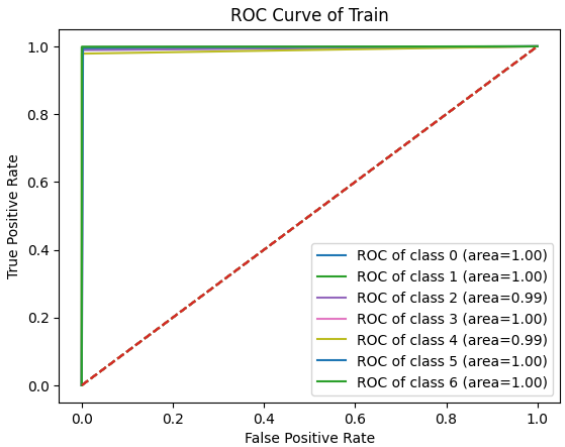
نمودارهای ROC رسم شده به صورت زیر هستند:

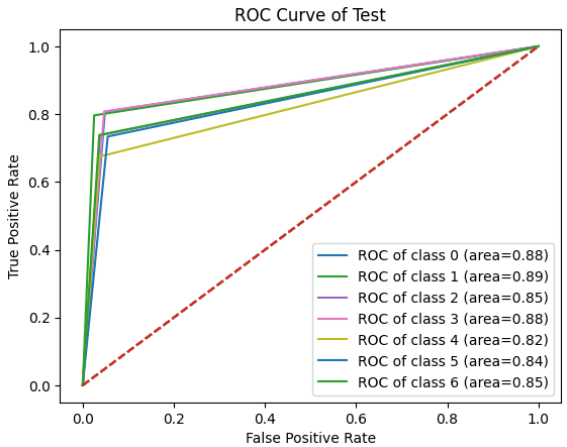
- Feature Set 1





- Feature Set 2





همان‌گونه که مشاهده می‌شود این روش طبقه‌بندی نیز مانند روش‌های قبلی دقت بسیار خوبی را در مجموعه train و test به ما می‌دهد. دقت این روش در هر دو مجموعه ویژگی برای داده‌های آموزش مشابه دقت طبقه‌بند svm است اما برای داده‌های تست svm کمی بهتر از این مدل عمل می‌کند که نشان می‌دهد این طبقه‌بند به دلیل پیچیدگی خود مدل کمی بیشت. این مدل نیز در مجموعه ویژگی‌های اولی عملکرد بهتری از خود نشان می‌دهد .

۵-۳-۳ Logistic Regression

Logistic regression یک الگوریتم طبقه بندی تحت نظارت است که از آن می‌توان برای پیش‌بینی احتمال یک متغیر هدف استفاده کرد. در این روش مجموع وزن دار ویژگی‌های ورودی به همراه یک مقدار خطا به یک تابع sigmoid داده می‌شوند و خروجی این تابع که یک عدد بین صفر و یک هست احتمال تعلق داده به کلاس مورد نظر را نشان می‌دهد. این الگوریتم برای پیدا کردن وزن‌های بهینه باید تابع هزینه زیر را کمینه کند.

$$J(\theta) = \frac{-1}{m}\sum\left[y^{(i)}log(h_\theta(x^{(i)})) + (1 - y^{(i)})log(1 - h_\theta(x^{(i)}))\right]$$

برای بهینه سازی این تابع فرم بسته‌ای وجود ندارد پس برای پیدا کردن پارامترهای بهینه مجبوریم از روش‌های گرادیان کاهشی استفاده کنیم. برای این کار در هر مرحله مشتق زنجیره‌ای هر پارامتر را نسبت به خطای به دست‌آمده حساب می‌کینم و در نهایت مقدار مشتق را از مقدار قبلی آن پارامتر کم می‌کنیم تا مقدار پارامتر به‌روزرسانی شود.

ما برای پیاده‌سازی این طبقه‌بند از Logistic Regression که در کتابخانه sklearn موجود است استفاده می‌کنیم. تعدادی از پارامترهای تاثیرگذار در این مدل به شرح زیر است:

- **Penalty**: یک روش برای جلوگیری از **overfitting** اضافه کردن یک ترم جریمه به عنوان **regulizer** به تابع هزینه است. اضافه شدن این هزینه باعث می‌شود مدل درجه آزادی کمتری برای تغییر وزن‌ها داشته باشد.

  در روش **ridge regression** یا **l2** این پنالتی برابر مجموع مجذور وزن‌ها ست و در روش **lasso** یا **l1** برابر با مجموع قدرمطلق وزن هاست. روش **l1** باعث می‌شود بعضی از وزن‌ها صفر شوند و در واقع مانند یک روش انتخاب ویژگی عمل می‌کند. این روش بیشتر برای وقتی مناسب است که داده‌ها خیلی پراکنده هستند. روش **l2** باعث می‌شود همه وزن‌ها به صفر میل کنند اما برابر صفر نشوند در نتیجه هیچ اطلاعاتی از دست نمی‌رود. ما در این پروژه برای این پارامتر مقدار **l1** را قرار دادیم.

- **C**: مقدار پنالتی مشخص شده ابتدا در یک ضریب ضرب می‌شود و سپس به تابع هزینه کل اضافه می‌شود. این ضریب تاثیر مقدار پنالتی را مشخص می‌کند و هر چه بزرگتر باشد باعث می‌شود وزن‌ها به صفر نزدیک تر شوند.

  پارامتر **C** برابر با معکوس این ضریب است و هر چه کوچکتر باشد به معنای **regularization** بیشتر است. در این پروژه بعد از بررسی مقادیر مختلف، مقدار ۰.۴۳ را برای این پارامتر انتخاب کردیم.

معیارهای **accuracy**، **precision**، **recall**، **F1-score** و **error** برای داده آموزش و آزمون مدل فیت‌شده به شرح زیر است:

- **Feature Set 1**

```
----------------------------Train----------------------------

              precision    recall  f1-score   support

           0       0.42      0.38      0.40       598
           1       0.50      0.52      0.51       562
           2       0.45      0.46      0.45       633
           3       0.44      0.48      0.46       651
           4       0.42      0.36      0.39       513
           5       0.49      0.47      0.48       623
           6       0.50      0.55      0.53       629

    accuracy                           0.46      4209
   macro avg       0.46      0.46      0.46      4209
weighted avg       0.46      0.46      0.46      4209

Error: 1.4846383125378444
```

```
-------------------------------Test-------------------------------

              precision    recall  f1-score   support

           0       0.38      0.34      0.36       192
           1       0.46      0.55      0.50       167
           2       0.41      0.40      0.40       221
           3       0.38      0.44      0.41       213
           4       0.45      0.35      0.39       212
           5       0.42      0.44      0.43       203
           6       0.45      0.45      0.45       195

    accuracy                           0.42      1403
   macro avg       0.42      0.42      0.42      1403
weighted avg       0.42      0.42      0.42      1403

Error: 1.569730328773351
```

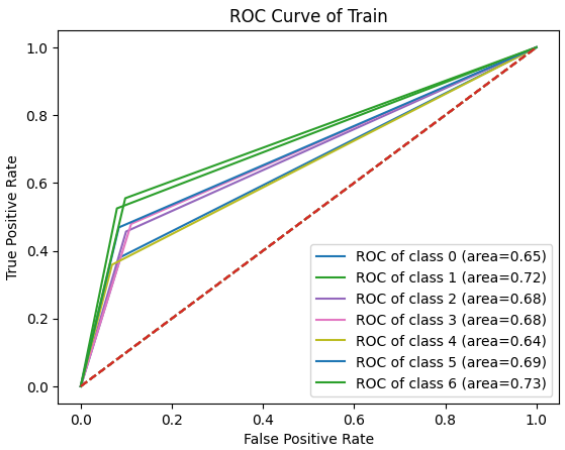
- **Feature Set 2**

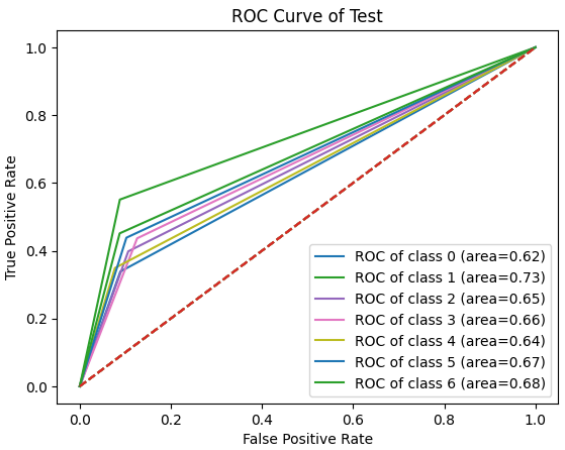
```
-----------------------------Train-----------------------------

              precision    recall  f1-score   support

           0       0.27      0.27      0.27       598
           1       0.27      0.22      0.24       562
           2       0.30      0.30      0.30       633
           3       0.27      0.35      0.31       651
           4       0.25      0.23      0.24       513
           5       0.31      0.35      0.33       623
           6       0.29      0.24      0.26       629

    accuracy                           0.28      4209
   macr o avg       0.28      0.28      0.28      4209
weighted avg       0.28      0.28      0.28      4209

Error: 1.8260369383457316
```

```
-----------------------------Test-----------------------------

              precision    recall  f1-score   support

           0       0.24      0.22      0.23       192
           1       0.19      0.19      0.19       167
           2       0.28      0.28      0.28       221
           3       0.26      0.38      0.31       213
           4       0.31      0.24      0.27       212
           5       0.32      0.33      0.33       203
           6       0.24      0.19      0.21       195

    accuracy                           0.26      1403
   macro avg       0.26      0.26      0.26      1403
weighted avg       0.27      0.26      0.26      1403

Error: 1.8581922051412463
```

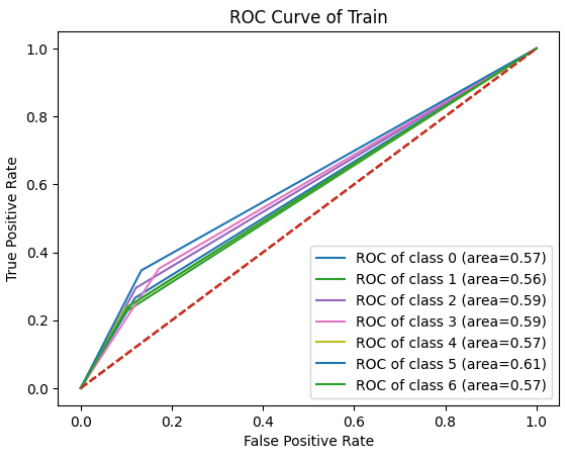
نمودارهای ROC رسم شده به صورت زیر هستند:

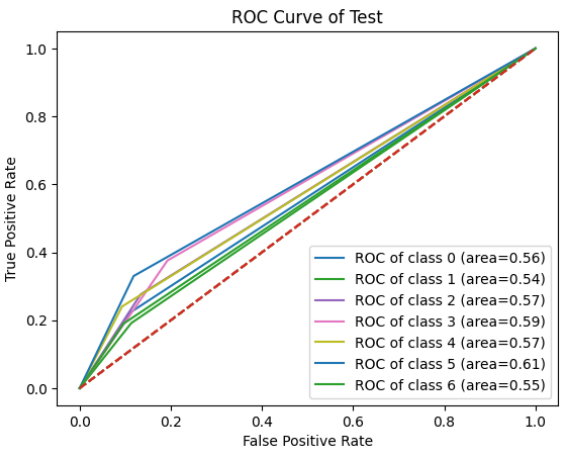
- **Feature Set 1**





- **Feature Set 2**

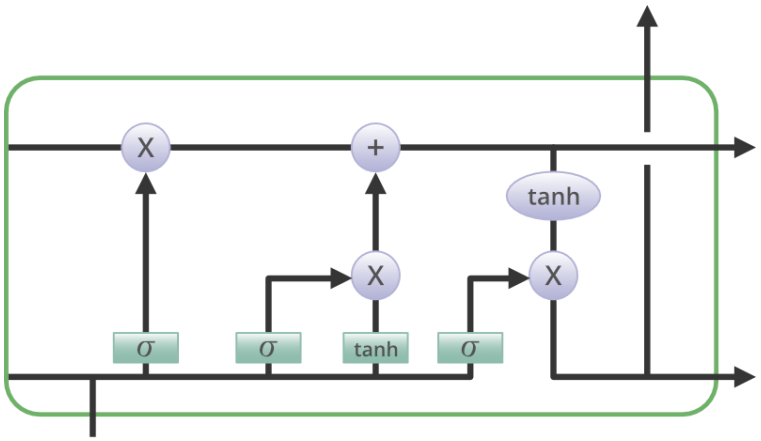




همان‌طور که مشاهده می‌شود دقت این طبقه‌بند از سایر الگوریتم‌های استفاده شده کمتر است زیرا سطح تصمیم‌گیری در logistic regression خطی است و فرض می‌کند بین متغیرهای وابسته و مستقل یک رابطه خطی وجود دارد. همچنین این الگوریتم برای یادگیری روابط پیچیده به داده‌های آموزش بیشتری نیاز دارد در حالی که مدل‌های پیچیده‌تر مانند شبکه‌های عصبی می‌توانند همان روابط را با داده‌های کمتری یاد بگیرند. این مدل نیز مشابه سایر الگوریتم‌های استفاده شده، در مجموعه ویژگی‌های اولی عملکرد بهتری از خود نشان می‌دهد.

۶-۳-۳ Long Short-Term Memory

LSTM نوعی شبکه عصبی RNN است. شبکه‌ی عصبی RNN نوعی شبکه عصبی است که حافظه‌ی داخلی دارد؛ به‌عبارت دیگر، این شبکه یک شبکه‌ی عصبی معمولی است که در ساختارش حلقه‌ای دارد که ازطریق آن در هر گام (Step) خروجی گام قبلی، به‌همراه ورودی جدید، به شبکه وارد می‌شود. این حلقه به شبکه کمک می‌کند تا اطلاعات قبلی را در کنار اطلاعات جدید داشته باشد و بتواند براساس این اطلاعات خروجی مدنظر را به ما بدهد. این ویژگی شبکه‌ی RNN این امکان را می‌دهد که بتوانیم با داده‌های ترتیبی (Sequential Data)، مانند متن، صدا و غیره، کار کند. LSTM مشکل حافظه‌ی بلندمدت شبکه‌ی RNN را حل می‌کند. شبکه‌ی LSTM سازوکارهایی داخلی به اسم گیت (Gate) دارد. این گیت‌ها جریان اطلاعات را کنترل می‌کنند؛ همین‌طور مشخص می‌کنند چه داده‌هایی در توالی مهم هستند و باید هم‌چنان حفظ شوند و چه داده‌هایی باید حذف شوند؛ به‌این شکل، شبکه‌ی اطلاعات مهم را در طول زنجیره‌ی توالی عبور می‌دهد تا خروجی مدنظر را داشته باشیم. LSTM یک ساختار زنجیره‌ای دارد که شامل بلوک‌های مموری مختلف به نام سلول (Cell) می‌باشد که اطلاعات در این سلول‌ها نگهداری می‌شوند. به عنوان مثال یک ساختار نمونه به صورت زیر می‌باشد:



در این پروژه ما ابتدا 10 درصد داده آموزش را به عنوان داده validation جدا می‌کنیم، سپس یک مدل Sequential ایجاد می‌کنیم. تعدادی از پارامترهای تاثیرگذار لایه LSTM از کتابخانه keras به شرح زیر است:

- **units:** عدد صحیح مثبت که نشان‌دهنده‌ی ابعاد خروجی است.

- **dropout:** یک عدد اعشاری بین صفر و یک است و نسبتی از تعداد unitهایی است که برای تبدیل خطی ورودی‌ها دراپ[1] می‌شوند.
- **recurrent_dropout:** یک عدد اعشاری بین صفر و یک است و نسبتی از تعداد unitهایی است که برای تبدیل خطی استیت recurrent دراپ می‌شوند.
- **return_sequences:** یک بولین است که مشخص می‌کند آیا آخرین خروجی را در دنباله خروجی برگردانیم یا دنباله کامل.

در این پروژه ما دو لایه LSTM با پارامترهای زیر را به مدل اضافه می‌کنیم:

- لایه اول:
  - **units:** 128
  - **dropout:** 0.1
  - **recurrent_dropout:** 0.35
  - **return_sequences:** True
- لایه دوم:
  - **units:** 64
  - **dropout:** 0.1
  - **recurrent_dropout:** 0.35
  - **return_sequences:** False

سپس یک لایه آخر Dense را به مدل اضافه می‌کنیم. تعدادی از پارامترهای تاثیرگذار لایه Dense از کتابخانه keras به شرح زیر است:

- **units:** عدد صحیح مثبت که نشان‌دهنده‌ی ابعاد خروجی است. در این پروژه چون 7 کلاس داریم، این پارامتر را به 7 ست می‌کنیم.
- **activation:** نوع تابع activation است که در این پروژه از softmax استفاده می‌کنیم تا احتمال تعلق به هر کلاس محاسبه شود.

در نهایت از Adam optimizer برای کامپایل‌کردن مدل استفاده می‌کنیم. همچنین سایز batch را 64 و تعداد ایپاک‌ها را 80 در نظر می‌گیریم.

معیارهای accuracy، precision، recall، F1-score و error برای داده آموزش و آزمون مدل فیت‌شده به شرح زیر است:

[1] drop

- **Feature Set 1**

```
-----------------------------Train-----------------------------

              precision    recall  f1-score   support

           0       0.78      0.73      0.75       598
           1       0.78      0.79      0.79       562
           2       0.76      0.74      0.75       633
           3       0.76      0.70      0.73       651
           4       0.81      0.62      0.70       513
           5       0.68      0.77      0.72       623
           6       0.67      0.81      0.73       629

    accuracy                           0.74      4209
   macro avg       0.75      0.74      0.74      4209
weighted avg       0.75      0.74      0.74      4209
```

```
-----------------------------Test-----------------------------

              precision    recall  f1-score   support

           0       0.53      0.50      0.51       192
           1       0.60      0.69      0.64       167
           2       0.67      0.63      0.65       221
           3       0.57      0.55      0.56       213
           4       0.69      0.43      0.53       212
           5       0.46      0.57      0.51       203
           6       0.52      0.61      0.56       195

    accuracy                           0.57      1403
   macro avg       0.58      0.57      0.57      1403
weighted avg       0.58      0.57      0.56      1403
```

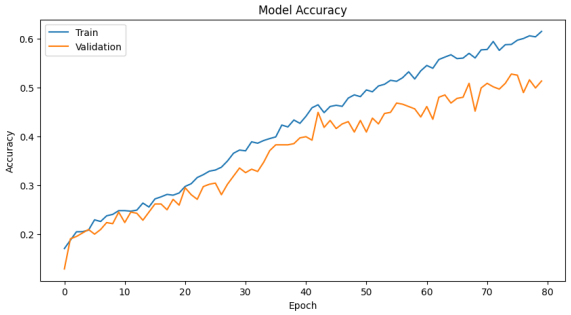
- **Feature Set 2**

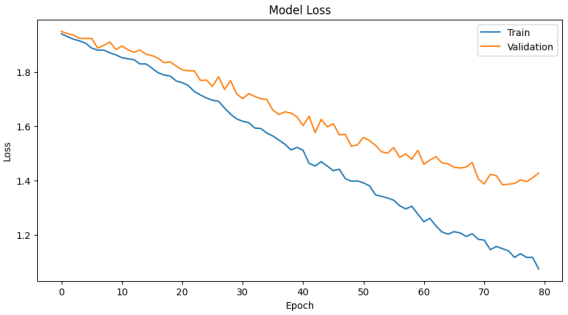
```
-----------------------------Train-----------------------------

              precision    recall  f1-score   support

           0       0.78      0.78      0.78       598
           1       0.75      0.78      0.77       562
           2       0.76      0.78      0.77       633
           3       0.73      0.72      0.73       651
           4       0.81      0.70      0.75       513
           5       0.76      0.78      0.77       623
           6       0.77      0.80      0.79       629

    accuracy                           0.76      4209
   macro avg       0.77      0.76      0.76      4209
weighted avg       0.76      0.76      0.76      4209
```

```
-----------------------------Test-----------------------------

              precision    recall  f1-score   support

           0       0.55      0.56      0.55       192
           1       0.59      0.67      0.63       167
           2       0.65      0.65      0.65       221
           3       0.55      0.58      0.56       213
           4       0.66      0.44      0.53       212
           5       0.51      0.58      0.54       203
           6       0.58      0.58      0.58       195

    accuracy                           0.58      1403
   macro avg       0.58      0.58      0.58      1403
weighted avg       0.58      0.58      0.58      1403
```

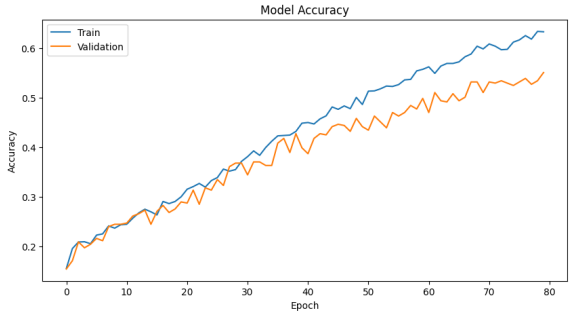
نمودارهای accuracy و loss بر اساس تعداد ایپاک به صورت زیر هستند:

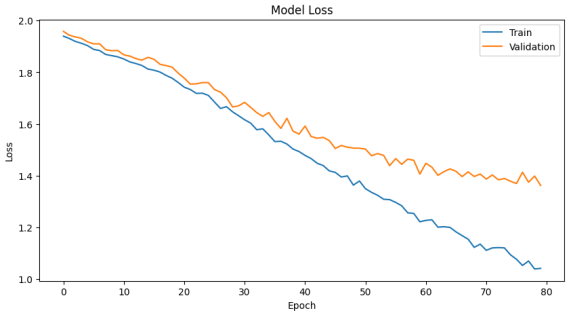
- **Feature Set 1**





- **Feature Set 2**





همان‌گونه که مشاهده می‌شود این روش طبقه‌بندی دقت خوبی در مجموعه train به ما می‌دهد اما دقت نسبتا پایینی در مجموعه test به ما می‌دهد. از طرف دیگر این مدل با افزایش تعداد ایپاک‌ها دچار overfitting شده است، یعنی در تعداد ایپاک بالا مجموعه train دقت بالا و loss پایین و مجموعه test، دقت پایین و loss بالا می‌گیرد. این مدل در مجموعه ویژگی‌های دومی کمی عملکرد بهتری از خود نشان می‌دهد و دقت کمی بالاتر در مجموعه test و train به ما می‌دهد.

۴-۳ کاهش ابعاد بردار ویژگی

کاهش ابعاد در فایل‌های صوتی به فرآیند کاهش تعداد ویژگی‌هایی که سیگنال صوتی را نشان می‌دهند اشاره دارد. هدف کاهش ابعاد حفظ مهمترین اطلاعات در سیگنال صوتی و در عین حال دور انداختن اطلاعات اضافی یا نامربوط است. این می تواند برای چندین هدف مفید باشد، مانند:

- **فشرده سازی داده ها:** با کاهش ابعاد سیگنال صوتی، ذخیره سازی داده های صوتی در اندازه فایل کوچکتر امکان پذیر می شود که برای ذخیره سازی و انتقال مفید است.
- **تجزیه و تحلیل داده ها:** کاهش ابعاد یک سیگنال صوتی، تجسم و درک داده ها را آسان تر می کند، که می تواند برای تجزیه و تحلیل سیگنال صوتی در برنامه هایی مانند تشخیص گفتار و طبقه بندی موسیقی مفید باشد.
- **کاهش نویز:** با حذف اطلاعات اضافی یا نامربوط از سیگنال صوتی، کاهش تاثیر نویز بر روی سیگنال آسان تر می شود.
- **جلوگیری از overfit:** با کاهش ابعاد سیگنال صوتی، می توان مدل را ساده کرد و خطر overfit را کاهش داد. سپس این نمایش کاهش یافته سیگنال صوتی می تواند به عنوان ورودی یک مدل یادگیری ماشینی استفاده شود، که می تواند با کاهش پیچیدگی مدل به جلوگیری از overfit کمک کند.

متدهایی که برای کاهش ابعاد بردار ویژگی پیاده سازی شده است، عبارتند از:

- **Principal Component Analysis:** یک روش آماری برای کاهش ابعاد و استخراج ویژگی است. یک تکنیک خطی است که داده‌های اصلی را به فضای جدیدی تبدیل می‌کند که در آن مهم‌ترین اطلاعات حفظ می‌شوند و ابعاد

آن کاهش می‌یابد. در زمینه سیگنال های صوتی، از PCA می توان برای کاهش تعداد ویژگی هایی که سیگنال صوتی را نشان می دهد و در عین حال مهم ترین اطلاعات را حفظ می کند، استفاده کرد.

- **Linear Discriminant Analysis:** تجزیه و تحلیل تشخیص خطی (LDA) یک روش آماری است که برای کاهش ابعاد و طبقه بندی استفاده می شود. این یک تکنیک خطی است که مجموعه جدیدی از ویژگی ها را پیدا می کند که به بهترین وجه می تواند بین کلاس های مختلف در داده ها تمایز قائل شود.
- **Forward Selection:** ایده اصلی این متد این است که با مجموعه ای خالی از ویژگی‌ها شروع کنید و به طور مکرر ویژگی‌ها را یک به یک بر اساس اهمیت آماری یا عملکرد پیش بینی آنها اضافه کنید تا زمانی که به تعداد قابل قبولی از ویژگی‌ها برسد. ترتیب اضافه شدن ویژگی‌ها با توانایی آنها در بهبود عملکرد مدل تعیین می شود.
- **Backward Elimination:** ایده اصلی این متد این است که با مجموعه ای کامل از ویژگی ها شروع شود و به طور مکرر ویژگی‌ها را یک به یک بر اساس اهمیت آماری یا عملکرد پیش بینی آنها حذف کنید تا زمانی که به تعداد قابل قبولی از ویژگی ها برسد. ترتیب حذف ویژگی ها با توانایی آنها در بهبود عملکرد مدل تعیین می شود.

متدهای گفته شده را در فایل dimension_reduction.py پیاده‌سازی شده‌اند. در ابتدا از روش‌های گفته شده پیش از پیاده‌سازی طبقه‌بندی‌ها استفاده کردیم اما با کاهش بعد دقت مجموعه کاهش می‌یافت بنابراین تصمیم گرفتیم که از آن‌ها استفاده نکنیم.

۴ خوشه‌بندی

خوشه‌بندی یک روش یادگیری بدون نظارت است که در آن داده‌ها بر اساس ویژگی‌های مشترکشان دسته‌بندی می‌شوند. این دسته‌بندی به گونه‌ای انجام می‌شود که داده‌های داخل هر دسته بیشترین شباهت را به هم داشته باشند و داده‌های داخل دسته‌های مختلف بیشترین تفاوت را داشته باشند. الگوریتم‌های مختلفی برای خوشه‌بندی وجود دارند که تفاوت آن‌ها اغلب در نحوه اندازه‌گیری شباهت و تفاوت داده‌ها و همچنین نحوه تخصیص داده به هر دسته است. این الگوریتم‌ها را می‌توان به چهار گروه اصلی خوشه‌بندی سلسله مراتبی، خوشه‌بندی تفکیکی، خوشه‌بندی بر مبنای چگالی و خوشه‌بندی مبتنی بر مدل تقسیم کرد. در ادامه دو روش خوشه‌بندی سلسله‌مراتبی و تفکیکی که در این پروژه مورد استفاده قرار گرفته‌اند را توضیح می‌دهیم. برای خوشه‌بندی‌ها از مجموعه تمام ویژگی‌ها (Feature Set 1) استفاده شده است زیرا بهترین امتیاز ممکن در این حالت به دست می‌آمد(با استفاده از تابع ارزیابی Fowlkes Mallows).

۴.۱ خوشه بندی تفکیکی (Partitional Clustering)

در این نوع خوشه‌بندی داده‌ها به k گروه تقسیم می‌شوند به طوری‌که هر داده فقط در یک گروه قرار می‌گیرد و هیچ گروهی نمی‌تواند خالی باشد. در این روش‌ها لازم است تعداد کل داده‌ها و تعداد خوشه‌های مد نظر از ابتدا مشخص باشد. الگوریتم خوشه‌بندی بر اساس این دو مقدار، از یک دسته‌بندی تصادفی شروع به کار می‌کند و در هر مرحله تلاش می‌کند با جابه‌جایی داده‌ها بین گروه‌ها، دقت خوشه‌بندی را افزایش دهد. بهبود دقت خوشه‌بند را می‌توان معادل با کمینه‌سازی یک تابع هزینه دانست. با تعریف تابع هزینه‌های مختلف می‌توان به الگوریتم‌های مختلفی مانند k-medoids ، k-means و k-modes رسید.

۴.۱.۱ الگوریتم k-means

در این الگوریتم تابع هزینه بر اساس میانگین فاصله اقلیدسی نقاط هر گروه با مرکز آن گروه مشخص می‌شود. که مرکز هر گروه نیز برابر با میانگین داده‌های آن گروه است.

$$E = \sum_{j=1}^{k} \sum_{x \in c_j} dist(x, \mu_{c_j})$$

کمینه کردن این تابع هزینه باعث می‌شود فاصله بین داده‌های داخل یک گروه حداقل و فاصله بین گروه‌های مختلف حداکثر شود. نحوه کار الگوریتم k-means به این شکل است که ابتدا k نقطه را به طور تصادفی به عنوان مرکز خوشه‌ها انتخاب می‌کند سپس برای هر داده فاصله آن را تا تمام مراکز حساب می‌کند و نهایتا داده در خوشه با کمترین فاصله قرار می‌گیرد. بعد از این‌که تمام داده‌ها در k خوشه قرار گرفتند، میانگین جدید هر خوشه را حساب کرده و آن را به عنوان مرکز خوشه در نظر می‌گیرد و این مراحل را تا زمانی تکرار می‌کند که هیچ تغییری در خوشه‌ها اتفاق نیفتد یا تابع هزینه از مقدار مشخصی کمتر شود.

الگوریتم k-means بسیار سریع است زیرا پیچیدگی زمانی خطی دارد اما این روش حساسیت زیادی به داده‌های نویز و دور از مرکز دارد و معمولا نمی‌تواند خوشه‌هایی با اشکال غیرمحدب را به خوبی تفکیک کند.

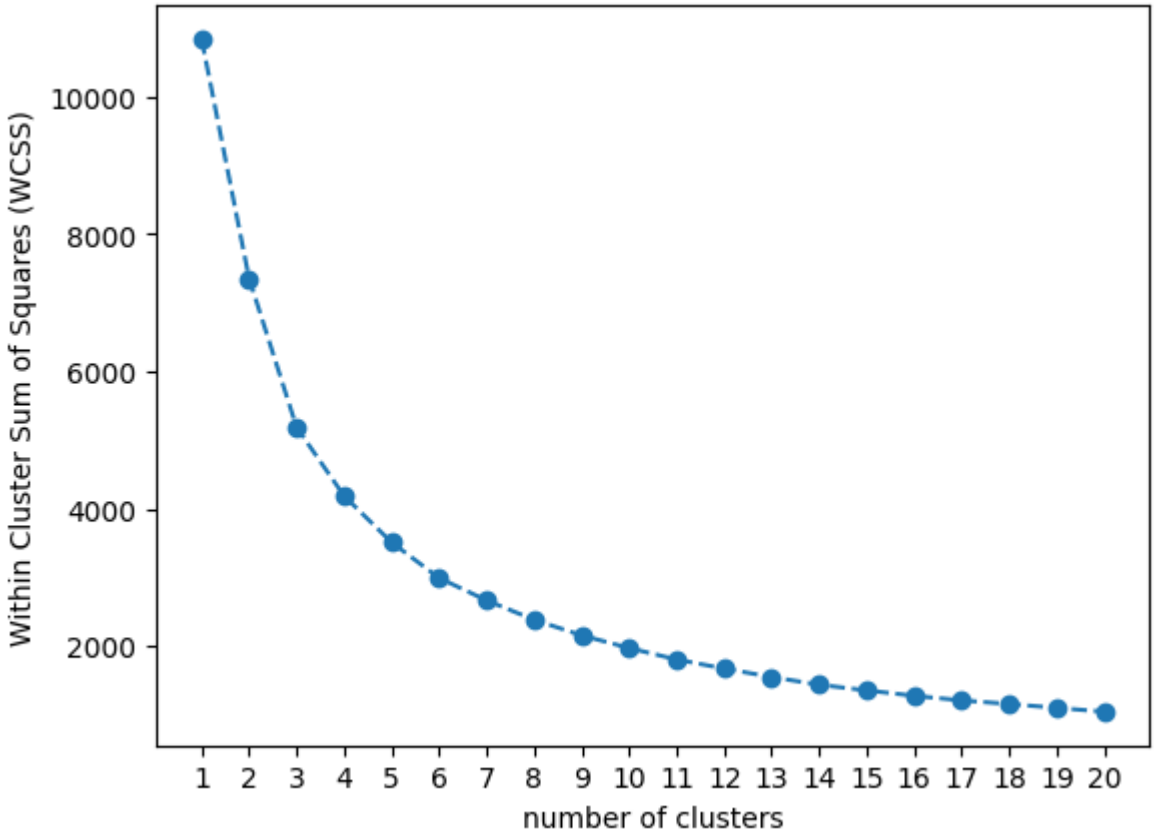
در این پروژه ما برای پیاده‌سازی الگوریتم k-means از KMeans موجود در کتابخانه sklearn استفاده می‌کنیم. تعدادی از پارامترهای تاثیرگذار در این مدل به شرح زیر است:

- **N_clusters:** این پارامتر تعداد خوشه‌ها را مشخص می‌کند. مقدار این پارامتر تاثیر زیادی روی عملکرد الگوریتم دارد. در این پروژه ما از روش elbow برای مشخص کردن تعداد بهینه خوشه‌ها استفاده کردیم و در نهایت نتایج خوشه بندی را به ازای مقادیر مختلف این پارامتر بررسی کردیم.
- **Init:** این پارامتر نحوه انتخاب مراکز خوشه اولیه را مشخص می‌کند که به صورت پیش‌فرض برابر 'random' است یعنی مراکز خوشه ها به طور تصادفی از بین داده‌ها انتخاب می‌شوند. همان‌طور که گفته شد این الگوریتم وابستگی زیادی به انتخاب مراکز اولیه دارد و اگر این مراکز به درستی انتخاب نشده باشند ممکن است باعث شود الگوریتم به‌جای بهینه سراسری، یک بهینه محلی را پیدا کند. ما برای حل این مشکل و همچنین برای افزایش سرعت همگرایی الگوریتم این پارامتر را برابر 'k-means++' قرار دادیم. در روش 'k-means++' اولین مرکز خوشه به طور تصادفی انتخاب می‌شود و k-1 مرکز دیگر به نحوی انتخاب می‌شوند که داده‌هایی با فاصله بیشتر از مراکز قبلی، احتمال انتخاب بیشتری داشته باشند بدین ترتیب احتمال انتخاب مراکز خوشه نزدیک به هم کاهش پیدا می‌کند.
- **Max_iter:** این پارامتر تعداد دفعات اجرای الگوریتم را مشخص می‌کند. اگر مقدار آن خیلی کوچک باشد باعث می‌شود الگوریتم قبل از رسیدن به همگرایی متوقف شود و دقت خوشه‌بندی پایین بیاید. ما مقدار این پارامتر را ۱۰۰ قرار دادیم.
- **Algorithm:** پیدا کردن بهترین موقعیت خوشه‌ها بر اساس بهینه سازی تابع هزینه مشخص شده از لحاظ محاسباتی بسیار پرهزینه است به همین دلیل معمولا از الگوریتم‌های تکرار شونده برای پیدا کردن نقطه بهینه استفاده می‌شود. یکی از این الگوریتم ها elkan است که هنگام اختصاص داده‌ها به خوشه ها از قضیه نامساوی مثلث استفاده می‌کند تا محاسبات کمتری انجام دهد.

مشخص کردن تعداد خوشه‌ها:

در الگوریتم k-means هر چه مقدار k بیشتر باشد، خوشه‌ها کوچک‌تر می‌شوند و میزان تفاوت خوشه‌های همسایه از یکدیگر کم‌تر می‌شود. به عبارت دیگر مدل overfit می‌شود. از طرفی اگر مقدار k خیلی کم باشد باعث می‌شود داده‌های متفاوت در یک خوشه قرار بگیرند و مدل دچار undefitting شود.

برای پیدا کردن تعداد بهینه خوشه‌ها، الگوریتم را به ازای k های مختلف اجرا می‌کنیم و به ازای هر k مجموع مجذور فاصله داده‌ها(WSS) با نزدیک‌ترین مرکز خوشه را حساب می‌کنیم. این عدد فشردگی خوشه‌ها را نشان می‌دهد و با افزایش تعداد خوشه‌ها کم می‌شود. نمودار زیر مجموع فواصل درون خوشه‌ها را بر اساس تعداد خوشه‌ها نشان می‌دهد. نقطه‌ای که سرعت کاهش WSS از آن به بعد کم شده‌است معمولا مقدار مناسبی برای تعداد خوشه‌هاست. با توجه به نمودار زیر به نظر می‌رسد k=6 تعداد خوشه‌های بهینه برای این مسئله باشد.

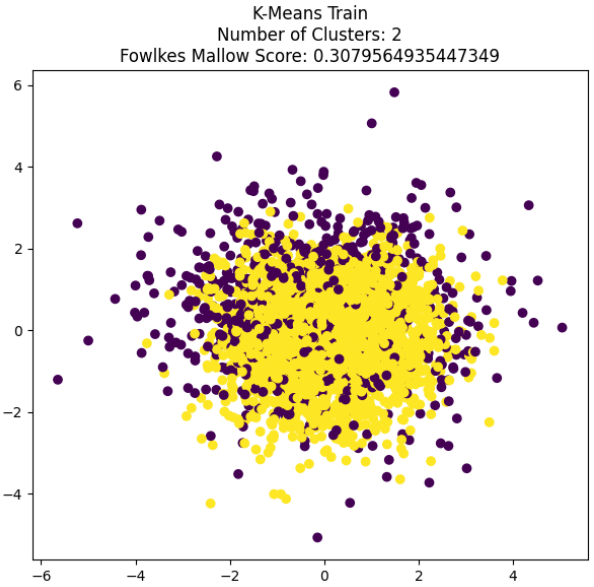


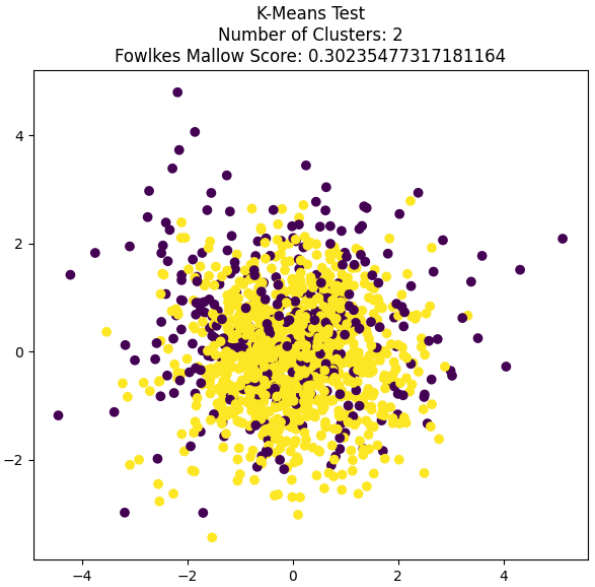
ارزیابی عملکرد خوشه‌بندی

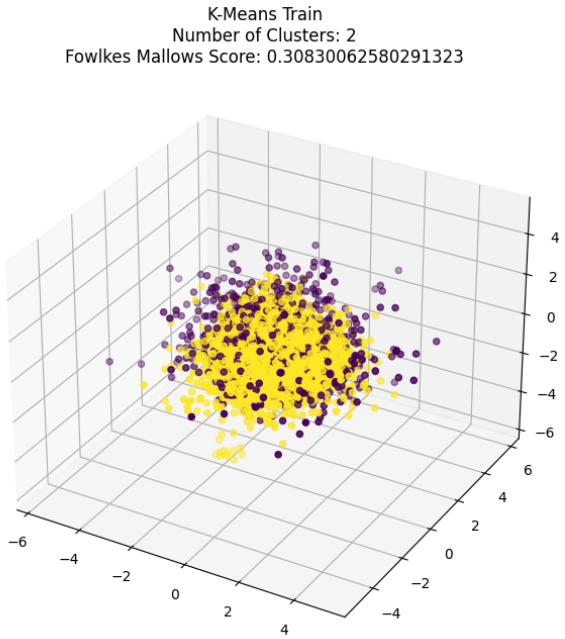
در این پروژه برای ارزیابی خوشه‌بندی از معیار Fowlkes Mallows استفاده کردیم. این معیار یک روش ارزیابی خارجی است که برای بررسی میزان شباهت میان دو خوشه‌بندی استفاده می‌شود. همچنین می‌توان از آن برای ارزیابی یک خوشه‌بندی با استفاده از لیبل‌های از قبل داده شده استفاده کرد. این روش بر اساس بررسی نحوه خوشه‌بندی جفت داده‌ها عمل می‌کند.

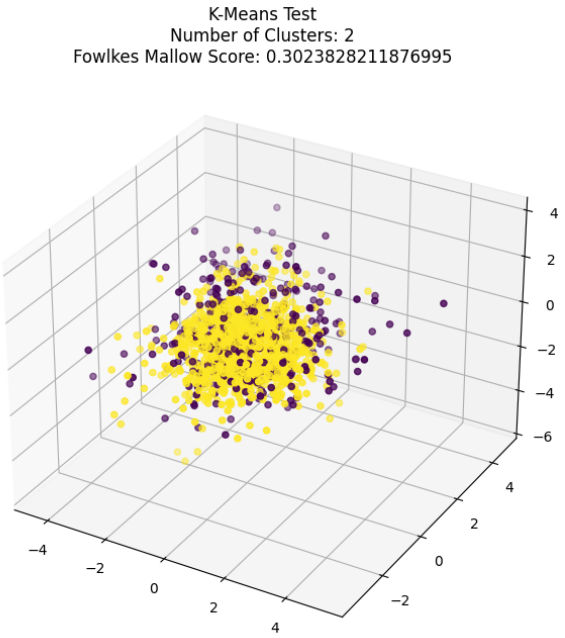
نتایج خوشه‌بندی k-means

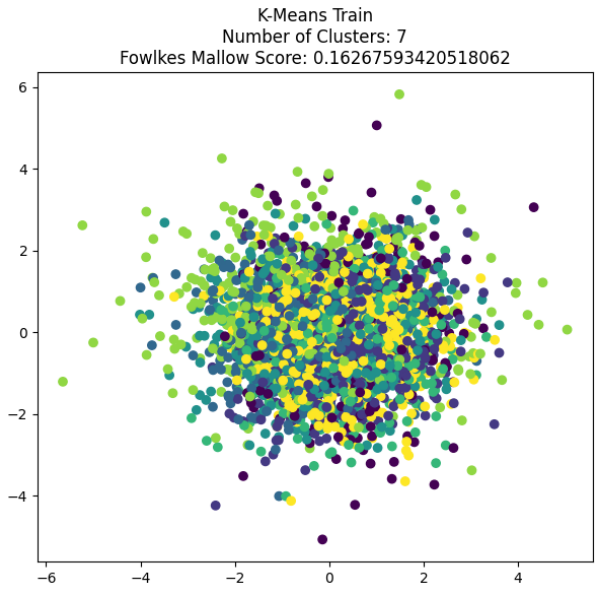
در این پروژه خوشه‌بندی k-means به ازای سه مقدار ۲و۷و۲۰ برای k یک بار بر اساس ۲ مولفه LDA و یک بار بر اساس ۳ مولفه LDA رسم شده‌اند. در شکل‌های زیر نتایج خوشه‌بندی قابل مشاهده است.

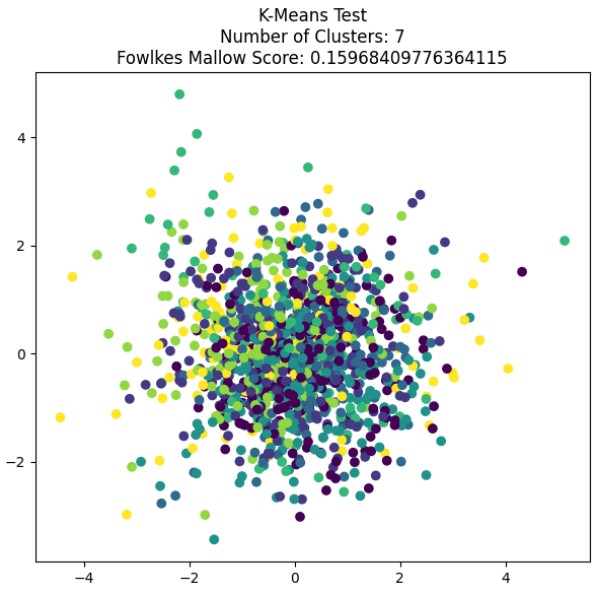


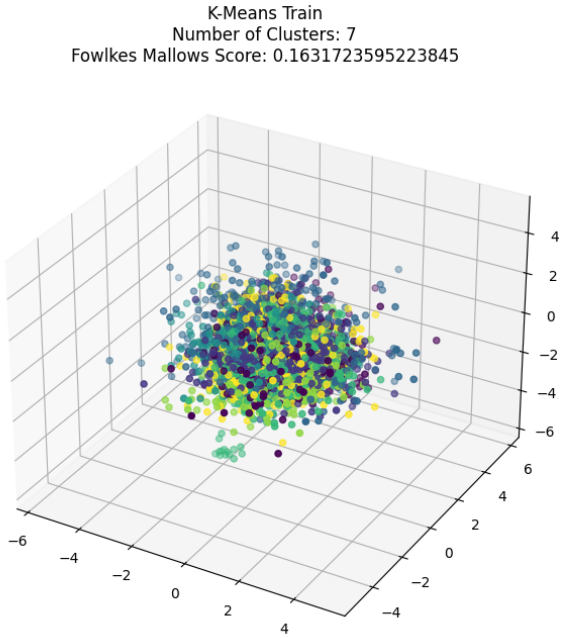


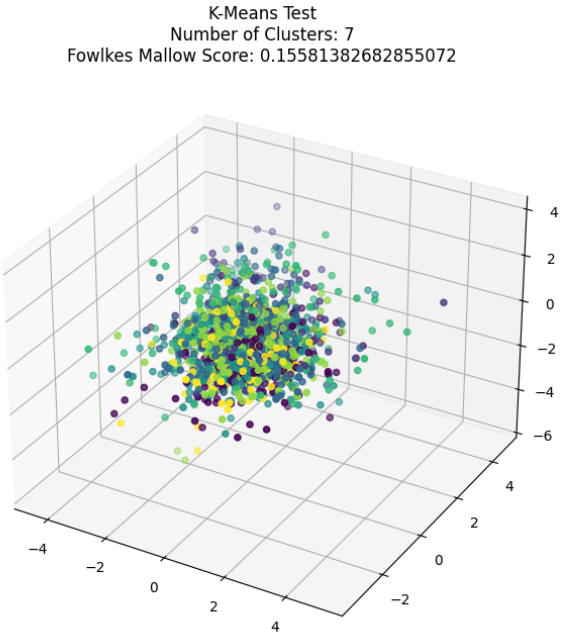


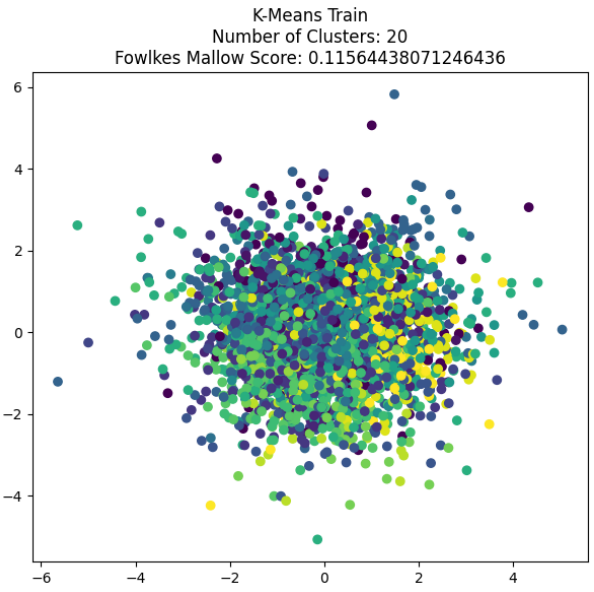


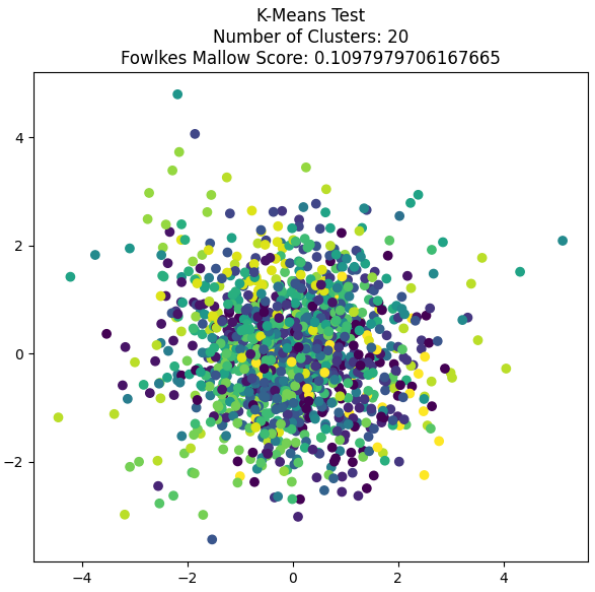

K-Means Train
Number of Clusters: 7
Fowlkes Mallow Score: 0.16267593420518062

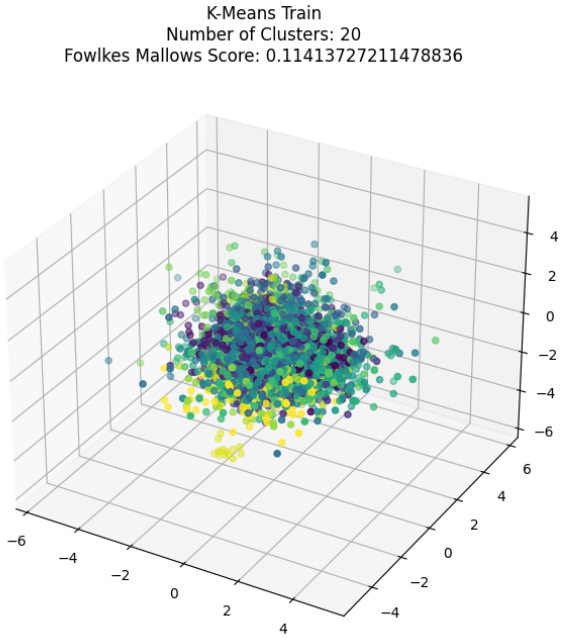

K-Means Test
Number of Clusters: 7
Fowlkes Mallow Score: 0.15968409776364115

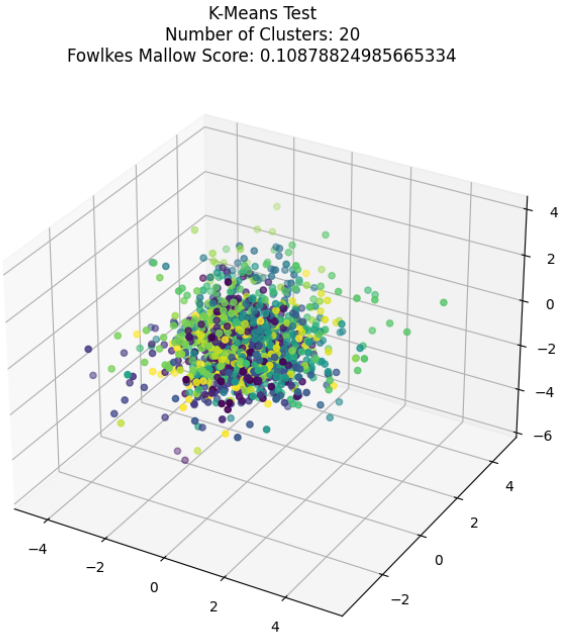

K-Means Train
Number of Clusters: 7
Fowlkes Mallows Score: 0.1631723595223845


K-Means Test
Number of Clusters: 7
Fowlkes Mallow Score: 0.15581382682855072









همان‌طور که در شکل‌ها مشاهده می‌شود در حالت اول که تنها از دو خوشه برای داده‌ها استفاده کرده‌ایم به علت نوع عملکرد تابع ارزیابی Fowlkes Mallows، که بر اساس جفت ویژگی‌ها که در کلاس یکسان قرار می‌گیرند عمل می‌کند، امتیاز بالاتری را به نسبت سایر حالت‌ها کسب کرده است زیرا جفت‌هایی که در حالت واقعی لیبل یکسان دارند و در خوشه یکسان نیز قرار گرفته‌اند بیشترند. در حالت بعدی نیز مشاهده می‌کنیم که با وجود ۷ خوشه امتیاز خوبی کسب نخواهیم کرد که نشان می‌دهد که داده‌ها به خوبی از یکدیگر جدا پذیر نمی‌باشند. در حالت سوم نیز که از ۲۰ خوشه استفاده شده است، امتیاز کمی از حالت قبل

پایین‌تر می‌آید. همچنین همان‌طور که در نمودارها مشاهده می‌شود با وجود استفاده از LDAها برای نمایش خوشه به خوبی دیده نمی‌شوند که به علت پیچیدگی بسیار زیاد داده‌ها می‌باشد که به تعداد زیادی ویژگی برای جداسازی خوشه‌های آن نیاز است.

۴.۲ خوشه بندی سلسه‌مراتبی (Hierarchical Clustering)

در این نوع خوشه‌بندی هدف اصلی ساخت یک سلسه‌مراتب از خوشه‌ها می‌باشد که هر سلسله‌مراتب خود یک دسته‌بندی از داده‌ها است. در این روش هر داده می‌تواند در بیش از یک خوشه قرار بگیرد و هر خوشه ممکن است زیر مجموعه یک خوشه دیگر در سطح بالاتر باشد و در هر مرحله دو خوشه‌ای که کمترین فاصله را دارند با هم ادغام می‌شوند و خوشه جدیدی را ایجاد می‌کنند. مزیت مهم این نوع خوشه‌بندی نسبت به خوشه‌بندی تفکیکی این است که در این روش نیازی به مشخص کردن تعداد خوشه‌ها نیست. الگوریتم‌های خوشه‌بندی سلسله مراتبی بر اساس جهت حرکت برای ایجاد خوشه‌ها، به دو دسته بالا به پایین(Divisible) و پایین به بالا (Agglomerative) تقسیم می‌شوند.

۴.۲.۱ الگوریتم Agglomerative

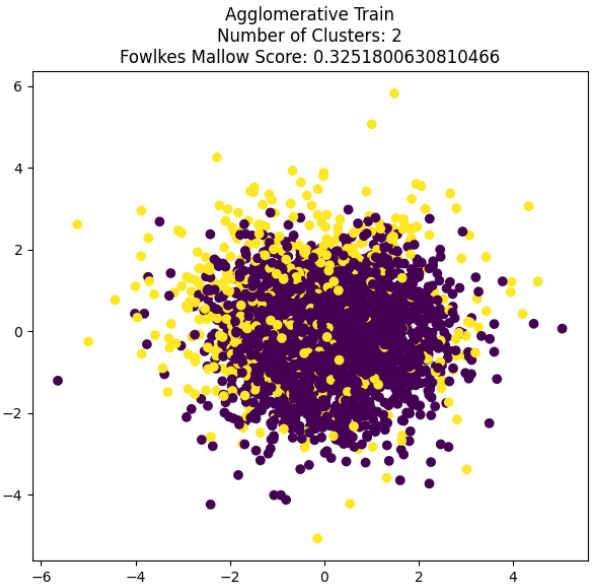
در این روش ابتدا هر داده به عنوان یک خوشه در نظر گرفته می‌شود و در یک فرآیند تکرار شونده در هر مرحله خوشه‌هایی که بیشترین شباهت را دارند با هم ترکیب می‌شوند.

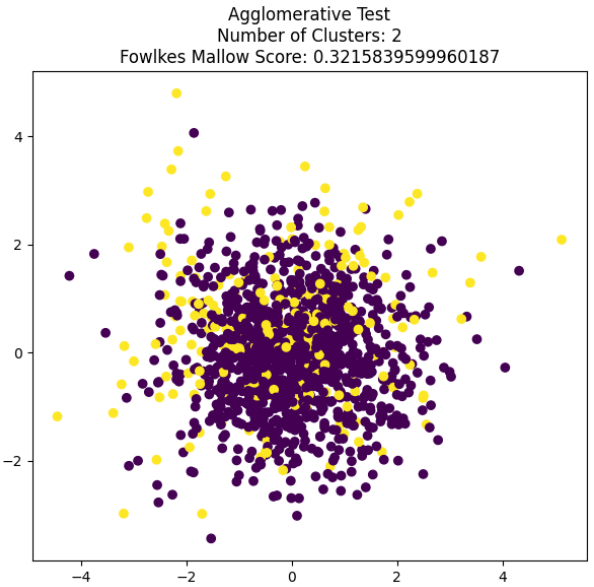
ما برای پیاده‌سازی این الگوریتم از AgglomerativeClustering موجود در کتابخانه sklearn استفاده می‌کنیم. تعدادی از پارامترهای تاثیرگذار در این مدل به شرح زیر است:

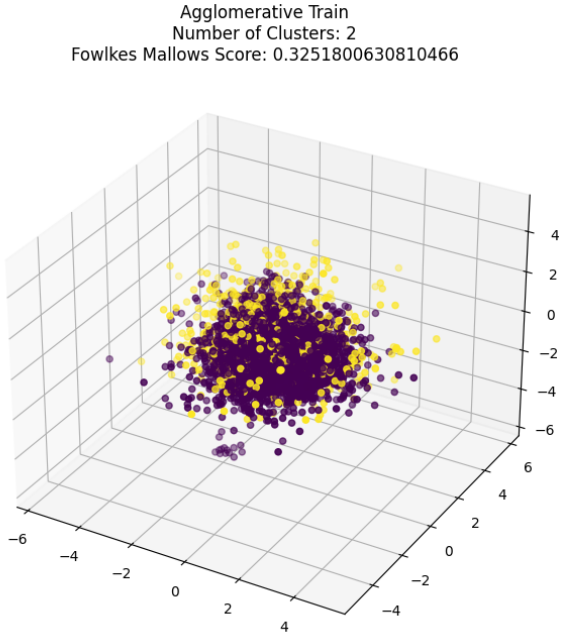
- **N_clusters:** همان طور که گفته شد در این الگوریتم نیازی نیست تعداد خوشه‌ها از ابتدا مشخص باشد اما از این پارامتر می‌توان برای تعیین شرط خاتمه الگوریتم استفاده کرد.
- **Linkage:** برای محاسبه شباهت بین خوشه‌ها از معیارهای مختلفی از جمله single-link، average-link و complete-link می‌توان استفاده کرد. در روش single-link شباهت دو خوشه کمترین فاصله بین دو عضو آن‌ها در نظر گرفته می‌شود. روش complete-link دقیقا برعکس روش قبلی است و شباهت را برابر با فاصله دورترین عضوها در نظر می‌گیرد. روش average-link میانگین فاصله تمام اعضای دو خوشه را به عنوان شباهت آن‌ها در نظر می‌گیرد این روش به محاسبات بیشتری نیاز دارد اما در مقایسه با دو روش قبلی حساسیت کمتری به نویز دارد. معیار دیگری که ما نیز از آن استفاده کردیم معیار ward است. در این روش برای هر دو خوشه ضریب ward محاسبه می‌شود که برابر است با مجذور انحراف مقادیر داده‌های درون خوشه‌ها از از میانگین داده‌ها. سپس دو خوشه‌ای که بر این اساس کمترین فاصله را دارند با هم ترکیب می شوند. معیار ward در مقایسه با سایر معیارها حساسیت کمتری به داده‌های نویز دارد.

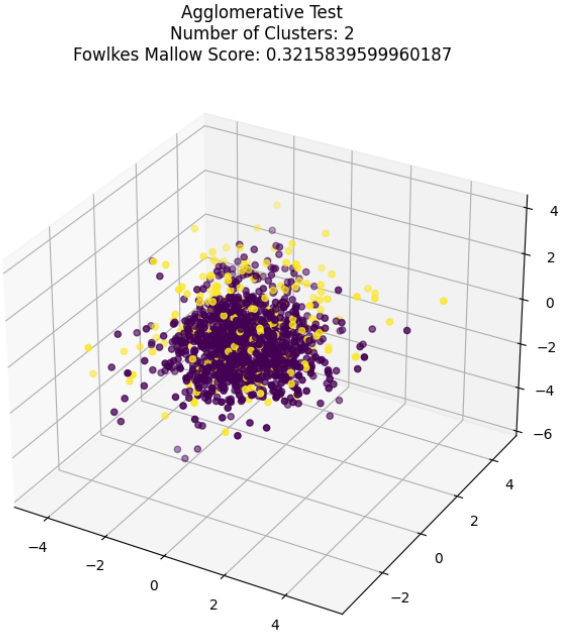
نتایج خوشه بندی Agglomerative

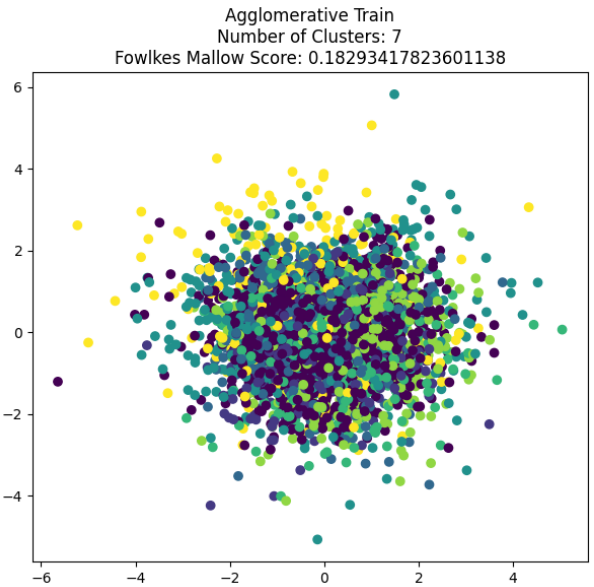
خوشه‌بندی Agglomerative به ازای تعداد خوشه‌های ۲و۷و۲۰ یک بار بر اساس ۲ مولفه LDA و یک بار بر اساس ۳ مولفه LDA رسم شده‌اند. در شکل‌های زیر نتایج خوشه‌بندی قابل مشاهده است.

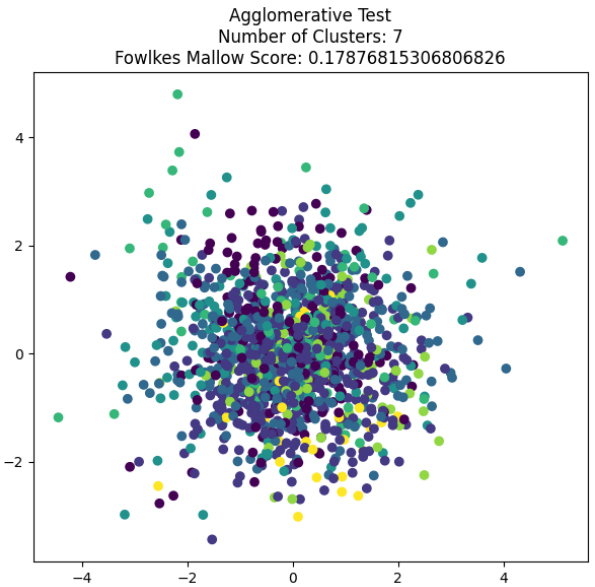


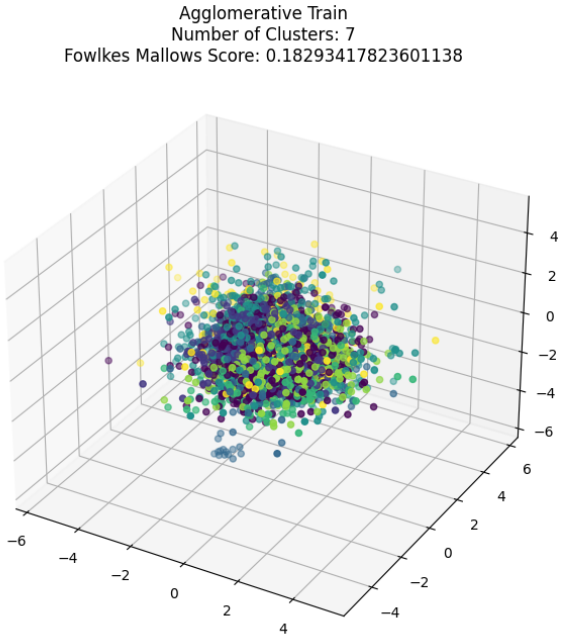


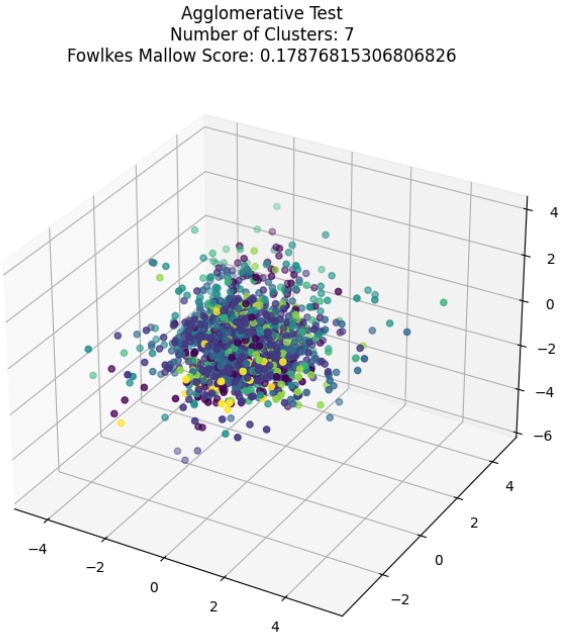


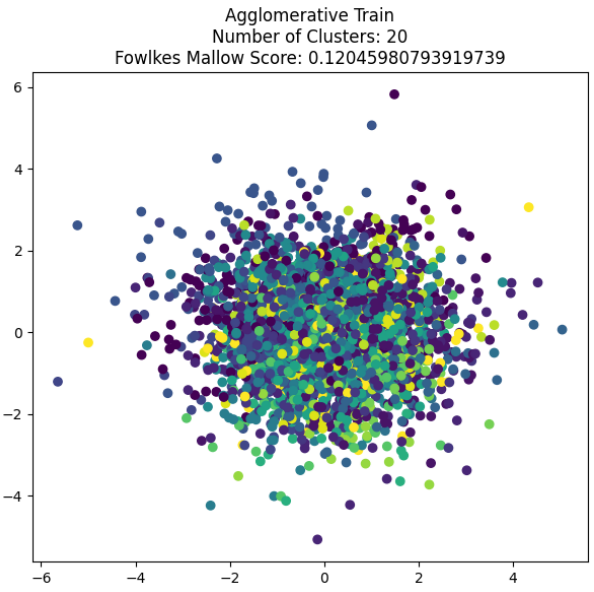


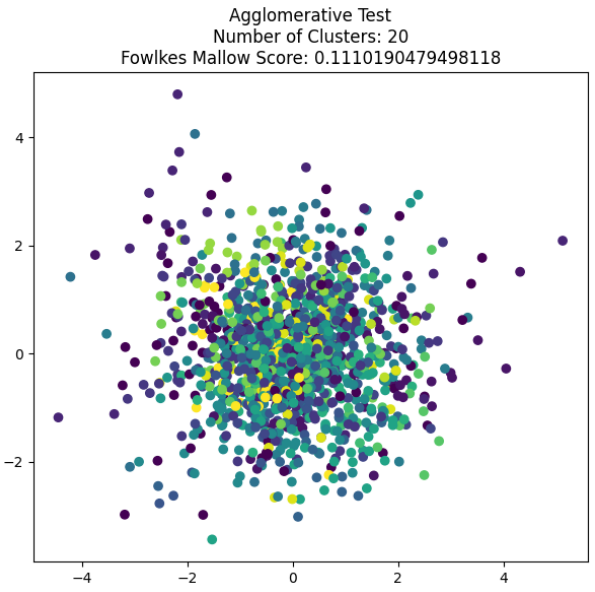

Agglomerative Train
Number of Clusters: 7
Fowlkes Mallow Score: 0.18293417823601138

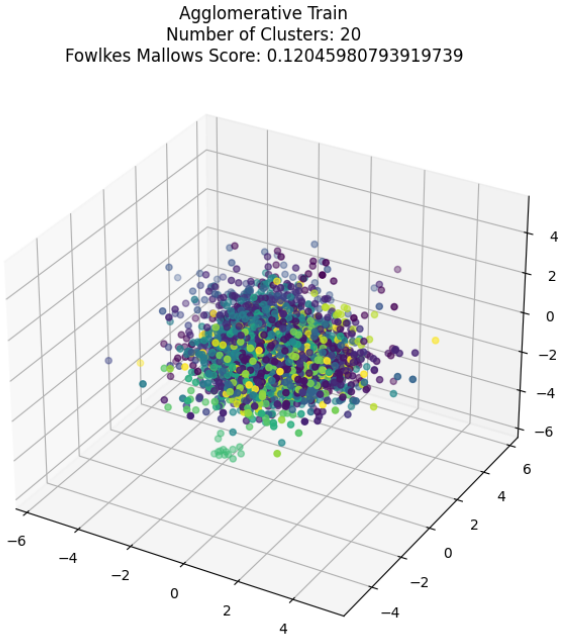

Agglomerative Test
Number of Clusters: 7
Fowlkes Mallow Score: 0.17876815306806826

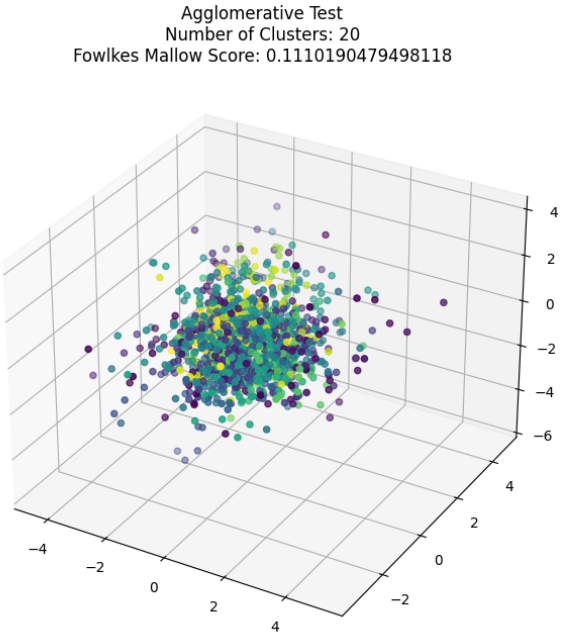

Agglomerative Train
Number of Clusters: 7
Fowlkes Mallows Score: 0.18293417823601138


Agglomerative Test
Number of Clusters: 7
Fowlkes Mallow Score: 0.17876815306806826



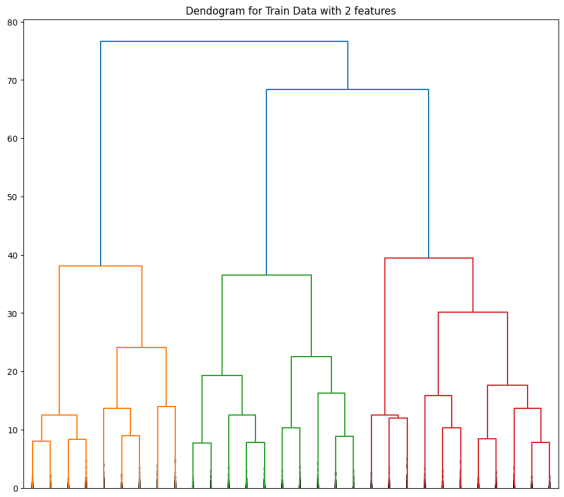


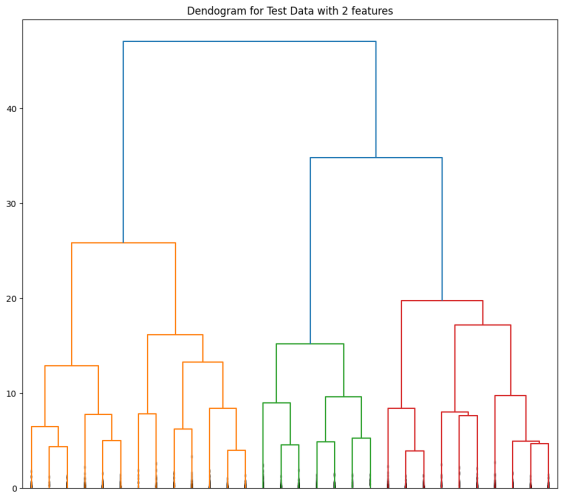


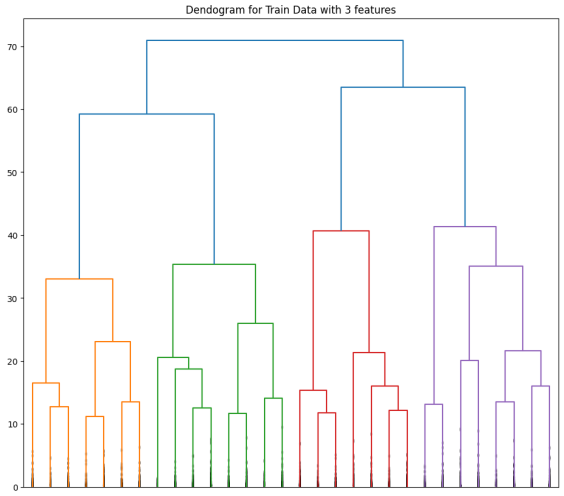


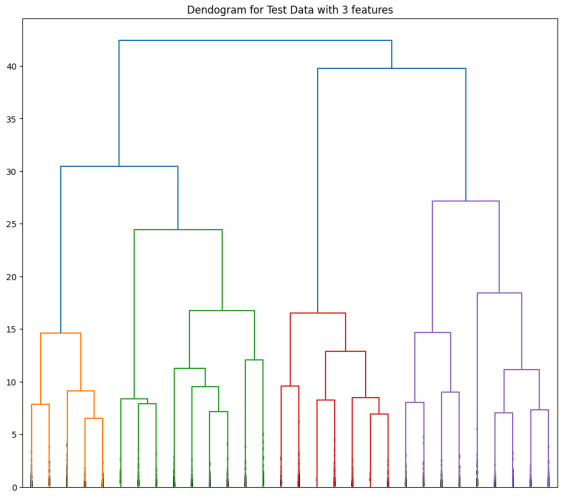
در این روش نیز به همان علتی که در روش قبل گفته شد، امتیاز خوشه‌بندی با ۲ دسته بالاتر از حالت ۷ دسته‌ای است. در حالت ۷ دسته‌ای این روش عملکرد بهتری را نسبت به روش قبل دارد و بهتر می‌تواند داده‌ها را از یکدیگر جدا کند.

در شکل زیر نمودار dendrogram در حالتی که از دو ویژگی و سه ویژگی استفاده شده قابل مشاهده است. همان طور که در تصویر مشاهده می‌شود با افزایش تعداد ابعاد داده‌ها شکل خوشه‌ها تغییر می‌کند و ممکن است داده‌های که در فضای دو بعدی نزدیک هم بودند و در یک خوشه قرار می‌گرفتند در بعدهای بالاتر در خوشه‌های متفاوتی قرار بگیرند.









۵ جمع‌بندی

طبقه بندی موسیقی‌های سنتی ایرانی براساس دستگاه‌های آن‌ها، کار پیچیده‌ای است که نیازمند رویکردی جامع بر تحلیل داده‌ها می‌باشد. در این پروژه رویکردی جامع برای طبقه‌بندی موسیقی سنتی ایرانی اتخاذ شد که شامل ترکیبی از تکنیک‌های پاکسازی و پیش‌پردازش داده‌ها، الگوریتم‌های یادگیری ماشین، روش‌های کاهش ابعاد و تکنیک‌های خوشه‌بندی بود. هدف توسعه یک طبقه‌بندی قوی و کارآمد بود که بتواند موسیقی‌های سنتی ایرانی را بر اساس ویژگی‌های مختلف موسیقایی، به دستگاه‌های مختلف دسته‌بندی کند.

همانگونه که در بخش‌های قبل مشاهده کردیم، مدل‌های طبقه‌بند SVM، KNN و XGBoost در مقایسه با سایر مدل‌ها دقت بهتری داشتند. همچنین این سه مدل در مجموعه ویژگی‌های اولی عملکرد بهتری از خود نشان می‌دهند. بر خلاف چیزی که انتظار داشتیم مدل LSTM دقت خوبی با استفاده از دو مجموعه ویژگی استفاده‌شده به ما نمی‌دهد. مدل MLP نیز دقت بسیار پایینی دارد و اصلا مدل خوبی برای پیاده‌سازی پروژه ما نمی‌باشد.

از طرفی دیگر خوشه‌بند Agglomerative با ۷ کلاستر امتیاز بالاتری در مقایسه با خوشه‌بند K-Means دارد. البته با این وجود مشاهده می‌شود که امتیاز آن در مقایسه با دقت طبقه‌بندها بسیار پایین‌تر می‌باشد. بنابراین ترجیح ما در این پروژه این است که با داشتن لیبل دستگاه‌ها و طبقه‌بند مناسب بتوانیم مدل خوبی برای پیش‌بینی دستگاه موسیقی یک آهنگ سنتی ایرانی پیاده‌سازی کنیم.