University of Tehran

School of Electrical and Computer Engineering

MACHINE LEARNING

Final Project Report

Sasan Keshavarz
810199253

Mohammad Roudbari
810100364

Reza Souri
810100380

Summer 2022

فهرست

مقدمه

هدف نهایی این پروژه طبقه بندی و خوشه بندی صدا بر اساس احساسات و جنسیت آن است. مراحل مختلف پروژه جمع آوری دادها، تحلیل و پیش پردازش آن ها و طبقه بندی و خوشه بندی است. ابتدا داده‌های مربوط به افراد متخلف با احساسات خواسته شده را جمع آوری کردیم. پس از تمیز سازی داده آن ها را طبقه بندی و خوشه بندی کردیم.

جمع آوری داده

در زمینه جمع آوری داده چالشهایی وجود داشت که ممترین آن این بود که به علت کوتاه بودن عبارات انتقال احساس برای سوژه ها دشوار بود. یکی دیگر از چالش ها وجود نویز در محیط است که سعی شده کاملا محیط آرام باشد که مشکل تمیز سازی داده ها در مراحل بعدی کمتر شود. جنسیت بر دامنه و فرکانس صدا تأثیر می گذارد و چون این ها ویژگی های اصلی تشخیص احساس هستند. به طور کلی صدای زنان زیر تراست و فرکانس بالاتری دارد اما مردها صدای بم تر با فرکانس پایین تری دارند. به طور میانگین دامنه ی صدای مردها بالاتر است. همچنین ویژگی های فردی هم موثر هستند. لحن بیان کردن جملات فارغ از جنسیت و سن، بسته به شرایط اجتماعی فرد و محل زندگی نیز تغییر می کند.

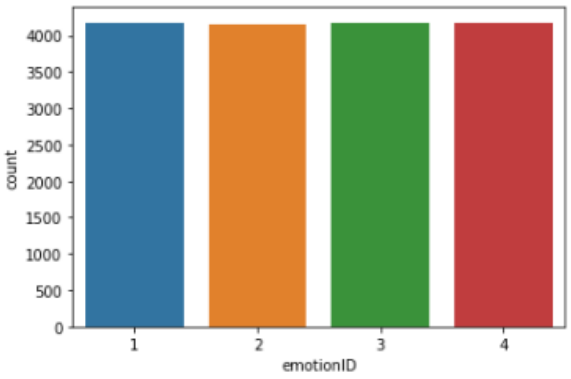
تمیزسازی داده ها

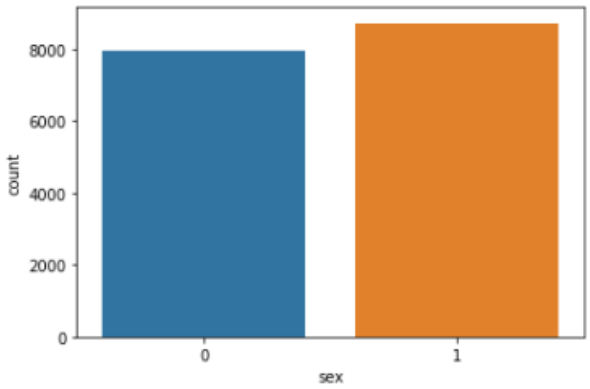
در ابتدای کار داده ها نویزی بودند و صدای ضبط شده فقط شامل صدای سوژه نبود در نتیجه باید داده هارا تمییز کردیم. برای این کار ابتدا صدا های موجود در فایل را تفکیک کردیم و صدا های اضافی را حذف کردیم. همچنین ابتدا و انتهای هر داده که سوژه در آن ساکت است باید بریده شد. بعضی از برچسب داده‌ها تکراری بود و این موارد اصلاح شدند.

در قدم اول تعداد داده های null در هر کلاس را شمارش می کنیم.

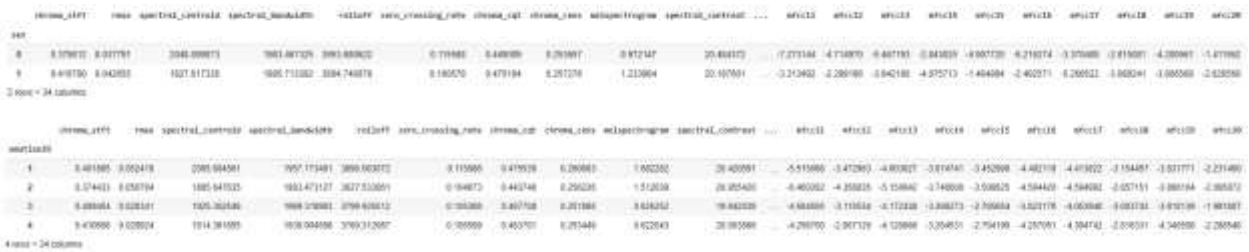
```
chroma_stft          0
rmse                 0
spectral_centroid    0
spectral_bandwidth   0
rolloff              0
zero_crossing_rate   0
chroma_cqt           0
chroma_cens          0
melspectrogram       0
spectral_contrast    0
spectral_flatness    0
poly_features        0
tonnetz              0
tempogram            0
mfcc1                0
mfcc2                0
mfcc3                0
mfcc4                0
mfcc5                0
mfcc6                0
mfcc7                0
mfcc8                0
mfcc9                0
mfcc10               0
mfcc11               0
mfcc12               0
mfcc13               0
mfcc14               0
mfcc15               0
mfcc16               0
mfcc17               0
mfcc18               0
mfcc19               0
mfcc20               0
emotionID            0
textID               0
sex                  0
age                  0
dtype: int64
```

در قدم بعد تعداد داده ها را در هر ستون جنسیت و احساس رسم می کنیم.





در قدم بعد میانگین هر ستون را بر اساس جنسیت و احساس دست می اوریم تا دید کلی نسبت به مقادیر هر ستون داشته باشیم.



در قدم بعد correlation ستون هارا بدست اورده و میزان وابستگی هر feature به برچسب های جنسیت و احساس را بدست می اوریم .

همانطور که در شکل فوق مشخص است ستون جنسیت وابستگی بیشتری به ویژگی های chroma stft ،spectral centroid ،mfcc نسبت به سایر ویژگی ها دارد و همچنین ستون احساسات وابستگی بیشتری به ویژگی هایrmse ، melspectrogram ،poly features ،mfcc دارد .

استخراج ویژگی

هر سیگنال صوتی شامل ویژگی‌های بسیاری است. در نتیجه، باید مشخصه‌هایی در تشخیص جنسیت و عاطفه از صدا مرتبط هستند را استخراج کرد. فرآیند استخراج ویژگی برای استفاده از آن‌ها در تحلیل‌ها، استخراج ویژگی نامیده می‌شود. اکنون برخی از ویژگی‌های سیگنال‌های صوتی همراه با جزئیات مورد بررسی قرار می‌گیرد. در این مرحله سیگنال‌های گفتاری پردازش شده به یک نمایش مختصر اما منطقی که متمایزتر و قابل اعتمادتر از سیگنال واقعی است تبدیل می‌شوند.

پس از انتخاب ویژگی های مورد نظر با قطعه کد زیر از داده های صوتی ویژگی هارا استخراج کرده و از هر کدام میانگین می گیریم و دیتاست را تشکیل می دهیم .

```
header = 'voice_id chroma_stft rmse spectral_centroid spectral_bandwidth rolloff zero_crossing_rate chroma_cqt chroma_cens
        |melspectrogram spectral_contrast spectral_flatness poly_features tonnetz tempogram'
for i in range(1, 21):
    header += f' mfcc{i}'
# header += ' label'
header = header.split()
```

```
file = open('voice_data.csv', 'w', newline='')
with file:
    writer = csv.writer(file)
    writer.writerow(header)

for filename in os.listdir(f'./drive/MyDrive/Voice'):
    songname = f'./drive/MyDrive/Voice/{filename}'
    y0, sr0 = librosa.load(songname, mono=True, duration=30)
    rmse = librosa.feature.rms(y=y0)
    chroma_stft = librosa.feature.chroma_stft(y=y0, sr=sr0)
    spec_cent = librosa.feature.spectral_centroid(y=y0, sr=sr0)
    spec_bw = librosa.feature.spectral_bandwidth(y=y0, sr=sr0)
    rolloff = librosa.feature.spectral_rolloff(y=y0, sr=sr0)
    zcr = librosa.feature.zero_crossing_rate(y0)
    mfcc = librosa.feature.mfcc(y=y0, sr=sr0)

    chroma_cqt = librosa.feature.chroma_cqt(y0, sr0)
    chroma_cens= librosa.feature.chroma_cens(y0, sr0)
    melspectrogram = librosa.feature.melspectrogram(y0, sr0)
    spectral_contrast = librosa.feature.spectral_contrast(y0, sr0)
    spectral_flatness = librosa.feature.spectral_flatness(y0)
    poly_features = librosa.feature.poly_features(y0, sr0)
    tonnetz = librosa.feature.tonnetz(y0, sr0)
    tempogram = librosa.feature.tempogram(y0, sr0)

    to_append = f'{filename} {np.mean(chroma_stft)} {np.mean(rmse)} {np.mean(spec_cent)} {np.mean(spec_bw)} {np.mean(rolloff)} {np.mean(zcr)}
                 {np.mean(chroma_cqt)} {np.mean(chroma_cens)} {np.mean(melspectrogram)} {np.mean(spectral_contrast)} {np.mean(spectral_flatness)}
                 {np.mean(poly_features)} {np.mean(tonnetz)} {np.mean(tempogram)}'
    for e in mfcc:
        to_append += f' {np.mean(e)}'

    file = open('voice_data.csv', 'a', newline='')
    with file:
        writer = csv.writer(file)
        writer.writerow(to_append.split())
```

```
data = pd.read_csv('voice_data.csv')
data.head()
```

در قطعه کد فوق ، در ابتدا نام های ویژگی هایی که استفاده کردیم را درون header قرار دادیم و در ادامه با استفاده ار پکیج librosa ، ویژگی های مورد نظر را از دیتای صوتی استخراج کردیم .

هر کدام از ویژگی ها در خروجی یک بردار به ما می دهد که نیاز است از ان میانگین بگیریم . لازم به ذکر است در ابتدا تمام ویژگی هایی که در بخش spectral features پکیج librosa وجود دارد را از داده ها استخراج کردیم و طبق دستور العمل پروژه داده هارا به دو دسته ی تقسیم کرده و مدل های مختلف را روی ان ها اموزش می دهیم و نتایج بدست امده را تحلیل و مقایسه می کنیم. در ادامه توضیح مختصری برای هر ویژگی ارائه شده است.

Chroma STFT

مقدار Chroma یک صدا اساساً نشان دهنده شدت دوازده کلاس صدای متمایز است که برای مطالعه موسیقی استفاده می شود. آنها را می توان در تمایز پروفیل های کلاس گام بین سیگنال های صوتی به کار برد. در واقع یک توصیفگر است که محتوای آهنگی سیگنال صوتی را به صورت فشرده نشان می دهد. بنابراین ویژگی‌های کروما را می‌توان به عنوان پیش‌نیاز مهم برای تحلیل معنایی سطح بالا، مانند تشخیص آکورد یا تخمین تشابه هارمونیک در نظر گرفت.

Chroma STFT مقدار Chroma صدا اساساً نشان دهنده شدت دوازده کلاس صدای متمایز است که برای مطالعه موسیقی استفاده می شود. آنها را می توان در تمایز پروفیل های کلاس گام بین سیگنال های صوتی استفاده کرد. در نتیجه هم برای تشخیص جنسیت و هم تشخیص عاطفه هم ویژگی مهمی است.

Chroma CQT

یکی دیگر از رویکردهای کروماگرام، تبدیل Chroma Constant-Q است که در آن STFT را به گونه ای تنظیم می کند که به صورت لگاریتمی فاصله بین bin های فرکانس ایجاد کند. با این حال، زمانی که فرکانس را در مقیاس لگاریتمی ترسیم می کنید، اندازه bin ثابت برای همه فرکانس ها منجر به مشکلاتی می شود. به دنبال حل این مشکل با افزایش اندازه بافر برای فرکانس‌های پایین‌تر و کاهش فشار محاسباتی ناشی از آن با کاهش اندازه بافر مورد استفاده برای فرکانس‌های بالا است.

CQT از دو جهت با STFT متفاوت است: ۱) مقیاس فرکانس لگاریتمی است نه خطی. ۲) طول پنجره برای هر فرکانس متفاوت است: فرکانس های پایین از پنجره های بلند استفاده می کنند، فرکانس های بالا از پنجره های کوتاه استفاده می کنند.

Chroma CENS (Chroma Energy Normalized Statistics)

توابع CENS را می توان به طور موثر به دلیل وضوح فضایی کم تفسیر کرد. ویژگی های Chroma فقط بر روی ۱۲ ویژگی صدای زیر و بمی متمرکز هستند که در نت موسیقی غربی آشنا هستند که در آن هر متغیر Chroma نشان می دهد که چگونه شدت صدا در میان ده ها باند Chroma پراکنده می شود. این توابع با در نظر گرفتن شاخصه های آمارهای کوتاه‌مدت بر روی توزیع انرژی در باندهای کروم، ویژگی‌های CENS (آمار نرمال شده انرژی کروم) را که خانواده‌ای از ویژگی‌های صوتی مقیاس‌پذیر و قوی را تشکیل می‌دهند، به دست می‌آورد.

ایده اصلی ویژگی‌های CENS این است که آمارگیری از پنجره‌های بزرگ انحرافات محلی در تمپو، بیان و تزیینات موسیقی مانند تریل‌ها و آکوردهای آرپگیت شده را هموار می‌کند. CENS به بهترین وجه برای کارهایی مانند تطبیق صدا و شباهت استفاده می شود.

Mel spectrogram

طیف‌نگار Mel ترکیبی از مقیاس Mel و طیف‌نگار است که در آن مقیاس Mel تبدیل غیرخطی مقیاس فرکانس را نشان می‌دهد. در این حالت ابتدا سیگنال صوتی به فریم های کوچکتر تقسیم می شود و یک پنجره

Hamming روی هر فریم اعمال می شود. سپس DFT برای تغییر از حوزه زمان به حوزه فرکانس اعمال می شود. یک لگاریتم در مرحله نهایی برای تولید طیف استفاده می شود که به تولید طیف نگار Mel کمک میکند.

طیف‌نگار mel مقادیر بر حسب هرتز را به مقیاس mel تبدیل می‌کند. طیف‌نگار صوتی خطی برای برنامه‌هایی که همه فرکانس‌ها اهمیت یکسانی دارند، ایده‌آل است، در حالی که طیف‌نگارهای mel برای برنامه‌هایی که نیاز به مدل‌سازی درک شنوایی انسان دارند، مناسب‌تر هستند.

طیف نگاری تجسم طیف فرکانس یک سیگنال است، که در آن طیف فرکانس سیگنال، محدوده فرکانسی است که توسط سیگنال موجود است. طیف نگار Mel طیفی است که به مقیاس Mel تبدیل می شود. مطالعات نشان داده است که انسان فرکانس ها را در مقیاس خطی درک نمی کند.

MFCC

Mel frequency cepstral coefficients از یک سیگنال، مجموعه‌ای کوچک از ویژگی‌ها (معمولا حدود ۱۰ الی ۲۰) است که به طور خلاصه شکل کلی یک طیف را توصیف می‌کند. MFCCها با تبدیل کسینوس گسسته (DCT) به یک طیف‌نگار مِل فرکانس محاسبه می‌شوند. این ویژگی مشخصه‌های صدای انسانی را مدل می‌کند. ضرایب مغزی فرکانس مل، نمایش های فشرده ای از طیف هستند که معمولاً برای شناسایی خودکار گفتار استفاده می شوند و همچنین به عنوان یک ویژگی اصلی در بسیاری از زمینه های تحقیقاتی که شامل سیگنال های صوتی می شود، استفاده می شود. همچنین می‌توان مقیاس‌دهی ویژگی‌ها را به گونه‌ای انجام داد که هر «بُعد ضریب (coefficient dimension) «دارای میانگین صفر و واریانس واحد باشد.

می توان آن را با نگاشت سیگنال تبدیل فوریه بر روی مقیاس mel با استفاده از پنجره های مثلثی یا کسینوسی به دست آورد. جایی که پس از گرفتن لاگ توان ها در هر یک از فرکانس های Mel و پس از تبدیل کسینوس گسسته توان های مل log دامنه یک طیف را می دهد. لیست دامنه MFCC است.

Tonnetz (Tonal Centroid Features)

Tonnetz با تشخیص تغییرات هارمونیک در کلیپ های صوتی موسیقی، ویژگی های مرکز تونال را محاسبه می کند. این یک نمایش مسطح شناخته شده از روابط زیر و بمی در یک کلیپ صوتی است که به عنوان یک صفحه بی نهایت در نظر گرفته می شود.همانطور که در قسمت ویژگی chroma گفته شد، یک فایل صوتی می تواند شامل ۱۲ کلاس گام صدا باشد، این پدیده با ادغام برخی از کلاس ها با هم در نظر می گیرد که فایل

صوتی از ۶ کلاس گام است. بنابراین نمودار ویژگی کروما ترسیم شده از این روش زمین ها را فقط در ۶ کلاس طبقه بندی می کند که ما آن را Tonnetz می نامیم.

پروفیل‌های کلاس گام زمانی - که معمولاً به آنها کروماگرام گفته می‌شود - نمایش سیگنال استاندارد بالفعل برای روش‌های مبتنی بر محتوا تحلیل هارمونیک موسیقی هستند. داده‌های صوتی در فضای Tonnetz، با نمایشی هندسی فواصل زیر و بمی همسان مبتنی بر تئوری موسیقی، پخش می‌شوند. نشان داده شده است که این رویکرد از دقت طبقه‌بندی نمایش‌های کرومای قبلی بهتر عمل می‌کند، در حالی که فضای ویژگی سطح متوسطی را فراهم می‌کند که چالش‌های ذاتی کروما را دور می‌زند.

Spectral Centroid

این ویژگی نشان می‌دهد که مرکز جرم برای یک صدا در کجا قرار گرفته و به عنوان میانگین وزنی برای فرکانس‌های موجود در صدا محاسبه می‌شود. مثلا بین دو اهنگ یکی از سبک بلوز و یکی از سبک متال، موسیقی بلوز در سراسر طولش یکسان است ولی موسیقی متال فرکانس‌های بیشتری تا پایان دارد. بنابراین، (Spectral Centroid) برای موسیقی بلوز جایی نزدیک اواسط طیف آن است، در حالیکه برای موسیقی متال در انتهای آن قرار دارد. دستور librosa.feature.spectral_centroid این ویژگی را برای هر فریم در سیگنال را محاسبه می‌کند.

در بعضی جاها می توان آن را میانه طیف در نظر گرفت اما بین اندازه گیری مرکز طیفی و میانه طیف تفاوت وجود دارد. مرکز طیفی مانند یک میانه وزنی و میانه طیف شبیه به میانگین است. هر دوی آنها تمایل مرکزی سیگنال را اندازه گیری می کنند. در برخی موارد، هر دوی آنها نتایج مشابهی را نشان می دهند.

Spectral Bandwidth

پهنای باند تفاوت بین فرکانس های بالا و پایین در یک باند فرکانس پیوسته است. همانطور که می دانیم سیگنال ها حول یک نقطه نوسان می کنند، بنابراین اگر نقطه مرکز سیگنال باشد، مجموع حداکثر انحراف سیگنال در دو طرف آن نقطه را می توان به عنوان پهنای باند سیگنال در آن فریم زمانی در نظر گرفت.

این مفهوم از مرکز طیفی مشتق شده است. این محدوده طیفی مورد توجه در اطراف مرکز است، یعنی واریانس از مرکز طیفی. همبستگی مستقیمی با صدای درک شده دارد. پهنای باند مستقیماً با انرژی پخش شده در باندهای فرکانسی متناسب است. از نظر ریاضی، میانگین وزنی فواصل باندهای فرکانسی از مرکز طیفی است.

Spectral Flatness

مسطح بودن طیفی که همچنین به عنوان آنتروپی وینر شناخته می شود، معیاری است که در پردازش سیگنال دیجیتال برای مشخص کردن طیف صوتی استفاده می شود. این ویژگی ، یکنواختی توزیع انرژی سیگنال در حوزه فرکانس را تخمین می زند. صافی طیفی معمولاً بر حسب دسی بل اندازه گیری می شود و روشی را برای تعیین کمیت یک صدا در مقایسه با نویز مانند بودن، ارائه می دهد.

مسطح بودن طیفی بالا (نزدیک به ۱,۰ برای نویز سفید) نشان می دهد که طیف در همه باندهای طیفی دارای مقدار مشابهی از قدرت است، این صدا شبیه نویز سفید است و نمودار طیف نسبتاً صاف و صاف به نظر می رسد و به معنی یکنواختی بیشتر در توزیع انرژی سیگنال در حوزه فرکانس است. در حالی که آنتروپی وینر پایین دلالت بر یکنواختی کمتر دارد. صافی طیفی کم (نزدیک به ۰,۰ برای یک تن خالص) نشان می دهد که توان طیفی در تعداد نسبتاً کمی از باندها متمرکز است.

این معیار را می توان برای تمایز یک سیگنال باند باریک از سیگنال باند پهن، به عنوان مثال، تمایز بین یک صدا و نویز سفید استفاده کرد. اما نمی توان از آنها برای تشخیص دو سیگنال پهن باند، به عنوان مثال، سیگنال LFM و نویز استفاده کرد.

Spectral Contrast

در یک سیگنال صوتی، کنتراست طیفی اندازه گیری انرژی فرکانس در هر پنجره زمانی است. از آنجایی که اکثر فایل های صوتی حاوی فرکانس هایی هستند که انرژی آن با گذشت زمان در حال تغییر است. اندازه گیری سطح انرژی دشوار می شود. کنتراست طیفی راهی برای اندازه گیری این تنوع انرژی است. کنتراست طیفی حداکثر طیفی، حداقل طیفی و نابرابری بین هر فرکانس زیر باند را در نظر می گیرد.مقادیر کنتراست بالا به طور کلی با سیگنال‌های باند باریک و شفاف مطابقت دارد، در حالی که مقادیر کنتراست پایین مربوط به نویز باند وسیع است. کنتراست انرژی با مقایسه انرژی متوسط در قاب انرژی اوج با قاب انرژی پایین یا دره اندازه‌گیری می‌شود.

Spectral Rolloff

این ویژگی پهنای باند سیگنال صوتی را با تعیین یک bin فرکانسی که در آن درصد معینی از کل انرژی وجود دارد، اندازه گیری می کند. در واقع فرکانس roll-off به عنوان فرکانسی تعریف می شود که در آن درصدی (قطع) از کل انرژی طیف قرار می گیرد. برای مثال ۸۰٪ یا هر درصد دلخواه دیگر. فرکانس رول آف را می توان برای تمایز بین صداهای هارمونیک (زیر رول آف) و نویزدار (بالای رول آف) استفاده کرد.

می توان آن را به عنوان عملکرد نوع خاصی از فیلتر تعریف کرد که برای کاهش فرکانس های خارج از محدوده خاص طراحی شده است. دلیل اینکه ما آن را رول آف می نامیم این است که یک رویه تدریجی است. دو نوع فیلتر وجود دارد: Hi-pass و Low Pass و هر دو می توانند فرکانس را از سیگنالی که خارج از محدوده آنها قرار می گیرد حذف کنند.

نقطه rolloff طیفی برای تمایز بین گفتار صدادار و بدون صدا، تمایز گفتار/موسیقی، طبقه‌بندی ژانر موسیقی، تشخیص صحنه آکوستیک و طبقه‌بندی حالت موسیقی استفاده شده است.

RMS

مقدار (Root-Mean-Square) همان مقدار مؤثر شکل موج کل است. این برابر با سطح سیگنال DC است که همان توان متوسط سیگنال دوره ای را ارائه می دهد. چون مربوط به ریشه میانگین مربعات است پس با RMS بالاتر به طور کلی به صدای قوی تر است. که برای تشخیص صدای مردان و همچنین تشخیص عاطفه عصبانیت میتواند مفید باشد.

این اندازه‌گیری عمداً یک اندازه‌گیری آهسته است که میانگین قله‌ها و پایین‌ترین زمان‌ها را برای انعکاس بلندی درک شده را نشان می‌دهد. RMS معیار تقریبی روشی است که گوش شما سطوح صدا را درک می کند. مثلا گوش شما معمولاً قله های تیز را به اندازه واقعی بلند نمی بیند.

Zero Crossing Rate

نرخ تغییر علامت‌ها در طول یک سیگنال و در واقع نرخی است که در آن سیگنال از مثبت به منفی یا بالعکس تغییر می‌کند. این ویژگی به طور سنگین هم در «بازشناسی گفتار (Speech Recognition)» و هم در «بازیابی اطلاعات موسیقی (Music Information Retrieval)» مورد استفاده قرار می‌گیرد. ویژگی مذکور معمولا دارای مقداری بیشتر برای صداهای بسیار کوبه‌ای است، مثلا در هنگام عصبانیت و خوشحالی احتمالا این ویژگی بیشتر از عواطف غم و خنثی باشد. همچنین می توان از آن برای الگوریتم های تشخیص زمین و همچنین برای تشخیص فعالیت های صوتی استفاده کرد.

Poly_Features

ضرایب برازش یک چند جمله ای مرتبه n را به ستون های یک طیف نگار برمی گرداند. این ویژگی را هم می توان به راحتی با استفاده از Librosa استخراج کرد . ضرایب چند جمله ای برای هر فریم را محاسبه میکند. به این ترتیب که اول ضرایب[۰] مربوط به بالاترین درجه ، ضرایب[۱] مربوط به بالاترین درجه بعدی تا ضرایب

ثابت را به عنوان خروجی برمیگرداند. سپس می توان میانگینی از ضرایب را اعمال کرد یا آماری را برای توصیف سری خروجی به دست آورد، در غیر این صورت مقادیر بیش از حد داده می شود.

پیش پردازش داده ها :

پس از مرحله ی استخراج ویژگی ها و بدست اوردن دیتا ست اولیه از داده ها باید داده ها را پیش پردازش کنیم و به ازای هر دیتا یک برچسب به ان نسبت دهیم.

دیتا ست اولیه داده ها بعد از استخراج ویژگی ها را Voice_data نام گذاری می کنیم که به صورت زیر می باشد:

	voice_id	chroma_stft	rmse	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mfcc2	mfcc3	...	mfcc11	mfcc12	mfcc13	mfcc14	mfcc15	m
0	9098.wav	0.369225	0.045809	2329.631156	2413.742289	4977.576382	0.116935	-394.132568	65.614784	-0.230034	...	-4.276749	-2.955371	-14.975864	-1.616190	-14.069487	-11.6
1	9099.wav	0.299851	0.041585	2258.563870	2382.174317	4477.544056	0.106513	-396.777618	77.251091	-1.967859	...	-6.351185	-2.557911	-19.613825	2.119483	-12.453156	-10.0
2	9100.wav	0.331110	0.027311	1928.588396	1460.443570	1579.451696	0.049447	-478.123016	82.608612	26.420359	...	-12.246988	-0.775021	-3.362810	-1.860957	-7.215577	-6.7
3	9100.wav	0.429410	0.011069	2268.009634	2696.274514	5271.647135	0.094634	-533.723877	80.416367	27.698174	...	-2.108526	-6.852947	-7.655149	-1.265465	-1.209886	-3.6
4	9101.wav	0.396582	0.016276	2322.139883	2855.396293	5161.143494	0.105826	-460.091223	67.926096	16.232029	...	-2.383848	-1.810102	-9.272352	1.916275	-6.315449	-6.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17152	10133.wav	0.478882	0.010691	2459.603144	2628.635417	5508.592122	0.093594	-511.655542	79.787819	4.267803	...	-7.007113	-7.512400	-2.180801	-2.388452	-5.206617	-1.5
17153	10134.wav	0.522729	0.001647	2777.438580	2724.236810	5861.376067	0.151424	-584.935242	78.813919	2.997258	...	-6.767550	-5.975239	-6.760041	1.310687	-2.112588	0.3
17154	10135.wav	0.544817	0.001796	2876.029520	2734.382058	6088.184077	0.165784	-582.691650	75.094543	1.962689	...	-4.628536	-3.738466	-6.230413	2.142874	-3.748188	-0.8
17155	10136.wav	0.333702	0.027438	2071.605226	1999.756791	4059.066265	0.113778	-478.583740	61.517368	8.281408	...	-11.922658	-2.483714	-5.218111	4.417655	-7.259745	-3.7
17156	10137.wav	0.309189	0.037861	1954.365253	1830.258652	3396.602314	0.078250	-433.365897	60.326766	3.264502	...	-11.350521	-1.512024	-10.716776	-5.300368	-5.766279	-6.6

17157 rows × 27 columns

ستون voice id به صورت String می باشد . به همین منظور باید ابتدا پسوندwav. را ازدیتا های این ستون حذف کنیم و ان ها را به مقادیر integer تبدیل کنیم .

در قطعه کد موارد فوق انجام شده است و دیتا برحسب ستون voice id به صورت صعودی مرتب شده است .

```
for i in range (0,17157 ):
  voice_data['voice_id'][i] = voice_data['voice_id'][i].replace('.wav', '')
```

```
voice_data['voice_id']=voice_data['voice_id'].astype(str).astype(int)
```

```
voice_data=voice_data.sort_values('voice_id')
```

در قدم بعد دیتاست متشکل از برچسب های داده ها را import می کنیم (datav2)که به صورت زیر می باشد:

	emotionID	textID	sex	age	voice id
0	1	1	m	21	15997
1	1	2	m	21	16001
2	1	3	m	21	16005
3	1	4	m	21	16009
4	1	5	m	21	16013
...	...	...	...	...	...
16860	4	6	f	54	10563
16861	4	7	f	54	10567
16862	4	8	f	54	10571
16863	4	9	f	54	10575
16864	4	10	f	54	10539

16865 rows × 5 columns

همانطور که مشاهده می کنید تعداد دیتا های موجود در دیتا ست برچسب ها ۱۶۸۶۵ و دیتاست داده ها ۱۷۱۵۷ می باشد ، در نتیجه دیتا باید پیش پردازش شوند و دیتاهای نادرست حذف شوند.

در قدم اول ، سطرهایی که در دیتاست متشکل از برچسب های داده ها، دارای voice id یکسان هستند را ، تشخیص داده و به صورت زیر حذف می کنیم .

```
same_id = []
for i in range(0,16865):
  for j in range(i,16865):
    if (dataV2['voice id'][i] == dataV2['voice id'][j]) & (i != j) :
      same_id.append(i)

# same_id
dataV2.drop(same_id , axis=0, inplace=True)
```

در قدم بعد ، متوجه شدیم که تعدادی از سطر های دیتاست داده ها و برچسب ها دارای voice id ای هستند که در دیتاست دیگر وجود ندارد ، در نتیجه با مقایسه ستون voice id دیتاست داده ها را با دیتاست برچسب ها ان ها را حذف می کنیم.

```
for j in voice_data['voice_id'] :
  if not ((datav2.voiceId == j).any()) :
    voice_data.drop(voice_data[voice_data['voice_id']== j].index, axis=0, inplace=True)
```

```
for j in datav2['voiceId'] :
  if not ((voice_data.voice_id == j).any()) :
    datav2.drop(datav2[datav2['voiceId']== j].index, axis=0, inplace=True)
```

تعدادی از داده های موجود در دیتاست برچسب ها ، چندین بار تکرار شده اند و نیاز است از سطر هایی که بیش از یک بار تکرار شده اند یکی از ان ها نگه داشته شود و باقی ان ها حذف شوند. برای این منظور به صورت زیر عمل می کنیم .

```
for j in datav2['voiceId'] :
  if datav2['voiceId'].value_counts()[j] > 1 :
    for i in range(datav2['voiceId'].value_counts()[j]-1) :
      datav2.drop(datav2[datav2['voiceId']== j].index[i], axis=0, inplace=True)
```

پس از اتمام تمیز سازی داده ها نیاز است دیتاست داده ها و برچسب ها را یکی کنیم . برای این منظور دیتاست هارا بر حسب ستون voide id مرتب کرده و سپس دو دیتاست را به هم پیوند می زنیم .

```
voice_data.reset_index(inplace=True)
voice_data=voice_data.drop(['index'],axis=1)
```

```
datav2.reset_index(inplace=True)
datav2=datav2.drop(['index'],axis=1)
```

```
Data_final = pd.concat([voice_data, datav2], axis=1, join='inner')
Data_final=Data_final.drop(['voiceId'],axis=1)
```

می دانیم ستون sex باید متشکل از دو نوع String باشد ، یکی’f‘ برای داده های دارای جنسیت زن ودیگری ’m‘ برای داده های دارای جنسیت مرد . اما این ستون دارای عناصر دیگری به جز f , m می باشد که نیاز است اصلاح شود . همچنین با استفاده از کتابخانه ی LabelEncoder به ازای f ، ۰ و به ازای m ، ۱ قرار می دهیم. برای این منظور به صورت زیر عمل می کنیم.

```
Data_final['sex'].unique()
```

```
array(['f', 'm', 'M', 'F', 'w', 'f '], dtype=object)
```

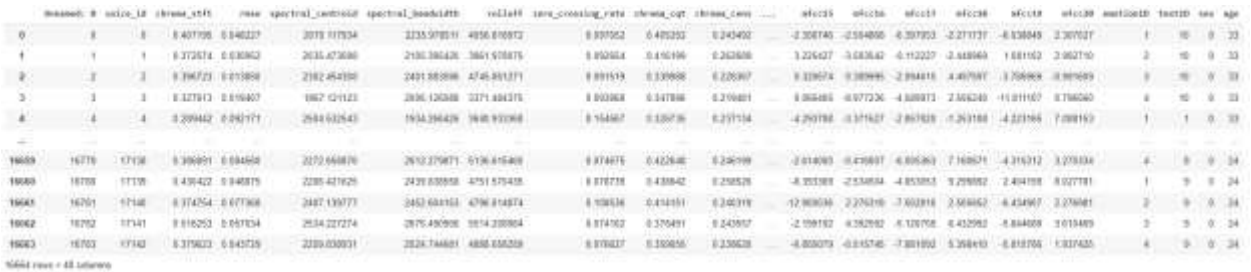
```
for i in range (0 , 16784 ) :
  if (Data_final['sex'][i] == 'f') :
    Data_final['sex'][i] = 'f'
  if (Data_final['sex'][i] == 'f ') :
    Data_final['sex'][i] = 'f'
  if (Data_final['sex'][i] == 'F') :
    Data_final['sex'][i] = 'f'
  if (Data_final['sex'][i] == 'M') :
    Data_final['sex'][i] = 'm'
  if (Data_final['sex'][i] == 'm') :
    Data_final['sex'][i] = 'm'
  if (Data_final['sex'][i] == 'w') :
    Data_final['sex'][i] = 'm'

LE=LabelEncoder()
Data_final['sex']=LE.fit_transform(Data_final['sex'])
```

ستون emotionID نیز دارای عناصری غیر از ۱و۲و۳و۴ می باشد. برای این منظور دیتا هایی که دارای emotionID غیر از موارد ذکر شده هستند را به صورت زیر حذف می کنیم.

```
for j in [5,6,7,8,9,10] :
  if  ((Data_final.emotionID == j).any()) :
    Data_final.drop(Data_final[Data_final['emotionID']== j].index, axis=0, inplace=True)
```

در نتیجه دیتاست نهایی ما پس از مراحل پیش پردازش و تمیز سازی داده ها به صورت زیر خواهد.



طبقه بندی :

طبق دستور عمل ویژگی های استخراج شده را به دو دسته تقسیم میکنیم. سعی شده است ویژگی های شبیه به هم در یک دسته قرار بگیرند.

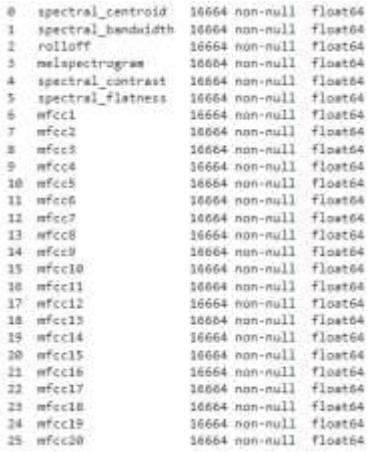
دسته ویژگی های اول :

ویژگی هایی که در دسته ی اول استفاده شده اند عبارت اند از :

```
0   chroma_stft          16664 non-null  float64
1   rmse                 16664 non-null  float64
2   zero_crossing_rate   16664 non-null  float64
3   poly_features        16664 non-null  float64
4   chroma_cqt           16664 non-null  float64
5   chroma_cens          16664 non-null  float64
```

دسته ویژگی های دوم :

ویژگی هایی که در دسته ی دوم استفاده شده اند عبارت اند از :



طبقه بندی جنسیت :

برای این منظور در ابتدا دیتاست X , y تعیین کرده و با استفاده از Standard Scaler ، داده ها را نرمالایز می کنیم. در قدم بعد داده را به دو دسته ی train و test تقسیم کرده و ۳۰ درصد داده ها را به تست اختصاص می دهیم.

مدل MLP برای طبقه بندی جنسیت:

دسته ای از شبکه های عصبی مصنوعی پیشخور است. یک MLP شامل حداقل سه لایه گره است: یک لایه ورودی، یک لایه پنهان و یک لایه خروجی. به جز گره‌های ورودی، هر گره یک نورون است که از یک تابع فعالسازی غیر خطی استفاده می‌کند. MLP از تکنیک یادگیری نظارت شده به نام بازپرداخت برای آموزش استفاده می‌کند. لایه‌های متعدد آن و فعال‌سازی غیر خطی آن MLP را از یک پرسپترون خطی متمایز می‌کند. در واقع می‌تواند داده‌هایی را متمایز کند که به صورت خطی قابل‌تفکیک نیستند. این شبکه ها شامل سه یا تعداد بیشتری از لایه‌ها است که از گره‌های غیرخطی فعال کننده هستند. از آنجا کهMLP ها به‌طور کامل متصل شده‌اند، هر گره در یک لایه با وزن مشخص w_{ij} درهر گره در لایه بعدی متصل می‌شود.

نودهای شبکه عصبی که به آن‌ها نورون نیز گفته می‌شود، واحدهای محاسباتی در یک شبکه عصبی محسوب می‌شوند. در این شبکه عصبی، از خروجی‌های لایه اول (ورودی)، به عنوان ورودی‌های لایه بعدی (نهان) استفاده می‌شود؛ این کار به همین شکل ادامه پیدا می‌کند، تا زمانی که، پس از تعداد خاصی از لایه‌ها، خروجی‌های آخرین لایه نهان به عنوان ورودی‌های لایه خروجی مورد استفاده قرار می‌گیرد. به لایه‌هایی که بین لایه ورودی و لایه خروجی قرار می‌گیرند، لایه‌های نهان گفته می‌شود. شبکه‌های پرسپترون چند لایه، مانند شبکه‌های عصبی پرسپترون تک لایه، حاوی مجموعه‌ای از وزن‌ها نیز هستند که باید برای آموزش و یادگیری شبکه عصبی تنظیم شوند.

مدل طراحی شده برای شبکه های عصبی mlp به صورت زیر می باشد . این مدل از۴ لایه تشکیل شده است که تابع فعال ساز سه لایه اول relu و لایه اخر softmax انتخاب شده است . تعداد نورون های در نظر گرفته شده برای هر لایه در قطعه کد زیر نمایش داده شده است .

```
model = Sequential()
model.add(layers.Dense(256, activation='relu', input_shape=(X_train.shape[1],)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(2, activation='softmax'))
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
classifier = model.fit(X_train,
                       y_train,
                       epochs=20,
                       batch_size=128 , validation_split = 0.2)
```

اعمال مدل MLP بر روی دسته ویژگی های اول :

دقت و خطای بدست امده برای داده های train و validation در epochs 5 اخر به شرح زیر است.

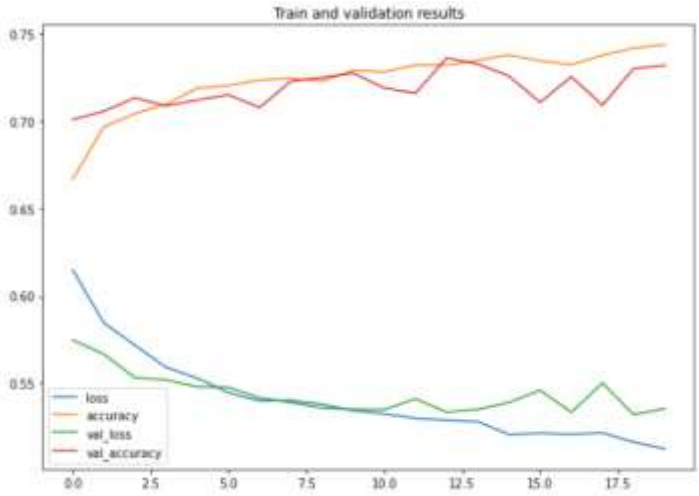
```
Epoch 16/20
73/73 [==============================] - 0s 5ms/step - loss: 0.5215 - accuracy: 0.7343 - val_loss: 0.5461 - val_accuracy: 0.7107
Epoch 17/20
73/73 [==============================] - 0s 5ms/step - loss: 0.5210 - accuracy: 0.7322 - val_loss: 0.5335 - val_accuracy: 0.7252
Epoch 18/20
73/73 [==============================] - 0s 5ms/step - loss: 0.5216 - accuracy: 0.7375 - val_loss: 0.5502 - val_accuracy: 0.7090
Epoch 19/20
73/73 [==============================] - 0s 5ms/step - loss: 0.5165 - accuracy: 0.7416 - val_loss: 0.5321 - val_accuracy: 0.7300
Epoch 20/20
73/73 [==============================] - 0s 5ms/step - loss: 0.5126 - accuracy: 0.7435 - val_loss: 0.5356 - val_accuracy: 0.7317
```

دقت و خطای مدل روی داده های تست برابر است با :

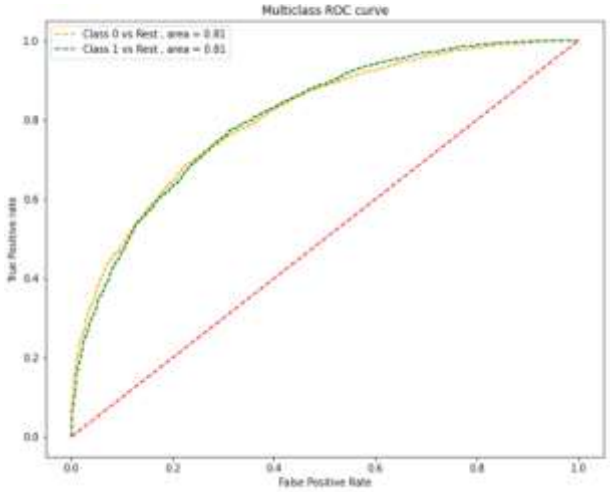
```
results = model.evaluate(X_test,  y_test, verbose = 0)
print('test loss, test accuracy:', results)
```

```
test loss, test accuracy: [0.5321431756019592, 0.7310000061988831]
```

نمودار دقت و خطا مدل :



نمودار ROC مدل :



اعمال مدل MLP بر روی دسته ویژگی های دوم :

دقت و خطای بدست امده برای داده های train و validation در 5 epochs اخر به شرح زیر است.

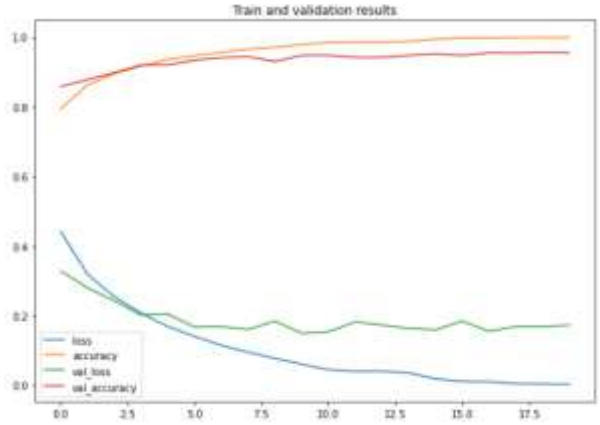
```
Epoch 16/20
73/73 [==============================] - 0s 5ms/step - loss: 0.0107 - accuracy: 0.9986 - val_loss: 0.1840 - val_accuracy: 0.9477
Epoch 17/20
73/73 [==============================] - 0s 5ms/step - loss: 0.0097 - accuracy: 0.9984 - val_loss: 0.1548 - val_accuracy: 0.9559
Epoch 18/20
73/73 [==============================] - 0s 5ms/step - loss: 0.0046 - accuracy: 0.9998 - val_loss: 0.1680 - val_accuracy: 0.9546
Epoch 19/20
73/73 [==============================] - 0s 5ms/step - loss: 0.0035 - accuracy: 0.9999 - val_loss: 0.1678 - val_accuracy: 0.9567
Epoch 20/20
73/73 [==============================] - 0s 5ms/step - loss: 0.0024 - accuracy: 1.0000 - val_loss: 0.1726 - val_accuracy: 0.9559
```

دقت و خطای مدل روی داده های تست برابر است با :

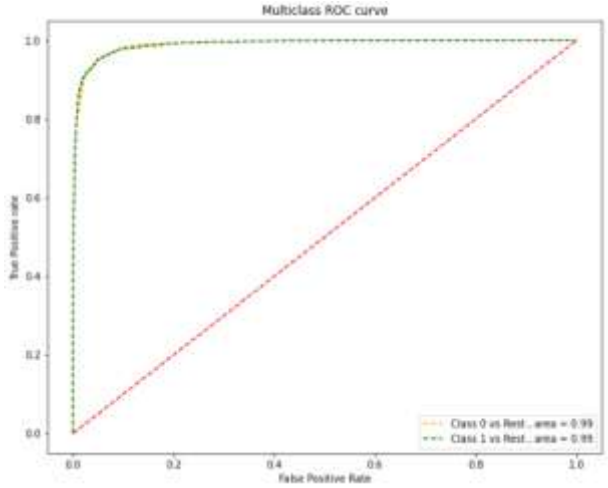
```
results = model.evaluate(X_test,  y_test, verbose = 0)
print('test loss, test accuracy:', results)
```

```
test loss, test accuracy: [0.17778711020946503, 0.9491999745368958]
```

نمودار دقت و خطا مدل :



نمودار ROC مدل :



تحلیل نتایج بدست امده :

دقت مدل روی دسته اول برای داده آموزش ۷۴٪ و برای داده تست ۷۳٪ بوده است. پس بیش برازش اتفاق نیفتاده است. دقت مدل روی دسته داده دوم به نحو چشمگیری افزایش افتاده است. روی دسته دوم دقت مدل برای داده آموزش ۱۰۰٪ و برای داده تست ۹۵٪ به دست آمده است. پس دسته ویژگی های دوم که شامل mfcc و چند ویژگی spectral بوده است(منظور از ویژگی spectral ویژگی های spectral rolloff، spectral centroid، spectral bandwidth و در کل ویژگی هایی است که با کلمه spectral شروع میشوند.) دقت بسیار بهتری به دست داده است. دسته اول ویژگی ها شامل RMSE، ZCR، poly_features و chroma ها است دقت کمتری داده است. علت این اس که داده های دسته اول مشخصه های طیفی و زیر و بم را بسیار بهتر نمایان میکنند. همچنین ویژگی های mfcc مدل بسیار نزدیکی به صدای انسان تولید میکنند که برای آموزش داده ها بهتر عمل کرده‌اند. در دسته داده دوم بیشتر ویژگی ها از نوع زمانی و معمولی بوده اند و برای تفکیک داده صوتی کافی نبوده اند. همان عملکرد نسبتا خوب دسته ویژگی دوم هم مربوط به chroma است که گام صدا را استخراج میکند.

مدل SVM برای طبقه بندی جنسیت:

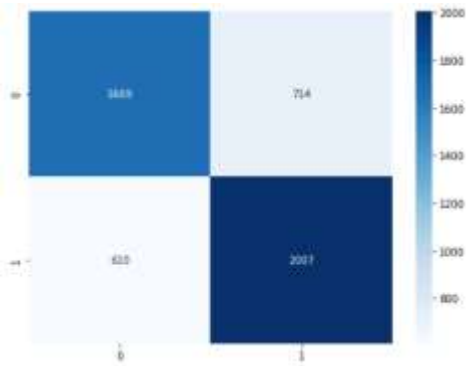
این روش از جمله روش‌های نسبتاً جدیدی است که در سال‌های اخیر کارایی خوبی نسبت به روش‌های قدیمی‌تر برای طبقه‌بندی نشان داده‌است. مبنای کاری دسته‌بندی کننده ی SVM دسته‌بندی خطی داده‌ها است و در تقسیم خطی داده‌ها سعی می‌کنیم خطی را انتخاب کنیم که حاشیه اطمینان بیشتری داشته باشد. حل معادله پیدا کردن خط بهینه برای داده‌ها به وسیله روش‌های QP که روش‌های شناخته شده‌ای در حل مسائل محدودیت‌دار هستند صورت می‌گیرد. قبل از تقسیمِ خطی برای اینکه ماشین بتواند داده‌های با پیچیدگی بالا را دسته‌بندی کند داده‌ها را به وسیله ی تابع phi به فضای با ابعاد خیلی بالاتر می‌بریم. برای اینکه بتوانیم مسئله ابعاد خیلی بالا را با استفاده از این روش‌ها حل کنیم از قضیه دوگانی لاگرانژ برای تبدیلِ مسئله ی مینیمم‌سازی مورد نظر به فرم دوگانی آن که در آن به جای تابع پیچیده ی phi که ما را به فضایی با ابعاد بالا می‌برد، تابعِ ساده‌تری به نامِ تابع هسته که ضرب برداری تابع phi است ظاهر می‌شود استفاده می‌کنیم. از توابع هسته مختلفی از جمله هسته‌های نمایی، چندجمله‌ای و سیگموید می‌توان استفاده نمود.

```
classifier = SVC(kernel='rbf',C=150)
classifier.fit(X_train,y_train)
y_pred_test = classifier.predict(X_test)
```

اعمال مدل SVM بر روی دسته ویژگی های اول :

مقادیر دقت روی داده های تست و اموزش و f1 score و jaccard score و همچنین ماتریس confusion به شرح زیر می باشد.

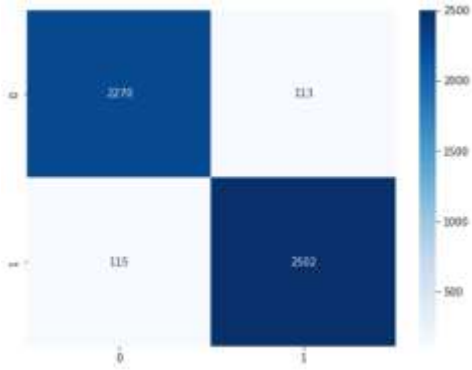
```
Train_accuracy    : 76.8004%
Test_accuracy     : 73.5200%
f1_score          : 73.398%
jaccard_score     : 58.008%
```



اعمال مدل SVM بر روی دسته ویژگی های دوم :

مقادیر دقت روی داده های تست و اموزش و f1 score و jaccard score و همچنین ماتریس confusion به شرح زیر می باشد.

```
Train_accuracy    : 99.9914%
Test_accuracy     : 95.4400%
f1_score          : 95.430%
jaccard_score     : 91.261%
```



تحلیل نتایج بدست امده SVM برای جنسیت روی دو دسته ویژگی :

دقت مدل روی دسته اول برای داده آموزش ۷۶/۸٪ و برای داده تست ۷۳/۵٪ بوده است. پس بیش برازش اتفاق نیفتاده است. دقت مدل روی دسته داده دوم به نحو چشمگیری افزایش افتاده است. روی دسته دوم دقت مدل برای داده آموزش ۱۰۰٪ و برای داده تست ۹۵/۴٪ به دست آمده است. برای دسته دوم ویژگی هم بیش برازش اتفاق نیفتاده است. برای مدل svm هم مانند مدل قبل دسته دوم بسیار بهتر عمل میکند. دلایل مشابه قبل است.

مدل Random Forest برای طبقه بندی جنسیت:

جنگل تصادفی یک روش یادگیری آنسمبل است که برای دسته بندی و رگرسیون به کار میرود. این الگوریتم بر اساس ساختاری متشکل از تعداد زیادی درخت تصمیم،

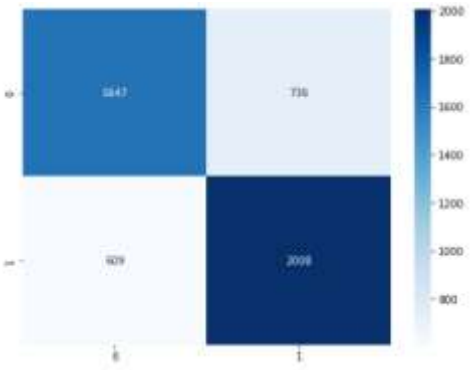
مدل مورد نظر به صورت زیر می باشد .

```
clf = RandomForestClassifier(n_estimators=1000 , random_state=42)
clf.fit(X_train,y_train)
y_pred_test=clf.predict(X_test)
```

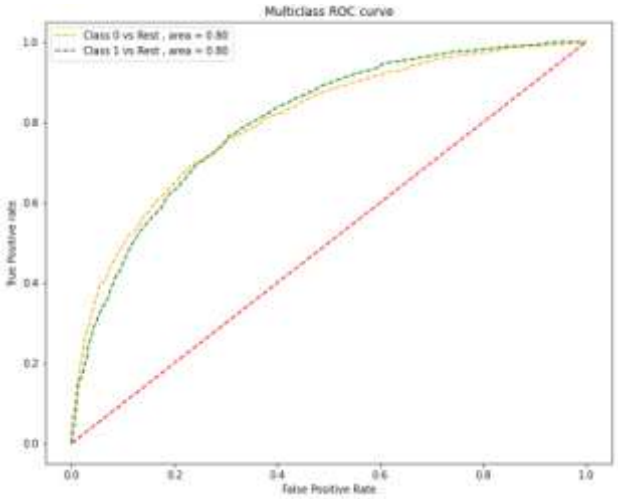
مقادیر دقت روی داده های تست و اموزش و f1 score و jaccard score و همچنین ماتریس confusion به شرح زیر می باشد.

اعمال مدل Random Forest بر روی دسته ویژگی های اول :

```
Train_accuracy    : 100.0000%
Test_accuracy     : 73.1000%
f1_score          : 72.959%
jaccard_score     : 57.467%
```

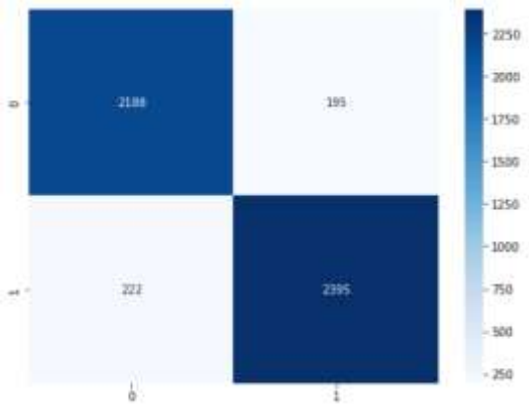


نمودار ROC مدل :

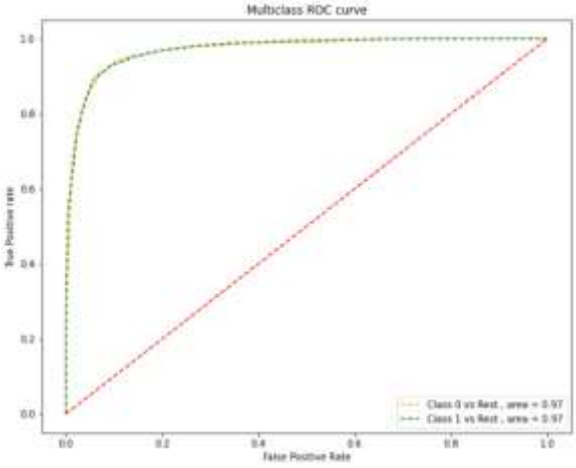


اعمال مدل Random Forest بر روی دسته ویژگی های دوم :

```
Train_accuracy    : 100.0000%
Test_accuracy    : 91.6600%
f1_score          : 91.646%
jaccard_score     : 84.582%
```



نمودار ROC مدل :



تحلیل نتایج بدست امده جنگل تصادفی روی دسته اول و دوم:

دقت مدل روی دسته اول برای داده آموزش ۱۰۰٪و برای داده تست ۷۳٪ بوده است. پس بیش برازش اتفاق افتاده است.با وجود اینکه روش جنگل تصادفی که امکان رخ دادن بیش برازش بسیار کم است باز هم این اتفاق افتاده است. دقت مدل روی دسته داده دوم به نحو چشمگیری افزایش یافته است. روی دسته دوم دقت مدل برای داده آموزش ۱۰۰٪ و برای داده تست ۹۱/۶٪ به دست آمده است. دسته دوم ویژگی به قدری برای تفکیک صدای زن و مرد خوب عمل میکند که حتی روی داده آموزش هم دقت عالی گرفته ایم. ویژگی های بر حسب طیف و mfcc برای امر حداسازی صدای زن و مرد بسیار خوب عمل میکند.

مقایسه مدلها برای جنسیت:

همانطور که در جدول زیر نمایش داده شده است ، همانطور که انتظار داشتیم، مدل SVM بر روی دسته ویژگی های دوم دارای عملکرد بهتری نسبت به سایر مدل ها دارا است. البته اختلاف میان دقت مدلها روی داده تست بسیار نزدیک بوده است به هر حال random forest ضعیف تر از بقیه مدلها عمل کرده است.

مدل	دقت روی دسته ویژگی اول		دقت روی دسته ویژگی دوم	
	Train	test	Train	Test
MLP	74.35%	73.1 %	100%	94.9%
SVM	76.89%	73.52%	99.99%	95.44%
Random Forest	100%	73.1%	100%	91.66%

طبقه بندی احساسات :

برای این منظور در ابتدا دیتاست y , X تعیین کرده و با استفاده از Standard Scaler ، داده ها را نرمالایز می کنیم. در قدم بعد داده را به دو دسته ی train و test تقسیم کرده و ۳۰ درصد داده ها را به تست اختصاص می دهیم.

مدل MLP برای طبقه بندی احساسات:

مدل طراحی شده برای شبکه های عصبی mlp به صورت زیر می باشد . این مدل از۴ لایه تشکیل شده است که تابع فعال ساز سه لایه اول relu و لایه اخر softmax انتخاب شده است . تعداد نورون های در نظر گرفته شده برای هر لایه در قطعه کد زیر نمایش داده شده است .

```
model = Sequential()
model.add(layers.Dense(256, activation='relu', input_shape=(X_train.shape[1],)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(4, activation='softmax'))
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

اعمال مدل MLP بر روی دسته ویژگی های اول :

دقت و خطای بدست امده برای داده های train و validation در 5 epochs اخر به شرح زیر است.

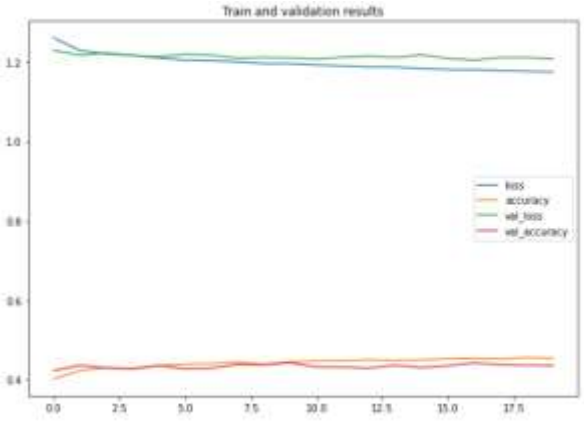
```
Epoch 16/20
73/73 [==============================] - 0s 4ms/step - loss: 1.1810 - accuracy: 0.4529 - val_loss: 1.2085 - val_accuracy: 0.4351
Epoch 17/20
73/73 [==============================] - 0s 5ms/step - loss: 1.1806 - accuracy: 0.4539 - val_loss: 1.2045 - val_accuracy: 0.4423
Epoch 18/20
73/73 [==============================] - 0s 5ms/step - loss: 1.1781 - accuracy: 0.4524 - val_loss: 1.2108 - val_accuracy: 0.4376
Epoch 19/20
73/73 [==============================] - 0s 5ms/step - loss: 1.1761 - accuracy: 0.4559 - val_loss: 1.2109 - val_accuracy: 0.4363
Epoch 20/20
73/73 [==============================] - 0s 5ms/step - loss: 1.1752 - accuracy: 0.4542 - val_loss: 1.2074 - val_accuracy: 0.4359
```

دقت و خطای مدل روی داده های تست برابر است با :

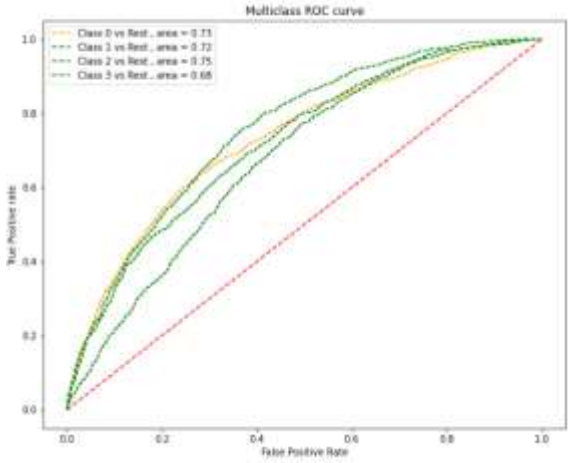
```
results = model.evaluate(X_test,  y_test, verbose = 0)
print('test loss, test accuracy:', results)
```

```
test loss, test accuracy: [1.2038077116012573, 0.4390000104904175]
```

نمودار دقت و خطا مدل :



نمودار ROC مدل :



اعمال مدل MLP بر روی دسته ویژگی های دوم :

دقت و خطای بدست امده برای داده های train و validation در 5 epochs اخر به شرح زیر است.

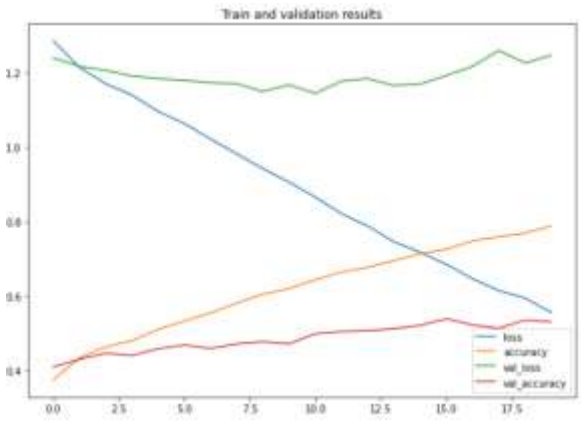
```
Epoch 16/20
73/73 [==============================] - 0s 6ms/step - loss: 0.6848 - accuracy: 0.7252 - val_loss: 1.1919 - val_accuracy: 0.5304
Epoch 17/20
73/73 [==============================] - 0s 5ms/step - loss: 0.6461 - accuracy: 0.7477 - val_loss: 1.2165 - val_accuracy: 0.5221
Epoch 18/20
73/73 [==============================] - 0s 5ms/step - loss: 0.6142 - accuracy: 0.7592 - val_loss: 1.2591 - val_accuracy: 0.5131
Epoch 19/20
73/73 [==============================] - 0s 6ms/step - loss: 0.5943 - accuracy: 0.7682 - val_loss: 1.2255 - val_accuracy: 0.5349
Epoch 20/20
73/73 [==============================] - 0s 5ms/step - loss: 0.5554 - accuracy: 0.7879 - val_loss: 1.2464 - val_accuracy: 0.5302
```

دقت و خطای مدل روی داده های تست برابر است با :

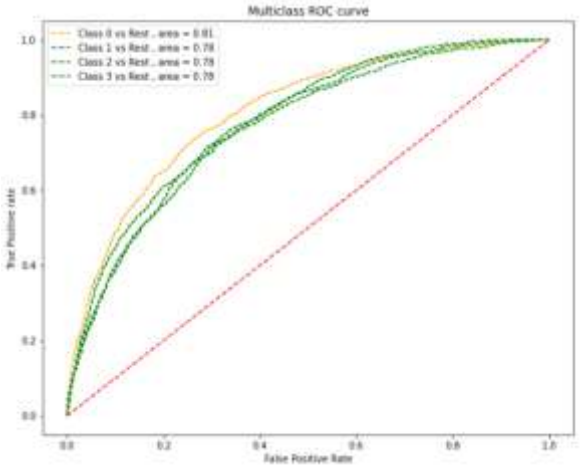
```
results = model.evaluate(X_test,  y_test, verbose = 0)
print('test loss, test accuracy:', results)
```

```
test loss, test accuracy: [1.2326005697250366, 0.5266000032424927]
```

نمودار دقت و خطا مدل :



نمودار ROC مدل :



تحلیل نتایج بدست امده mlp روی دسته ویژگی اول برای تفکیک احساسات:

دقت مدل روی دسته اول برای داده آموزش ۴۵٪ و برای داده تست ۴۳٪ بوده است. پس بیش برازش اتفاق نیفتاده است. اما دقت مدل روی این دسته ویژگی برای تفکیک احساسات حتی از دقت رندوم هم پایینتر است و اصلا مناسب نیست. دقت مدل برای دسته ویژگی دوم بهبود یافته است چون ویژگی های این دسته حتی برای تفکیک احساس هم بهتر هستند. روی دسته دوم دقت مدل برای داده آموزش ۷۸/۸٪ و برای داده تست ۵۳٪ به دست آمده است. روی قسمت آموزش دقت قابل قبول گرفته ایم ولی روی داده تست باز هم تفاوت چشمگیر

نیست و از دسته بدی رندوم تصادفی تنها ۳٪ بهتر شده است. در مورد این دسته بیش برازش اتفاق افتاده است که در انتهای این بخش در این مورد بحث شده است.

مدل SVM :

```
classifier = SVC(kernel='rbf',C=3.8 , gamma=0.1)
classifier.fit(X_train,y_train)
y_pred_test = classifier.predict(X_test)
```

اعمال مدل SVM بر روی دسته ویژگی های اول :

مقادیر دقت روی داده های تست و اموزش و f1 score و jaccard score و همچنین ماتریس confusion به شرح زیر می باشد.

```
Train_accuracy     : 45.7562%
Test_accuracy      : 44.4600%
f1_score           : 44.381%
jaccard_score      : 28.622%
```

اعمال مدل SVM بر روی دسته ویژگی های دوم :

مقادیر دقت روی داده های تست و اموزش و f1 score و jaccard score و همچنین ماتریس confusion به شرح زیر می باشد.

```
Train_accuracy     : 96.3820%
Test_accuracy      : 63.3800%
f1_score           : 63.343%
jaccard_score      : 46.429%
```

تحلیل نتایج بدست امده :

دقت مدل روی دسته اول برای داده آموزش ۴۵/۷٪ و برای داده تست ۴۴/۴٪ بوده است. پس بیش برازش اتفاق نیفتاده است. اما دقت این مدل هم مثل مدل قبل روی این دسته ویژگی برای تفکیک احساسات اصلا مناسب نیست. دقت مدل برای دسته ویژگی دوم بهبود یافته است چون ویژگی های این دسته حتی برای تفکیک احساس هم بهتر هستند. روی دسته دوم دقت مدل برای داده آموزش۹۶/۳٪ و برای داده تست ۶۳/۴٪ به دست آمده است. دقت روی داده آموزش و تست بسیار بهتر از مدل mlp است. اما باز هم بیش برازش اتفاق افتاده است که در این مورد هم در انتها بحث خواهد شد. دقت روی داده تست قابل قبول است.

مدل Random Forest برای طبقه بندی احساسات :

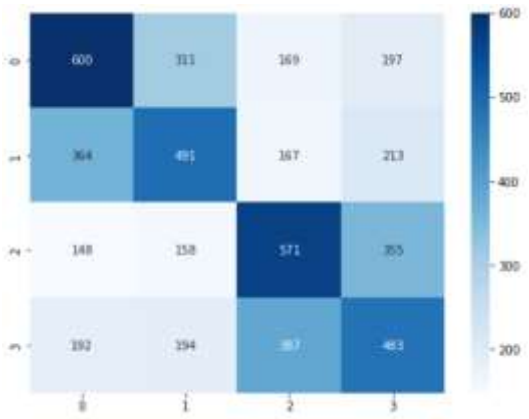
مدل مورد نظر به صورت زیر می باشد .

```
clf = RandomForestClassifier(n_estimators=2000 , random_state=42 , max_depth=50 )
clf.fit(X_train,y_train)
y_pred_test=clf.predict(X_test)
```

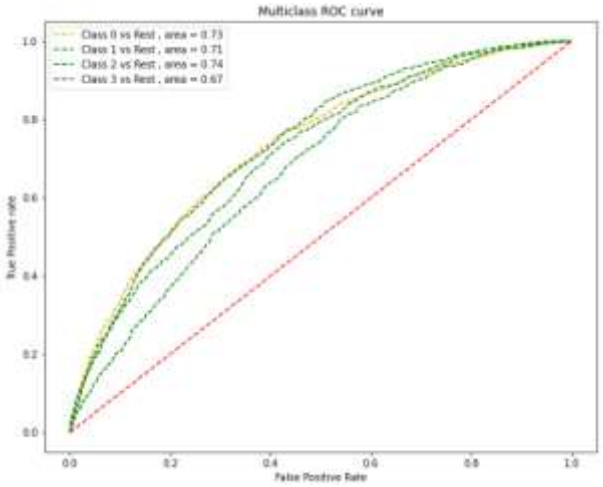
مقادیر دقت روی داده های تست و اموزش و f1 score و jaccard score و همچنین ماتریس confusion به شرح زیر می باشد.

اعمال مدل Random Forest بر روی دسته ویژگی های اول :

```
Train_accuracy    : 99.9228%
Test_accuracy    : 42.9000%
f1_score          : 42.847%
jaccard_score     : 27.316%
```

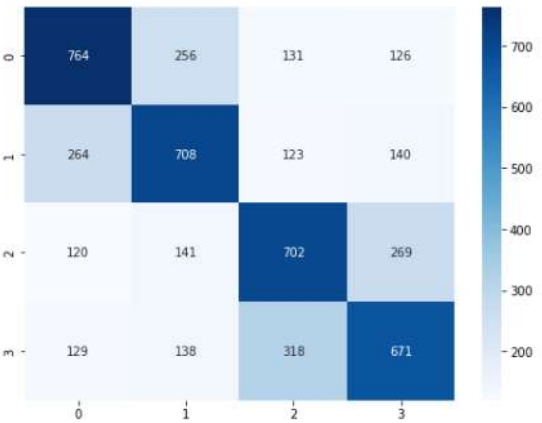


نمودار ROC مدل :

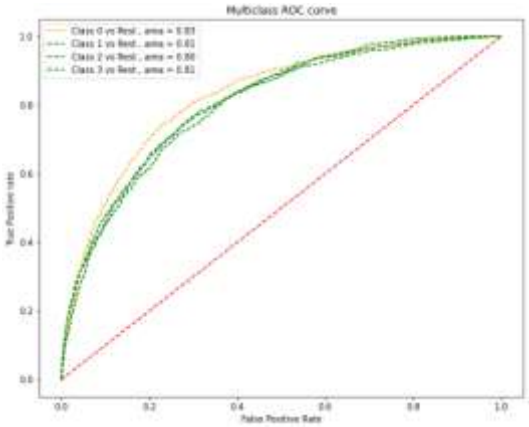


اعمال مدل Random Forest بر روی دسته ویژگی های دوم :

```
Train_accuracy    : 99.9228%
Test_accuracy    : 56.9000%
f1_score          : 56.876%
jaccard_score     : 39.765%
```



نمودار ROC مدل :



تحلیل نتایج بدست امده :

دقت مدل روی دسته اول برای داده آموزش ۹۹/۹٪ و برای داده تست ۴۲/۹٪ بوده است. با وجود استفاده از مدل جنگل تصادفی بیش برازش بسیار زیادی اتفاق افتاده است. دقت مدل روی این داده تست دسته ویژگی برای تفکیک احساسات حتی از دقت رندوم هم پایینتر است و اصلا مناسب نیست. روی دسته دوم دقت مدل برای داده آموزش ۹۹/۹٪ و برای داده تست ۵۶/۹٪ به دست آمده است. روی دسته دوم ویژگی هم بیش برازش زیادی رخ داده اما حداقل دقت روی داده تست بیشتر از طبقه بند رندوم است و میتوان به آن اتکا کرد.

در مورد بیش برازش در داده احساسات میتوان تحلیل کرد که در روشهای انتخاب ویژگی ۱۰ صدای هر سوژه با یک عاطفه مشخص برچسب گذاری شده است. پس همبستگی این وویسها برای یک شخص زیاد است. ممکن است دو شخص که یک جمله را با یک عاطفه را گفته اند همبسگی آنچنان زیادی نداشته باشند. پس داده های یک عاطفه خاص برای هر شخص شباهت زیادی دارند. این ممکن است منجر به overfit شود. تعداد داده های مشابه در قسمت آموزش است و همبستگی بالایی وجود دارد و انگار چند داده کاملا مثل هم را وارد مدل کردیم و مدل دارد به جای یادگیری حفظ میکند. به ازای هر نفر ۴۰ داده داریم ولی اگر داده های ورودی متفاوت بودند و مثلا از هر فرد فقط ۴ داده میرفتیم و از هر عاطفه برای هر فرد یک وویس در داده موجود بود دیگر این شباهت و اورفیتینگ پیش نمی امد. اورفیتینگ در جایی اتفاق میافتد که دیتاست بالانس نباشد. مدلهای ساده احتمال اورفیتینگ بیشتری دارد. ولی مثلا cnn اگر میزدیم اینطور نمیشد.

مقایسه مدلها برای تفکیک احساسات :

همانطور که در جدول زیر نمایش داده شده است ، مدل SVM بر روی دسته ویژگی های دوم دارای عملکرد بهتر قابل توجهی نسبت به سایر مدل ها دارا است. دقت مدل جنگل تصادفی از mlp هم بهتر است. باز هم این مورد انتظار بود چون قبل از ابداع روش cnn و روشهای یادگیری عمیق بهترین مدل همین svm بوده است.

مدل	دقت روی دسته ویژگی اول		دقت روی دسته ویژگی دوم	
	train	test	train	test
MLP	45.42%	43.9%	78.79%	٪۵۲,۶۶
SVM	45.75%	٪۴۴,۴۶	96.38%	٪۶۳,۳۸
Random Forest	92.92%	٪۴۲,۹	92.92%	٪۵۶,۹

خوشه‌بندی

با اینکه دیتاست پروژه برچسب‌گذاری شده است در این بخش قصد داریم برچشبها را نادیده بگیریم و داده‌ها را بر اساس دو روش k-means و GMM خوشه بندی کنیم. در نهایت با توجه به معیارهای ارزیابی عملکرد خوشه‌بند را بررسی خواهیم کرد. برای ارزیابی عملکرد خوشه‌بند از معیار silhouette استفاده کردیم. این معیار هم به پیوستگی (Cohesion) درون خوشه‌ها و هم به میزان تفکیک‌پذیری آن‌ها بستگی دارد. مقدار نیم‌رخ برای هر نقطه، میزان تعلق آن را به خوشه‌اش در مقایسه با خوشه مجاور اندازه می‌گیرد. محدوده سیلوئت از ۱− تا ۱+ است، که در آن مقدار زیاد نشان می‌دهد که شی به خوبی با خوشه خود مطابقت دارد و با خوشه‌های همسایه همسان نیست. اگر بیشتر اشیا از مقدار بالایی برخوردار باشند، ساختار خوشه بندی مناسب است. اگر بسیاری از نقاط دارای مقدار کم یا منفی باشند، در این صورت ممکن است ساختار خوشه بندی دارای خوشه‌های بسیار زیاد یا بسیار کم باشد.

مشابه قسمت طبقه بندی در این بخش هم دو دسته ویژگی در نظر گرفتیم و هر بار یک دسته را به خوشه بند مورد نظر دادیم.

روش k-means

برای ورودی این قسمت همه فضای ویژگی‌های استخراج شده را در نظر میگیریم. همانطور که خواسته شده الگوریتم را برای ۲، ۴، ۱۰ و تعداد کل شرکت‌کننده ها که ۴۰۰ تخمین زده شده است پیاده سازی میکنیم. برای این کار از دستور KMeans استفاده میکنیم و تعداد خوشه را مشخص میکنیم. سپس پارامترهای max_iter که مربوط به تعداد ماکسیمم تکرار الگوریتم است را ۳۰۰ قرار میدهیم و n_init که مربوط به دفعات تکرار کل الگوریتم برای در نظر گرفتن انواع نقاط شروع رندوم برای شروع الگوریتم است را ۱۰ قرار میدهیم.

در این پروژه از الگوریتم K-Means++ استفاده شده است. در الگوریتمK-Means++ ، کاربر می‌خواهد که مرکزوارها در مقداردهی اولیه تا حد ممکن از هم دور باشند. ایده نهفته در پس این روش آن است که مرکزوارها به مرکز خوشه‌های واقعی نزدیک‌تر باشند و بنابراین سریع‌تر به همگرایی برسند. در واقع یک الگوریتم تخمین برای مسائل KMeans ، NP-hard است.

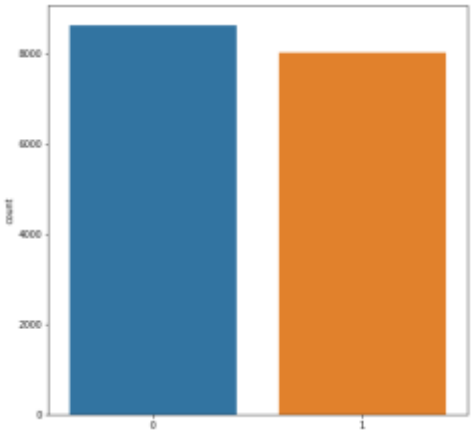
K=2 دو خوشه

برای دسته ویژگی اول:

توقع داریم این ۲ دسته متناظر با توزیع برچسب جنسیت زن و مرد در داده اولیه باشد.

```
model = KMeans(n_clusters = 2, init = "k-means++", max_iter = 300, n_init = 10, random_state = 0)
y_clusters = model.fit_predict(X1)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X1,y_clusters)
silhouette
```

```
0.2349702368059524
```

در جدول زیر فراوانی جنسیت داده ها در هر خوشه نمایش داده شده است.

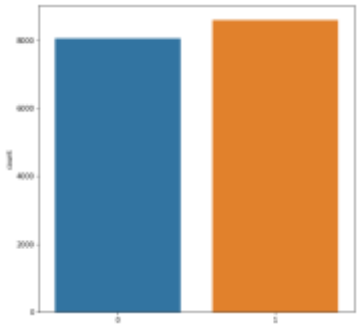
Actual label / Cluster label	0	1
0	3669	4964
1	4278	3753

برای دسته ویژگی دوم:

مدل اموزش داده به صورت زیر می باشد.

```
model = KMeans(n_clusters = 2, init = "k-means++", max_iter = 300, n_init = 10, random_state = 0)
y_clusters = model.fit_predict(X2)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X2,y_clusters)
silhouette
```

```
0.10781839934972111
```

در جدول زیر فراوانی جنسیت داده ها در هر خوشه نمایش داده شده است.

Actual label / Cluster label	0	1
0	3549	4513
1	4398	4204

مقدار silhouette برای دسته ویژگی اول بسیار بهتر است و این برخلاف دقت در طبق بندها میباشد. دسته دوم ویژگی ابعاد بسیار بزرگتری دارد و چون الگوریتم بر حسب فاصله کار میکند و فاصله در ابعاد بالا خیلی زیاد میشود نتیجه خوشه بندی بدتر میشود.

K=4 دو خوشه

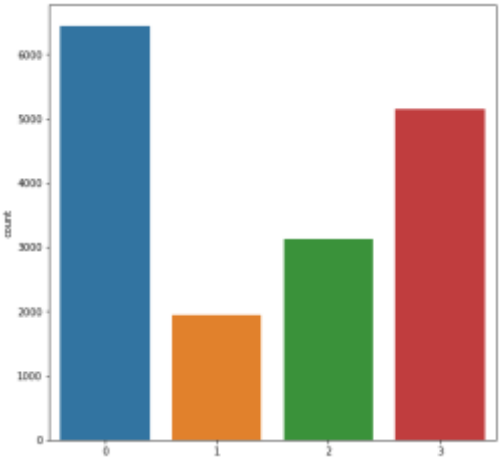
برای دسته ویژگی اول:

توقع داریم این ۴ دسته متناظر با توزیع برچسب ۴ احساس خشم، غم، شادی و خنثی در داده اولیه باشد.

مدل اموزش داده به صورت زیر می باشد.

```
model = KMeans(n_clusters = 4, init = "k-means++", max_iter = 300, n_init = 10, random_state = 0)
y_clusters = model.fit_predict(X1)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X1,y_clusters)
silhouette
```

```
0.24170465861133994
```

در جدول زیر فراوانی نوع احساس داده ها در هر خوشه نمایش داده شده است.

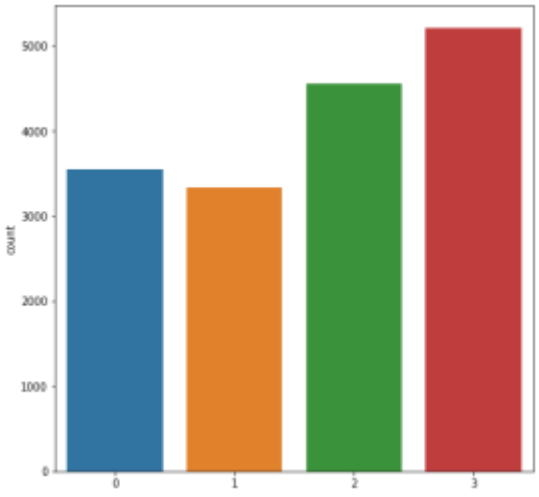
Actual label / Cluster label	1	2	3	4
0	1439	1265	1858	1885
1	575	350	504	511
2	1366	1165	285	310
3	784	1364	1532	1471

برای دسته ویژگی دوم:

مدل اموزش داده به صورت زیر می باشد.

```
model = KMeans(n_clusters = 4, init = "k-means++", max_iter = 300, n_init = 10, random_state = 0)
y_clusters = model.fit_predict(X2)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X2,y_clusters)
silhouette
```

```
0.07514463033931845
```

در جدول زیر فراوانی نوع احساس داده ها در هر خوشه نمایش داده شده است.

Actual label / Cluster label	1	2	3	4
0	827	673	1065	990
1	1138	993	614	593
2	1227	1409	928	989
3	972	1069	1572	1605

برای تعداد ۴ خوشه هم مقدار silhouette برای دسته ویژگی اول بسیار بهتر است و این برخلاف دقت در طبق بندها میباشد. تحلیل این رخداد هم مانند حالت دو خوشه ای به خاطر ابعاد بالای دسته ویژگی دوم است.

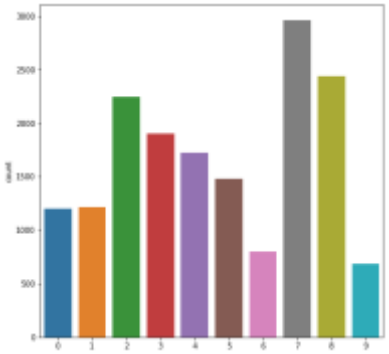
K=10 دو خوشه

برای دسته ویژگی اول:

توقع داریم این ۱۰ دسته متناظر با توزیع برچسب ۱۰ TEXT_ID در داده اولیه باشد. مدل اموزش داده به صورت زیر می باشد.

```
model = KMeans(n_clusters = 10, init = "k-means++", max_iter = 300, n_init = 10, random_state = 0)
y_clusters = model.fit_predict(X1)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X1,y_clusters)
silhouette
```

```
0.1858674891168136
```

در جدول زیر فراوانی text id داده ها در هر خوشه نمایش داده شده است.

Actual label / Cluster label	1	2	3	4	5	6	7	8	9	10
0	129	108	149	92	155	248	80	99	65	75
1	164	220	108	88	96	76	152	106	103	104
2	222	161	185	251	262	234	223	212	255	237

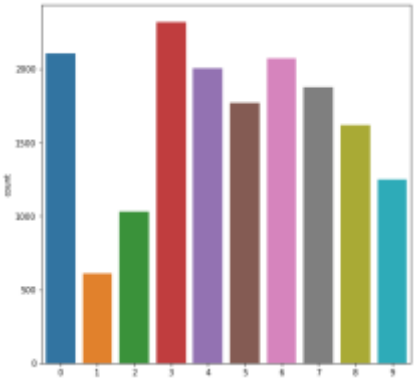
3	199	234	197	157	189	148	182	199	190	202
4	154	106	233	255	132	263	137	197	134	109
5	170	219	121	119	149	74	134	136	145	186
6	67	76	30	82	108	62	107	65	87	116
7	260	246	336	342	267	275	291	309	343	292
8	249	260	258	218	234	149	260	265	287	260
9	73	36	89	84	62	127	57	65	43	52

برای دسته ویژگی دوم:

مدل اموزش داده به صورت زیر می باشد.

```
model = KMeans(n_clusters = 10, init = "k-means++", max_iter = 300, n_init = 10, random_state = 0)
y_clusters = model.fit_predict(X2)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X2,y_clusters)
silhouette
```

```
0.06152897869811592
```

در جدول زیر فراوانی text id داده ها در هر خوشه نمایش داده شده است.

Actual label / Cluster label	1	2	3	4	5	6	7	8	9	10
0	208	172	315	210	188	334	170	218	155	134
1	58	67	66	58	73	63	57	53	56	57
2	97	107	43	101	113	68	150	103	102	148
3	204	202	321	240	244	259	219	221	206	202
4	213	254	125	203	241	135	224	167	226	214
5	194	231	102	137	159	89	181	177	256	242
6	214	189	245	249	215	243	197	191	166	164
7	197	184	179	200	133	175	184	226	212	183
8	190	139	148	160	135	143	163	196	162	184
9	112	121	162	130	153	147	108	101	111	105

باز هم مقدار silhouette برای دسته ویژگی اول بسیار بهتر است و این هم برخلاف دقت در طبق بندها میباشد. علت مشابه قبل است.

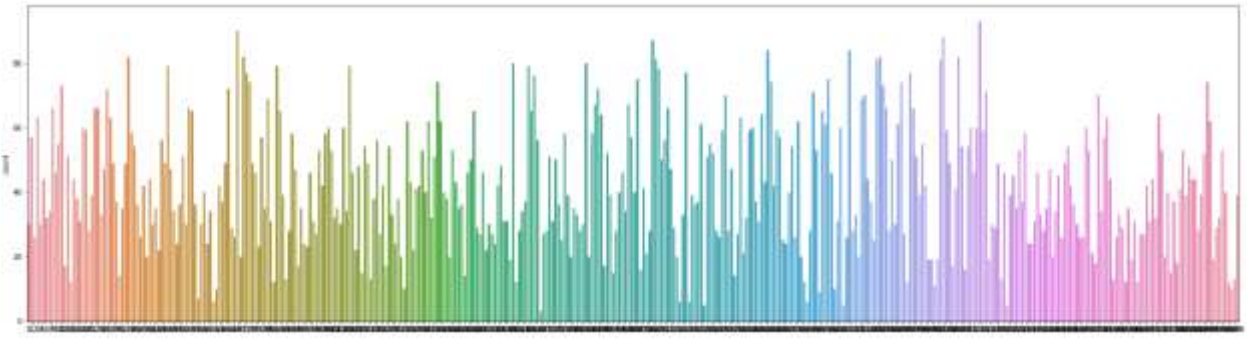
K=400 دو خوشه

برای دسته ویژگی اول:

توقع داریم این ۱۰ دسته متناظر با توزیع برچسب ۴۰۰ شرکت کننده در داده اولیه باشد. مدل اموزش داده به صورت زیر می باشد.

```
model = KMeans(n_clusters = 400, init = "k-means++", max_iter = 300, n_init = 10, random_state = 0)
y_clusters = model.fit_predict(X1)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X1,y_clusters)
silhouette
```

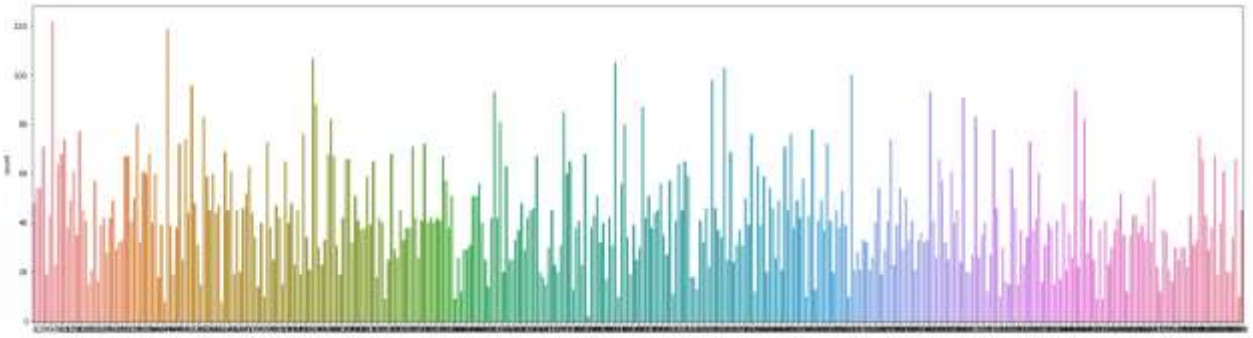
```
0.14528973950497076
```

برای دسته ویژگی دوم:

. که این مدل اموزش داده به صورت زیر می باشد.

```
model = KMeans(n_clusters = 400, init = "k-means++", max_iter = 300, n_init = 10, random_state = 0)
y_clusters = model.fit_predict(X2)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X2,y_clusters)
silhouette
```

```
0.1231195596116675
```

برای این تعداد خوشه مقدار silhouette برای دسته ویژگی اول کمی بهتر است و این هم برخلاف دقت در طبق بندها میباشد.

به طور تقریبی مقدار silhouette با افزایش تعداد خوشه کاهش یافته است. k-means برای داده های کروی خوش شکل مناسب است و بر اساس فاصله عمل میکند. برای همین برای خوشه بندی این دیتاست انتخاب بهینه ای نیست. میدانیم با افزایش بیش از حد تعداد خوشه ها این امر قابل انتظار است.

الگوریتم GMM

در این شیوه خوشه‌بندی، فرض بر این است که هر خوشه از داده‌هایی با توزیع نرمال (گوسی) تشکیل شده و در حالت کلی نیز داده‌ها نمونه‌ای از توزیع آمیخته نرمال هستند. هدف از خوشه‌بندی مدل آمیخته گوسی یا نرمال، برآورد پارامترهای توزیع هر یک از خوشه‌ها و تعیین برچسب برای مشاهدات است. به این ترتیب مشخص می‌شود که هر مشاهده به کدام خوشه تعلق دارد. چنین روشی را در یادگیری ماشین، خوشه‌بندی برمبنای مدل می‌نامند. روش‌های خوشه‌بندی مانند kmeans قادر به شناسایی خوشه‌های کروی هستند و اگر ساختار خوشه‌ها خارج از این شکل باشد، عمل خوشه‌بندی توسط این الگوریتم‌ها به خوبی صورت نمی‌گیرد. به عنوان یک راه حل در چنین مواردی می‌توان از خوشه‌بندی برمبنای مدل یا همان مدل آمیخته گوسی استفاده کرد. البته با توجه به ساختار خوشه‌های می‌توان از توزیع‌های دیگری نیز در این میان کمک گرفت ولی به طور معمول فرض می‌شود که مشاهدات مربوط به خوشه‌ها دارای توزیع نرمال هستند.

K=2

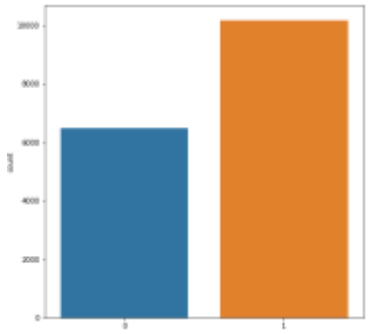
برای دسته ویژگی اول:

توقع داریم این ۲ دسته متناظر با توزیع برچسب ۲ جنسیت زن و مرد در داده اولیه باشد. مدل اموزش داده به صورت زیر می باشد.

```
gmm = GaussianMixture(n_components=2, covariance_type='full').fit(X1)

y_clusters = gmm.predict(X1)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X1,y_clusters)
silhouette
```

```
0.20176181348124786
```

در جدول زیر فراوانی جنسیت داده ها در هر خوشه نمایش داده شده است.

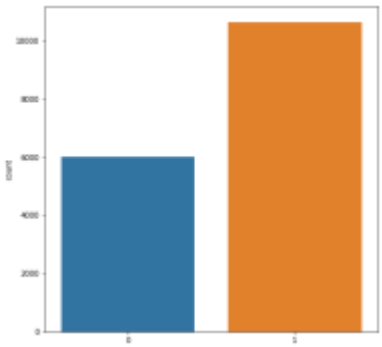
Actual label / Cluster label	0	1
0	۲۸۶۰	۳۶۳۱
1	۵۰۸۷	۵۰۸۶

برای دسته ویژگی دوم:

تعداد ۸۶۳۳ در خوشه اول قرار میگیرد و تعداد ۸۰۳۱ خوشه در دسته دوم. در جدول زیر اطلاعات مربوط به این دو خوشه گزارش شده است. معیار silhouette برای روش GMM دو خوشه ای برابر با ۰/۲۳ شد. که این مقدار بالاترین مقدار در روش GMM بود. مدل اموزش داده به صورت زیر می باشد.

```
gmm = GaussianMixture(n_components=2, covariance_type='full').fit(X2)
y_clusters = gmm.predict(X2)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X2,y_clusters)
silhouette
```

```
0.09620584599600727
```

در جدول زیر فراوانی جنسیت داده ها در هر خوشه نمایش داده شده است.

Actual label / Cluster label	0	1
0	5278	5375
1	2669	3342

مانند خوشه بندی kmeans برای GMM مقدار silhouette برای دسته ویژگی اول بسیار بهتر است و این برخلاف دقت در طبق بندها میباشد. دسته دوم ویژگی ابعاد بسیار بزرگتری دارد و دچار نحصی ابعاد میشویم.

K=4

برای دسته ویژگی اول:

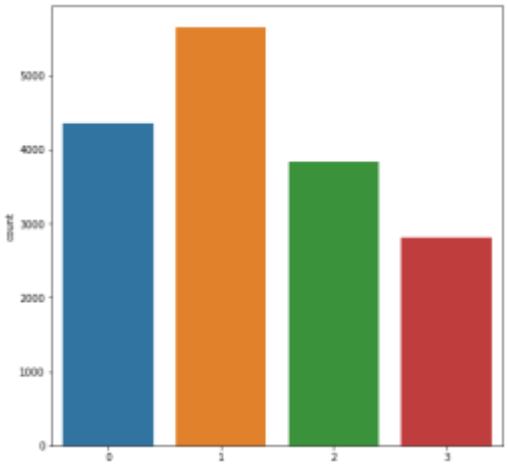
توقع داریم این ۴ دسته متناظر با توزیع برچسب ۴ احساس خشم، غم، شادی و خنثی در داده اولیه باشد.

مدل اموزش داده به صورت زیر می باشد.

```
gmm = GaussianMixture(n_components=4, covariance_type='full').fit(X1)

y_clusters = gmm.predict(X1)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X1,y_clusters)
silhouette
```

```
0.11373828056357054
```

در جدول زیر فراوانی نوع احساس داده ها در هر خوشه نمایش داده شده است.

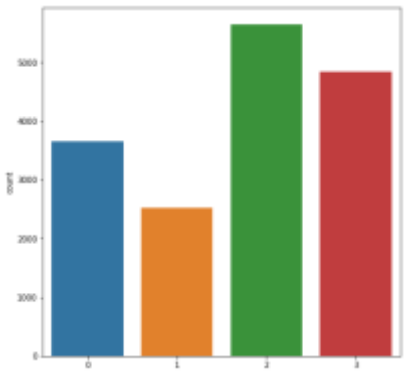
Actual label / Cluster label	1	2	3	4
0	۸۵۳	۱۲۹۰	۱۱۱۰	۱۱۰۱
1	۸۲۸	۹۰۰	۲۰۳۳	۱۸۹۹
2	۱۳۵۳	۹۴۲	۶۸۶	۸۵۹
3	۱۱۳۰	۱۰۱۲	۳۵۰	۳۱۸

برای دسته ویژگی دوم:

تعداد ۸۶۳۳ در خوشه اول، تعداد ۸۰۳۱ عدد در خوشه دوم، تعداد * در خوشه سوم و تعداد در خوشه چهارم قرار میگیرد. در جدول زیر اطلاعات مربوط به این دو خوشه گزارش شده است. معیار silhouette برای روش GMM دو خوشه ای برابر با ۰/۲۳ شد. مدل اموزش داده به صورت زیر می باشد.

```
gmm = GaussianMixture(n_components=4, covariance_type='full').fit(X2)
y_clusters = gmm.predict(X2)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X2,y_clusters)
silhouette
```

```
0.012251905721506032
```

در جدول زیر فراوانی نوع احساس داده ها در هر خوشه نمایش داده شده است.

Actual label / Cluster label	1	2	3	4
0	1300	1269	585	537
1	1224	1045	1247	1346
2	889	988	1906	1796
3	751	842	441	498

برای این تعداد خوشه هم مقدار silhouette برای دسته ویژگی اول بسیار بهتر است و این برخلاف دقت در طبق بندها میباشد. معیار ارزیابی خوشه بندی برای دسته دوم نزدیک به صفر است و خوشه بندی اصلا قابل قبل نیست. دسته دوم ویژگی ابعاد بسیار بزرگتری دارد و دچار نحصی ابعاد میشویم.

K=10

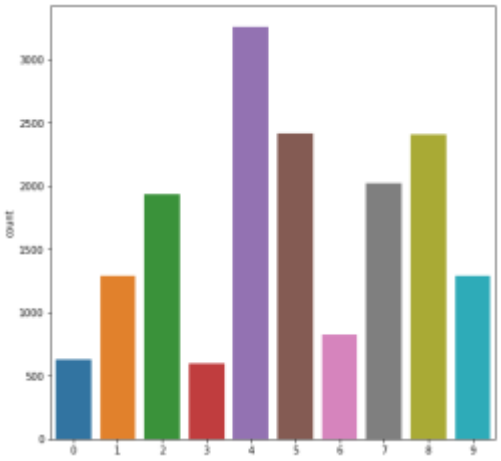
برای دسته ویژگی اول:

توقع داریم این ۱۰ دسته متناظر با توزیع برچسب ۱۰ TEXT_ID در داده اولیه باشد مدل اموزش داده به صورت زیر می باشد.

```
gmm = GaussianMixture(n_components=10, covariance_type='full').fit(X1)

y_clusters = gmm.predict(X1)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X1,y_clusters)
silhouette
```

```
0.05637521668995065
```

در جدول زیر فراوانی text id داده ها در هر خوشه نمایش داده شده است.

Actual label / Cluster label	1	2	3	4	5	6	7	8	9	10
0	۸۱	۱۰۵	۶۳	۴۲	۶۰	۴۶	۵۹	۶۷	۴۱	۶۱
1	۱۳۱	۶۹	۱۷۰	۱۷۹	۹۸	۲۲۷	۱۰۵	۱۲۳	۱۰۶	۸۱
2	۲۰۵	۲۵۶	۱۵۰	۱۵۷	۱۷۴	۸۰	۲۳۷	۲۱۰	۲۱۲	۲۵۴
3	۵۸	۳۳	۷۹	۶۸	۵۹	۱۱۶	۴۸	۵۶	۳۷	۴۴
4	۳۲۳	۳۳۱	۳۶۸	۳۳۰	۳۱۱	۳۲۸	۳۳۰	۳۵۰	۳۱۲	۲۷۳
5	۲۲۷	۲۱۹	۲۶۷	۲۶۸	۲۲۴	۲۱۳	۲۲۲	۲۵۳	۲۶۵	۲۵۵

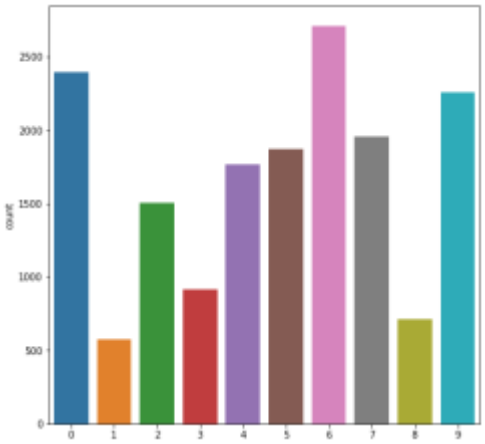
6	۸۱	۸۸	۴۵	۷۶	۹۴	۷۲	۱۰۹	۷۶	۶۸	۱۱۳
7	۲۰۰	۲۵۱	۱۸۸	۱۷۹	۱۷۷	۹۶	۲۳۲	۲۰۲	۲۴۶	۲۴۷
8	۲۳۹	۱۸۴	۲۱۵	۲۸۰	۲۹۰	۲۵۸	۲۱۰	۲۰۶	۲۹۲	۲۲۹
9	۱۴۲	۱۳۰	۱۶۱	۱۰۹	۱۶۷	۲۲۰	۱۰۱	۱۱۰	۷۳	۷۶

برای دسته ویژگی دوم:

مدل اموزش داده به صورت زیر می باشد.

```
gmm = GaussianMixture(n_components=10, covariance_type='full').fit(X2)
y_clusters = gmm.predict(X2)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X2,y_clusters)
silhouette
```

```
0.0019084484354461647
```

در جدول زیر فراوانی text id داده ها در هر خوشه نمایش داده شده است.

Actual label / Cluster label	1	2	3	4	5	6	7	8	9	10
0	۲۳۱	۲۵۶	۲۳۲	۲۴۳	۲۴۹	۱۸۱	۲۵۱	۲۳۴	۲۵۲	۲۶۷
1	۶۱	۶۰	۵۳	۵۶	۵۶	۵۳	۶۰	۵۵	۶۰	۶۰
2	۱۴۰	۱۵۹	۱۱۵	۱۳۹	۱۶۵	۱۳۲	۱۷۲	۱۴۹	۱۵۳	۱۷۹
3	۹۴	۸۷	۱۰۷	۹۶	۸۴	۹۰	۸۱	۹۹	۸۵	۹۵
4	۱۹۶	۱۵۲	۱۵۳	۱۹۵	۱۶۳	۱۶۸	۱۷۸	۲۰۷	۱۶۳	۱۸۵
5	۱۸۱	۱۵۱	۲۳۶	۱۷۹	۱۷۸	۲۱۱	۱۸۲	۱۹۳	۱۹۰	۱۷۱
6	۲۷۳	۲۷۸	۲۹۶	۲۹۷	۲۶۰	۲۶۵	۲۵۰	۲۶۶	۲۶۷	۲۵۹
7	۲۱۲	۲۲۶	۱۸۳	۱۷۶	۲۰۸	۲۳۲	۱۹۰	۱۵۲	۱۹۰	۱۸۵
8	۷۰	۶۷	۸۳	۷۲	۷۰	۷۹	۶۹	۶۹	۶۹	۶۲
9	۲۲۹	۲۳۰	۲۴۸	۲۳۵	۲۲۱	۲۴۵	۲۲۰	۲۲۹	۲۲۳	۱۷۰

ایین برای این تعداد خوشه silhouette برای هر دو دسته ویژگی . معیار ارزیابی خوشه بندی برای دسته دوم نزدیک به صفر است ولی باز هم برای دسته ویژگی اول بهتر عمل کرده است. برای هر دسته خوشه بندی قابل قبول نیست.

K=400

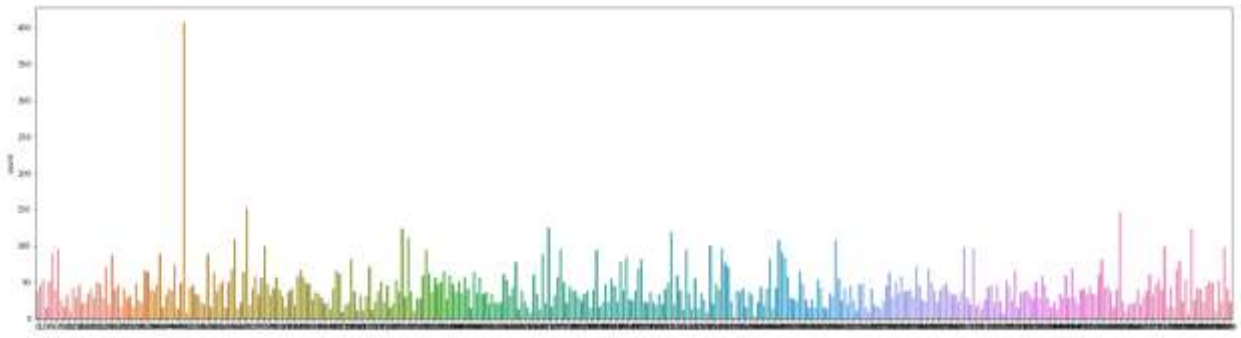
برای دسته ویژگی اول:

توقع داریم این ۱۰ دسته متناظر با توزیع برچسب ۴۰۰ شرکت کننده در داده اولیه باشد. مدل اموزش داده به صورت زیر می باشد.

```
gmm = GaussianMixture(n_components=400, covariance_type='full').fit(X1)

y_clusters = gmm.predict(X1)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X1,y_clusters)
silhouette
```

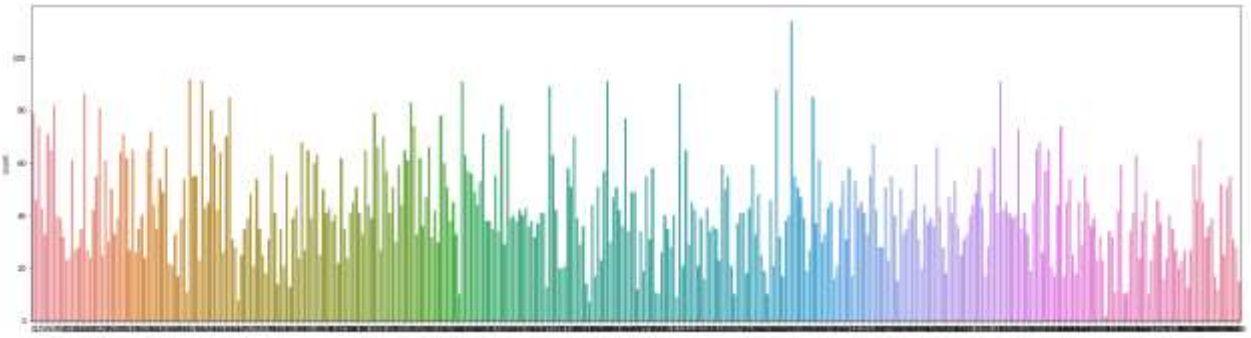
```
-0.04428584678315022
```

برای دسته ویژگی دوم:

. معیار silhouette برای روش GMM دو خوشه ای برابر با ۰/۲۳ شد. مدل اموزش داده به صورت زیر می باشد.

```
gmm = GaussianMixture(n_components=400, covariance_type='full').fit(X2)
y_clusters = gmm.predict(X2)
```

تعداد داده های موجود در هر خوشه :



مقدار silhouette بدست امده برای مدل :

```
silhouette = silhouette_score (X2,y_clusters)
silhouette
```

```
0.11932491431225853
```

برای این خوشه بندی معیار ارزیابی خوشه بندی برای دسته اول منفی است که اصلا قابل قبول نیست. silhouette دسته دوم مثبت شد ولی بسیار نزدیک به صفر بود. به طور کلی روی این تعداد خوشه الگوریتم اصلا خوب عمل نکرد.

در این الگوریتم خوشه بندی هم به طور تقریبی با افزایش تعداد خوشه ها مقدار silhouette کاهش یافته است. میدانیم با افزایش بیش از حد تعداد خوشه ها این امر قابل انتظار است.

به طور کلی نتایج k-means برای همه تعداد خوشه ها از GMM بهتر بود. علت احتمالا این است که الگوریتم GMM تعدادی گاوسی را برای تووزیع کل نمونه ها در نظر میگیرد و داده های ما به خوبی از این توزیع های گاوسی پیروی نمیکنند. پس روش مبتنی بر فاصله k-means جواب بهتر به دست داده است.