پروژه نهایی درس یادگیری ماشین

Siavash Razmi, Mohsen Ahmand, Mahyar Maleki, Amin Qahramani

LaTeX

فهرست مطالب

فصل ۱

مقدمه

به طور کلی هدف از این پروژه انجام صفر تا صد یک پروژه واقعی یادگیری ماشین است. در این پروژه ما با چالشهایی روبه رو خواهیم شد که بعضاً در دیگر پروژه های دانشگاهی یادگیری ماشین با آن روبه رو نبوده ایم. به عنوان مثال در اکثر پروژه ها داده (بعضاً پیش پردازش شده) در اختیار ما قرار می گرفت اما ما در این پروژه بخش جمع آوری داده را خواهیم داشت.

فصل ۲

تعریف مسئله و شرح پروژه

۱.۲ تعریف کلی مسئله

هدف نهایی این پروژه طبقه بندی و خوشه بندی ساز ها به صورت تکنوازی در ۶ دسته بندی ویولون، تار، سه تار، سنتور، نی و پیانو است. که ما برای هر کدام از این دسته ها داده های مورد نظر(تکنوزی) را جمع آوری کرده ایم.

۲.۲ جمع آوری داده

هر دانشجو بصورت انفرادی برای هر گروه مشخص شده ۵ آهنگ جمع آوری کرده است(یعنی هر نفر ۳۰ آهنگ) که هیچکدام از آهنگ ها تکراری نیست و برای یکدست بودن داده ها همگی آهنگ ها در فرمت (128 kb mp3) انتخاب شده اند.

۳.۲ گزارش اولیه

در این مرحله بصورت گروهی گزارشی تهیه کردیم و در آن راجع به انواع این ساز ها و آهنگ های ساخته شده با آن ها اطلاعاتی را شرح دادیم.

۴.۲ گروه بندی

نام	نام خانوادگی	شماره دانشجویی
سیاوش	رزمی	۸۱۰۱۰۰۳۵
محسن	آهمند	۸۱۰۱۰۰۲۹۵
مهیار	ملکی	۸۱۰۱۰۰۴۷۶
امین	قهرمانی	۸۱۰۱۰۰۴۳۹

۵.۲ تمیز سازی داده ها

حال پس از جمع آوری داده ها لازم است با پردازش اولیه داده ها آن ها را آماده برای مراحل بعدی پروژه کنیم[1]
تمیز سازی داده ها نیز بخشی حیاتی از فرایند تجزیه وتحلیل داده ها است.

اگر داده های ما مغایرت یا خطایی داشته باشد، می توانیم پیش بینی کنیم که نتایج ما نیز ناقص خواهند بود.

تمیزسازی داده‌ها فقط مورد حذف داده‌های اشتباه نیست، اگرچه این مهم اغلب بخشی از آن است. در مبحث تمیزسازی داده‌ها، تمرکز اصلی بیشتر بر روی شناسایی داده‌های متقلب و (در صورت امکان) اصلاح آنها، انجام می‌شود.

"داده‌های متقلب" شامل مواردی مانند داده‌های ناقص، نادرست، بی‌ربط، خراب یا با قالب‌بندی نادرست هستند. این فرایند همچنین شامل تکثیر یا "برداشتن" است که به طور موثر به معنی ادغام یا حذف نقاط داده یکسان می‌باشد.

اما چرا اصلاح این نوع خطاها تا این حد مهم است؟

پاسخ به اندازه کافی ساده است: اگر این کار را نکنیم، آنها بر نتایج تجزیه و تحلیل ما تأثیر می‌گذارند. از آنجا که تجزیه و تحلیل داده‌ها معمولا برای اطلاع رسانی تصمیمات کسب و کار مورد استفاده قرار می‌گیرد، نتایج باید دقیق باشند در این حالت، به سادگی حذف داده‌های متقلب یا ناقص ایمن‌تر به نظر می‌رسد.

اما این مسئله نیز مشکلاتی را ایجاد می‌کند: "یک مجموعه داده ناقص نیز بر نتایج تجزیه و تحلیل ما تأثیر می‌گذارد". به همین دلیل یکی از اهداف اصلی تمیزسازی داده‌ها، سالم نگه داشتن هر چه بیشتر یک مجموعه داده است. این امر به بهبود قابلیت اطمینان بینش ما، کمک می‌کند.

[1] Preparing Data for Machine Learning

که در این پروژه ما داده های خراب، بی کیفیت و آهنگ هایی که ترکیبی از چند ساز بودن را از آهنگ های سالم بصوت دستی جداسازی کردیم.

۶.۲ پیش پردازش

تمییز سازی : تعداد زیادی از نمونه فایل های ارسالی مناسب برای استفاده در پروژه نبودند به طور مثال دارای صدای های اضافه (صدای ساز یا صدای آواز)، یا مربوط به ساز های غیر از سازهای مشخص شده (مثلا به جای نی صدای دودوک یا فلوت و یا ویالن سل بجای ویالون) که تمامی اینها توسط اعضای گروه بررسی و نمونه های نامناسب جداسازی شد.

بارگذاری :

۱ - اسکن کردن دایرکتوری نمونه ها:

درابتدا نام دایرکتوری که نمونه ها در آن ذخیره شده اند در متغیر path ذخیره می شود، اسم تمام ساز ها در آرایه ایی به نام instruments ذخیره می گردد سپس یک حلقه برای تمامی فولدر هایی که نامشان در instrument هست اجرا می شود و نام فایل هایی که با پسوند mp3 در این سابفولدر ها هستند به ارایه files و همچنین نام ساز را نیز به آرایه labels اضافه میکند.

Label encoding - 2

برای استفاده از کتابخانه های sklearn نیاز است که لیبل های فایل به فرمت عددی به ماژول ها داده شود بنابرین با استفاده labelEncoder این لیبل هارا به عدد تبدیل میکنیم.

برای بارگذاری و پیش پردازش آهنگ ها از پکیج librosa استفاده شد، این پکیج مشخصاً برای کاربرد های MIR (Music information retrieval) طراحی شده است.
برای بارگذاری یک فایل صوتی نیاز به مشخص کردن نرخ نمونه برداری یا Sampling rate هست که در ثانیه از فایل نمونه گرفته می شود، هرچه این نرخ دقیق تر باشد نمونه برداری ما دقیق تر اما زمانبرتر است و فضای بیشتری احتیاج دارد، با توجه به این موضوع نرخ ۴۴۱۰۰ نمونه در ثانیه برای اینکار انتخاب شد.

سیگنال صوتی پس از باگذاری توسط librosa به صورت یک آرایه یک بعدی تبدیل می شود که به همراه نرخ نمونه برداری آن به خروجی داده می شود.

به طور پیش فرض در صورتی که فایل ورودی به شکل استریو باشد (خروجی صدا برای دو اسپیکر طراحی شده باشد) متد load در این کتابخانه ابتدا این دو خروجی را با همدیگر میانگین میگیرد و سپس از آن نمونه برداری می کند.

به این جهت که حجم فایل ها بسیار زیاد بود وبرای بارگذاری آنها مدت زمان مدیدی صرف میشد پس بررسی دقت نتیجه، تصمیم گرفته شد که از هر فایل به شکل نمونه های ۳۰ ثانیه ایی رندم انتخاب شود بدین شکل مدت زمان باگذاری به طور چشمگیری کاهش یافت البته با مشخص کردن پارامتر n تعداد نمونه هایی که از هر فایل انتخاب می شود قابل تغییر است.

پس از بارگذاری فایل در قدم بعد هر آرایه را نرمالایز کردیم تا تمام ورودی ها به شکل یکسانی تبدیل شوند و اگر نمونه ایی دارای شدت صدای بالاتر یا پایین تری بود در روند train مدل تداخل ایجاد نکند

حال پس از بارگذاری نمونه به تابعی به نام get_features داده می شود تا ویژگی های آن استخراج گردد.

۳- استخراج ویژگی :

برای استخراج ویژگی های هر نمونه فایل یک تابع تعریف شد که با گرفتن یک نمونه به عنوان ورودی با استفاده از توابع کتابخانه librosa اقدام به استخراج ویژگی ها کرده و در نهایت ۲۶ ویژگی مختلف را به عنوان خروجی تحویل میدهد، در ادامه به شرح هر کدام از این ویژگی ها می پردازیم:

Chromogram : chromo در زبان انگلیسی به معنای نت های ۱۲ گانه موسیقی است و chromogram در واقع نحوه توزیع نت ها در فایل صوتی را در بازه زمان نشان می دهد در کتابخانه librosa با استفاده از متد chroma_stft میتوان این ویژگی را استخراج کرد.

Root-meat-square: که همان جذر میانگین مربعات مقادیر سیگنال صوتی است و با متد rms به دست می آید

Spectral centroid : این مقدار مکان مرکز طیف موج را نشان میدهد که به عنوان معیاری برای مشخص کردن طیف صوت به کار می رود و با دستور spectral_centroid به دست می آید

Spectral_bandwidth : این مقدار میزان انحراف مقادیر از مرکز طیف موج را نمایش می دهد و با دستور spectral_bandwidth به دست می آید.

Spectral rolloff: فرکانسی است که مقدار معینی از انرژی موج در فرکانس های کمتر از آن قرار می گیرد ، به طور مثال فرکانسی را میدهد که بیشتر از ۸۵ درصد از انرژی موج در فرکانس های کمتر از آن قرار دارد و با دستور spectral_rolloff به دست می آید.

Zero crossing rate یا ZCR : میانگین نرخی است که موج در واحد زمان از مقدار مثبت به مقدار

منفی و بالعکس تغییر میکند و در تشخیص گفتار و موسیقی کاربرد زیادی دارد و با دستور zero_crossing_rate به دست می آید

Mel-frequency cepstral coefficients : cepstrum mel-frequency یا mfc نمایش توان طیف موج صوتی در بازه زمان هست که بر حسب تبدیل خطی کسینوس بر روی طیف توان لگاریتمی به دست می آید

MFCC یا coefficients MFC ضرایبی هستند که MFC را تشکیل می دهند، این ضرایب توسط تابع mfcc به دست می آیند و طور پیش فرض این تابع تعداد ۲۰ عدد از این ضرایب را به عنوان خروجی می دهد

لازم به ذکر است که تمامی این توابع برای محاسبه این مقادیر پنجره هایی به طول مشخص را از نمونه انتخاب و این مقادیر را برای آن محاسبه میکنند به طور پیشفرض هر پنجره ۲۰۴۸ sample طول دارد و هر دو پنجره به اندازه ۵۱۲ sample از هم فاصله دارند که این مقادیر با توجه به نرخ نمونه برداری sampling rate که ما برای تابع مشخص میکنیم ممکن است مدت زمان متفاوتی داشته باشند، در اینجا ما مقادیر پیشرفض را تغییر ندادیم

بنابرین هر تابع خروجی را به شکل ماتریسی از مقادیر که برای هر frame محاسبه شده به عنوان خروجی میدهد حال ما برای اینکه یک مقدار را به عنوان ویژگی برای هر فایل صوتی نگهداری کنیم مقادیر را برای تمام پنجره ها میانگین میگیریم.

در نهایت تمامی این مقادیر برای هر فایل نمونه به عنوان یک ردیف در ماتریس feature_vectors ذخیره می شود.

برای راحتی کار و به دلیل اینکه با هردفعه اجرای کد مجبور به انجام دوباره این کار نباشیم پس از محسابه ماتریس feature_vector را با استفاده از ماژول pickle در یک مکان ذخیره میکنیم تا در دفعات بعد تنها لازم به بازگذاری همین فایل باشد.

4 - Standardization :

حال پس از محاسبه ماتریس ویژگی ها برای یکسان کردن مقادیر همه ستون ها اقدام به Standardization آنها میکنیم برای اینکار از ماژول Standardscaler کتابخوانه sklearn استفاده کردیم و پس از fit کردن آن مقادیر جدید را در متغیر Scaled_feature_vector ذخیره کردیم

۷.۲ طبقه بندی

الگوریتم نزدیک‌ترین همسایه[۲]

این الگوریتم (k) نزدیکترین همسایه، یکی از الگوریتم های پر استفاده‌ی یادگیری ماشین است که جزء الگوریتم های بدون پارامتر (یعنی هیچ فرضی در مورد توزیع داده‌ها ندارد) و lazy Learning (زمان یادگیری کوتاه اما زمان حدس زدن طولانی) است. هدف این الگوریتم استفاده از دیتاست هایی است که نقاط داده در آن ها بصورت مجزا بوده و در چندین دسته قرار گرفته‌اند.
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 گزینه ها برای کلاس بندی، استفاده از الگوریتم KNN است.

مزایا و معایب :

از جمله مزایای الگوریتم Knn می توانیم به سادگی این الگوریتم، عدم نیاز به در اختیار داشتن فرضیات درباره‌ی داده (که مخصوصا در خصوص داده های غیرخطی بسیارکاربردی است)، دقت بسیار بالا، کاربردی برای انواع مسائل (کلسیفیکیشن و رگرسیون) اشاره کنیم.

با این وجود این الگوریتم نقاط ضعفی هم دارد. یکی از مهمترین نقاط ضعف آن، حجم زیاد محاسبات است زیرا این الگوریتم داده های آموزشی را نگهداری می‌کند. همین موضوع باعث می‌شود تا به حافظه‌ی بسیار زیادی هم نیاز داشته باشم. زمان حدس زدن (Prediction) این الگوریتم هم طولانی خواهد بود. از طرفی ویژگی های غیر مرتبط واندازه داده‌ها هم روی این الگوریتم تاثیر میگذارد.

تعیین پارامتر (k)

این پارامتر که نقش اساسی در تعیین مرز های تصمیم دارد، به عنوان Smoothing Factor محسوب میشود. هر چقدر مقدار این پارامتر بزرگتر باشد مرز های تصمیم Smoothing تر خواهند بود. با کوچک انتخاب کردن مقدار کا ممکن است به داده‌های Train فیت بشویم و در بررسی داده‌های تست با مشکلاتی از جمله Overfitting روبرو شویم.

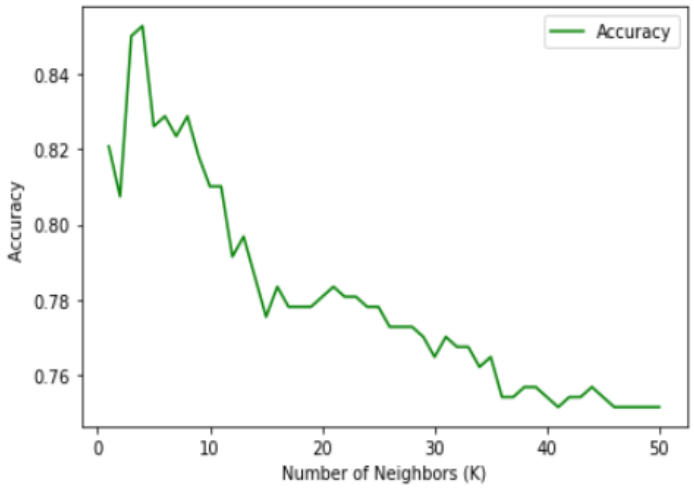
اما بهترین مقدار برای k چقدر است ؟ در واقع بسته به مساله ما دارد و هیچ معیار از پیش تعیین شده ای برای انتخاب بهینه ی پارامتر نداریم.
ایده ی استفاده شده برای یافتن بهینه ترین حالت این پارامتر بررسی Accuracy & Error rate برای مقادیر مختلف Knn و رسم نمودار آنها بود تا با توجه به نمودار بدست آمده بهترین مقدار را برای این مساله تشخیص دهیم

[2] Knn Algotithm

Finding best parameters

```
K = 50
acc = []
for n in range(1,K+1):
    clf = KNeighborsClassifier(n_neighbors = n).fit(X_train, y_train)
    yhat = clf.predict(X_test)
    acc.append(metrics.accuracy_score(y_test, yhat))

acc = np.array(acc)
print( "The best accuracy is", round(acc.max()*100, 2), "% with k =", acc.argmax()+1)
plt.plot(range(1,K+1),acc,'g',label='Accuracy')
plt.legend()
plt.ylabel('Accuracy ')
plt.xlabel('Number of Neighbors (K)')
plt.tight_layout()
plt.show()
```

```
The best accuracy is 85.29 % with k = 4
```



باتوجه به نمودار فوق بهترین مقدار برای پارامتر k برابر ۴ میباشد که دقت این کلسیفایر به ۸۵ درصد میرسد.

معیار های ارزشیابی این الگوریتم :

ابتدا به معرفی ۶ معیار می پردازیم که بر اساس آنها دقت و صحت الگوریتم ها را سنجیده‌ایم.

Precision : عبارت است از $tp/(tp+fp)$ که در آن tp معادل True Positive و Fp معادل False Positive است. در واقع معنی این عبارت توانایی کلسیفایر در عدم نامگذاری نمونه‌های نفی به عنوان نمونه مثبت است.

Recall : عبارت است از $tp/(tp+fn)$ که در آن fn معادل false negative است.معنی این عبارت این است تواناییکلسیفایر در یافتن تمام نمونه های مثبت چقدر است.

F-BETA : به معنی میانگین وزنی Recall و Precision است. بهترین مقدار بتا برابر ۱ و بدترین آن برابر ۰ است. یک بودن بتا بیانگر این است که هر دپی معیار فوق الذکر درجه اهمیت یکسانی دارند.

Support : تعداد وقوع هر کلاس در y_True را نشان می‌دهد.(تعداد نمونه های مورد بررسی در هر کلاس)

Macro : معیار های هر برچسب را محاسبه می کنیم و میانگین غیروزنی آنها را می‌یابیم.

Weighted : معیار های هر برچسب را محاسبه می کنیم و میانگین وزنی آنهارا بر حسبSupport بیان می کنیم.

حال به بررسی معیار های فوق که برای الگوریتم KNN بدست آمده اند، می پردازیم.

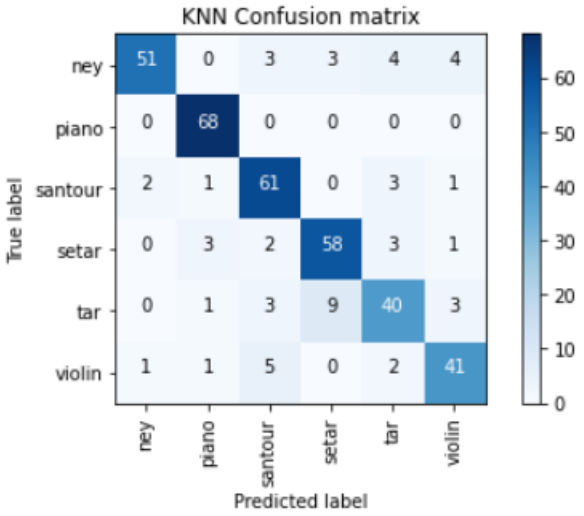
Scores

```
f1_score.append(metrics.f1_score(y_test, y_pred, average='macro'))
precision.append(metrics.precision_score(y_test, y_pred, average='macro'))
recall.append(metrics.recall_score(y_test, y_pred, average='macro'))
accuracy.append(metrics.accuracy_score(y_test, y_pred))
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.78      0.86        65
           1       0.92      1.00      0.96        68
           2       0.82      0.90      0.86        68
           3       0.83      0.87      0.85        67
           4       0.77      0.71      0.74        56
           5       0.82      0.82      0.82        50

    accuracy                           0.85       374
   macro avg       0.85      0.85      0.85       374
weighted avg       0.85      0.85      0.85       374
```

مقادیر بدست آمده برای ۶ کلاس نی، پیانو، سنتور، سه‌تار، تار و ویولن به ترتیب به شرح فوق است.

ماتریس Confusion



از نمودار فوق میتوان دقت تشخیص هر ساز را فهمید. همانطور که از نمودار فوق پیداست ضعیفترین عملکرد مربوط به ساز تار است و الگویتم کا به اشتباه ساز تار را به عنوان سه تار تشخیص داده است. که این امر بابت شبیه بودن صدا های تار پ سه تاربه یکدیگر است.بهترین عملکرد تشخیص مربوز به ساز پیانو است که تمام موارد را به درستی تشخیص داده است.

Cross Validation

```
clf = make_pipeline(preprocessing.StandardScaler(), knn_clf)
scores = cross_val_score(clf, X, y, cv=10, scoring='accuracy')
accuracy_cv.append([scores.min(),scores.mean(),scores.max()])

print('lowest accuracy is', scores.min())
print('highest accuracy is', scores.max())
print('average accuracy is', scores.mean())
```

```
lowest accuracy is 0.784
highest accuracy is 0.888
average accuracy is 0.8352
```

الگوریتم رگرسیون لجستیک[3]

رگرسیون لجستیک یک روش یادگیری ماشین است و یکی از محبوب ترین تکنیک ها برای طبقه بندی داده‌ها است.در مسئله طبقه بندی هنگامی که باید کلاس را از کلاس دیگر تشخیص داد، استفاده می‌شود. این الگوریتم برای پیش‌بینی متغیر وابسته طبقه‌ای با استفاده از یک مجموعه داده شده از متغیر های مستقل استفاده می‌شود.

مزایای رگرسیون لجستیک :

الگوریتم رگرسیون لجستیک یک تکنیک بسیار پرکاربرد و کارآمد است، به منابع محاسباتی زیادی احتیاج ندارد. بسیار قابل تفسیر است. خروجی احتمالات پیش بینی شده را به خوبی کالیبره می کند و رگرسیون یک خط مبنای خوب است که می‌توان از آن برای اندازه‌گیری عملکرد الگوریتم های پیچیده‌تر استفاده کرد. مزیت دیگر رگرسیون لجستیک این است که اجرای آن بسیار آسان است و آموزش آن بسیار کارآمد است. مانند رگرسیون خطی، رگرسیون لجستیک هنگامی که ویژگی هایی را که با متغیر خروجی ارتباط ندارند و همچنین ویژگی هایی که بسیار شبیه به یکدیگر هستند را حذف می کنید، بهتر عمل می کند. بنابراین مهندسی ویژگی نقش مهمی در عملکرد رگرسیون لجستیک و خطی دارد.

تعیین پارامتر ها :

Max_iter : حداکثر تعداد انجام شده برای همگرایی

Logistic Regression

Finding best parameters

```
model = LogisticRegression(class_weight='balanced', random_state=0)
clf = GridSearchCV(model, param_grid={'C':[.1, 10, 100],
                                      'max_iter':[50, 100, 500]})

clf.fit(X_sclr,y)
print('best parameters of the model are:',clf.best_params_)
```

```
best parameters of the model are: {'C': 10, 'max_iter': 100}
```

حال به بررسی معیار های ارزیابی که برای الگوریتم رگرسیون لجستیک بدست آمده‌اند، می‌پردازیم.

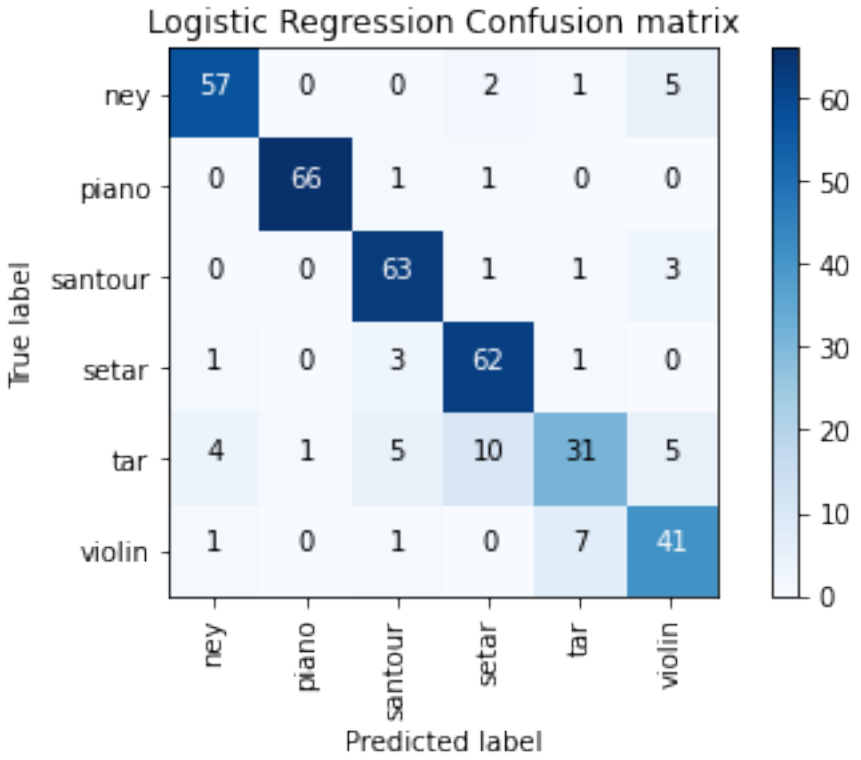
[3] : Logistic Regression Algorithm

Scores

```
f1_score.append(metrics.f1_score(y_test, y_pred, average='macro'))
precision.append(metrics.precision_score(y_test, y_pred, average='macro'))
recall.append(metrics.recall_score(y_test, y_pred, average='macro'))
accuracy.append(metrics.accuracy_score(y_test, y_pred))
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.88      0.89        65
           1       0.99      0.97      0.98        68
           2       0.86      0.93      0.89        68
           3       0.82      0.93      0.87        67
           4       0.76      0.55      0.64        56
           5       0.76      0.82      0.79        50

    accuracy                           0.86       374
   macro avg       0.85      0.85      0.84       374
weighted avg       0.85      0.86      0.85       374
```

ماتریس Confusion :



همانطور که از شکل پیداست ضعیف ترین تفکیک ساز مرتبط با ساز های تار و سه‌تار است . بهترین عملکرد هم برای تشخیص صدای پیانو است . این الگوریتم در تشخیص صدا های تار و پیانو نسبت به الگوریتم Knn عملکرد ضعیف‌تری داشته است. اما در سایر ساز ها عملکرد به نسبت بهتری نشان داده است. دقت هر دو الگوریتم را در این مساله می‌توان معادل هم دانست.

Cross Validation

```
clf = make_pipeline(preprocessing.StandardScaler(), lr_clf)
scores = cross_val_score(clf, X, y, cv=10, scoring='accuracy')
accuracy_cv.append([scores.min(),scores.mean(),scores.max()])

print('lowest accuracy is', scores.min())
print('highest accuracy is', scores.max())
print('average accuracy is', scores.mean())
```

```
lowest accuracy is 0.8387096774193549
highest accuracy is 0.944
average accuracy is 0.8809548387096774
```

الگوریتم SVM[4]

ماشین بردار پشتیبان SVM یک الگوریتم نظارت‌شده یادگیری ماشین است که هم برای مسائل طبقه‌بندی و هم مسائل رگرسیون قابل استفاده است؛ با این حال از آن بیشتر در مسائل طبقه‌بندی استفاده می‌شود.

مزایا و معایب الگوریتم ماشین بردار پشتیبان

مزایا

حاشیه جداسازی برای دسته‌های مختلف کاملاً واضح است.

در فضاهای با ابعاد بالاتر کارایی بیشتری دارد.

در شرایطی که تعداد ابعاد بیش از تعداد نمونه‌ها باشد نیز کار می‌کند.

یک زیر مجموعه از نقاط تمرینی را در تابع تصمیم‌گیری استفاده می‌کند (که به آن‌ها بردارهای پشتیبان گفته می‌شود)، بنابراین در مصرف حافظه نیز به صورت بهینه عمل می‌کند.

[4]Support Vector Machine algorithm

معایب

هنگامی که مجموعه داده‌ها بسیار بزرگ باشد، عملکرد خوبی ندارد، زیرا نیازمند زمان آموزش بسیار زیاد است.
هنگامی که مجموعه داده نویز زیادی داشته باشد، عملکرد خوبی ندارد و کلاس‌های هدف دچار همپوشانی می‌شوند.
ماشین بردار پشتیبان به طور مستقیم تخمین‌های احتمالاتی را فراهم نمی‌کند و این موارد با استفاده از یک اعتبارسنجی متقابل (Cross Validation) پرهزینه پنج‌گانه انجام می‌شوند. این امر با روش SVM موجود در کتابخانه،scikit-kearn پایتون مرتبط است.

تعیین پارامتر ها:

کرنل: این پارامتر پیش از این مورد بررسی قرار گرفت. گزینه‌های گوناگونی شامل «linear»،«poly» rbf برای کرنل وجود دارند و در حالت پیش‌فرض کرنل روی پارامتر rbf قرار دارد. rbf و poly برای خط جداساز غیر راست مفید هستند.

گاما (gamma) : ضریب کرنل برای rbf، poly و sigmoid است. هرچه مقدار گاما بیشتر باشد، الگوریتم تلاش می‌کند برازش را دقیقاً بر اساس مجموعه داده‌های تمرینی انجام دهد و این امر موجب تعمیم یافتن خطا و وقوع مشکل بیش برازش (Over-Fitting) می‌شود.

C : پارامتر پارامتر جریمه C، برای جمله خطا است. این پارامتر همچنین برقراری تعادل بین مرزهای تصمیم‌گیری هموار و طبقه‌بندی نقاط داده تمرینی را کنترل می‌کند.

Finding best parameters

```
model = svm.SVC(class_weight='balanced', random_state=0)
clf = GridSearchCV(model, param_grid={'C':[.1, 10, 50],
                                      'gamma':[.001, .01, .1],
                                      'kernel':['rbf', 'poly', 'sigmoid']})
clf.fit(X_sclr,y)
print('best parameters of the model are:',clf.best_params_)
```

```
best parameters of the model are: {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
```

همانطور که در شکل مشخص است ، پارامتر های این الگوریتم نشان داده شده است.

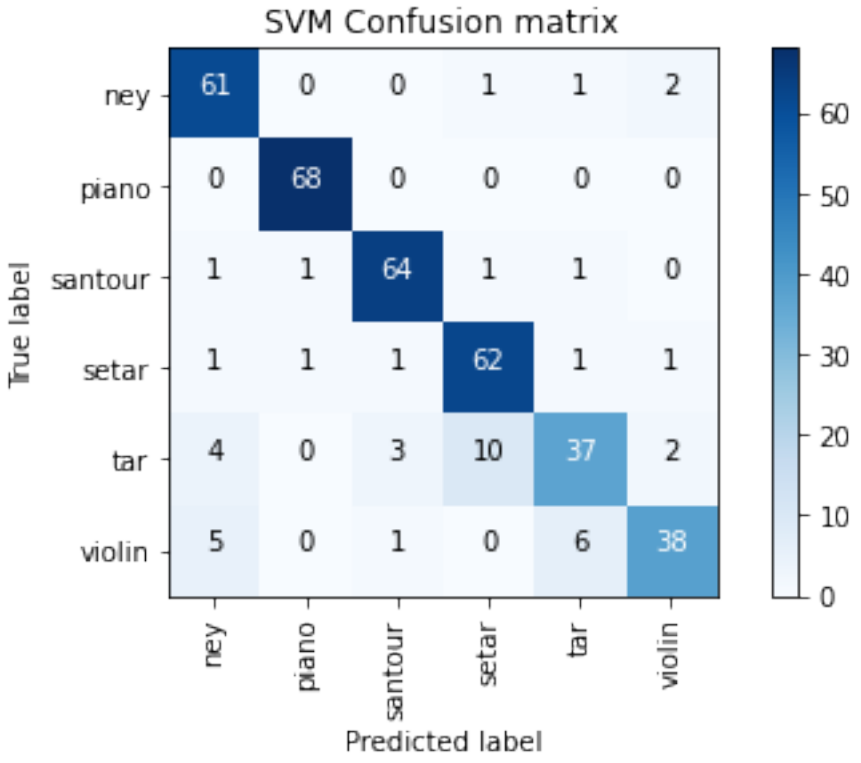
حال به بررسی معیار های ارزیابی که برای الگوریتم SVM بدست آمده اند ، می پردازیم.

Scores

```
f1_score.append(metrics.f1_score(y_test, y_pred, average='macro'))
precision.append(metrics.precision_score(y_test, y_pred, average='macro'))
recall.append(metrics.recall_score(y_test, y_pred, average='macro'))
accuracy.append(metrics.accuracy_score(y_test, y_pred))
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.94      0.89        65
           1       0.97      1.00      0.99        68
           2       0.93      0.94      0.93        68
           3       0.84      0.93      0.88        67
           4       0.80      0.66      0.73        56
           5       0.88      0.76      0.82        50

    accuracy                           0.88       374
   macro avg       0.88      0.87      0.87       374
weighted avg       0.88      0.88      0.88       374
```

ماتریس Confusion :



با توجه به نمودار ضعیف ترین عملکرد همانند روش های قبل مربوط به تشخیص ساز تار و بهترین متعلق به پیانو می باشد.با توجه به اعداد بدست آمده متوجه میشویم که عملکرد الگوریتم SVM از دو الگوریتم قبلی کمی کارآمد تر است و برای مساله ما روش بهتری محسوب می‌شود. با اعتبار سنجی انجام شده به دقت ماینگین ۹۰ درصد میرسیم که آمار بسیار خوبی محسوب می‌شود.

به شکل زیر توجه کنید.

Cross Validation

```
clf = make_pipeline(preprocessing.StandardScaler(), svm_clf)
scores = cross_val_score(clf, X, y, cv=10, scoring='accuracy')
accuracy_cv.append([scores.min(),scores.mean(),scores.max()])

print('lowest accuracy is', scores.min())
print('highest accuracy is', scores.max())
print('average accuracy is', scores.mean())
```

```
lowest accuracy is 0.8548387096774194
highest accuracy is 0.944
average accuracy is 0.9051419354838709
```

الگوریتم درخت تصمیم [۵]

درخت تصمیم که هدف اصلی آن، دسته‌بندی داده‌هاست، مدلی در داده‌کاوی است که مشابه فلوچارت، ساختاری درخت مانند را جهت اخذ تصمیم و تعیین کلاس و دسته یک داده خاص به ما ارائه می‌کند. چنانچه متغیری وابسته عددی باشد دسته بندی ما یک مساله رگرسیون و چنانچه طبقه‌ای باشد، دسته بندی از نوع، رده‌بندی (Classification) است.

درخت تصمیم یک مدل خودتوصیف است یعنی به‌تنهایی و بدون حضور یک فرد متخصص در آن حوزه، نحوه دسته‌بندی را به صورت گرافیکی نشان می‌دهد و به دلیل همین سادگی و قابل‌فهم بودن، روش محبوبی در داده‌کاوی محسوب می‌شود. البته در مواردی که تعداد گره‌های درخت زیاد باشد، نمایش گرافیکی و تفسیر آن می‌تواند کمی پیچیده باشد.

مزایا و معایب درخت تصمیم:

مزایا :

احتیاجی به تخمین تابع توزیع نیست. آماده‌سازی داده‌ها برای یک درخت تصمیم، ساده یا غیرضروری است.

[5] Decision tree algorithm

(روش‌های دیگر اغلب نیاز به نرمال‌سازی داده یا حذف مقادیر خالی یا ایجاد متغیرهای پوچ دارند)
درخت تصمیم یک روش غیرپارامتریک است و نیاز به تنظیم خاصی برای افزایش دقت الگوریتم ندارد.

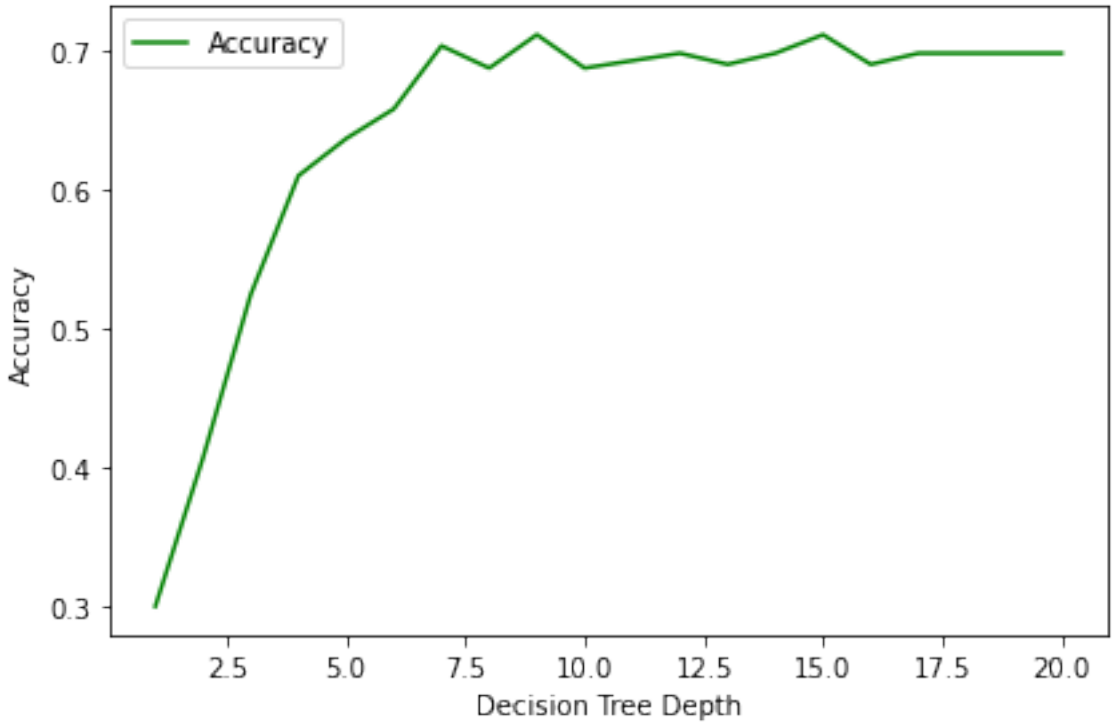
معایب درخت تصمیم:

در موارد با تعداد دسته‌های زیاد و نمونه آموزشی کم، احتمال خطا بالاست. تولید درخت تصمیم‌گیری، هزینه محاسباتی بالا دارد.
در مسائلی که دسته‌ها شفاف نباشند و همپوشانی داشته باشند، خوب عمل نمی‌کنند.
درصورتی‌که درخت بزرگ باشد امکان است خطاها از سطحی به سطحی دیگر جمع می‌شوند (انباشته شدن خطای لایه‌ها و تاثیر بر روی یکدیگر)

یافتن پارامتر بهینه :

Finding best parameters

```
depths = 20
DTCA = []
for i in range(1,depths+1):
    clf = DecisionTreeClassifier(max_depth = i, class_weight='balanced', random_state=0)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    DTCA.append(accuracy_score(y_test, y_pred))

DTCA = np.array(DTCA)
print("The accuracy of Decision Tree with max_depth =", DTCA.argmax()+1 , " is : ", round(DTCA.max()*100, 2), "%")
plt.plot(range(1,depths+1), DTCA,'g',label='Accuracy')
plt.legend()
plt.ylabel('Accuracy')
plt.xlabel('Decision Tree Depth')
plt.tight_layout()
plt.show()
```



با توجه به نمودار فوق بیشترین دقت این الگوریتم در عمق ۹ درخت وبه میزان ۱۲.۷۱ درصد است.

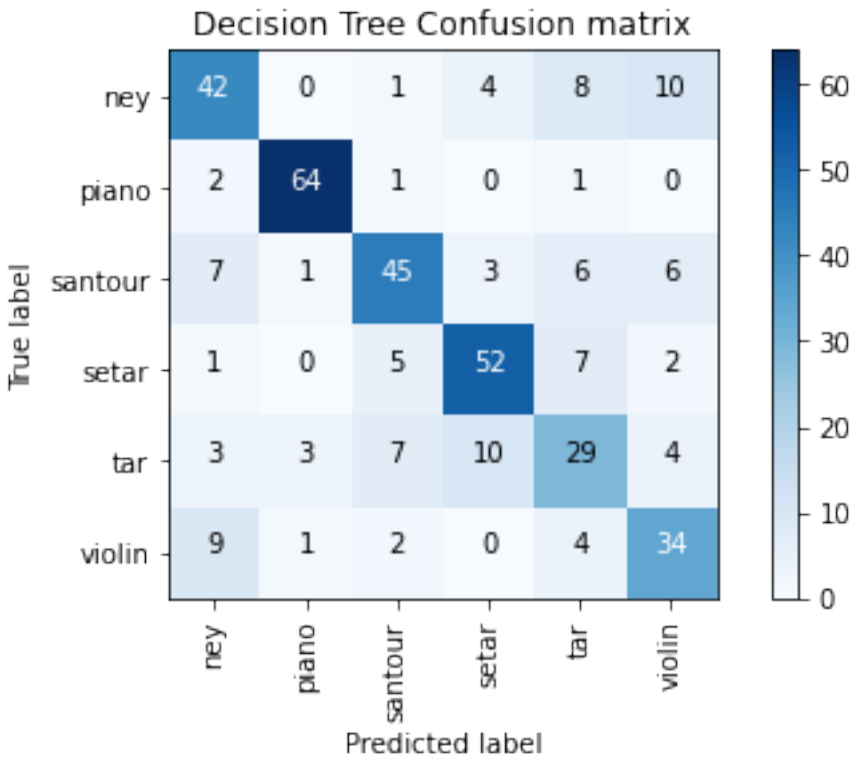
حال در ادامه به برسی معیار های ارزیابی که برای الگوریتم (decision tree) بدست آمده اند ، میپردازیم.

Scores

```
f1_score.append(metrics.f1_score(y_test, y_pred, average='macro'))
precision.append(metrics.precision_score(y_test, y_pred, average='macro'))
recall.append(metrics.recall_score(y_test, y_pred, average='macro'))
accuracy.append(metrics.accuracy_score(y_test, y_pred))
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.66      0.65      0.65        65
           1       0.93      0.94      0.93        68
           2       0.74      0.66      0.70        68
           3       0.75      0.78      0.76        67
           4       0.53      0.52      0.52        56
           5       0.61      0.68      0.64        50

    accuracy                           0.71       374
   macro avg       0.70      0.70      0.70       374
weighted avg       0.71      0.71      0.71       374
```

ماتريس Confusion :



با توجه به نمودار بدست آمده و مقایسه مقادیر معیار های ارزیابی الگوریتم ، واضح است که این روش به نسبت روش های قبل از دقت کمتری برخوردار بوده و بیشترین میزان خطا را دارد.

Cross Validation

```
clf = make_pipeline(preprocessing.StandardScaler(), dt_clf)
scores = cross_val_score(clf, X, y, cv=10, scoring='accuracy')
accuracy_cv.append([scores.min(),scores.mean(),scores.max()])

print('lowest accuracy is', scores.min())
print('highest accuracy is', scores.max())
print('average accuracy is', scores.mean())
```

```
lowest accuracy is 0.6370967741935484
highest accuracy is 0.792
average accuracy is 0.7322451612903226
```

در بهترین عملکرد خود نهایتا به دقت ۷۹ درصد می توانیم برسیم که حتی از بدترین عملکرد سایر الگوریتم ها نیز کمتر می‌باشد.

الگوریتم MLP [6]

پرسپترون چند لایه، به انگلیسی: (Multilayer perceptron) دسته ای از شبکه‌های 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 Backpropagation 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.

مزایای استفاده از الگوریتم شبکه عصبی:

از فواید استفاده از این الگوریتم می توان به موارد زیر اشاره کرد:

یادگیری انطباق پذیر (Adaptive Learning): یادگیری انطباق پذیر یعنی قابلیت یادگیری و نحوه انجام وظایف بر پایه اطلاعات داده شده برای تمرین و تجربه های مقدماتی.

سازماندهی توسط خود (Self Organization) : سازماندهی توسط خود یعنی یک شبکه هوش مصنوعی سازماندهی یا ارائه اش را برای اطلاعاتی که در طول دوره یادگیری دریافت می کند، خودش ایجاد کند.

[6] **Multilayer perceptron algorithm**

عملکرد به هنگام (Real Time Operation) : در عملکرد به هنگام و به موقع، محاسبات شبکه هوش مصنوعی می تواند به صورت موازی انجام شود و سخت افزار های مخصوصی طراحی و ساخته شده که می تواند از این قابلیت استفاده کنند.

تعیین پارامتر ها:

Finding best parameters

```
model = MLPClassifier(random_state=0, learning_rate='adaptive')
clf = GridSearchCV(model, param_grid={'activation':['logistic', 'tanh', 'relu'],
                                      'learning_rate_init':[.001, .01, .1],
                                      'hidden_layer_sizes':[(200,), (100,50), (200,100), (300,200)]})
clf.fit(X_sclr,y)
print('best parameters of the model are:',clf.best_params_)
```

```
best parameters of the model are: {'activation': 'tanh', 'hidden_layer_sizes': (200,), 'learning_rate_init': 0.001}
```

(activation) : تابعی است که به روی (hidden layer) اعمال می شود که ۴ تابع اصلی برای این الگوریتم عبارتند از : 'identity'، 'logistic'، 'tanh'، 'relu' که تابع انتخاب شده ما tanh می باشد که همان hyperbolic tan است.

(hidden_layer_sizes) : بیانگر تعداد نورون ها در (hidden layer) می باشد.

(learning_rate_init) : همان نرخ یادگیری میباشد که برای کنترل وزن ها و ضرایب به کار میرود.

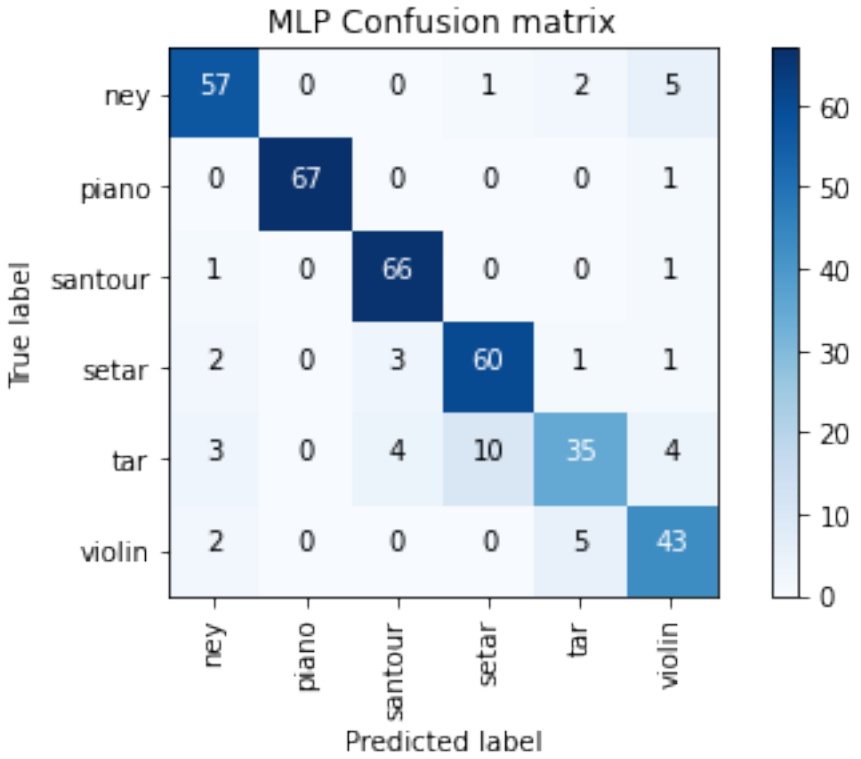
حال به بررسی معیار های ارزیابی که برای الگوریتم MLP بدست آمده اند ، میپردازیم.

Scores

```
f1_score.append(metrics.f1_score(y_test, y_pred, average='macro'))
precision.append(metrics.precision_score(y_test, y_pred, average='macro'))
recall.append(metrics.recall_score(y_test, y_pred, average='macro'))
accuracy.append(metrics.accuracy_score(y_test, y_pred))
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.88      0.88        65
           1       1.00      0.99      0.99        68
           2       0.90      0.97      0.94        68
           3       0.85      0.90      0.87        67
           4       0.81      0.62      0.71        56
           5       0.78      0.86      0.82        50

    accuracy                           0.88       374
   macro avg       0.87      0.87      0.87       374
weighted avg       0.88      0.88      0.87       374
```

ماتریس Confusion :



دقت بالا و صحت داده های تخمینی توسط این روش کاملا قابل شهود است.همانند متد های قبل ضعیف ترین عملکرد مربوط به تشخیص صدای تار و سه تار است.

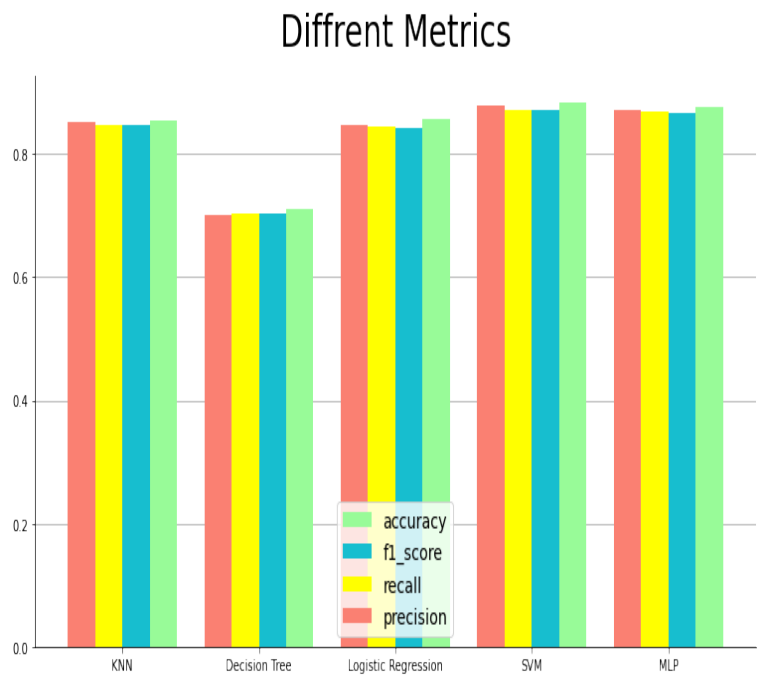
Cross Validation

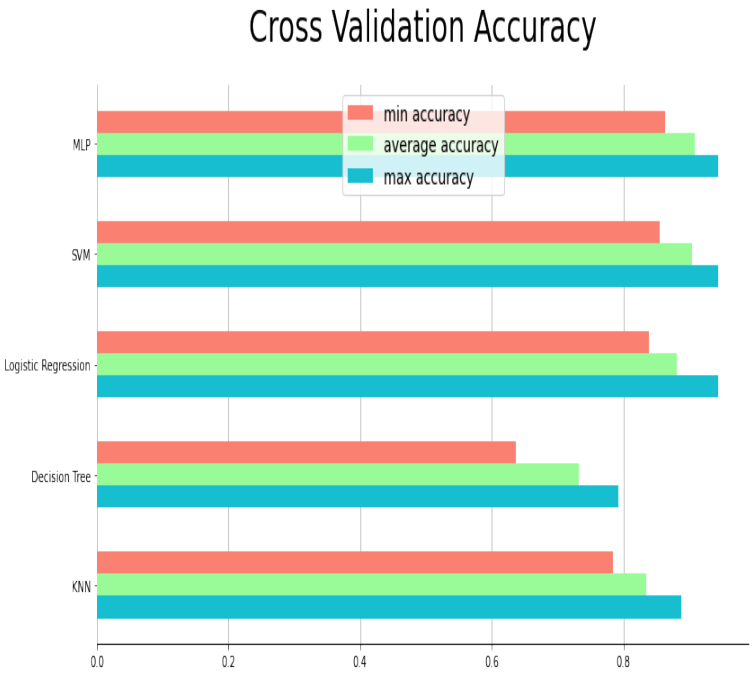
```
clf = make_pipeline(preprocessing.StandardScaler(), mlp_clf)
scores = cross_val_score(clf, X, y, cv=10, scoring='accuracy')
accuracy_cv.append([scores.min(),scores.mean(),scores.max()])

print('lowest accuracy is', scores.min())
print('highest accuracy is', scores.max())
print('average accuracy is', scores.mean())
```

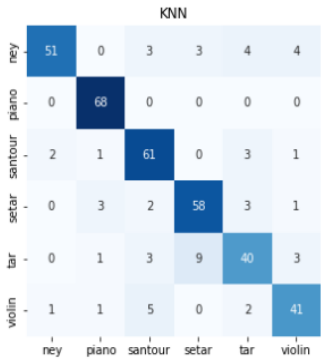
```
lowest accuracy is 0.8629032258064516
highest accuracy is 0.9435483870967742
average accuracy is 0.9083419354838711
```

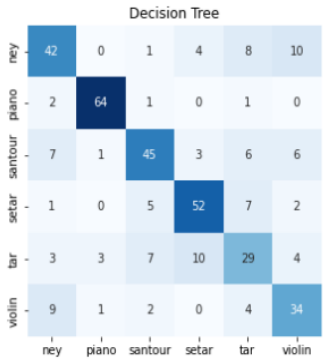
بدترین عملکرد این الگوریتم حتی از بهترین های برخی دیگر از الگوریتم ها بهتر عمل می کند.و در حالت ایده آل دقت ۹۴ درصدی داشته است که تقریبا از همه ی الگوریتم ها بهتر محسوب می‌شود.

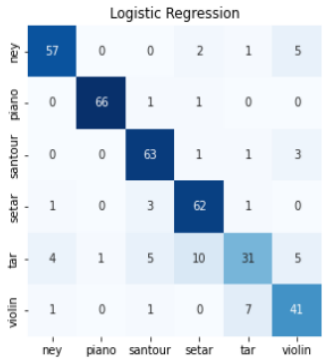
جمع بندی

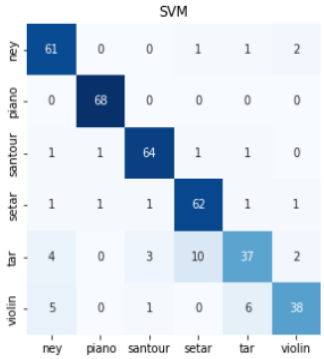


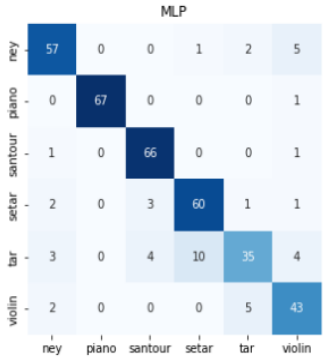

Cross Validation Accuracy
min accuracy
average accuracy
max accuracy
MLP
SVM
Logistic Regression
Decision Tree
KNN
0.0
0.2
0.4
0.6
0.8


KNN
ney 51 0 3 3 4 4
piano 0 68 0 0 0 0
santour 2 1 61 0 3 1
setar 0 3 2 58 3 1
tar 0 1 3 9 40 3
violin 1 1 5 0 2 41
ney piano santour setar tar violin


Decision Tree
ney 42 0 1 4 8 10
piano 2 64 1 0 1 0
santour 7 1 45 3 6 6
setar 1 0 5 52 7 2
tar 3 3 7 10 29 4
violin 9 1 2 0 4 34
ney piano santour setar tar violin


Logistic Regression
ney 57 0 0 2 1 5
piano 0 66 1 1 0 0
santour 0 0 63 1 1 3
setar 1 0 3 62 1 0
tar 4 1 5 10 31 5
violin 1 0 1 0 7 41
ney piano santour setar tar violin


SVM
ney 61 0 0 1 1 2
piano 0 68 0 0 0 0
santour 1 1 64 1 1 0
setar 1 1 1 62 1 1
tar 4 0 3 10 37 2
violin 5 0 1 0 6 38
ney piano santour setar tar violin

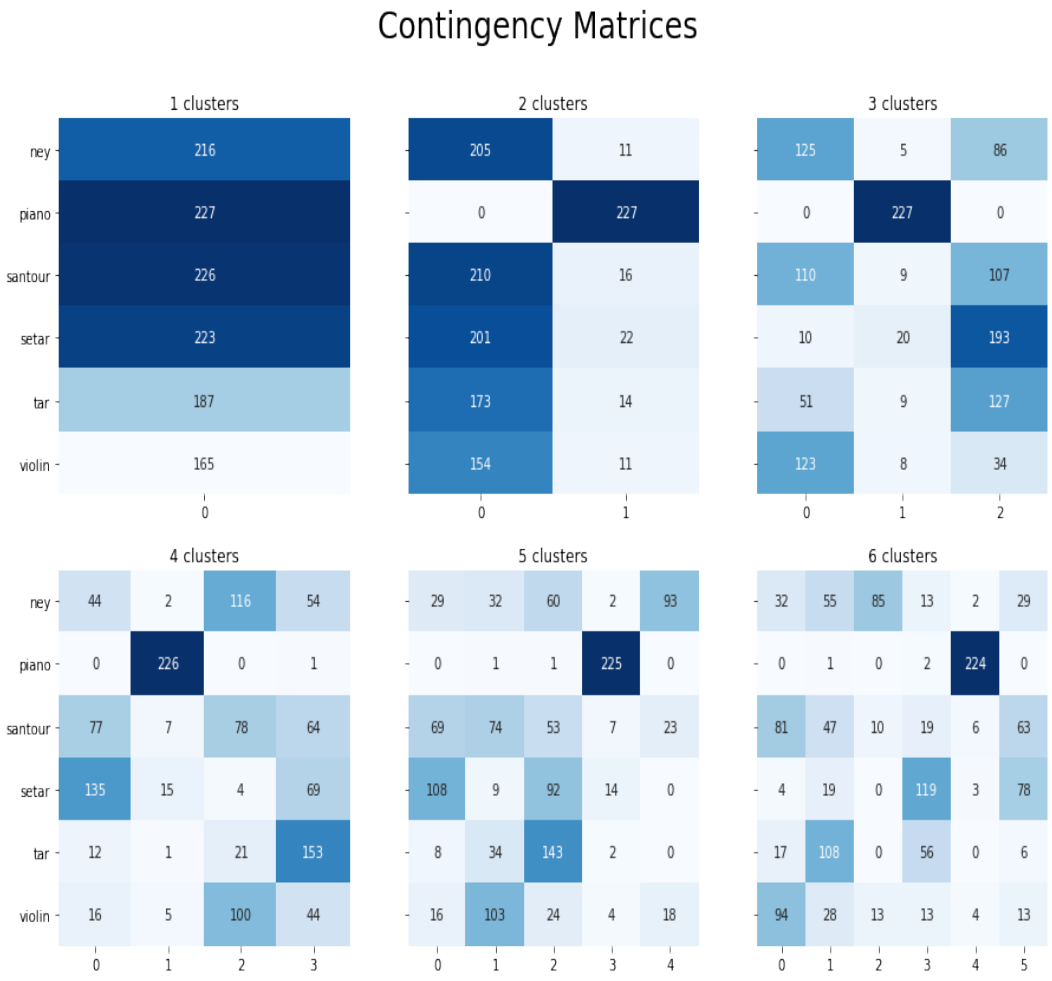

MLP
ney 57 0 0 1 2 5
piano 0 67 0 0 0 1
santour 1 0 66 0 0 1
setar 2 0 3 60 1 1
tar 3 0 4 10 35 4
violin 2 0 0 0 5 43
ney piano santour setar tar violin

۸.۲ خوشه بندی

Kmean

روش Kmean جزء روش های یادگیری بدون نظارت هست که هدف آن خوشه بندی کردن داده ها بر اساس ویژگی های آن هاست. در این روش نیاز به یک تابع امتیاز برای ارزیابی میزان کیفیت خوشه بندی داریم.

الگوریتم خوشه‌بندی Kmean میانگین از گروه روش‌های خوشه‌بندی تفکیکی (Partitioning Clustering) محسوب می‌شود و درجه پیچیدگی محاسباتی آن برابر با $O(ndk+1)$ است، به شرطی که n تعداد اشیاء، d بعد ویژگی‌ها و k تعداد خوشه‌ها باشد. همچنین پیچیدگی زمانی برای این الگوریتم برابر با $O(nkdi)$ است، که البته منظور از i تعداد تکرارهای الگوریتم برای رسیدن به جواب بهینه است.



شکل فوق روند خوشه بندی ساز ها را مشخص می‌کند.
ابتدا یک خوشه داریم و همه ی داده‌ها از ساز های مختلف باهم در خوشه شماره ۰ هستند. پس از افزایش تعداد خوشه‌ها به دو عدد، می‌بینیم که در خوشه شماره یک غالب داده های تشکیل دهنده‌ی خوشه مربوط به ساز پیانو است. در واقع می‌توان گفت با در نظر گرفتن ۲ خوشه ، یک خوشه کاملا به پیانو تعلق می‌گیرد و خوشه دیگر به بقیه ساز ها. نکته ی مهم این است که از همین دو نمودار می‌توان فهمید صدای ساز پیانو و ویژگی هایی که در فرکانس های آن نهفته شده، بسیار متمایز با سایر ساز ها هستند.

در ادامه با افزایش تعداد خوشه ها به ۳ عدد ، خوشه شماره یک‌ام را می توان متعلق به پیانو دانست زیرا که همچنان داده های این ساز در این خوشه تفاوت چشمگیری با سایرین دارد. خوشه های شماره ۰ و ۲ به طور تقریبا یکنواختی از مابقی ساز ها هستند. ساز هایی مثل (تار و سه‌تار، نی و ویولن) به دلیل شباهت، در یک کلاستر قرار گرفته‌اند.با افزایش خوشه ها به ۴ عدد میبینیم که تقریبا داده های غالب در خوشه های ۱،۰و۳ چشمگیر هستند. به عنوان مثال در خوشه شماره ۳ ، ۱۵۳ عدد از داده‌ها مربوط به تار است و در خوشه شماره ۰، ۱۳۵ عدد داده‌ی ساز سه‌تار وجود دارد.

نهایتا در خوشه بندی آخر همانطور که قابل مشاهده است برخی خوشه ها را نمی توان صراحتا و با اطمینان گفت که متعلق به چه سازی هستند.

متریک های بدست امده از خوشه‌های مختلف به شکل زیر است .

Metrics

```
print('\n')
print('type\t\t\ttime\tR-Score\tAMI\tNMI\tHomo\tComp\tV-meas\tSilh\tCH-score  DB-score')
print(110 * '_')

clustering_fit_stats(KMeans(n_clusters= 2, random_state= 0), name="K-means with k=2", data= X)
clustering_fit_stats(KMeans(n_clusters= 3, random_state= 0), name="K-means with k=3", data= X)
clustering_fit_stats(KMeans(n_clusters= 4, random_state= 0), name="K-means with k=4", data= X)
clustering_fit_stats(KMeans(n_clusters= 5, random_state= 0), name="K-means with k=5", data= X)
clustering_fit_stats(KMeans(n_clusters= 6, random_state= 0), name="K-means with k=6", data= X)
clustering_fit_stats(KMeans(n_clusters= 6, random_state= 0), name="K-means with k=6 & UMAP", data= reduced_X)
```

type	time	R-Score	AMI	NMI	Homo	Comp	V-meas	Silh	CH-score	DB-score
K-means with k=2	0.111s	0.148	0.291	0.292	0.191	0.618	0.292	0.150	225.527	1.745
K-means with k=3	0.094s	0.247	0.339	0.341	0.272	0.456	0.341	0.155	208.216	2.163
K-means with k=4	0.144s	0.296	0.365	0.368	0.326	0.423	0.368	0.140	187.359	2.054
K-means with k=5	0.104s	0.307	0.373	0.376	0.352	0.403	0.376	0.148	170.683	2.017
K-means with k=6	0.153s	0.340	0.394	0.398	0.395	0.401	0.398	0.140	155.776	2.011
K-means with k=6 & UMAP	0.099s	0.439	0.470	0.473	0.473	0.473	0.473	0.109	128.474	2.312

معرفی پارامتر ها :

Shorthand	full name
homo	homogeneity score
compl	completeness score
v-meas	V measure
ARI	adjusted Rand index
AMI	adjusted mutual information
silhouette	silhouette coefficient

Homogeneity : A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class.

completeness : A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster.

Nmi : Normalized Mutual Information between two clusterings.

Ami : Adjusted Mutual Information between two clusterings.

Ch-score : (Calinski and Harabasz score) The score is defined as ratio between the within-cluster dispersion and the between-cluster dispersion.

Db score : (Davies-Bouldin score)The score is defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. Thus, clusters which are farther apart and less dispersed will result in a better score.

به عنوان مثال معیار ch-score بیانگر پراکندگی درون کلاسی به پراکندگی بین کلاسی را نشان می دهد.
پس در واقع هرچه این عدد کوچکتر شود به معنی این است که به حالت ایده آل نزدیک تر می‌شویم .
با مشاهده ی مقادیر این معیار برای خوشه های متفاوت می‌بینیم که با افزایش خوشه ها این نسبت کمتر شده و به مقادیر مطلوب تر نزدیک می‌شویم وهمینطور با بررسی معیار db-score که بیانگر این است که هرچقدر کلاستر ها از هم دورتر باشند و پراکندگی کمتری داشته باشند ، نمره ی بهتری از این معیار را میگیرند .
با بررسی خوشه های مختلف شاهد افزایش این مقدار در حالت ۶ کلاستره کاهش ابعاد یافته به کمک umap

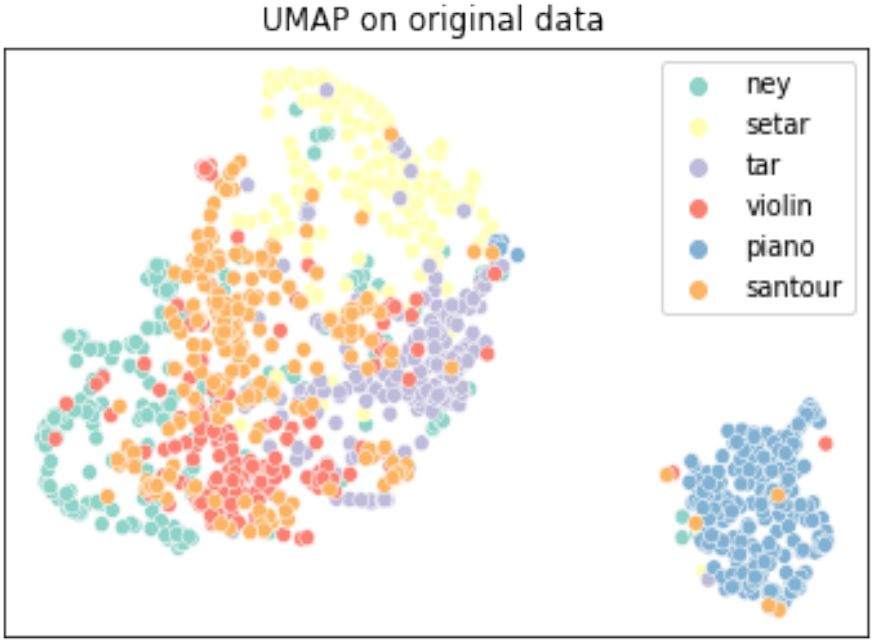
هستیم.

معرفی umap [7]

همانطور که از نامش پیداست تکنیکی برای کاهش ابعاد است و به ما کمک می‌کند که پس از کاهش دادن ابعاد بتوانیم داده را در نموداری ۲ بعدی نشان بدهیم.

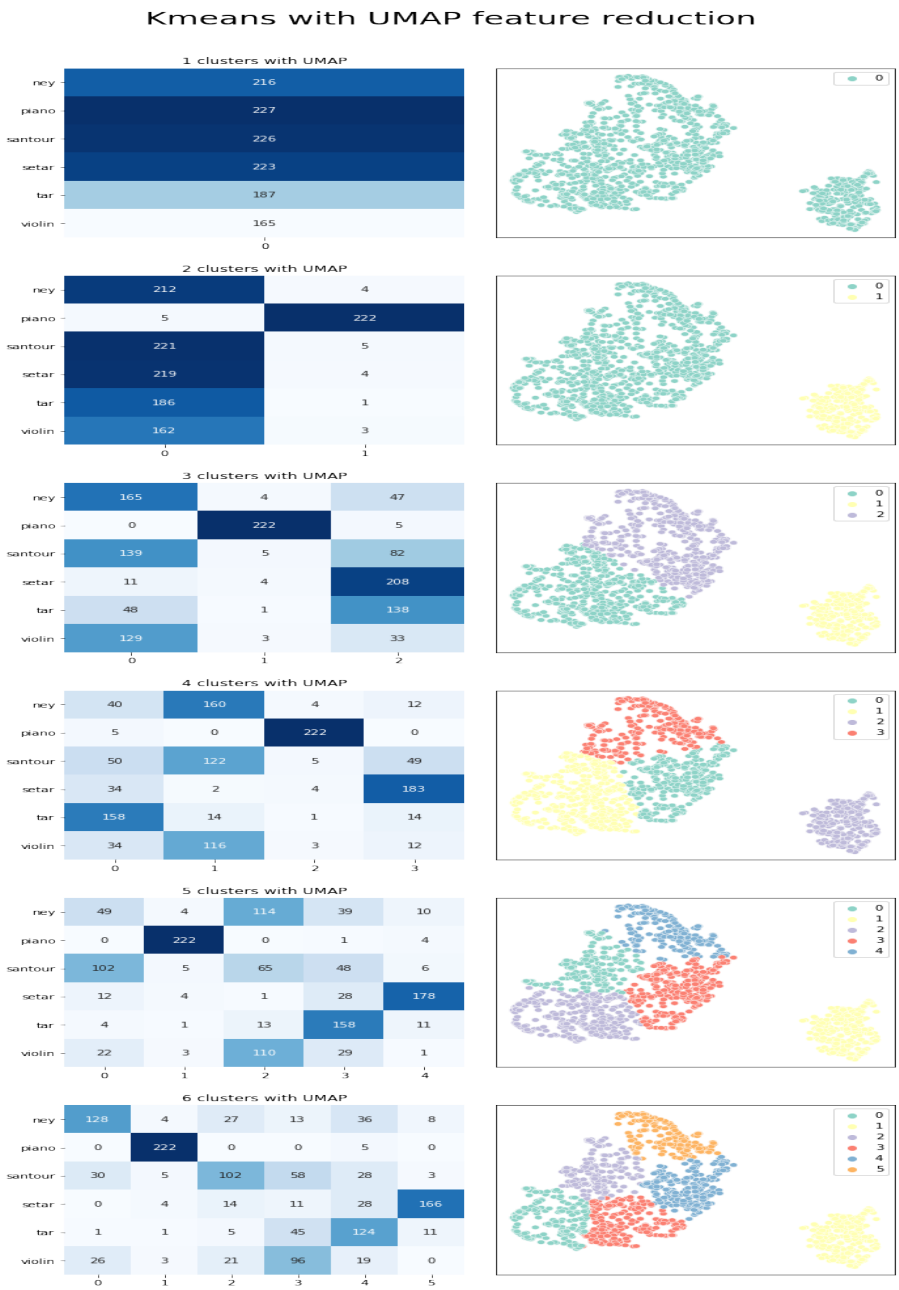
در این پروژه سعی بر آن شده که یک بار بدوت استفاده از umap عمل خوشه بندی را انجام دهیم و مقادیر متریک های ان را بدست آوریم ، سپس کلاسترینگ را یکبار دیگر بر مبنای umap انجام دهیم.

در ادامه به تاثیر و عملکرد این روش برای افزایش دقت خوشه بندی ها پی می بریم .



پس از کاهش ابعاد به دو بعد نمودار فوق بدست میاید.همانطور که مشخص است داده ها در هم تنیده شده هستند . در ادامه به پیاده سازی الگوریتم خوشه بندی Kmean با استفاده از umap می‌پردازیم.

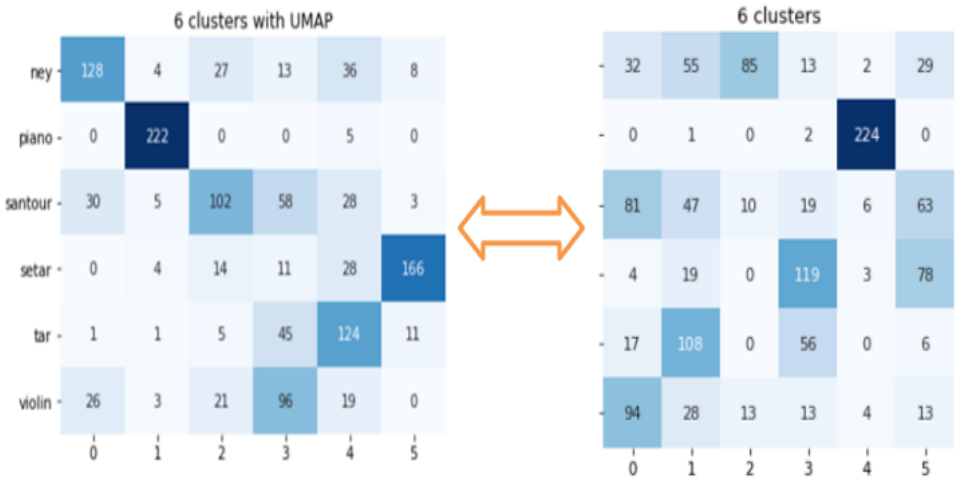
[7] **Uniform Manifold Approximation and Projection for Dimension Reduction**



همانطور که از نمودار ها پیداست با استفاده از umap برای کاهش ابعاد، خوشه بندی ها دقت بیشتری پیدا کرده اند .در ابتدا چون خوشه ها یکی هستند ، همه داده ها به یک رنگ درامده اند . هرچند که اختلاف فاصله ای زیادی با داده های متراکم سمت راست پایین تصویر دارند. با افزایش تعداد خوشه ها بلافاصله نقاط دورتر به یک

کلاستر جداگانه تبدیل می‌شوند. به نظر می‌رسد که این کلاستر همان داده های مرتبط با ساز پیانو باشد . زیرا در ماتریس ها هم درایه های بدست آمده ، بیانگر وجه تمایز خاص پیانو با سایر ساز ها بودند.با افزایش کلاستر ها به شش عدد به شکل اخر می‌رسیم که تقریبا میتواند بیانگر این باشد که هر کلاستر به کدام ساز اختصاص دارد.

مقایسه قبل و بعد از feature reduction :



همانطور که ملاحظه میکنید پس از انجام feature reduction اختلاف تعداد داده ها از هر ساز بیشتر میشود و داده ی Max هر خشوه نمایان تر می‌شود.

از این رو می توان برای آن خوشه راحتر تصمیم گیری کرد و دسته ها را با دقت بیشتری از هم تمیز داد.

خوشه‌بندی سلسله مراتبی [8]

یکی از روش‌های «یادگیری ماشین» Machine Learning که به «آموزش بدون نظارت» (Un-supervised Learning) شهرت دارد، تحلیل خوشه‌بندی (Clustering Analysis) است.
در این روش، برعکس خوشه بندی k میانگین، هر مشاهده ممکن است در بیش از یک خوشه قرار گیرد زیرا براساس سطوح مختلف فاصله، خوشه‌ها تشکیل می‌شود. بنابراین هر خوشه ممکن است زیر مجموعه خوشه دیگر در سطحی از فاصله قرار گیرد.
به هر حال خوشه‌بندی روش است که به کمک «ویژگی‌ها» Features یا «صفت‌ها» Attributes مشاهدات، آن را به گروه‌های مشابه طبقه‌بندی می کند.

[8] **Hierarchical Clustering**

مزایای خوشه‌بندی سلسله مراتبی

نمایش‌های حاصل از خوشه‌بندی سلسله مراتبی می‌تواند حاوی اطلاعات مفید و سودمندی باشد.

دندروگرام‌ها روش جالب و آموزنده‌ای برای مصورسازی هستند.

دندروگرام‌ها به ویژه زمانی سودمند هستند که دیتاست‌ها شامل روابط سلسله مراتبی واقعی باشند.

معایب خوشه‌بندی سلسله مراتبی

این روش نسبت به داده‌های پرت بسیار حساس هستند و در صورت وجود این‌گونه داده‌ها عملکرد مدل تا حد زیادی کاهش پیدا می‌کند.

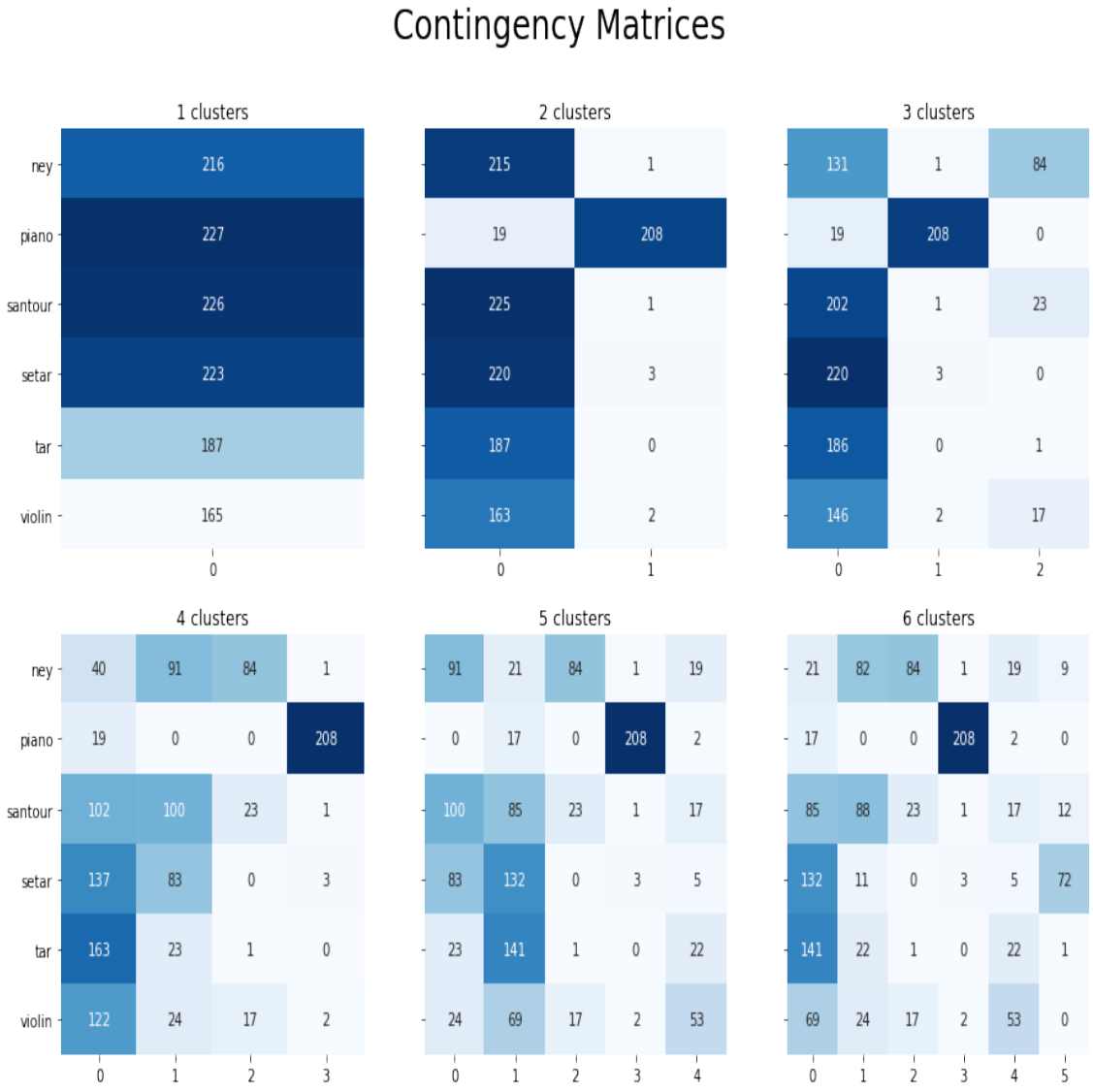
به لحاظ محاسباتی بسیار گران است.

پیچیدگی زمانی الگوریتم خوشه‌بندی سلسله مراتبی تجمیعی

در الگوریتم HAC پیچیدگی زمانی برابر با $O(n3)O(n3)$ و فضای مورد نیاز حافظه نیز برابر با $O(n2)O(n2)$ است.

بنابراین با افزایش حجم داده‌ها، سرعت و فضای حافظه برای اجرای عملیات خوشه‌بندی به شدت افزایش می‌یابد. به همین دلیل معمولا از این الگوریتم برای خوشه‌بندی «کلان داده» Big Data استفاده نمی‌شود.

: Contingency Matrix of Diffrent Clusters



همانطور که از روال خوشه بندی بالا مشخص است وقتی تعداد دسته ها به ۲ عدد میرسد ،غالب داده ی کلاستر شماره ۱ را داده های پیانو تشکیل میدهند. با افزایش خوشه ، خوشه نی به وجود میاد .(دقت شود وقتی میگویم خوشه نی ، منظور این است که داده های نی مربوط به آن کلاستر از سایر داده ها بیشتر هستند پس احتمال اینکه در آینده آن خوشه متعلق به نی شود بیشتر از سایر ساز هاست)

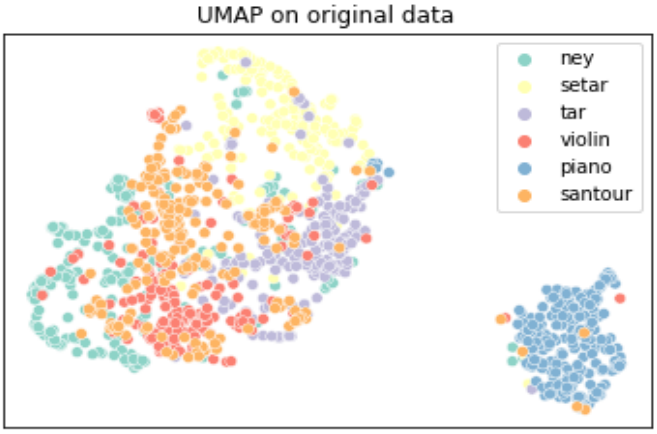
در خوشه ۴ تایی دیگر نمیتوان مشخص کرد داده غالب هر دسته کدام اند. داده ها در هم تنیده شده هستند واحتمال همه ی ساز ها باهم برابر است.

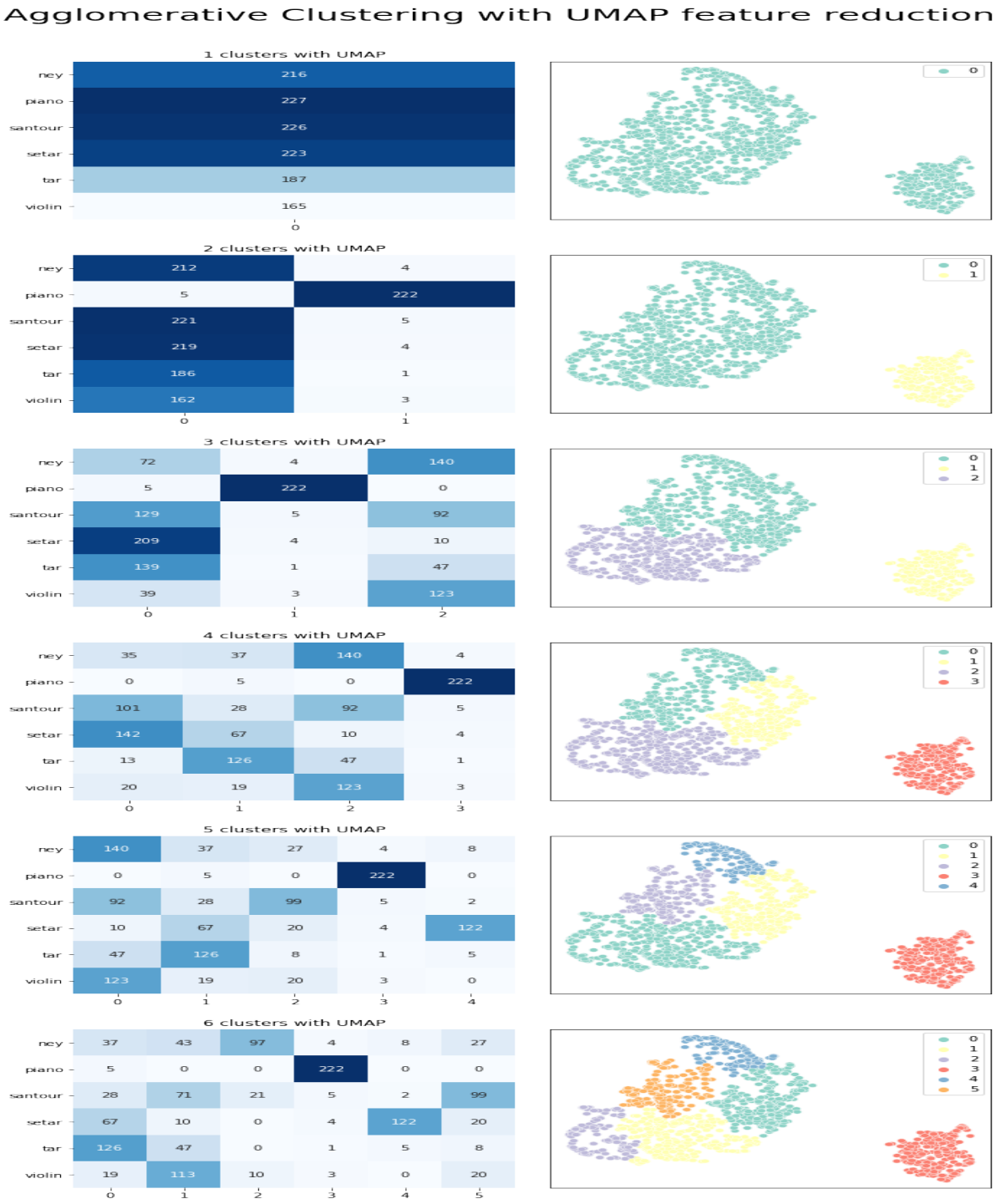
نهایتا هم در خوشه ۶ تایی همانند خوشه ۴ تایی اطمینانی برای تشخیص هر دسته وجود ندارد.

با مقایسه نمودار فوق و نمودار مربوط به kmean متوجه میشویم که نمودار kmean خوشه بندی ها را بهتر انجام داده است و فاصله ی بین کلاسی خوشه ها بیشتر بود.
شکل زیر متریک های بدست آمده این الگوریتم می باشد.

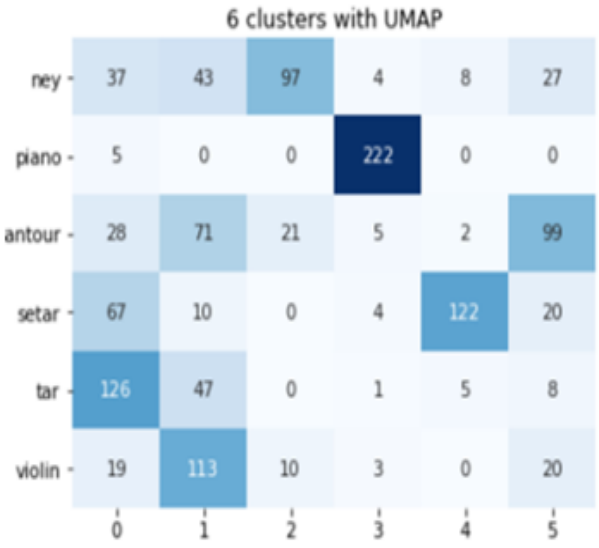
type	time	R-Score	AMI	NMI	Homo	Comp	V-meas	Silh	CH-score	DB-score
Agglomerative with 2 clusters	0.143s	0.128	0.334	0.335	0.211	0.817	0.335	0.141	200.262	1.522
Agglomerative with 3 clusters	0.131s	0.154	0.357	0.359	0.256	0.598	0.359	0.151	157.979	2.030
Agglomerative with 4 clusters	0.089s	0.216	0.346	0.349	0.296	0.426	0.349	0.114	149.179	2.337
Agglomerative with 5 clusters	0.088s	0.245	0.342	0.345	0.315	0.382	0.345	0.099	133.072	2.299
Agglomerative with 6 clusters	0.090s	0.267	0.369	0.373	0.357	0.390	0.373	0.100	120.492	2.395
Agglomerative with 6 clusters & UMAP	0.101s	0.367	0.431	0.434	0.429	0.440	0.434	0.114	128.927	2.334

در ادامه با اعمال umap همین روند را تکرار میکنیم تا ببینیم به نتایج بهتر میرسیم یا خیر.




Agglomerative Clustering with UMAP feature reduction
1 clusters with UMAP
ney 216
piano 227
santour 226
setar 223
tar 187
violin 165
0
2 clusters with UMAP
ney 212 4
piano 5 222
santour 221 5
setar 219 4
tar 186 1
violin 162 3
0 1
3 clusters with UMAP
ney 72 4 140
piano 5 222 0
santour 129 5 92
setar 209 4 10
tar 139 1 47
violin 39 3 123
0 1 2
4 clusters with UMAP
ney 35 37 140 4
piano 0 5 0 222
santour 101 28 92 5
setar 142 67 10 4
tar 13 126 47 1
violin 20 19 123 3
0 1 2 3
5 clusters with UMAP
ney 140 37 27 4 8
piano 0 5 0 222 0
santour 92 28 99 5 2
setar 10 67 20 4 122
tar 47 126 8 1 5
violin 123 19 20 3 0
0 1 2 3 4
6 clusters with UMAP
ney 37 43 97 4 8 27
piano 5 0 0 222 0 0
santour 28 71 21 5 2 99
setar 67 10 0 4 122 20
tar 126 47 0 1 5 8
violin 19 113 10 3 0 20
0 1 2 3 4 5

مقایسه خوشه های ۶ تایی و تاثیر feature reduction



در این روش داده های غالب مشخص نیستند و نمیتوان اظهار نظر کرد که هرخوشه متعلق به کدام ساز است.

به عنوان مثال در خوشه شماره ۰ داده های سه تار و تار بسیار احتمال نزدیکی بهم دارند

در خوشه شماره ۱ داده های نی و سنتور نزدیک هم دیگرند

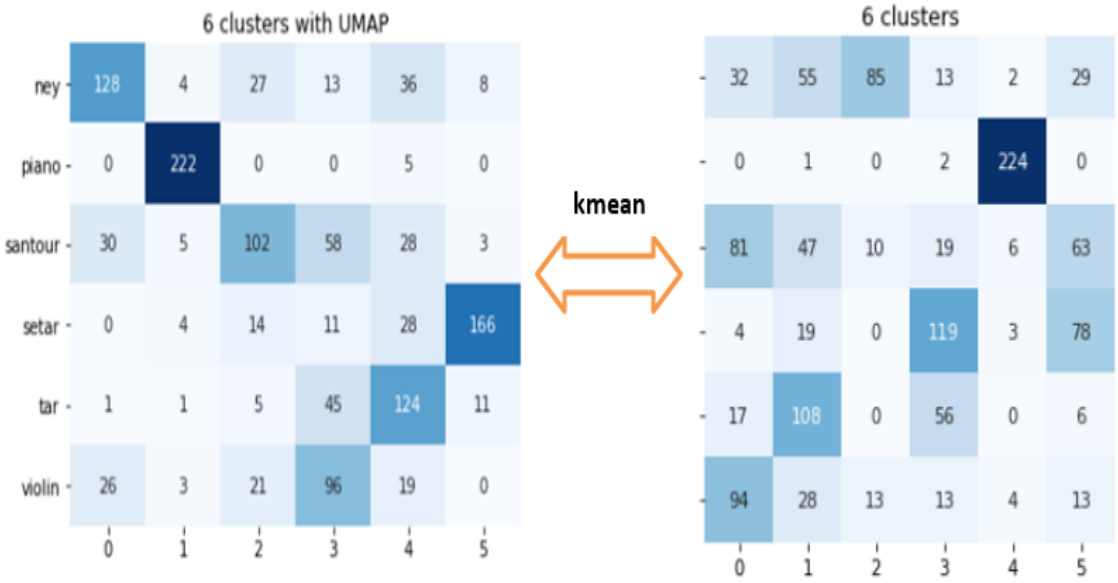
در این روش که از Umap استفاده شده ،داده های غالب کلاس ها به ترتیب عبارتند از :

۱۲۶ ، ۱۱۳ ، ۹۷ ، ۲۲۲ ، ۱۲۲ ،۹۹

که با سایر داده ها اختلاف بسیاری دارند.

با مقایسه مقادیر فوق تاثیر چشمگیر feature reduction قابل شهود است.

مقایسه روش Agglomerative و kmean :

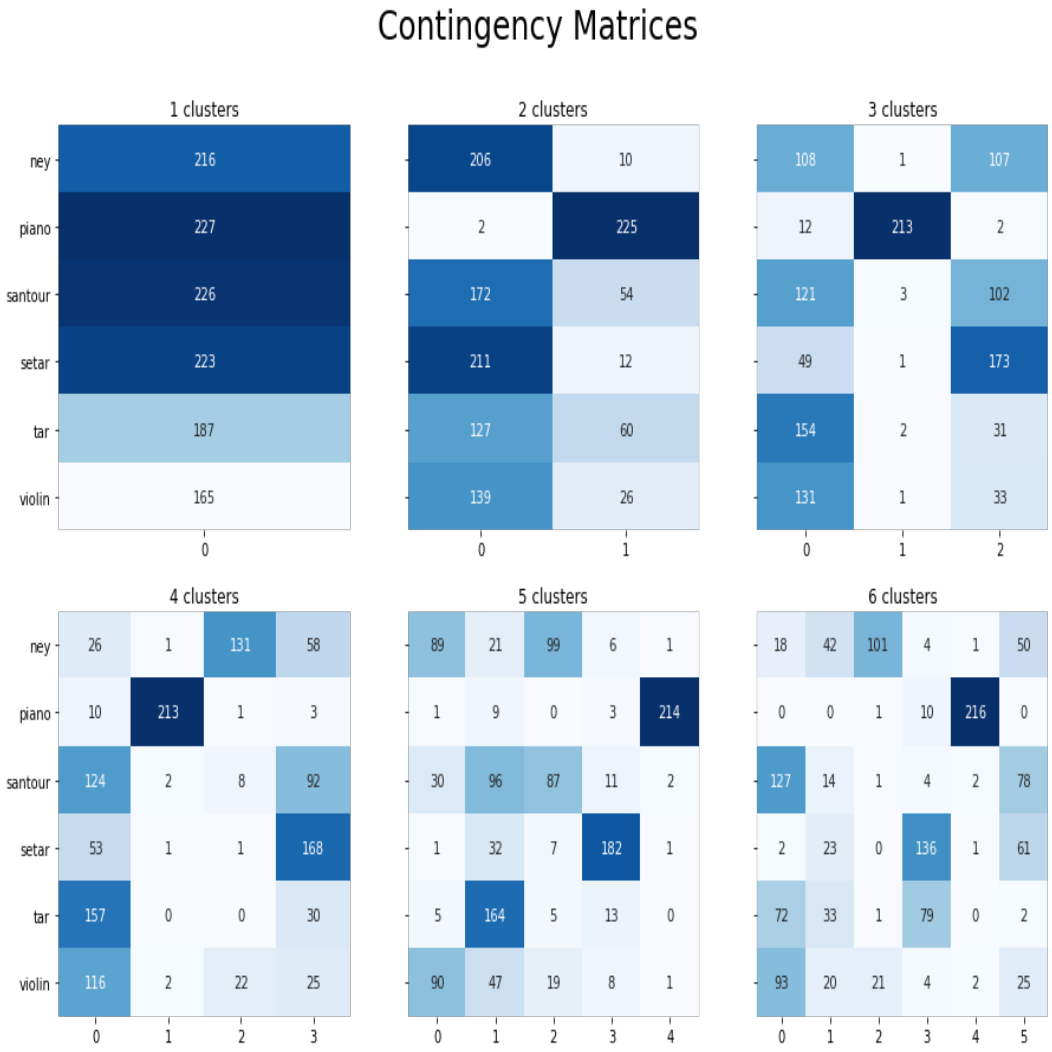

6 clusters with UMAP
ney 128 4 27 13 36 8
piano 0 222 0 0 5 0
santour 30 5 102 58 28 3
setar 0 4 14 11 28 166
tar 1 1 5 45 124 11
violin 26 3 21 96 19 0
0 1 2 3 4 5
kmean
6 clusters
32 55 85 13 2 29
0 1 0 2 224 0
81 47 10 19 6 63
4 19 0 119 3 78
17 108 0 56 0 6
94 28 13 13 4 13
0 1 2 3 4 5


6 clusters with UMAP
ney 37 43 97 4 8 27
piano 5 0 0 222 0 0
antour 28 71 21 5 2 99
setar 67 10 0 4 122 20
tar 126 47 0 1 5 8
violin 19 113 10 3 0 20
0 1 2 3 4 5
agglo
6 clusters
21 82 84 1 19 9
17 0 0 208 2 0
85 88 23 1 17 12
132 11 0 3 5 72
141 22 1 0 22 1
69 24 17 2 53 0
0 1 2 3 4 5

الگوریتم GMM

مزایای الگوریتم GMM

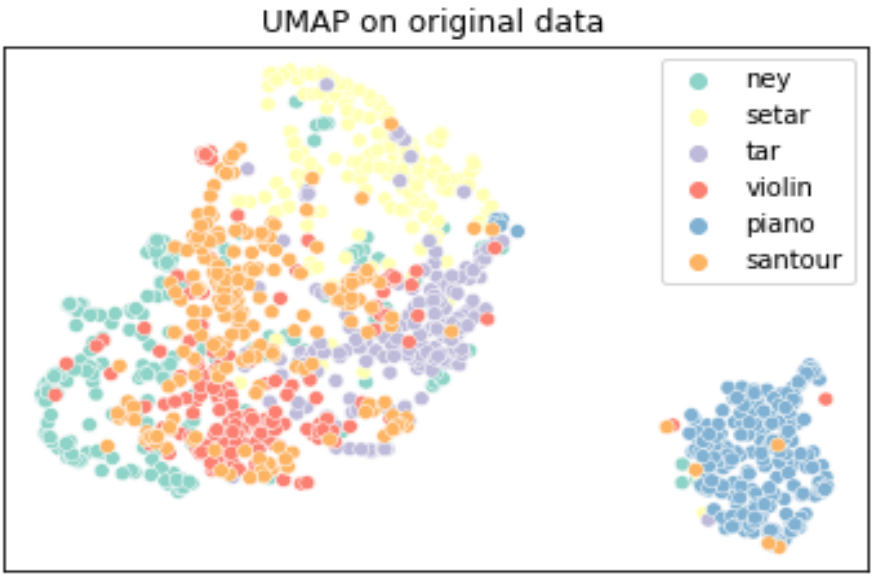
GMM یک روش خوشه‌بندی نرم است که نقاط نمونه را به چندین خوشه نسبت می‌دهد. این ویژگی موجب شده الگوریتم GMM به سریع‌ترین الگوریتم در یادگیری مدل‌های مخلوط تبدیل شود.
در این روش خوشه‌ها به لحاظ تعداد و اشکال متفاوت و انعطاف‌پذیر هستند.



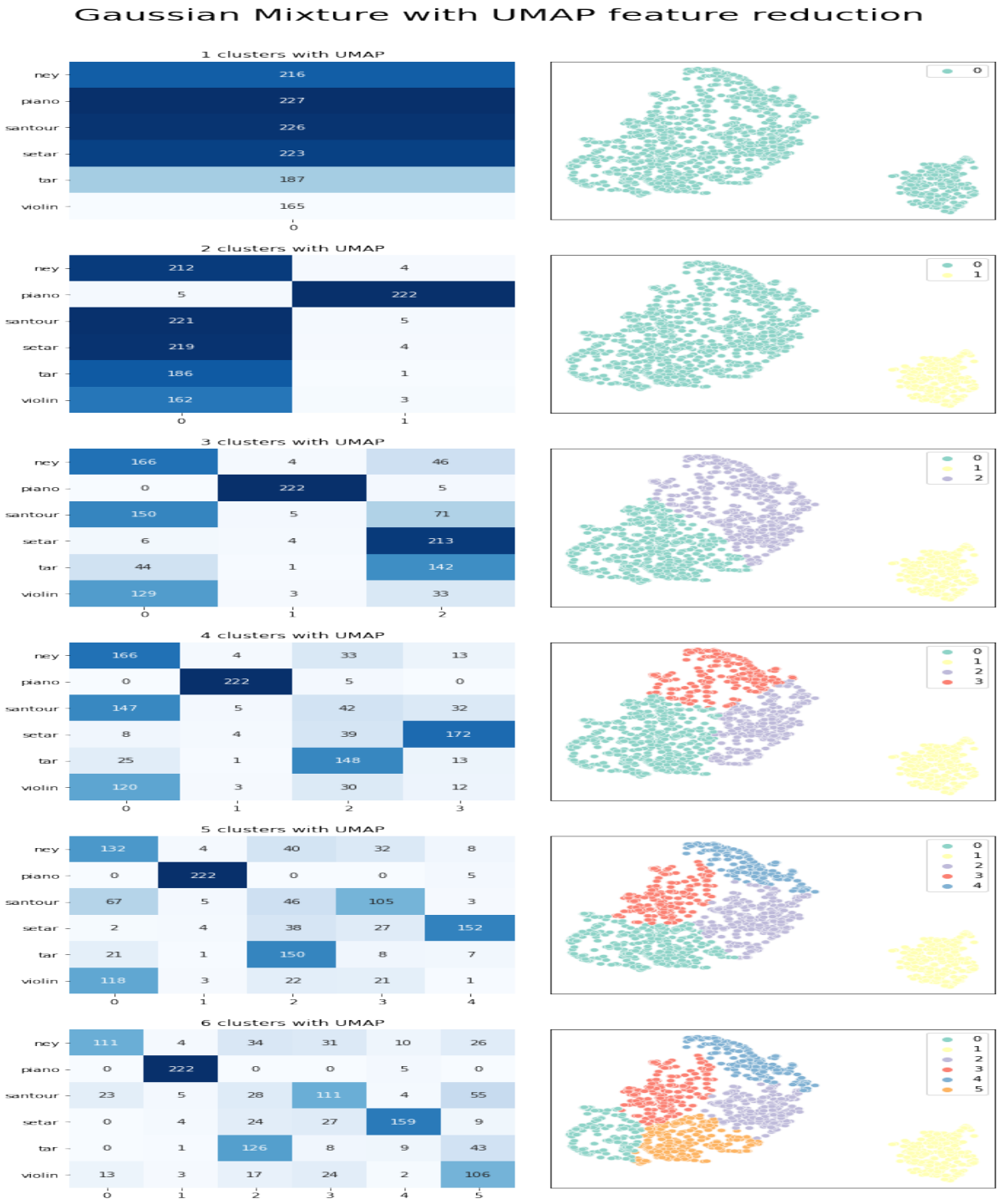
در این روش هم همانند روش قبل تعداد خوشه ها به مرور افزایش یافته و روند دسته بندی داده ها قابل مشاهده است.

type	time	R-Score	AMI	NMI	Homo	Comp	V-meas	Silh	CH-score	DB-score
Gaussian Mixture with 2 clusters	0.123s	0.143	0.239	0.240	0.162	0.466	0.240	0.116	188.899	2.152
Gaussian Mixture with 3 clusters	0.203s	0.219	0.337	0.338	0.267	0.462	0.338	0.111	165.675	2.591
Gaussian Mixture with 4 clusters	0.215s	0.303	0.415	0.418	0.361	0.495	0.418	0.109	144.080	2.403
Gaussian Mixture with 5 clusters	0.241s	0.411	0.474	0.476	0.449	0.507	0.476	0.106	132.806	2.400
Gaussian Mixture with 6 clusters	0.344s	0.362	0.443	0.446	0.441	0.451	0.446	0.094	112.217	2.908
Gaussian Mixture with 6 clusters & UMAP	0.043s	0.426	0.458	0.461	0.460	0.462	0.461	0.113	130.404	2.219

تاثیر استفاده از umap در روش GMM بیشتر از هر پارامتر دیگری روی زمان همگرایی اثر گذاشته است.



استفاده از umap و کاهش ابعاد بر روی داده های اصلی


Gaussian Mixture with UMAP feature reduction
1 clusters with UMAP
ney 216
piano 227
santour 226
setar 223
tar 187
violin 165
0
2 clusters with UMAP
ney 212 4
piano 5 222
santour 221 5
setar 219 4
tar 186 1
violin 162 3
0 1
3 clusters with UMAP
ney 166 4 46
piano 0 222 5
santour 150 5 71
setar 6 4 213
tar 44 1 142
violin 129 3 33
0 1 2
4 clusters with UMAP
ney 166 4 33 13
piano 0 222 5 0
santour 147 5 42 32
setar 8 4 39 172
tar 25 1 148 13
violin 120 3 30 12
0 1 2 3
5 clusters with UMAP
ney 132 4 40 32 8
piano 0 222 0 0 5
santour 67 5 46 105 3
setar 2 4 38 27 152
tar 21 1 150 8 7
violin 118 3 22 21 1
0 1 2 3 4
6 clusters with UMAP
ney 111 4 34 31 10 26
piano 0 222 0 0 5 0
santour 23 5 28 111 4 55
setar 0 4 24 27 159 9
tar 0 1 126 8 9 43
violin 13 3 17 24 2 106
0 1 2 3 4 5

معایب GMM

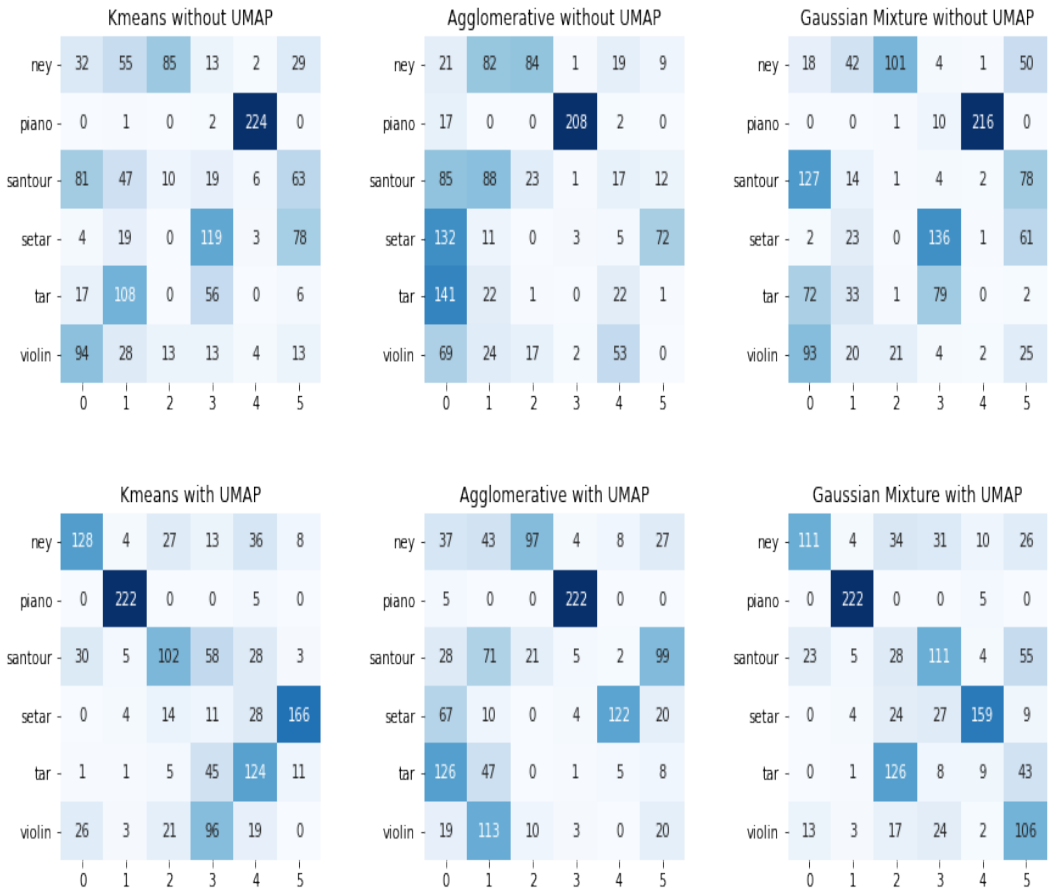
الگوریتم GMM نسبت به مقادیر اولیه بسیار حساس است و این مقادیر می‌تواند کیفیت عملکرد آن را تحت تأثیر قرار دهند.

GMM ممکن است با کمینه محلی همگرا شود و همین امر باعث می‌شود این الگوریتم کمتر بهینه باشد.

زمانی که به ازای هر مخلوط نقطه کافی وجود نداشته باشد، الگوریتم واگرا می‌شود و راهکارهایی با احتمال درستنمایی‌های بی‌نهایت پیدا می‌کند، مگر این‌که کوواریانس میان داده‌ها را به صورت مصنوعی تنظیم کنیم.

جمع بندی نهایی

type	time	R-Score	AMI	NMI	Homo	Comp	V-meas	Silh	CH-score	DB-score
K-means	0.141s	0.340	0.394	0.398	0.395	0.401	0.398	0.140	155.776	2.011
K-means with UMAP	0.108s	0.439	0.470	0.473	0.473	0.473	0.473	0.109	128.474	2.312
Agglomerative	0.086s	0.267	0.369	0.373	0.357	0.390	0.373	0.100	120.492	2.395
Agglomerative with UMAP	0.063s	0.367	0.431	0.434	0.429	0.440	0.434	0.114	128.927	2.334
GMM	0.349s	0.362	0.443	0.446	0.441	0.451	0.446	0.094	112.217	2.908
GMM with UMAP	0.041s	0.426	0.458	0.461	0.460	0.462	0.461	0.113	130.404	2.219



GMs یک الگوریتم احتمالی است. با تخصیص احتمالات به نقاط داده، می توانیم بیان کنیم که چقدر حدس ما به اینکه یک نقطه داده معین به یک خوشه خاص تعلق دارد، قوی است. اگر هر دو الگوریتم را مقایسه کنیم، به نظر می‌رسد که GM ها قوی‌تر هستند. با این حال، GM ها معمولاً نسبت به kmean کندتر هستند، زیرا برای رسیدن به همگرایی، تکرارهای بیشتری از الگوریتم EM نیاز است. آنها همچنین می توانند به سرعت به یک حداقل محلی همگرا شوند که راه حل چندان مطلوبی نیست. هرچند kmean الگوریتم خوشه بندی بسیار مناسبی است، اما بیشتر مناسب مواقعی است که از قبل تعداد دقیق خوشه‌ها را می‌دانیم و با توزیع‌های کروی شکل سروکار داریم.

LaTeX