**بسمه تعالی**

**دانشکده مهندسی مکانیک**

**پردیس دانشکده های فنی**

**دانشگاه تهران**

# هوش مصنوعی

پروژه 4


دانشجو:

شهریار نامداری

810098043


استاد: دکتر فدایی

خرداد 1401

## فاز صفر:

1. متد های info و describe را بررسی میکنیم:
   - در متد info می‌بینیم که دیتاست دارای 16 ستون است که type یازده تا از آنها float و 5 تا object میباشد و دیتاست حافظه 3.7 مگابایت را اشغال کرده.
   - در متد describe برای هر کدام از ستون ها اطلاعاتی کلی از داده ها به دست می‌آوریم. از جمله اطلاعاتی که میتوانیم کسب کنیم شامل مقدار مینیمم، ماکسیمم، میانگین و انحراف معیار است.
2. فقط در سه ستون برخی داده ها حذف شده اند که به ترتیب زیر است:
   - نام هنرمند: 1494 داده که میشود 4.98 درصد
   - مدت زمان: 3010 داده که میشود تقریبا 10 درصد
   - تمپو: 2933 داده که میشود تقریبا 9.7 درصد
3. نمودار histogram برای داده های عددی رسم شد و برای ستون های tempo ، energy ، valence ، danceability و popularity از توزیع نرمال پیروی می‌کردند.

## فاز اول:

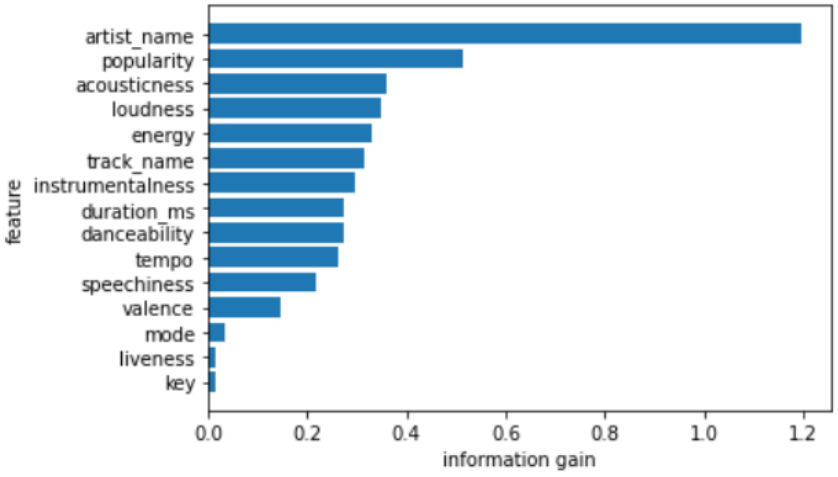
1. روش‌های مختلفی برای پر کردن داده‌های گم شده موجود است. از جمله جایگذاری داده ها با میانگین آن داده ها یا مود آن داده ها. همچنین میتوان طبق باقی داده ها و سطر هایی که داده هایشان موجود است میزان داده گم شده رو پیشبینی و محاسبه کرد و سپس آن را جایگذاری کرد. روش میانگین گیری روشیست که دقت آن چنان بالایی ندارد اما روشیست که به نسبت راحت میباشد و زمان و انرژی زیادی نمیگیرد. در کنار این روش برای داده های categorical استفاده از روش مود بهتر میباشد. در زمان هایی که تعداد داده ها کم است یا به طریقی ارزش داده های گم شده و تاثیر گذاری و آن‌ها بیشتر شود استفاده از این روش ها شاید مناسب نباشد و بهتر است با استفاده از باقی داده های موجود مقادیر گم شده را با دقت خوبی محاسبه کرد یا حدس زد.
2. مقادیر گم شده با روش های mean و mode جایگذاری شدند.
3. این کار به جهت هم اسکیل کردن داده‌ها و انجام میشود تا بتوان وزن های شبکه را هنگام به روز رسانی راحت تر به هدف رساند معمولا با این دو روش داده ها بین 0 تا 1 یا -1 تا 1 اسکیل میشوند.
   رابطه normalization :

```
X_new = (X - X_min)/(X_max - X_min)
```

رابطه standardization :

```
X_new = (X - mean)/Std
```

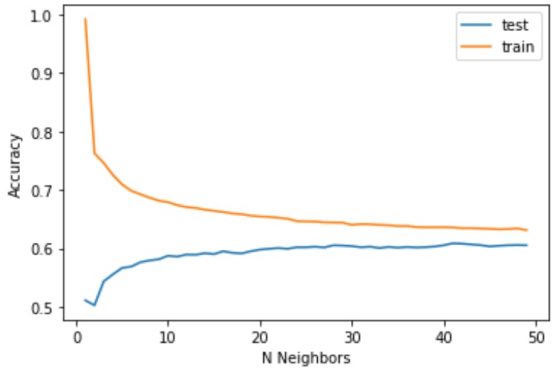
4. با توجه به اینکه توزیع داده های نرمال بوده روش standardization برای اسکیل کردن داده ها مناسب میباشد زیرا معمولا از روش standardization برای زمان‌هایی که داده ها از توزیع گاوسی پیروی میکنند استفاده میشود.
5. دو روش برای کار با داده های دسته ای موجود است. یکی اینکه از روش encoding استفاده کنیم و هر کدام از داده ها را سیستم به یک عدد map کند. روش دیگر map کردن دستی این داده ها به اعداد است که انعطاف بیشتری را به ما میدهد اما از آن جا که ما در این جا نیاز خاصی به انعطاف نداریم و صرفا جداسازی داده ها برایمان مهم است روش encoding روش مناسبی است که در بخش 3 فاز صفر این کار انجام شد(جهت نمایش توزیع داده ها).
6. میتوان از این مورد در پر کردن داده های گم شده استفاده کرد. برای مثال اگر نام هنرمندی گم شده ببینیم ژانر آن اهنگ چه بوده و از بین موسیقی های در آن ژانر مود گرفته و جایگذاری داده گم شده کنیم، نه اینکه از تمام داده ها مود بگیریم. در مورد سوال نیز باید گفت به همین دلیلی که در صورت سوال مطرح شد نام هنرمند میتواند ویژگی خوبی برای تشخیص ژانر باشد و کمک کننده خوبیست. پس این ستون را نگه می‌داریم.
7. محاسبات information gain انجام شد نمودار نهایی در زیر آمده است.




8. با توجه به نمودار آمده در سوال قبل میتوان ترتیب اهمیت اطلاعاتی که هر ویژگی به ما میدهد متوجه شد. منطقا نگه داشتن ویژگی هایی که به درد ما نمیخورند و اطلاعات مفیدی در راستای شناخت ژانر موسیقی به ما نمیدهند فقط پیچیدگی زمانی و محاسباتی را بیشتر می‌کند. همانطور که از نمودار بالا پیداست مواردی که در بالاتر آمده اند ویژگی هایی هستند که بیشتر به شناخت ژانر موسیقی کمک میکنند و هرچه پایین تر می‌آییم این کمک کنندگی کمتر میشود. برای مثال ویژگی هایی که پایین هستند زیاد کمک کننده نیستند و اگر قرار بر حذف ستونی از ویژگی ها باشد این موارد برای حذف مناسب هستند.

## فاز دوم:

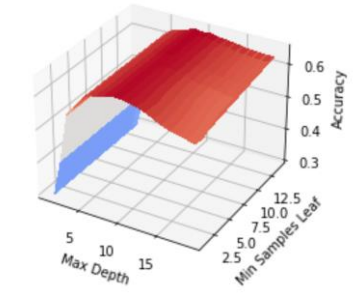
1. معمولا میزان 70 تا 80 درصد داده ها را برای train جدا کرده مقدار 20 تا 30 درصد را برای تست جدا میکنند. اگر درصد داده های آموزش بیشتر باشد آنگاه داده کافی برای تست مدل نخواهیم داشت و اگر کمتر باشد مدل نمیتواند به خوبی آموزش داده شود. هم چنین باید در داده های train و test به میزان تقریبا یکسان از همه کلاس ها داشته باشیم تا مدل به سمت کلاسی خاص سوگیری نکند. پارامتر stratify به ما کمک میکند تا این کار را انجام دهیم و در هنگام تقسیم بندی داده ها از هر کلاس تقریبا به اندازه مساوی داشته باشیم.
2. به کمک پارامتر stratify جدا سازی داده های آموزش و تست انجام شد.
3. شبکه KNN طراحی شد و مدل آموزش داده شد و نمودار خواسته شده رسم شد که مطابق شکل زیر است:



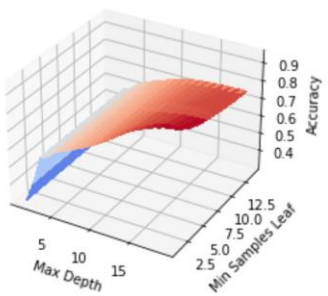

```
Best N Neighbors for test data: 41
Best Accuracy for test data: 61.133333333333326 %
```

همانطور که در نمودار بالا دیده میشود در تعداد همسایه پایین overfitting رخ میدهد و با زیاد شدن تعداد همسایه به یک میزان بهینه میرسیم و پس از آن وارد مرحله underfitting میشویم.

4. نمودار سه بعدی این مدل برای حالات تست و آموزش رسم شد که در زیر آمده است.




```
Best Accuracy for test data: 65.08333333333334 %
```

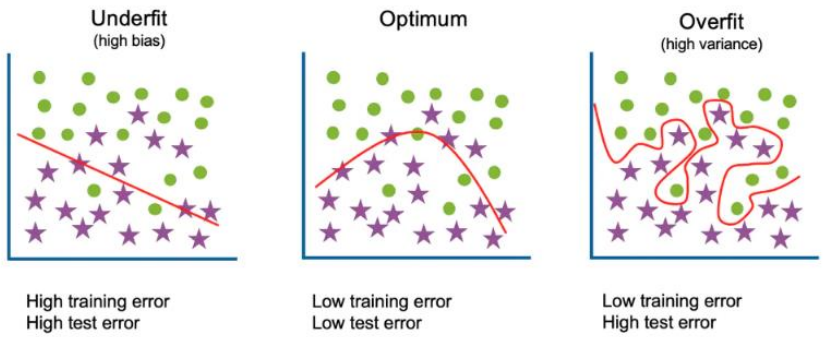



```
Best Accuracy for train data: 95.675 %
```

*نمودار سمت راست برای داده آموزش و سمت چپ برای داده تست است*

همانطور که مشاهده می‌شود داده در حالت آموزش دقت حتی تا 95 درصد هم بالا رفته ولی در حالت تست این دقت در حالت ماکسیمم به 65 درصد رسیده که این خود نشان دهنده وجود overfitting در فرایند آموزش میباشد. همچنین به نظر بهینه ترین حالت آموزش برای مقادیر max_depth برابر با 9 و مقدار min samples leaf برابر با 5 است.

5. به طور خلاصه به بررسی underfitting و overfitting میپردازیم:



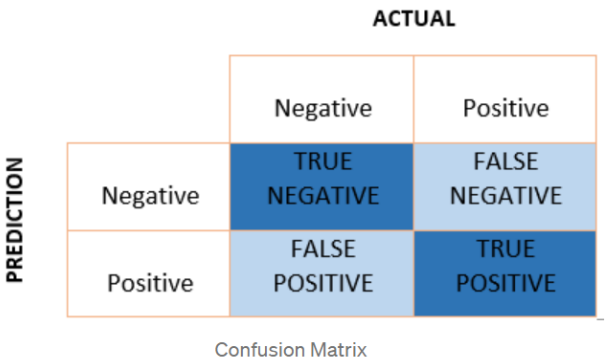

همانطور که در تصویر بالا میبینید وقتی که دقت داده های آموزش و تست هر دو کم باشد به معنی underfitting است و یعنی مدل نتوانسته رابطه میان داده های ورودی و خروجی آموزش را خوب یاد بگیرد و زمانی که overfitting رخ میدهد به این معناست که مدل حتی نویز ها را نیز یاد گرفت و به نوعی حفظ کردن رخ داده که مطلوب نخواهد بود و درست است که دقت داده های آموزش بالا میرود اما دقت داده های تست پایین می‌آید. همانطور که در بالاتر هم ذکر شد در مدل KNN باید نقطه بهینه را انتخاب کنیم تا از overfitting و underfitting جلوگیری کنیم. همچنین در D-Tree دیدیم که overfitting برای نقاط خاصی رخ می‌دهد. و باید حواسمان باشد که دقت داده های تست مهم است نه داده های train .

6.

برای سادگی انتقال مفاهیم، به بیان مفاهیم در دسته بندی باینری می‌پردازیم.

ماتریس آشفتگی:

طبقه بندی باینری ماتریس آشفتگی یک ماتریس دو در دو و به شکل زیر می‌باشد:




Confusion Matrix

فرض کنیم می‌خواهیم پارامتری که یا مثبت یا منفی است را بررسی کنیم.
بخش true positive(TP): در این بخش نشان داده می‌شود که چه مقدار از پیشبینی ها و مقادیر واقعی به طور همزمان مثبت بوده‌اند. (درست حدس زده‌اند)
بخش true negative(TN): در این بخش نشان داده می‌شود که چه مقدار از پیشبینی ها و مقادیر واقعی به طور همزمان منفی بوده‌اند. (درست حدس زده‌اند)
بخش false positive(FP): در این بخش نشان داده می‌شود که چه مقدار از پیشبینی ها مثبت بوده‌اند ولی در واقع منفی بوده‌اند. (اشتباه حدس زده‌اند)
بخش false negative(FN): در این بخش نشان داده می‌شود که چه مقدار از پیشبینی ها منفی بوده‌اند ولی در واقع مثبت بوده‌اند. (اشتباه حدس زده‌اند)
به کمک این پارامترها می‌توانیم چهار مقدار کاربردی را بیابیم که در زیر روش محاسبه آن‌ها آمده است:

$$TPR = \frac{TP}{Actual\ Positive} = \frac{TP}{TP + FN}$$

$$FNR = \frac{FN}{Actual\ Positive} = \frac{FN}{TP + FN}$$

$$TNR = \frac{TN}{Actual\ Negative} = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{Actual\ Negative} = \frac{FP}{TN + FP}$$

در کل ماتریس آشفتگی به ما کمک می‌کند تا متوجه شویم که شبکه ما تا چه حد خوب یا بد کار می‌کند. برای این که شبکه خوب کار کند باید مقادیر TNR و TPR زیاد باشند و مقادیر FNR و FPR کم باشند.

در ادامه برای precision و recall آن بخش مثبت قضیه اهمیت بیشتری دارد. زیرا وقتی دنبال چیزی می‌گردیم ما اهمیتی به مواردی که نمی‌خواهیم و شبکه درست آن‌ها را تشخیص داده که نمی‌خواهیم (TN) نمی‌دهیم و بیشتر سر و کارمان با TP,FP و FN است.
Percision: از رابطه زیر محاسبه می‌شود (که مقداری بین 0 تا 1 دارد):

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

Recall: از رابطه زیر محاسبه می‌شود:

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

Accuracy : می‌شود مجموع تمام حدس‌های درست تقسیم بر کل حدس‌ها

به کمک این مقادیر و پارامترها می‌توان ارزیابی خوبی برای رسیدن به مطلوب شبکه داشت.
F1score: از رابطه زیر محاسبه می‌شود:

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

هم چنین پارامتر F1 score وزن دار نیز داریم که به صورت زیر محاسبه می‌شود:

$$F_\beta = (1 + \beta^2) * \frac{(Precision * Recall)}{(\beta^2 * Precision) + Recall}$$

در رابطه بالا بتا نشان می‌دهد که مقدار recall چند برابر مهم‌تر است از precision. حال اگر مانند مساله ای که الان با آن مواجه هستیم مساله باینری نباشد و چندین کلاس داشته باشیم:

| | | True/Actual | | |
|---|---|---|---|---|
| | | Cat (🐱) | Fish (🐟) | Hen (🐔) |
| Predicted | Cat (🐱) | 4 | 6 | 3 |
| | Fish (🐟) | 1 | 2 | 0 |
| | Hen (🐔) | 1 | 2 | 6 |

مثلاً در تصویر بالا سه کلاس داریم که به کمک 25 تصویر این دسته بندی صورت گرفته است.

- مقدار precision می‌شود تعداد تمام گربه‌های درست حدس زده شده تقسیم بر کل حدس‌های زده شده برای گربه که می‌شود 4 تقسیم بر 13
- مقدار recall می‌شود تعداد تمام گربه‌های درست حدس زده شده تقسیم بر تعداد کل تصاویر موجود برای گربه که می‌شود 4 تقسیم بر 6

مقدار accuracy که با توجه به اینکه تقسیم کلاس ها در داده تست یکسان انجام شده پارامتر مناسبی است برا اندازه گیری محاسبه شده و برای مدل KNN مقدار 61 درصد و برای D-Tree مقدار 65 درصد شد.

D-Tree:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.630 | 0.674 | 0.651 | 1000 |
| 1 | 0.537 | 0.388 | 0.450 | 1000 |
| 2 | 0.626 | 0.506 | 0.560 | 1000 |
| 3 | 0.752 | 0.769 | 0.760 | 1000 |
| 4 | 0.505 | 0.721 | 0.594 | 1000 |
| 5 | 0.883 | 0.837 | 0.859 | 1000 |
| accuracy | | | 0.649 | 6000 |
| macro avg | 0.655 | 0.649 | 0.646 | 6000 |
| weighted avg | 0.655 | 0.649 | 0.646 | 6000 |

KNN:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.640 | 0.589 | 0.613 | 1000 |
| 1 | 0.447 | 0.372 | 0.406 | 1000 |
| 2 | 0.464 | 0.627 | 0.533 | 1000 |
| 3 | 0.733 | 0.728 | 0.731 | 1000 |
| 4 | 0.528 | 0.501 | 0.514 | 1000 |
| 5 | 0.877 | 0.836 | 0.856 | 1000 |
| accuracy | | | 0.609 | 6000 |
| macro avg | 0.615 | 0.609 | 0.609 | 6000 |
| weighted avg | 0.615 | 0.609 | 0.609 | 6000 |

*پارامتر های خواسته شده برای دو مدل KNN و D-Tree*

7. از آنجایی که داده های گم شده از سه ستون tempo، artist_name و duration بودند و از آنجایی که این موارد (به جز نام هنرمند) ستون هایی با information gain تقریبا پایین بودند

پس میتوان تقریبا گفت که تغییر این ستون ها در نتیجه نهایی تاثیر زیادی نخواهد داشت. اما تغییر نام هنرمند چون gain بالایی داشته احتمالا تاثیر زیادی بتواند بگذارد. برای تلاش دوم بخش پر کردن داده های گم شده را جور دیگری انجام دادیم. این بار داده های گم شده را به کمک هم کلاسی های خود آن داده گم شده در ژانر موسیقی پر کردیم. دقت KNN روی 61 ثابت ماند و دقت D-Tree از 65 به 66 افزایش یافت. همانطور که انتظار داشتیم زیاد تغییری حاصل نشد. پیش پردازش دیگری که انجام داده بودم نرمال سازی داده ها بود که اگر داده ها را نرمال نکنیم نتیجه KNN از 60 به 33 کاهش می‌یابد و نتیجه D-Tree به 65 کاهش می‌یابد.

## فاز سوم:

1. مدل Random Forest آموزش داده شده و دقت تقریبا 70 درصد با تغییر اندک دستی هایپرپارامترها حاصل شد.
2. به توضیح هایپرپارامترهای پرسیده شده می‌پردازیم:

- n_estimators : تعداد درخت های داخل جنگل را نمایش میدهد. به زبان دیگر نشان دهنده تعداد درخت های تصمیمی که به کمک آن ها تصمیم نهایی را میگیرد است.
- max_depth : حداکثر عمق درخت. که برای مثال مقدار آن با توجه به ستون ویژگی ها 10 قرار داده شده.
- min_samples_leaf : حداقل تعداد نمونه مورد نیاز برای قرار گرفتن در یک گره برگ. به عنوان مثال، min_samples_leaf = 10، به هر درخت می‌گوید اگر splitting باعث می‌شود که گره انتهایی هر شاخه‌ای کمتر از 10 برگ داشته باشد، splitting را متوقف کند. این پارامتر اثر هموارسازی برای مدل دارد؛ به خصوص در مسائل رگرسیون.

همچنین پس از محاسبه برای چندین هایپر پارامتر، هایپرپارامتر های زیر برای دقت 70 درصد مناسب می‌باشند:

n_estimator = 103
max_depth = 10
min_samples_leaf = 7

3. نتایج چنین است:

Random Forest:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.724 | 0.717 | 0.721 | 1000 |
| 1 | 0.636 | 0.396 | 0.488 | 1000 |
| 2 | 0.696 | 0.623 | 0.658 | 1000 |
| 3 | 0.750 | 0.791 | 0.770 | 1000 |
| 4 | 0.533 | 0.794 | 0.638 | 1000 |
| 5 | 0.917 | 0.870 | 0.893 | 1000 |
| accuracy | | | 0.699 | 6000 |
| macro avg | 0.709 | 0.699 | 0.695 | 6000 |
| weighted avg | 0.709 | 0.699 | 0.695 | 6000 |

4. ماتریس confusion نیز رسم شد که در شکل زیر قابل مشاهده است: