فایل ضمیمه : hw2.ipynb

فایل داده ها به نام pd_speech_features.csv را در قالب بک دیتافریم pandas باز میکنم. دادگان ۷۵۶ سطر با ۷۵۵ ستون است که از این ستون ها، ستون class را به عنوان متغیر هدف میگیرم که یک مقدار ۰ یا ۱ دارد. مساله یک کلاس بندی دوکلاسه است. از بقیه ستون ها، ستون id هم کنار می‌گذارم چون فقط کد نمونه ای است که داده از آن جمع آوری شده و ربطی به فضای مساله ندارد. این ۷۵۳ ستون را به عنوان مجموعه متغیرهاس وابسته می‌گیرم. تمام ستون ها مقدار عددی دارند. با استفاده از sklearn.model_selection تابع train_test_split دادگان را درهم می‌کند و با نسبت ۳۰ به ۷۰ داده های تست و آموزش را جدا می‌کند. برای اینکه نتایج بهتری از کلاس بندی و عملکرد مدل ها بگیرم، قبل از ساخت مدل ها داده ها را استاندارد می‌کنم. تابع StandardScaler از sklearn.preprocessing با استفاده از میانگین و واریانس مجموعه ی آموزش، استاندارد سازی را روی هر دو مجموعه ی آموزش و تست انجام می‌دهد.

Decission Tree

اولین مدل ساخته شده روی داده ها clf6 یک درخت تصمیم است با معیار information gain است که در جهت بهترین شاخه سازی برای کاهش انتروپی درخت را می‌سازد. این مدل با توابع sklearn ساخته شده. عمق 8 دارد. روی داده های آموزش به طور تقریبا کامل fit شده و دقت ۹۷ دارد. در هر برگ حداقل ۱ داده جای گرفته و روی هر گره برای شاخه سازی حداقل ۲ داده وجود داشته اند و معیارهای هرس درخت کاملا حداقلی تنظیم شده اند. اولین تلاش برای ساخت درخت این مساله درختی با عمق ۱۰ ساخته شد که روی تست نتیجه نزدیک ۷۰ داشت و برای کاهش خطای اورفیت برای عمق محدودیت ۸ گذاشتم و در مدل نهایی داده های تست را با clf6 دسته بندی می‌کنم. نتایج زیر به دست آمده به ترتیب روی مجموعه تست و بعد آموزش :

	precision	recall	f1-score	support
0	1.00	0.89	0.94	123
1	0.97	1.00	0.98	406
accuracy			0.97	529
macro avg	0.98	0.94	0.96	529
weighted avg	0.97	0.97	0.97	529

	precision	recall	f1-score	support
0	0.71	0.51	0.59	69
1	0.81	0.91	0.86	158
accuracy			0.79	227
macro avg	0.76	0.71	0.73	227
weighted avg	0.78	0.79	0.78	227

```
[[ 35  34]
 [ 14 144]]
```

ضعف اصلی مدل مربوز به تشخیص کلاس صفر هست. چند بار مدلسازی انجام شد و هر بار دقت حدود همین ۸۰ درصد بود. مدل عملکرد بدی ندارد.

Random Forest

دومین مدل ساخته شده روی داده ها clf7 یک جنگل تصادفی است که در آن ۱۰۰ درخت با معیار information gain به عنوان دسته بند وجود دارند و هر کدام با نمونه گیری bootstrap روی داده ها و روی ویژگی ها در جهت بهترین شاخه سازی برای کاهش انتروپی ساخته شده اند. ساخت درخت با استفاده از sklearn انجام شده. دسته بندی آموزش با دقت ۱۰۰ درصد انجام شده. با براورد برچسب داده های تست با clf7 نتایج زیر به دست می‌آید.

	precision	recall	f1-score	support
0	0.87	0.49	0.63	69
1	0.81	0.97	0.88	158
accuracy			0.82	227
macro avg	0.84	0.73	0.76	227
weighted avg	0.83	0.82	0.81	227

```
[[ 34  35]
 [  5 153]]
```

نسبت به درخت تصمیم نتیجه بهتر شده. تفاوت در accuracy کم است و تغیر بیشتر در تشخیص کلاس صفر پیدا شده و بهبود مدل مربوط به همین بخش precision است.

XGBoost

مدل سوم یا clf8 یک مدل XGBoost است که با کتابخانه ی xgboost ساخته شده. برای این مدل از پارامترهای پیش فرض استفاده شده و تعداد درخت ها ۱۰۰ ، نرخ یادگیری ۰,۳۰۰۰۰۰۰۱۲ هست و حداکثر عمق هر درخت ۶ است. هربار هر درخت به روش تکراری از روی درخت قبلی ساخته می‌شود طوری که درخت قبلی را بهبود بدهد. مدل روی داده های آموزش بدون خطا فیت شده. نتایج تست به صورت زیر است:

	precision	recall	f1-score	support
0	0.95	0.59	0.73	69
1	0.85	0.99	0.91	158
accuracy			0.87	227
macro avg	0.90	0.79	0.82	227
weighted avg	0.88	0.87	0.86	227

```
[[ 41  28]
 [  2 156]]
```

همان طور که قابل انتظار هم بود clf8 نسبت به هر دو مدل قبلی وضعیت بهتری دارد. این مدل دقت ۸۷ دارد و به زور واضحی در precision کلاس صفر بهتر عمل می‌کند. این مدل بهبود یافته ی همان جنگل تصادفی است به صورتی که ویژگی های مثبت آن را دارد از جمله اینکه از اجماع رای چند دسته بند استفاده می‌کند که به صورت تصادفی روی نمونه هایی از ویژگی ها و داده ها ساخته شده اند و این مزیت را دارد که تنها درخت هایی را شامل می‌شود که در نمونه های ساخته شده یکدیگر را کامل می کنند.

SVM

مدل clf9 یک ماشین بردار پشتیبان است. با مقدار ضریب منظم سازی ۵ و برای انتقال داده ها به فضای جدید از تابع radial basis به عنوان kernel استفاده می‌شود. داده های آموزش با دقت کامل دسته بندی می‌شوند و روی مجموعه تست نتایج مدل به صورت زیر هستند :

	precision	recall	f1-score	support
0	0.90	0.52	0.66	69
1	0.82	0.97	0.89	158
accuracy			0.84	227
macro avg	0.86	0.75	0.78	227
weighted avg	0.85	0.84	0.82	227

```
[[ 36  33]
 [  4 154]]
```

کارکرد svm از xgboost کمتر است و در خد و اندازه ی جنگا تصادفی است. یعنی از مدل clf6 بهتر است اما به دلیل نزدیکی اعداد بهترین راه مقایسه این ۳ مدل روش میانگین k fold cross validation است اما چون مدل clf8 که د رهمین سطح هم مشخصا از دوتای دیگر وضعیت بهتری دارد را در دست داریم این مقایسه نتیجه ای برای پیدا کردن بهترین مدل ندارد.

MLP

مدل اول clf1 یک شبکه پرسپترون چندلایه است که با sklearn.neural_network.MLPClassifier ساخته شده. به عنوان تابع فعالساز، از تابع غیرخطی سیگموید یا لوجستیک استفاده کردم. شبکه دو لایه دارد و تعداد نودهای لایه ی پنهان شبکه ۲۰ عدد است و حداکثر ۱۰۰۰ ایپاک برای آموزش صرف می‌شود. نرخ یادگیری برابر مقدار پیش قرض 0.001 گرفته شده. این مدل بعد از آموزش، روی داده های آموزشی به طور کامل فیت می‌شود و بعد از کلاس بندی داده های تست و اندازه گیری مقدارهای خطا عملکرد مدل به صورت زیر است :

	precision	recall	f1-score	support
0	0.68	0.71	0.70	69
1	0.87	0.85	0.86	158
accuracy			0.81	227
macro avg	0.78	0.78	0.78	227
weighted avg	0.81	0.81	0.81	227

همچنین confusion matrix برای داده های تست به صورت زیر می‌باشد:

```
[[ 49  20]
 [ 23 135]]
```

عملکرد مدل clf1 در تشخیص کلاس دوم (کلاس ۱) با توجه تمام معیارهای f1و recall و precision بهتر است و در کل عملکرد ۸۱ درصد است. با توجه به اختلاف accuracy روی مجموعه تست و آموزش، احتمال دادم که کمی خطای اورفیت داشته باشد. همین مدل را با پارامترهای متفاوت تری ساختم و طی چند بار آزمایش و خطا مدل هایی با ایپاک کمتر آموزش دادم یا تعداد نودهای لایه میانی را کم و زیاد کردم. ولی بهترین نتیجه های به دست آمده در حدود همین ۸۰ تا ۸۵ درصد بود و بهبودی در نتیجه ی تست حاصل نمی شد. پس همین را به عنوان نتیجه ی مدل اول گزارش کردم. این مدل به لحاظ accuracy در حد مدل های clf6 و clf7 و clf9 هست اما به لحاظ مقدار recall در کلاس صفر خیلی بهتر از آن ها عمل می‌کند

ELM

مدل دوم باز هم یک شبکه ی عصبی است که این بار به روش ELMآموزش داده می‌شود. کد ساخت و آموزش را خودم در توابع مختلف نوشتم و اجرا کردم. تابع ELM تابع ساخت و آموزش شبکه است که اول به تعداد فیچرها نود ورودی تعین می‌کند، سپس برای بردار وزن های بین ورودی ها با لایه پنهان اعداد تصادفی می‌گیرد. وزن های بایاس را هم تصادفی تعین می‌کند و در آخر از راه حل معادله ی ماتریس و محاسبه ی ماتریس وارون، وزن های لایه میانی به خروجی را پیدا می‌کند طوری که بردار خروجی برابر مقدار متغیرهای هدف بشود. برای محاسبه ی مقدار در نودهای لایه پنهان از تابع hidden_nodes استفاده می‌کند. این تابع قبل از ارسال مقادیر به لایه بعدی(خروجی) یک تابع ریاضی به عنوان تابع فعالساز روی مقادیر اعمال می‌کند. من اینجا از تابع سیگموییید استفاده کردم که در تابع sigmoid تعریف شده. نهایتا predict براورد را انجام می‌دهد و کلاس بندی داده ها روی مجموعه ورودی و خروجی را انجام می‌دهد. مدل این قسمت را با ۳۵ نود در لایه پنهان و به صورت دو لابه ساختم. نتایج آموزش و confusion matrix آن به صورت زیر است :

	precision	recall	f1-score	support
0	0.62	0.45	0.52	123
1	0.85	0.92	0.88	406
accuracy			0.81	529
macro avg	0.74	0.68	0.70	529
weighted avg	0.79	0.81	0.80	529

```
[[ 55  68]
 [ 33 373]]
```

و بعد از کلاس بندی داده های x_test مقدار دقت مدل به صورت زیر است :

	precision	recall	f1-score	support
0	0.69	0.45	0.54	69
1	0.79	0.91	0.85	158
accuracy			0.77	227
macro avg	0.74	0.68	0.70	227
weighted avg	0.76	0.77	0.75	227

```
[[ 31  38]
 [ 14 144]]
```

دقت clf2 کم است و حتی با تغیر پارامترها روی داده های آموزش هم به طور کامل یا حتی با نتیجه هایی که به طور چشمگیر بهتر از این باشد فیت نمی‌شود. البته به دلیل تصادفی بودن نیمی از وزن های مدل، می توان با پارامترهای ثابت نتایج متفاوتی گرفت ولی حتی در بهترین حالت باز هم نسبت به clf1 ضعیف تر عمل می‌کند. از روی نتایج جدول بالا، clf2 هم روی کلاس ۱ عملکرد بهتری دارد و حتی این عملکرد روی recall از clf1 هم بهتر است. یعنی در اطمینان به برچسب ۱ بهتر است ولی روی کلاس صفر دقت قابل قبول دارد اما اط طرف دیگر recall یا به تعبیری همان اطمینان به برچسب صفر از حالت شانس ۵۰-۵۰ هم بدتر است! و همین مقدار f1 را هم متاثر می‌کند و در کل مدل قابل استفاده و اتکا نیست. توجه شود که در clf6 هم accuracy فقط کمی بهتر از این بود اما به لحاظ بقیه معیارهایی که ذکر شد، درخت تصمیم از clf2 خیلی بهتر عمل می‌کند.

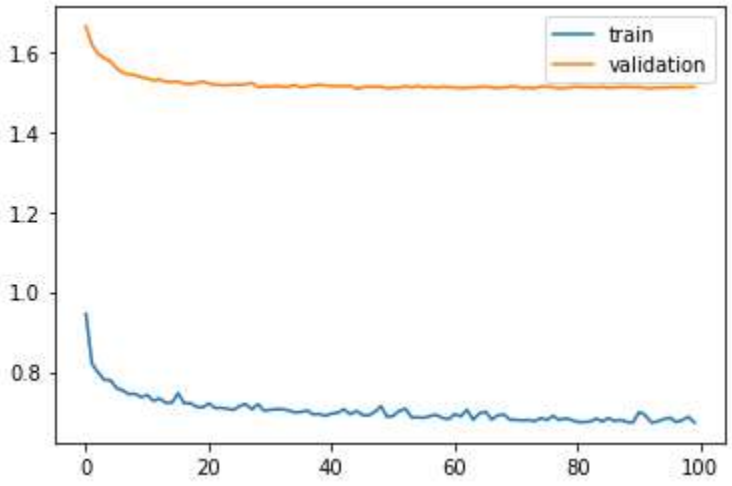
AutoEncoder

برای این قسمت داده ها را با استفاده از یک شبکه ی عصبی encode و decode کردم. تمام این عملیات شامل تعریف شبکه و معماری لایه ها و آموزش با استفاده از توابع و کلاس های tensorflow.keras انجام شده. در هر مرحله از یک شبکه دو لایه استفاده کردم و در بار اول داده ها را ¼ و در بخش رابط یا bottleneck شبکه تعداد فیچیرها را به ۸/۱ مقدار اولیه بردم. ودر بحش دیکودر به صورت برعکس اول داده ها را ۸ و بعد ۴ برابر کردم تا بازسازی مقادیر اولیه انجام شود. به عنوان تابع فعالساز خروجی از سیگوید استفاده کردم. سپس مدل را با همان x_train و در ۱۰۰ ایپاک با mse به عنوان تابه خطا آموزش دادم و برای محاسبه ی خطا هم از مجموعه ی x_test استفاده کردم. تمام این بخش ها به صورت بدون ناظر یعنی بدون حضور

متغیرهای هدف کلاس انجام می‌شود بنابراین می توان داده ها و نتایج انکودر را به صورت یک مساله جدا در نظر گرفت و استفاده از داده های تست برای ارزیابی انکودر تاثیر روی نتایج کلاسیفایرهای این بخش ندارد. شبکه ی ساخته شده به صورت زیر است :

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 753)]             0
_________________________________________________________________
dense (Dense)                (None, 753)               567762
_________________________________________________________________
batch_normalization (BatchNo (None, 753)               3012
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 753)               0
_________________________________________________________________
dense_1 (Dense)              (None, 188)               141752
_________________________________________________________________
batch_normalization_1 (Batch (None, 188)               752
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 188)               0
_________________________________________________________________
dense_2 (Dense)              (None, 94)                17766
=================================================================
Total params: 731,044
Trainable params: 729,162
Non-trainable params: 1,882
```

حالا از بخش encoder این شبکه استفاده کردم تا به عنوان یک فشرده ساز، تعداد متغیرهای مستقل کلاسیفایر را کمتر کند. این encoder تعداد ویژگی ها را بدون توجه به کلاس آن ها در مساله اصلی از ۷۵۳ به ۹۴ ویژگی می‌برد. نمودار روند آموزش شبکه ی autoencoder به صورت زیر است :



حالا روی این مجموعه داده ی فشرده شده ی x_train_encode ، سه مدل clf3 و clf4 و clf5 را ساختم و آموزش دادم. و بعد داده های تست را با همین مدل فشرده کردم و دسته بندی را روی x_test_encode انجام دادم تا دقت محاسبه شود.

مدل اول این بخش یا clf3 یک پرسپترون است. این مدل ساده را با sklearn.linear_model.Perceptron ساختم و آموزش دادم. نتایج دسته بندی آموزش به صورت زیر است :

	precision	recall	f1-score	support
0	0.64	0.82	0.72	123
1	0.94	0.86	0.90	406
accuracy			0.85	529
macro avg	0.79	0.84	0.81	529
weighted avg	0.87	0.85	0.86	529

```
[[101  22]
 [ 56 350]]
```

و با دسته بندی داده های تست با clf3 نتیجه ی عملکرد مدل به صورت زیر به دست میآید :

	precision	recall	f1-score	support
0	0.70	0.68	0.69	69
1	0.86	0.87	0.87	158
accuracy			0.81	227
macro avg	0.78	0.78	0.78	227
weighted avg	0.81	0.81	0.81	227

```
[[ 47  22]
 [ 20 138]]
```

روی داده های تست نتیجه ی کلاسیفایر خطی پرسپترون برای داده های فشرده شده از این مقدار بین ۷۵ تا ۸۲ بهتر نمی‌شد و در این اجرا، دقت 81 به دست آمد. این داده ها به صورت خطی بهتر از این قابلیت جداسازی با یک ابرصفحه را ندارند. یک تفاوت بارز این مدل با clf2 بهبود precision و recall کل است. clf3 از clf2 بهتر عمل می‌کند. مدل همچنان در کلاس ۱ وضع کلی بهتری دارد اما نسبت به clf2 دقت یکنواخت تری روی دو کلاس دارد.

مدل دوم این بخش clf4 همان شبکه ی پرسپترون چندلایه ای است که برای clf1 استفاده کردم و این بار با داده های فشرده شده به عنوان ورودی کار می‌کند (معماری و روش آموزش عینا همان است فقط تعداد ورودی ها که تعداد فیچرهای داده است کمتر شده). مدل روی داده های آموزش کاملا فیت می‌شود و دقت ۱۰۰ می دهد. روی مجموعه ی x_test_encode دسته بندی انجام می‌دهم. دقت clf4 به صورت زیر است :

```
              precision    recall  f1-score   support

           0       0.77      0.62      0.69        69
           1       0.85      0.92      0.88       158

    accuracy                           0.83       227
   macro avg       0.81      0.77      0.78       227
weighted avg       0.82      0.83      0.82       227

[[ 43  26]
 [ 13 145]]
```

در تمام معیارهای گزارش شده عملکرد clf4 و clf1 مشابه همدیگر است. در بعضی کمی بهتر و در بعضی دیگر مقداری افت داشته. هدفم از استفاده از این مدل این بود که تاثیر فشرده کردن داده ها را روی مدل اول ببینم و چون معیارها بسیار به هم نزدیک اند حدس می‌زنم که فشرده سازی تااثیری روی نتیجه دسته بندی این مدل ندارد اما گزارش و بررسی نهایی این مساله را به نتایج به دست آمده از میانگین k fold cross validation در بخش بعدی واگذار می‌کنم.

مدل سوم این بخش clf5 یک شبکه است که با اگوریتم ELM روی داده های فشرده شده ی مجموعه ی x_train_encode آموزش داده می‌شود. شرح ساخت مدل clf5 همان است که برای clf2 آمد. نتایج آموزش به صورت زیر است :

```
              precision    recall  f1-score   support

           0       0.77      0.48      0.59       123
           1       0.86      0.96      0.90       406

    accuracy                           0.84       529
   macro avg       0.81      0.72      0.75       529
weighted avg       0.84      0.84      0.83       529

[[ 59  64]
 [ 18 388]]
```

و نتیجه عملکرد مدل روی تست :

```
              precision    recall  f1-score   support

           0       0.60      0.41      0.48        69
           1       0.77      0.88      0.82       158

    accuracy                           0.74       227
   macro avg       0.68      0.64      0.65       227
weighted avg       0.72      0.74      0.72       227

[[ 28  41]
 [ 19 139]]
```

در تمام بخش ها و تمام معیارها افت کیفیت مدل مشهود است. اگر clf2 را با clf5 مقایسه کنیم، فشرده سازی داده ها برای این مساله روی مدل ELM نه تنها بهبود حاصل نمی‌کند که در تمام موارد ارزیابی باعث می‌شود مدل از قبل ناکارامدتر باشد.

5 Fold Cross Validation

تا اینجا 9 مدل ساختم که ۳ تای آن ها از داده های فشرده ی حاصل autoencoder استفاده می‌کرد. در بین این ۹ مدل ۳ تا clf6 و clf5 و clf2 که از بقیه ضعیف تر بودند و clf8 هم از بقیه نتیجه بهتری نشان داد ولی انتخاب بین بقیه و اینکه کدام مدل بهتری هست از روی این نتایج کار راحتی نیست چون مقادیر به دست آمده نزدیک هم بودند. در این قسمت تمام مدل ها را دوباره این بار با تمام داده های آموزش به صورت یکجا و با روش cross validation و cv =5 ساختم و معیارهای دقت را روی هرکدام میانگین گرفتم. برای ۳ مدلی که با داده های انکود شده کار میکرد با encoder بخش قبلی فشرده سازی هم صورت گرفته. استانداردسازی روی تمام داده ها انجام شده. یعنی در قبل از اینکه مجموعه ی validation جدا شود استاندارد سازی انجام شده. این کار دقت را ممکن است متاثر کند چون در تمام پیش پردازش های آموزش و تست باید قاعدتا جدا از هم انجام شوند ولی اینجا از این تاثیر صرف نظر کردم. با accuracy بهترین مدلی که به دست آمد clf7 بود که از روش random forest ساخته شد. البته clf9 و clf8 که هم هملکرد نزدیک به همین داشتند. نتایج در جدول قسمت بعدی آمده. برای این مدل مقادیر confusion matrix :

```
[[102  90]
 [ 35 529]]
```

مقایسه نتایج با مقاله

A comparative analysis of speech signal processing algorithms for Parkinson’s disease classification and the use of the tunable Q-factor wavelet transform

https://doi.org/10.1016/j.asoc.2018.10.022

در این مقاله هدف پیش بینی و تشخیص بیماری پارکینسون از روی بررسی سیگنال های صوتی است که روی داده های در این تمرین آن کار شد. در این کار هدف انتخاب یک سری ویژگی از بین داده ها به عنوان baseline feature و سپس انتخاب یک سری ویژگی دیگر با روش های feature selection است. اسن مجموعه های انتخاب شده در مدل های مختلف دسته بندی آزموده می‌شوند. در بهترین مدلی که در این تحقیق به دست آمده یک مجموعه ی ۵۰ تایی از ویژگی ها انتخاب شده و در ردیف آخر جدول زیر آوردم. در این مقاله تنها معیارهای f1 و accuracy و MCC گزارش و بررسی شده. نتایج این مدل از تمام مدل هایی که من در این گزارش به دست آوردم بهتر است. تصور می‌کنم علت اصلی این باشد که از بین این ویژگی ها تعدادی هستند که تفاوت معناداری در تشخیص بیماری پارکینسون ندارند. و هدف این مقاله هم شناسایی ویژگی هایی از نوارهای صوتی پردازش شده ی نمونه ها است که به بیماری ارتباط پیدا می‌کند. و اگر فرض درست باشد که تعدادی از این ستون ها، اصلا تاثیری روی نتیجه کلاس بندی ندارند، پس وجودشان در مدل ها باعث کاهش عملکرد مدل های شبکه عصبی من می‌شود. دلیل دوم می‌تواند پیش پردازش ها باشد. من به جز استاندارد سازی کار دیگری برای مقیاس داده ها یا تغیر آنها انجام ندادم. انجام این نوع تغیرات یا استفاده از استراتژی های کاهش بعد ممکن است نتایج بهتری برای مدل های شبکه ای من بدهد. اما برای xgboost و جنگل

تصادفی و svm عملکرد تقریبا همان معادل مدل انتخابی مقاله ست با این تفاوت که مقاله با استفاده از یک استراتژی انتخاب فیچر مناسب، با ۵۰ فیچر به این عملکرد رسیده و من با تمام ستون های داده ها و یک مدل حجیم به این جواب رسیدم.

Method	Accuracy	Precision	Recall	f-measure
MLP	0.75	0.78	0.75	0.76
ELM	0.78	0.77	0.78	0.77
AutoEncoder with Perceptron classifier	0.77	0.78	0.77	0.77
AutoEncoder with MLP classifier	0.80	0.80	0.80	0.80
AutoEncoder with ELM classifier	0.80	0.78	0.80	0.79
Decision tree	0.78	0.77	0.78	0.77
Random Forest	**0.85**	**0.84**	**0.85**	**0.83**
XGBoost	0.83	0.83	0.83	0.82
SVM	0.82	0.81	0.82	0.81
	0.86	Not reported	Not reported	0.84