تمرین سوم NLP

سوال اول

a) برای محاسبه prior probability از فرمول رو به رو استفاده میکنیم:

$$P(\text{کلاس}) = \frac{\text{تعداد جمله های آن کلاس}}{\text{کل جمله های بخش } train}$$

تعداد جمله های مثبت = 3

تعداد جمله های منفی = 2

تعداد کل جمله ها = 5

$$P(\text{مثبت}) = \frac{3}{5}$$

$$P(\text{منفی}) = \frac{2}{5}$$

مفهوم prior probability به توزیع داده ها به صورت پیش فرض اشاره دارد که تاثیر زیادی بر طبقه بندی یک جمله جدید دارند به صورتی که اگر جمله جدید شامل اطلاعات خاصی نباشد مدل تمایل دارد که جمله جدید را به کلاسی نسبت دهد که احتمال بیشتری دارد.

b) برای ایجاد vocabulary باید کلمه های یونیک را از متن پیدا کنیم.

unique words: I , love , this , movie , is , fantastic , hate , boring , enjoy

حال تعداد تکرار این کلمات را برای هر کلاس محاسبه می کنیم.

برای کلاس مثبت: (جمله های 1 و 2 و 5)

I	2
love	1
this	3
movie	3
is	1
fantastic	1
enjoy	1

برای کلاس منفی: (جمله های 3 و 4)

I	1
hate	1
this	2
movie	2
is	1
boring	1

برای محاسبه likelihood با Laplace Smoothing باید از فرمول زیر استفاده کنیم:

$$P(\text{کلمه} \mid \text{کلاس}) = \frac{\text{تعداد کلمه های درون کلاس} + 1}{\text{تعداد کل کلمه های کلاس} + Vocabulary\ Size}$$

تعداد کل کلمات مثبت: 12

تعداد کل کلمات منفی: 8

Vocabulary Size: 9

$$P(\text{کلمه} \mid \text{کلاس مثبت}) = \frac{\text{تعداد کلمه های درون کلاس} + 1}{9 + 12}$$

I	2	$\frac{2+1}{21} = 0.143$
love	1	$\frac{1+1}{21} = 0.095$
this	3	$\frac{3+1}{21} = 0.19$
movie	3	$\frac{3+1}{21} = 0.19$
is	1	$\frac{1+1}{21} = 0.095$
fantastic	1	$\frac{1+1}{21} = 0.095$
enjoy	1	$\frac{1+1}{21} = 0.095$
hate	0	$\frac{0+1}{21} = 0.048$
boring	0	$\frac{0+1}{21} = 0.048$

$$P(\text{کلاس منفی} \mid \text{کلمه}) = \frac{\text{تعداد کلمه های درون کلاس} + 1}{9 + 8}$$

I	1	$\frac{1+1}{17} = 0.118$
love	0	$\frac{0+1}{17} = 0.059$
this	2	$\frac{2+1}{17} = 0.176$
movie	2	$\frac{2+1}{17} = 0.176$
is	2	$\frac{2+1}{17} = 0.176$
fantastic	0	$\frac{0+1}{17} = 0.059$
hate	1	$\frac{1+1}{17} = 0.118$
boring	1	$\frac{1+1}{17} = 0.118$
enjoy	0	$\frac{0+1}{17} = 0.059$

c) با این فرض وابستگی های کلمات به یکدیگر از بین می رود و مدل ممکن است به اشتباه طبقه بندی کند.

برای مثال جمله This movie is fantastic but boring and I hate it ممکن است به دلیل وجود کلمه fantastic جزو جملات مثبت طبقه بندی شود در صورتی که مفهوم کلی منفی است.

d) برای محاسبه posterior probability از فرمول زیر استفاده می کنیم:

$$P(\text{کلاس} \mid \text{جمله}) = P(\text{جمله} \mid \text{کلاس}) * P(\text{کلاس})$$

P(کلاس | جمله) :

از ضرب likelihood هر کلمه درون جمله حساب می کنیم.

P(کلاس):

بالاتر حساب کردیم که برای کلاس مثبت شد 0.6 و برای کلاس منفی 0.4.

جمله تست اول :

I love this amazing movie

P(کلاس | جمله) = P(کلاس) *P(I | کلاس) * P(love | کلاس) * P(this | کلاس) * P(amazing | کلاس) * P(movie | کلاس)

از آنجایی که amazing در بخش آموزش نبود باید P(amazing) را نسبت به دو کلاس حساب کنیم:

P(amazing | مثبت) = $\frac{0+1}{21}$ = 0.048

P(amazing | منفی) = $\frac{0+1}{17}$ = 0.59

برای کلاس مثبت:

P(جمله | مثبت) = 0.6 * 0.143 * 0.095 * 0.190 * 0.048 * 0.190 = 0.000146

برای کلاس منفی:

P(جمله | منفی) = 0.4 * 0.118 * 0.059 * 0.176 * 0.059 * 0.176 = 0.000020

اینجا چون احتمال کلاس مثبت بیشتر شد مدل آنرا به کلاس مثبت طبقه می کند.

جمله تست دوم :

This movie is amazing but boring and I hate it

P(کلاس | جمله) = P(کلاس) * P(This | کلاس) * P(movie | کلاس) * P(is | کلاس) * P(amazing | کلاس) * P(but | کلاس) * P(boring | کلاس) * P(and | کلاس) * P(I | کلاس) * P(hate | کلاس) * P(it | کلاس)

کلمات and ، but و it در بخش train نیست و برای هر یک نسبت به دو کلاس داریم:

P(and | مثبت) = $\frac{0+1}{21}$ = 0.048

P(and | منفی) = $\frac{0+1}{17}$ = 0.59

P(but | مثبت) = $\frac{0+1}{21}$ = 0.048

P(but | منفی) = $\frac{0+1}{17}$ = 0.59

P(it | مثبت) = $\frac{0+1}{21}$ = 0.048

$P(it \mid \text{منفی}) = \frac{0+1}{17} = 0.59$

برای amaing هم بالا حساب کردیم.

برای کلاس مثبت:

$P(\text{مثبت} \mid \text{جمله}) = 0.6 * 0.190 * 0.190 * 0.095 * 0.048 * 0.048 * 0.048 * 0.048 * 0.143 * 0.048 * 0.048 = 0.0000000000035$

برای کلاس منفی:

$P(\text{منفی} \mid \text{جمله}) = 0.4 * 0.176 * 0.176 * 0.176 * 0.059 * 0.059 * 0.118 * 0.059 * 0.118 * 0.118 * 0.059 = 0.0000000000434$

اما اینجا چون احتمال کلاس منفی بیشتر شد مدل آنرا به کلاس منفی نیز اختصاص می دهد.

e) تکنیک Laplace Smoothing برای جلوگیری از صفر شدن احتمال کلمات جدید است و با اینکار مدل به کلماتی که تاکنون ندیده احتمالی نزدیک به صفر اختصاص میدهد تا روند محاسبات مختل نشود.
اگر از این روش استفاده نکنیم در مدل naive bayes که از ضرب احتمال کلمات استفاده می کنیم اگر کلمه جدیدی در جمله بیاید احتمال کل جمله صفر می شود.
برای مثال در جمله تست اول کلمه amazing در بخش training وجود ندارد و اگر از این قاعده استفاده نکنیم در صفر ضرب می شود و کلا از بین می رود.

f) اگر تعداد داده های آموزشی کلاس منفی بیشتر بود مقدار prior probability آن نیز افزایش می یافت و از آنجایی که برای محاسبه posterior probability احتمال کلمات در آخر در P(class) آن ضرب میشود، اگر اختلاف 2 کلاس بالا باشد مدل به سمت کلاس منفی میل میکند مگر آن که احتمال کلمات به شدت نفع کلاس مثبت باشد.

سوال دوم

a) برای محاسبه فاصله Levenshtein برای هر کلمه از 4 عمل زیر میتوانیم استفاده کنیم:

1- Insertions (افزودن) : هزینه 1

2- Deletions (حذف) : هزینه 1

3- Substitutions (تعویض) : هزینه 1

4- transposition (جابجایی) : هزینه 1

1- کلمه distance

مقدار اولیه: distnace

قدم اول: جابجایی a و n ← distance

1 : Edit Distance

2- Resistance

مقدار اولیه: distnace

قدم اول: تعویض d به r ← ristnace

قدم دوم: افزودن e بعد r ← reistnace

قدم سوم: افزودن s بعد e ← resistnace

قدم چهارم: جابجایی a و n ← resistance

4 : Edit Distance

3- Insistence

مقدار اولیه: distnace

قدم اول: تعویض d به i ← iistnace

قدم دوم: افزودن n بعد i ← inistnace

قدم سوم: افزودن s بعد n ← insistnace

قدم چهارم: جابجایی a و n ← insistance

قدم پنجم: تعویض a به e ← insistence

5 : Edit Distance

4- instance

مقدار اولیه: distnace

قدم اول: تعویض d به n ← nistnace

قدم دوم: جابجایی i و n ← instnace

قدم سوم: جابجایی n و a ← instance

3 : Edit Distance

5- substance

مقدار اولیه: distnace

قدم اول: افزودن s در اول ← sdistnace

قدم دوم: تعویض d به u ← suistnace

قدم سوم: تعویض i به b ← substnace

قدم چهارم: جابجایی n و a ← substance

4 : Edit Distance

6- assistance

مقدار اولیه: distnace

قدم اول: افزودن a در اول ← adistnace

قدم دوم: تعویض d به s ← asistnace

قدم سوم: افزودن s بعد s ← assistnace

قدم چهارم: جابجایی n و a ← assistance

4 : Edit Distance

7- persistence

مقدار اولیه: distnace

قدم اول: افزودن p در اول ← pdistnace

قدم دوم: تعویض d به e ← peistnace

قدم سوم: افزودن r بعد e ← peristnace

قدم چهارم: افزودن s بعد r ← persistnace

قدم پنجم: جابجایی n و a ← persistance

قدم ششم: تعویض a به e ← persistence

6 : Edit Distance

b) اگر دو یا چند کلمه Edit Distance یکسانی داشته باشند روش های متفاوتی برای انتخاب کلمه مناسب وجود دارد.

یکی از این روش ها Semantic Similarity است که به این معنی است که اگر کلمه اشتباهی در جمله باشد بررسی می شود که هر یک از کلمات پیشنهادی چقد با جمله در ارتباط است و کلمه ای که بیشترین تطابق را داشته باشد انتخاب میشود.

روش دیگر Word Frequency هست که به کلمه ای که بیشتر در یک دیتاست بزرگ آمده باشد انتخاب میشود.

Part-of-Speech Alignment: یکی دیگر از این روش ها است که بر اساس ساختار جمله به کلماتی که مناسب آن بخش جمله هستند توجه میکند. برای مثال اگر قرار است در آن جای جمله فعل بیاید از میان افعال کلمه مناسب را انتخاب میکند.

روش دیگری که وجود دارد انتخاب کلمه مناسب بر اساس شباهت طولی رشته است و کلمه ای که از نظر تعداد کاراکتر به کلمه مربوطه نزدیک تر است انتخاب میشود. این روش برای اشتباهات تایپی مناسب است.

c) هزینه های جدید برای عملیات Levenshtein

1. Substitutions (تعویض) = 2
2. transposition (جابجایی) = 1
3. Insertions (افزودن) برای حروف پر تکرار (a,e,s) برابر 0.5 و برای سایر حروف 1
4. Deletions (حذف) برای حروف پر تکرار (a,e,s) برابر 0.5 و برای سایر حروف 1

1- کلمه distance

مقدار اولیه: distnace

قدم اول: جابجایی a و n ← distance (1)

1 : Edit Distance

2- Resistance

مقدار اولیه: distnace

قدم اول: تعویض d به r ← ristnace (2)

قدم دوم: افزودن e بعد r ← reistnace (0.5)

قدم سوم: افزودن s بعد e ← resistance (0.5)

قدم چهارم: جابجایی a و n ← resistance (1)

4 : Edit Distance

3- Insistence

مقدار اولیه: distnace

قدم اول: تعویض d به i ← iistnace (2)

قدم دوم: افزودن n بعد i ← inistnace (1)

قدم سوم: افزودن s بعد n ← insistnace (0.5)

قدم چهارم: جابجایی a و n ← insistance (1)

قدم پنجم: افزودن e بعد a ← insistaence (0.5)

قدم ششم: حذف a ← insistence (0.5)

5. 5 : Edit Distance

4- instance

مقدار اولیه: distnace

قدم اول: تعویض d به n ← nistnace (2)

قدم دوم: جابجایی i و n ← instnace (1)

قدم سوم: جابجایی n و a ← instance (1)

4 : Edit Distance

5- substance

مقدار اولیه: distnace

قدم اول: افزودن s در اول ← sdistnace (0.5)

قدم دوم: تعویض d به u ← suistnace (2)

قدم سوم: تعویض i به b ← substnace (2)

قدم چهارم: جابجایی n و a ← substance (1)

Edit Distance : 5.5

6- assistance

مقدار اولیه: distnace

قدم اول: افزودن a در اول ← adistnace (0.5)

قدم دوم: افزودن s بعد d ← adsistnace (0.5)

قدم سوم: حذف d ← asistnace (1)

قدم چهارم: افزودن s بعد s ← assistnace (0.5)

قدم پنجم: جابجایی n و a ← assistance (1)

Edit Distance : 3.5

7- persistence

مقدار اولیه: distnace

قدم اول: افزودن p در اول ← pdistnace (1)

قدم دوم: افزودن e بعد d ← pdeistnace (0.5)

قدم سوم: حذف d ← peistnace (1)

قدم چهارم: افزودن r بعد e ← peristnace (1)

قدم پنجم: افزودن s بعد r ← persistnace (0.5)

قدم ششم: جابجایی n و a ← persistance (1)

قدم هفتم: جابجایی a و e ← persistence (1)

Edit Distance : 6

سوال سوم

a) برای محاسبه prior probability کلمات از فرمول زیر استفاده میکنیم:

$P(word_{corrcet})$ = $\frac{\text{تعداد تکرار کلمه}}{\text{تعداد کل کلمه ها}}$

P(there) = $\frac{1000}{4500}$ = 0.22

P(their) = $\frac{400}{4500}$ = 0.088

P(they're) = $\frac{100}{4500}$ = 0.022

P(the) = $\frac{3000}{4500}$ = 0.66

b) برای تعریف $P(word_{misspelled} \mid word_{correct})$ به صورت زیر عمل میکنیم:

مفروضات اولیه :

هزینه تعویض، حذف و افزودن برابر 1

هزینه جابجایی حروف مجاور برابر 0.5

تابع محاسبه احتمال:

$$P(word_{misspelled} \mid word_{correct}) = \frac{1}{edit\ distance+1}$$

محاسبه edit distance کلمه "their " تا هر کلمه:

1- their

مقدار اولیه: their

Edit Distance : 0

there -2

مقدار اولیه: their

قدم اول: تعویض i به e ← theer (1)

قدم دوم: جابجایی e و r ← there (0.5)

Edit Distance : 1.5

they’re -3

مقدار اولیه: their

قدم اول: تعویض i به y ← theyr (1)

قدم دوم: افزودن ‘ بعد y ← they’r (1)

قدم سوم: افزودن e بعد r ← they’re (1)

Edit Distance : 3

the -4

مقدار اولیه: their

قدم اول: حذف r ← thei (1)

قدم دوم: حذف i ← the (1)

Edit Distance : 2

محاسبه احتمال برای هر کلمه

$P(\text{their} \mid \text{there}) = \frac{1}{1.5 + 1} = 0.4$

$P(\text{their} \mid \text{their}) = \frac{1}{0 + 1} = 1$

P(their | they’re) = $\frac{1}{3+1}$ = 0.25

P(their | the) = $\frac{1}{2+1}$ = 0.33

c) برای محاسبه محتمل ترین کلمه صحیح با استفاده از مدل Noisy Channel از فرمول زیر استفاده میکنیم:

$$P(word_{correct} \mid thier) = \frac{P(thier \mid word_{correct}) * P(word_{correct})}{\Sigma_{word_{correct}} P(thier \mid word_{correct}) * P(word_{correct})}$$

در سوال های قبل ما $P(word_{correct})$ را برای هر کلمه حساب کردیم که به شکل زیر است:

P(there) = $\frac{1000}{4500}$ = 0.22

P(their) = $\frac{400}{4500}$ = 0.088

P(they’re) = $\frac{100}{4500}$ = 0.022

P(the) = $\frac{3000}{4500}$ = 0.66

حال برای تعریف $P(thier \mid word_{correct})$ همانند بخش b عمل میکنیم:

مفروضات اولیه :

هزینه تعویض، حذف و افزودن برابر 1

هزینه جابجایی حروف مجاور برابر 0.5

تابع محاسبه احتمال:

$$P(word_{misspelled} \mid word_{correct}) = \frac{1}{edit\ distance+1}$$

محاسبه edit distance کلمه thier تا هر کلمه:

1- their

مقدار اولیه: thier

قدم اول: جابجایی i و e ← their (0.5)

Edit Distance : 0.5

there -2

مقدار اولیه: thier

قدم اول: تعویض i به e ← theer (1)

قدم دوم: جابجایی e و r ← there (0.5)

Edit Distance : 1.5

they’re -3

مقدار اولیه: thier

قدم اول: تعویض i به y ← thyer (1)

قدم دوم: جابجایی e و y ← theyr (0.5)

قدم سوم: افزودن ‘ بعد y ← they’r (1)

قدم چهارم: افزودن e بعد r ← they’re (1)

Edit Distance : 3.5

the -4

مقدار اولیه: thier

قدم اول: حذف r ← thie (1)

قدم دوم: حذف i ← the (1)

Edit Distance : 2

محاسبه احتمال برای هر کلمه

P(their | their) = $\frac{1}{0.5+1}$ = 0.66

P(their | there) = $\frac{1}{1.5+1}$ = 0.4

P(their | they’re) = $\frac{1}{3.5+1}$ = 0.22

P(their | the) = $\frac{1}{2+1}$ = 0.33

حال طبق فرمول زیر عمل میکنیم:

$P(word_{correct} \mid their) = \frac{P(their \mid word_{correct}) * P(word_{correct})}{\sum_{word_{correct}} P(their \mid word_{correct}) * P(word_{correct})}$

P(their | their) * P(their) = 0.66 * 0.088 = 0.05808

P(their | there) * P(there) = 0.4 * 0.22 = 0.088

P(their | they’re) * P(they’re) = 0.22 * 0.022 = 0.00484

P(their | the) * P(the) = 0.33 * 0.66 = 0.217

$\sum_{word_{correct}} P(their \mid word_{correct}) * P(word_{correct})$ = 0.05808 + 0.088 + 0.00484 + 0.217 = 0.36792(

P(their | their) = $\frac{0.05808}{0.36792}$ = 0.1578

P(there | their) = $\frac{0.088}{0.36792}$ = 0.2391

P(they’re | their) = $\frac{0.00484}{0.36792}$ = 0.0131

P(the | their) = $\frac{0.217}{0.36792}$ = 0.5898

در اینجا مدل ما با احتمال 59 درصد کلمه ای که غلط بود را به the نسبت داده است. زیرا مدل ما ترکیب edit distance و P(word) که همان تعداد تکرار کلمه در دیتاست است را در نظر گرفته و این موضوع باعث می شود که هر دو در احتمال تاثیر داشته باشند.

برای بهبود مدل میتوانیم از یک تابع احتمال دیگر برای edit distance استفاده کنیم تا اهمیت بیشتری به این فاصله بده.

d) اگر کلمه ای که اشتباه وارد شده بود there می بود مدل ما آن را با احتمال 23 درصد پیش بینی می کرد.(که اشتباه کرده)

برای رفع این ابهام راه های زیادی برای بهتر کردن مدل پیشنهاد می شود:

1- برای پیش بینی کلمه ای که به اشتباه تایپ شده، نباید به خود کلمه بسنده کنیم و میتوانیم با استفاده از مدل های n-gram به جایگاه کلمه توجه کنیم
2- برای رفع این ابهام یک راه دیگر توجه به POS Tagging است که به ساختار کلمات در جمله توجه می کند
3- یکی دیگر از روش ها این است که برای edit distance های خاص که رایج ترند مثل اشتباه تایپی حرف m و n توجه کنیم و یا در این مثال جابجایی حرف i و e در their بسیار محتمل است.

سوال 4

پس از دریافت دیتاست و ایمپورت کردن کتابخانه های مربوطه که در فایل انجام شده بود به سراغ ادامه کد می رویم.

```
from datasets import concatenate_datasets
train_data = emotion_data['train'].filter(lambda x: x['label'] in [0, 1])
val_data = emotion_data['validation'].filter(lambda x: x['label'] in [0, 1])
merged_data = concatenate_datasets([train_data, val_data])
```

در ابتدا داده های دو کلاس 0 و 1 (طبق صورت سوال) را جدا می کنیم و در ادامه با استفاده از کتابخانه concatenate_datasets داده های Train و Validation را ترکیب می کنیم.

در ادامه عملیات مربوط به data cleaning را انجام میدهیم.

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
punctuations = string.punctuation

def preprocess_text(text):
    text = text.lower()
    text = re.sub(r"http\S+|www\S+|https\S+", "", text, flags=re.MULTILINE)
    text = re.sub(r"<.*?>", "", text)
    text = re.sub(r"[^a-zA-Z0-9\s]", "", text)
    text = " ".join(word for word in text.split() if word not in stop_words)
    text = " ".join(lemmatizer.lemmatize(word) for word in text.split())

    return text
```

در کد بالا یک تابع برای تمیز کردن داده هایمان ایجاد کردیم که ابتدا متن را به حروف کوچک تبدیل میکند. در ادامه با استفاده از Regex لینک ها و تگ های html را که درون داده ها هستند را حذف میکنیم. پس از آن هر کاراکتری به غیر از حروف کوچک و بزرگ البفای انگلیسی ، اعداد و فاصله در متن وجود دارد را حذف میکنیم.

در ادامه stopword هایی که در زبان انگلیسی هستند را که بالاتر لود کردیم از متن حذف میکنیم. سپس با استفاده از lemmatizer کلمات را به ریشه اصلی آن ها تبدیل می کنیم.

```
texts = []
for text in merged_data['text']:
  texts.append(preprocess_text(text))
labels = merged_data['label']
```

در ادامه برای هر یک سمپل دیتاست یکبار تابع را صدا میزنیم تا داده هایمان clean شوند و اطلاعات غیر ضروری ای نداشته باشند.

برچسب های داده ها را نیز درون labels می ریزیم.

```
X_train, X_test, y_train, y_test = tts(texts, labels, test_size=0.2, random_state=42)
```

داده های train و test را با نسبت 0.2 و random_state 42 تقسیم میکنیم تا هر سری با همان الگوریتم داده ها تقسیم بشوند.

```
vectorizer = TfidfVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
```

در اینجا با استفاده از این کلاس داده های متنی را به ماتریس از ویژگی های عددی تبدیل میکنیم تا بتوانیم محاسبات روی آنها انجام بدهیم. در fit_transform ویژگی های داده های آموزشی را محاسبه می کنیم و بردارهای TF-IDF را تولید می کنیم در ادامه از transform برای تبدیل متون تست به بردارهای TF-IDF با استفاده از ویژگ های محاسبه شده در مرحله قبل استفاده می کنیم.

```
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train_vectorized, y_train)

y_train_pred = nb_classifier.predict(X_train_vectorized)
print("Train Accuracy:", accuracy_score(y_train,y_train_pred))
```

نوبت به آموزش مدل می رسد که یک مدل بیز ساده چند جمله ای ایجاد میکنیم و داده هایمان را روی مدل فیت می کنیم.

در ادامه پس از آموزش مدل پیش بینی را برای داده های آموزشی انجام می دهیم که به صورت زیر است:

```
Train Accuracy: 0.9864819944598338
```

```
y_pred = nb_classifier.predict(X_test_vectorized)
print("Test Accuracy:", accuracy_score(y_test, y_pred))
```

برای داده های تست نیز میزان دقت به شرح زیر است:

```
Test Accuracy: 0.9539211342490032
```

```
classes = sorted(set(y_test))
conf_matrix = np.zeros((len(classes), len(classes)), dtype=int)

for true, pred in zip(y_test, y_pred):
    conf_matrix[true][pred] += 1
print("Manuel Confusion Matrix:")
print(conf_matrix)
print("SKlearn Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

به سراغ معیار های ارزیابی می رویم که اولی Confustion matrix است که به دو صورت دستی و با کتابخانه sklearn پیاده سازی شده است. در مدل دستی ابتدا یک ماتریس مربعی به ابعداد تعداد برچسب ها میزنیم. در ادامه منطق مقدار conf_matrix[true][pred] نشان دهنده تعداد نمونه هایی است که کلاس واقعی آن ها true بوده ولی به کلاس pred پیش بینی شده است.

که نتیجه به شکل رو به رو است که هر دو یکسان هستند.

```
Manuel Confusion Matrix:
[[ 964   69]
 [  35 1189]]
SKlearn Confusion Matrix:
[[ 964   69]
 [  35 1189]]
```

```
TP_0 = conf_matrix[0, 0]
FP_0 = conf_matrix[1, 0]
precision_0 = TP_0 / (TP_0 + FP_0) if (TP_0 + FP_0) > 0 else 0
print(f"Manuel Precision (Class sadness): {precision_0}")
print(f"SKlearn Precision (Class sadness): {precision_score(y_test, y_pred,labels=[0], average='macro')}"

TP_1 = conf_matrix[1, 1]
FP_1 = conf_matrix[0, 1]
precision_1 = TP_1 / (TP_1 + FP_1) if (TP_1 + FP_1) > 0 else 0
print(f"Manuel Precision (Class joy): {precision_1}")
print(f"SKlearn Precision (Class joy): {precision_score(y_test, y_pred,labels=[1], average='macro')}")
```

پس از Confusion Matrix نوبت به Precision می رسد که برای هر کلاس به صورت جداگانه تعریف کرده ایم. در اینجا نیز مقدار Precision طبق فرمول آن به صورت دستی محاسبه میشود:

```
Manuel Precision (Class sadness): 0.964964964964965
SKlearn Precision (Class sadness): 0.964964964964965
Manuel Precision (Class joy): 0.9451510333863276
SKlearn Precision (Class joy): 0.9451510333863276
```

در اینجا نیز از هر دو مدل استفاده کردیم که باهم برابر شدند.

```python
FN_0 = conf_matrix[0, 1]
recall_0 = TP_0 / (TP_0 + FN_0) if (TP_0 + FN_0) > 0 else 0
print(f"Manuel Recall (Class sadness): {recall_0}")
print(f"SKlearn Recall (Class sadness): {recall_score(y_test, y_pred,labels=[0], average='macro')}")

FN_1 = conf_matrix[1, 0]
recall_1 = TP_1 / (TP_1 + FN_1) if (TP_1 + FN_1) > 0 else 0
print(f"Manuel Recall (Class joy): {recall_1}")
print(f"SKlearn Recall (Class joy): {recall_score(y_test, y_pred,labels=[1], average='macro')}")
```

برای Recall نیز همانند Precision عمل کردیم و آن را برای هر دو کلاس به صورت جداگانه تعریف کردیم که به شکل زیر است:

```
Manuel Recall (Class sadness): 0.9332042594385286
SKlearn Recall (Class sadness): 0.9332042594385286
Manuel Recall (Class joy): 0.9714052287581699
SKlearn Recall (Class joy): 0.9714052287581699
```

در اینجا هم با هم برابر شدند.

```python
f1_0 = 2 * (precision_0 * recall_0) / (precision_0 + recall_0) if (precision_0 + recall_0) > 0 else 0
print(f"Manuel F1-Measure (Class sadness): {f1_0}")
print(f"SKlearn F1-Measure (Class sadness): {f1_score(y_test, y_pred,labels=[0], average='macro')}")

f1_1 = 2 * (precision_1 * recall_1) / (precision_1 + recall_1) if (precision_1 + recall_1) > 0 else 0
print(f"Manuel F1-Measure (Class joy): {f1_1}")
print(f"SKlearn F1-Measure (Class joy): {f1_score(y_test, y_pred,labels=[1], average='macro')}")
```

و در آخر نوبت به f1_score می رسد که از 2 برابر ضرب Precision و Recall تقسیم بر مجموعشان بدست می آید که برای هر کلاس محاسبه شده است:

```
Manuel F1-Measure (Class sadness): 0.9488188976377953
SKlearn F1-Measure (Class sadness): 0.9488188976377953
Manuel F1-Measure (Class joy): 0.9580983078162771
SKlearn F1-Measure (Class joy): 0.9580983078162773
```

محمد حقیقت - 403722042