گزارش پروژه پردازش زبان طبیعی

مقدمه

این پروژه به بررسی و پردازش مجموعه داده‌های متنی پرداخته و فرآیندهای پیش‌پردازش، تحلیل داده‌ها و آماده‌سازی برای مدل یادگیری عمیق را شامل می‌شود .هدف اصلی این پروژه آموزش یک مدل برای تحلیل احساسات متن است.

بارگذاری داده‌ها

در این بخش، داده‌ها تنها از درایو بارگذاری شده‌اند. ابتدا تعداد کل داده‌ها بررسی شده است تا مشخص شود چه حجمی از داده‌ها در اختیار داریم. علاوه بر این، یک زیرمجموعه از داده‌ها ایجاد شده است تا مدل‌های مختلف را روی آن آزمایش کنیم. این کار با هدف افزایش سرعت پردازش و کاهش زمان مورد نیاز برای آموزش مدل‌ها انجام شده است.

```
[6]  print(len(data_sampled))

     200664
```

تحلیل داده‌ها و پردازش اولیه

تحلیل داده‌ها و پردازش اولیه

در این بخش، چندین مرحله برای بررسی و پردازش اولیه داده‌ها انجام شده است:

۱. **نمایش نمونه‌هایی از داده‌ها** :ابتدا با چاپ چند سطر اولیه از داده‌ها، ساختار کلی آن‌ها، از جمله برچسب‌ها (labels) و نحوه سازماندهی اطلاعات، بررسی شد.

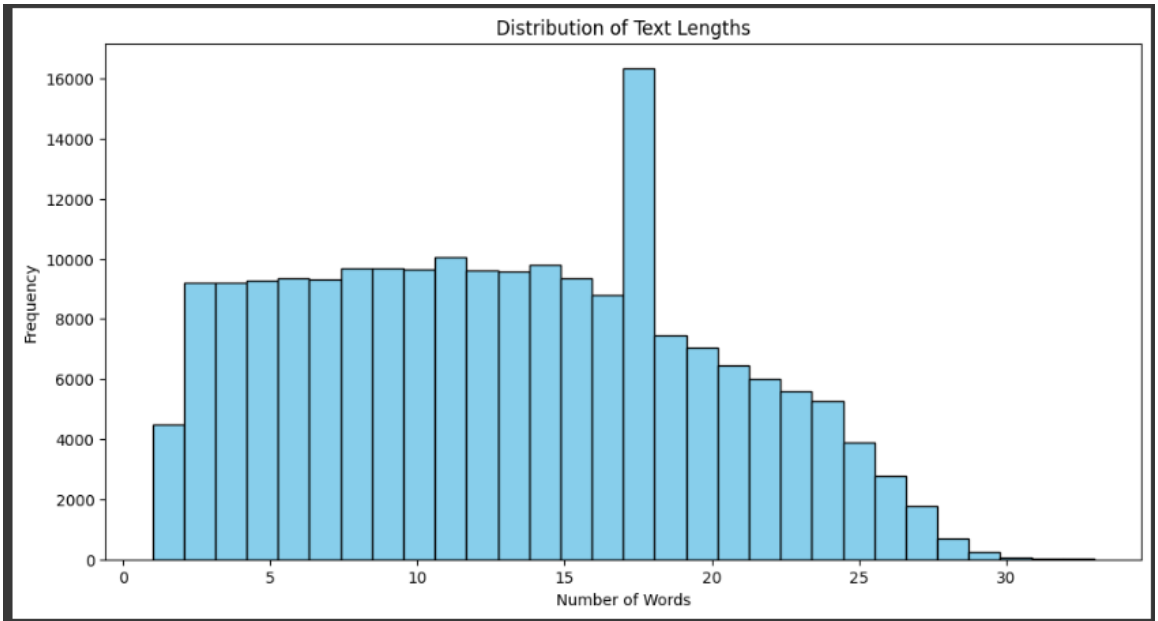
```
First few rows of the dataset:
   Unnamed: 0                                               text  praise  \
0           0  Is there some scripture you could quote me? I'...       1
1           1  Good. Now we just need people to dislike commi...       1
2           2                          This was driving me NUTS!       0
3           3                         Thank you for your advice!       0
4           4  Some do. Some don't. Blanket generalizations a...       0

   amusement  anger  disapproval  confusion  interest  sadness  fear  joy  \
0          0      0            0          0         1        0     0    0
1          0      0            0          0         0        0     0    0
2          1      0            0          0         0        0     0    0
3          0      0            0          0         0        0     0    0
4          0      1            1          0         0        0     0    0

   love
0     0
1     0
2     0
3     1
4     0
```

۲. **حذف ستون اضافی** :مشخص شد که یک ستون اضافی برای شناسه داده‌ها وجود دارد که نیازی به آن نبود، بنابراین حذف شد.

۳. **تحلیل طول متن‌ها** :توزیع طول متن‌ها بررسی شد که نتایج آن به صورت زیر بود :



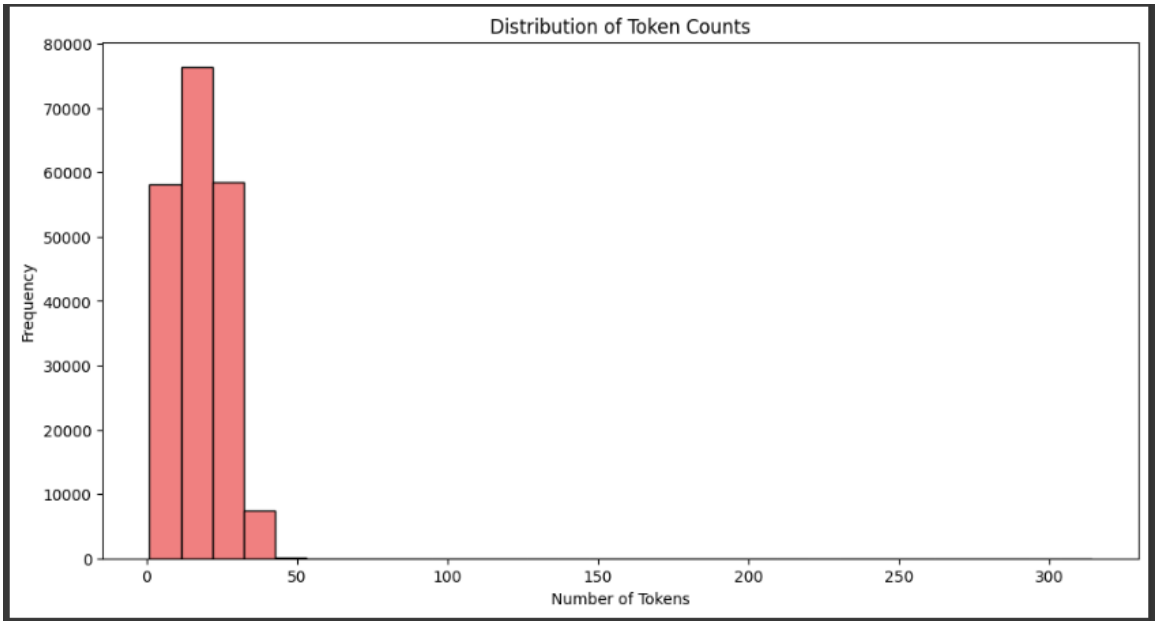

```
Text Length Statistics:
Mean Length: 13.00
Median Length: 13.00
Max Length: 33
Min Length: 1
```

میانگین (۱۳,۰۰) و میانه (۱۳,۰۰) برابر هستند این نشان می‌دهد که توزیع طول متن‌ها تقریباً متقارن است، زیرا میانگین و میانه معمولاً در یک توزیع نرمال نزدیک به هم قرار دارند.

حداکثر طول (۳۳) و حداقل طول دامنه طول متن‌ها نسبتاً گسترده است، اما مقدار حداکثر خیلی زیاد نیست. این نشان می‌دهد که بیشتر متون کوتاه هستند، اما برخی نمونه‌های بلندتر نیز وجود دارند.

عدم وجود مقادیر پرت خیلی بزرگ اگر حداکثر طول متن‌ها چند صد یا هزاران کاراکتر بود، نشان‌دهنده پراکندگی بسیار زیاد و وجود داده‌های پرت (outliers) بود، اما مقدار ۳۳ یک مقدار معقول است.

بررسی تعداد توکن‌ها در هر متن :




```
Token Count Statistics:
Mean Tokens: 17.40
Median Tokens: 17.00
Max Tokens: 314
Min Tokens: 1
```

میانگین (۱۷,۴۰) و میانه (۱۷,۰۰) نزدیک به هم هستند این مشابه بخش قبل نشان می‌دهد که توزیع تعداد توکن‌ها نیز تقریباً متقارن است.

حداکثر تعداد توکن‌ها (۳۱۴) این مقدار به نسبت میانگین (۱۷,۴۰) خیلی بزرگ‌تر است، که نشان می‌دهد در داده‌ها نمونه‌هایی وجود دارند که بسیار طولانی‌تر از اکثر متون دیگر هستند. احتمالاً این داده‌ها پرت هستند یا شامل اطلاعات خاصی می‌شوند که نیاز به پردازش بیشتر دارد.

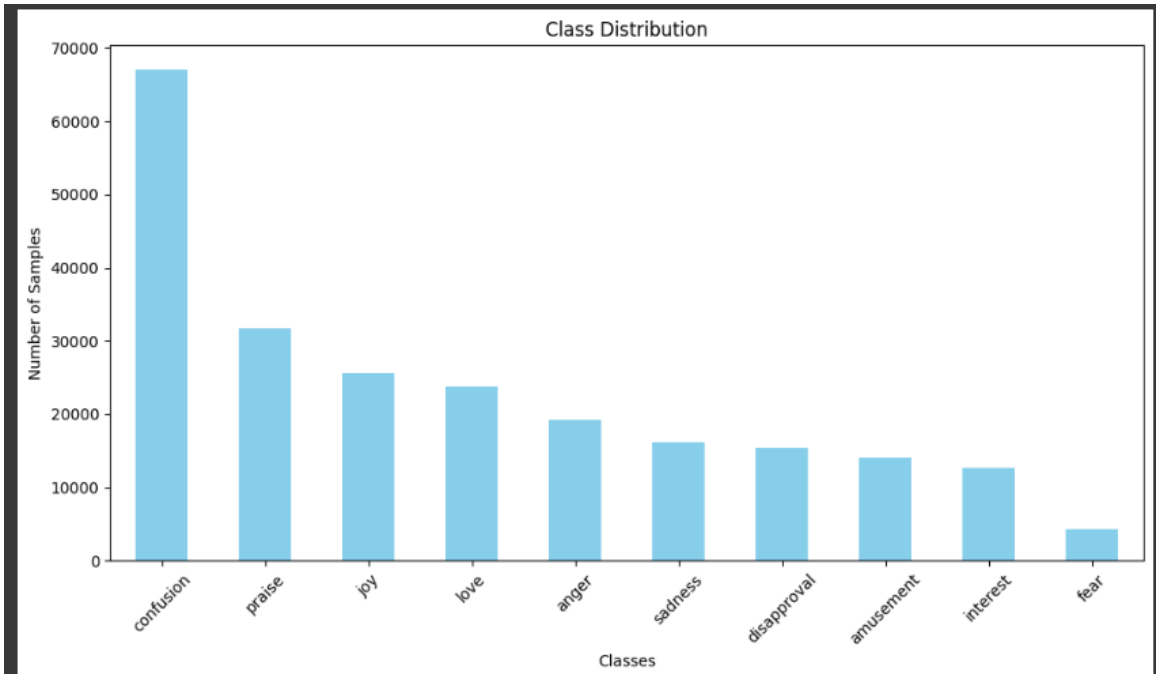
حداقل تعداد توکن‌ها برخی از متون تنها شامل یک کلمه یا علامت هستند، که ممکن است نیاز به فیلتر شدن داشته باشد.

۴. **انتخاب مقدار مناسب برای پدینگ :**مقدار **24**به عنوان بهترین مقدار برای پدینگ انتخاب شد.

```
Recommended Padding Length (95th Percentile): 24
```

۵. **بررسی مقادیر Null**: بررسی شد که آیا در داده‌های متنی مقدار Null وجود دارد یا نه، که مشخص شد هیچ مقدار Nullوجود ندارد.

۶. **بررسی توزیع کلاس‌ها :**مشاهده شد که کلاس‌های داده نامتوازن هستند، بنابراین در مراحل بعدی باید تدابیری برای متعادل‌سازی آن‌ها در نظر گرفته شود.



۷. **تعداد واژگان منحصربه‌فرد در داده‌ها** :تعداد کلمات مختلف موجود در مجموعه داده **31,399**بود که نشان‌دهنده گستردگی واژگان مورد استفاده در داده‌ها است.

آماده‌سازی داده‌ها برای مدل

در این بخش، داده‌ها برای استفاده در مدل یادگیری عمیق پردازش شده‌اند. فرآیند آماده‌سازی شامل چندین مرحله کلیدی بوده است:

تبدیل متون به توکن‌های عددی

برای پردازش متن‌ها از مدل Bert,DistilBERT استفاده شده است. در این مرحله، متن‌ها به مقادیر عددی تبدیل شده‌اند تا مدل بتواند آن‌ها را پردازش کند. مراحل پردازش شامل موارد زیر بوده است:

اعمال پدینگ (padding) تا طول ۲۴: تمام جملات کوتاه‌تر از این مقدار با توکن‌های خاص تکمیل شده‌اند.

برش جملات طولانی :(truncation) جملات بلندتر کوتاه شده‌اند.

تبدیل داده‌ها به فرمت :PyTorch داده‌ها به قالبی تبدیل شده‌اند که با PyTorch سازگار باشد.

بررسی روش‌های متعادل‌سازی داده‌ها

یکی از چالش‌های این پروژه نامتوازن بودن کلاس‌ها بود. در طی این مرحله، روش‌های مختلفی برای متعادل‌سازی داده‌ها آزمایش شد. با این حال، بسیاری از این روش‌ها عملکرد مناسبی نداشتند و مدل در مواجهه با این داده‌های نامتوازن دچار مشکل می‌شد. برخی از روش‌های تست‌شده عبارت بودند از:

Undersampling(کاهش تعداد نمونه‌های کلاس پرتکرار): این روش باعث از دست رفتن حجم زیادی از داده‌های آموزشی شد و تأثیر منفی بر عملکرد مدل داشت.

Oversampling(افزایش نمونه‌های کلاس کم‌تعداد): این روش با تکرار بیش از حد داده‌های کلاس‌های کم‌تعداد باعث بیش‌برازش (overfitting) مدل شد.

استفاده از وزن‌دهی کلاس‌ها: این روش نسبتا بهتر عمل کرد اما همچنان مدل در یادگیری توزیع صحیح کلاس‌ها مشکل داشت.

آماده‌سازی نهایی برای آموزش مدل

داده‌ها در مجموعه‌های آموزشی، اعتبارسنجی و آزمایشی تقسیم شدند.

تمامی متون به فرم برداری تبدیل شدند تا بتوانند توسط مدل پردازش شوند.

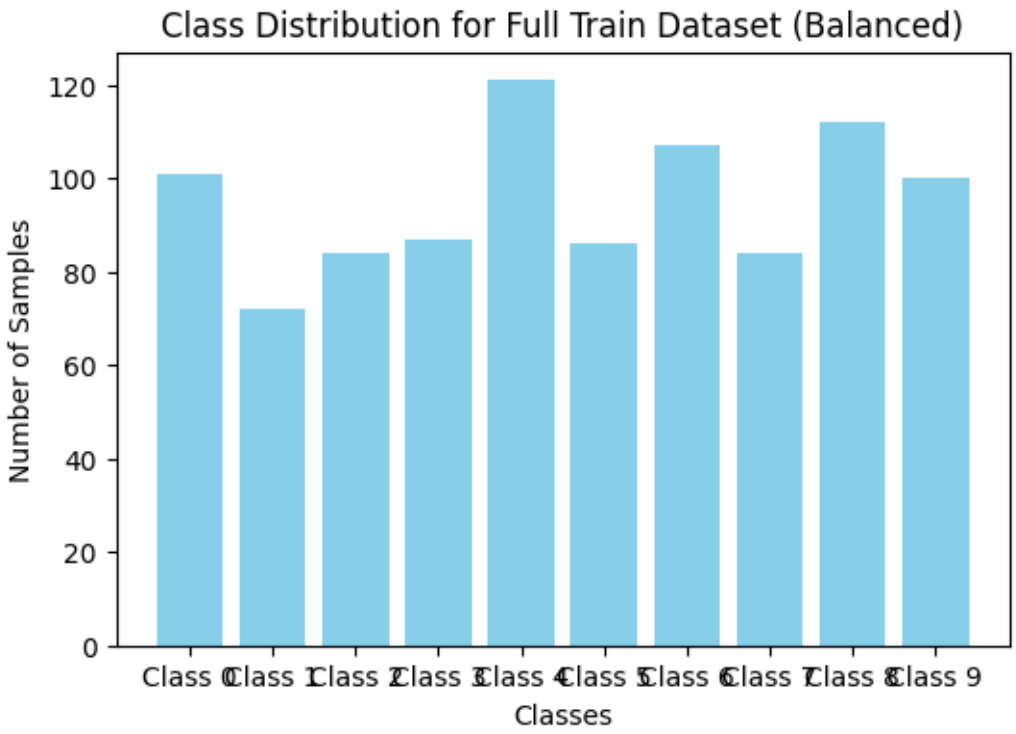
در نهایت، مدل برای یادگیری آماده شده است و باید روی داده‌های پردازش‌شده اجرا شود.

تمام آزمایش‌ها با استفاده از **۱٪ از داده‌های خام** بر روی تمامی مدل‌ها انجام شده است. پس از بررسی عملکرد مدل‌ها روی این بخش کوچک از داده‌ها، **بهترین مدل انتخاب شد** و سپس با داده‌های مختلف مورد آزمایش قرار گرفت تا عملکرد نهایی آن ارزیابی شود. همچنین تمام روش های بالانس کردن فقط روی داده آموزش انجان شده اند.

روش های مختلف برای بالانس کردن دیتا

روش اول)

در **اولین روش متعادل‌سازی**، کمترین تعداد نمونه بین کلاس‌ها محاسبه شد و سپس مجموعه داده تغییر یافت به‌گونه‌ای که **تمام کلاس‌ها دارای تعداد نمونه یکسان باشند** .این روش با کاهش تعداد نمونه‌های کلاس‌های پرتکرار، تعادل را برقرار کرد، اما باعث از دست رفتن حجم زیادی از داده‌های آموزشی شد، که می‌توانست بر عملکرد مدل تأثیر منفی بگذارد.



روش دوم)

در **روش دوم متعادل‌سازی**، به هر کلاس **وزنی متناسب با تعداد نمونه‌های آن کلاس** اختصاص داده شد. این وزن‌ها بر اساس فرمول **معکوس فراوانی کلاس** محاسبه شدند، به این صورت که:

- مجموع تعداد نمونه‌های هر کلاس محاسبه شد.
- وزن هر کلاس به‌صورت **معکوس تعداد نمونه‌های آن کلاس** تعیین گردید، یعنی کلاس‌هایی که نمونه‌های کمتری داشتند وزن بیشتری دریافت کردند.
- یک مقدار کوچک **1e−10 1e-10** به مخرج اضافه شد تا از تقسیم بر صفر جلوگیری شود.

این روش باعث شد که مدل در هنگام آموزش، کلاس‌های کم‌تعداد را با اهمیت بیشتری پردازش کند، اما همچنان چالش‌هایی در عملکرد مدل باقی ماند.

روش سوم)

در **روش سوم متعادل‌سازی**، ترکیبی از **Oversampling** و **Undersampling** برای تنظیم تعداد نمونه‌های کلاس‌ها به یک مقدار متعادل استفاده شد. در این روش:

- **کلاس‌هایی که تعداد نمونه‌های کمی داشتند، Oversample شدند** تا تعداد نمونه‌های آن‌ها افزایش یابد. این کار با نمونه‌گیری تصادفی از همان داده‌های کلاس و ایجاد داده‌های تکراری انجام شد.

- **کلاس‌هایی که نمونه‌های زیادی داشتند، Undersampleشدند** تا تعدادشان کاهش یابد. برای این کار، تعداد مشخصی از نمونه‌ها به‌صورت تصادفی انتخاب شدند.
- در نهایت، تمام کلاس‌ها به تعداد نسبتاً برابری از نمونه‌ها رسیدند تا مدل با داده‌ای متعادل آموزش ببیند.

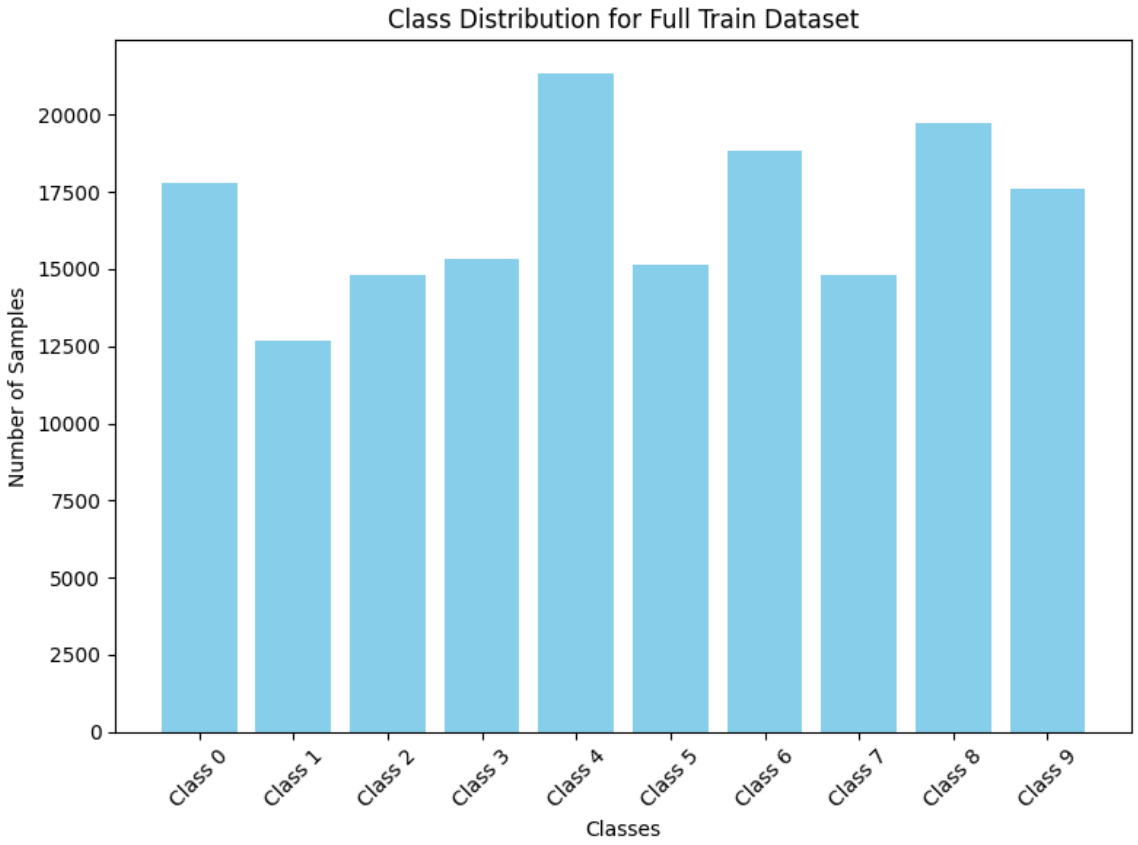
این روش نسبت به روش‌های قبلی عملکرد بهتری داشت، زیرا بدون حذف بیش از حد داده‌ها، کلاس‌های نادر را تقویت کرد و از تأثیر بیش از حد کلاس‌های پرتکرار جلوگیری نمود.

با این حال، **عملکرد این روش روی مدل همچنان مطلوب نبود.**

با وجود اینکه این روش توانست عدم تعادل بین کلاس‌ها را تا حدی برطرف کند، اما مشکلاتی ایجاد شد:

- **Oversamplingباعث بیش‌برازش (Overfitting) مدل روی کلاس‌های کم‌نمونه شد،** زیرا داده‌های تکراری مدل را گمراه می‌کرد.
- **Undersamplingباعث از دست رفتن اطلاعات مفید در کلاس‌های پرتکرار شد،** که ممکن است در تصمیم‌گیری مدل تأثیر منفی گذاشته باشد.
- **مدل همچنان در تشخیص کلاس‌های کم‌تعداد دچار مشکل بود** و نتایج پیش‌بینی به اندازه کافی بهبود نیافت.

در نتیجه، این روش نیز نتوانست مشکل نامتوازن بودن داده‌ها را به‌طور کامل حل کند.



روش چهارم)

در **روش چهارم متعادل‌سازی**، به جای تکرار داده‌ها(Duplicating) ، از یک رویکرد جدید **Oversampling**استفاده شد. این روش شامل چندین مرحله بود:

۱. **پیش‌پردازش داده‌ها:**

- در این مرحله، داده‌های متنی از اطلاعات غیرضروری پاکسازی شدند. این موارد شامل :
  - حذف لینک‌ها
  - حذف کلمات توقف(Stop Words)
  - حذف علائم و کاراکترهای اضافی

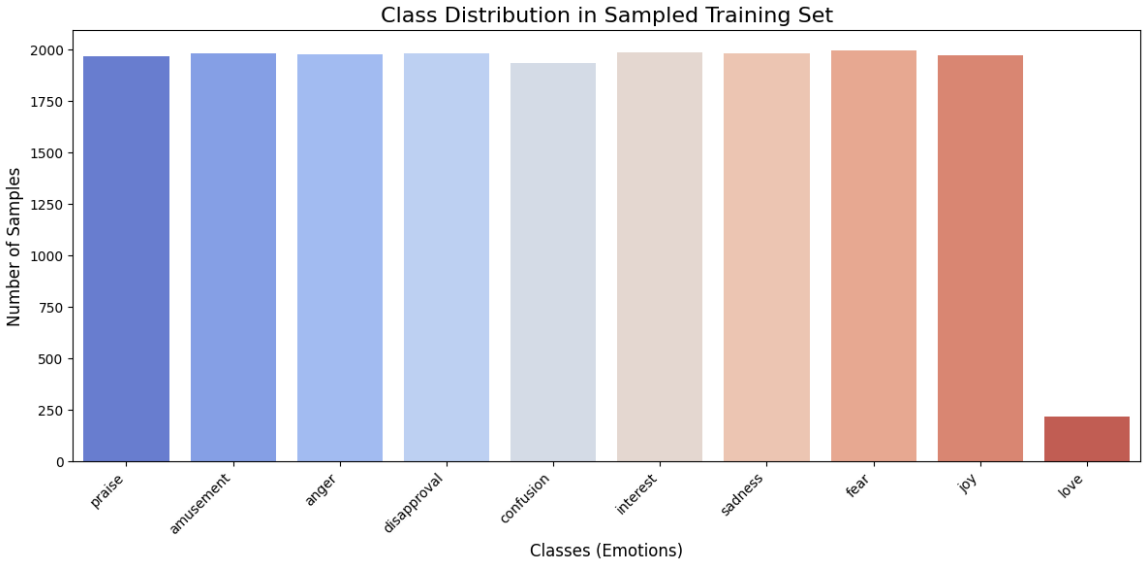
۲. **تولید نمونه‌های جدید:**

- پس از پیش‌پردازش داده‌ها، **پرتکرارترین کلمات در هر کلاس** استخراج شدند.

- بر اساس این کلمات پرتکرار، نمونه‌های جدیدی برای هر کلاس تولید شدند که به مدل کمک کند تنوع داده‌ها افزایش یابد.

این روش سعی داشت مشکل ایجاد داده‌های تکراری (که باعث بیش‌برازش می‌شد) را برطرف کند و به جای آن داده‌های متنوع‌تر و مبتنی بر ویژگی‌های اصلی هر کلاس تولید کند.

با این حال، این روش نیز نتایج کاملاً مطلوبی ارائه نداد. در برخی کلاس‌ها، داده‌های تولیدشده نتوانستند به خوبی نماینده داده‌های واقعی باشند و همچنان مدل در تشخیص برخی کلاس‌ها دچار مشکل شد.




```
Top words for class 'praise':
['name' 'thats' 'good' 'awesome' 'one' 'like' 'im' 'great' 'love'
 'amazing' 'looks' 'best' 'know' 'pretty' 'hes' 'actually' 'nice' 'well'
 'guy' 'people' 'way' 'also' 'right' 'never' 'ive' 'want' 'say' 'comment'
 'dont' 'even' 'done' 'still' 'weird' 'already' 'post' 'thought' 'ever'
 'welcome' 'yes' 'would' 'lmao' 'something' 'exactly' 'though' 'thank'
 'think' 'really' 'wish' 'better' 'make' 'maybe' 'come' 'man' 'seen' 'get'
 'cause' 'didnt' 'point' 'need' 'religion' 'yeah' 'going' 'lol' 'agree'
 'always' 'fix' 'brave' 'oh' 'hate' 'totally' 'favorite' 'got' 'hope'
 'greatest' 'us' 'wow' 'makes' 'cool' 'ok' 'completely' 'neat' 'worth'
 'games' 'game' 'friend' 'appreciate' 'subreddit' 'reddit' 'left' 'cant'
 'okay' 'glad' 'see' 'read' 'time' 'price' 'thing' 'dog' 'play'
 'highlight']
--------------------------------------------------
Top words for class 'amusement':
['lol' 'name' 'haha' 'good' 'one' 'like' 'im' 'funny' 'man' 'lmao' 'thats'
 'joke' 'look' 'got' 'big' 'yeah' 'hahaha' 'way' 'id' 'game' 'happy'
 'still' 'would' 'know' 'thought' 'feel' 'moment' 'make' 'really' 'get'
 'fuck' 'fun' 'give' 'ha' 'better' 'people' 'cant' 'thanks' 'last' 'thank'
 'expect' 'oooh' 'right' 'much' 'epic' 'gamer' 'excited' 'year' 'even'
 'video' 'think' 'knew' 'bad' 'goodlooking' 'mloled' 'day' 'nike' 'well'
 'sound' 'internet']
--------------------------------------------------
Top words for class 'anger':
['name' 'dont' 'im' 'like' 'people' 'hate' 'really' 'feel' 'would'
 'doesnt' 'thats' 'fucking' 'fuck' 'get' 'know' 'even' 'want' 'worst'
 'bad' 'much' 'yet' 'make' 'see' 'oh' 'way' 'one' 'many' 'stupid' 'yes'
 'yeah' 'men' 'good' 'go' 'hell' 'shit' 'look' 'bullshit' 'someone'
 'things' 'post' 'makes' 'still' 'time' 'always' 'mean' 'sub' 'weird'
 'think' 'isnt' 'use' 'going' 'left' 'youre' 'didnt' 'hated' 'never' 'hes'
 'right' 'something' 'damn' 'nothing' 'let' 'getting' 'call' 'shes'
 'anything' 'play' 'man' 'important' 'annoying' 'ever' 'away' 'pretty'
```

```
 'thread' 'dirty' 'long' 'put' 'work' 'exactly' 'reading' 'game' 'dumb'
 'sometimes' 'actually' 'terrible']
--------------------------------------------------
Top words for class 'disapproval':
['name' 'dont' 'like' 'thats' 'im' 'people' 'one' 'weird' 'want' 'worst'
 'looks' 'really' 'seen' 'make' 'right' 'way' 'cant' 'would' 'youre'
 'disgusting' 'ive' 'probably' 'another' 'even' 'either' 'know' 'time'
 'guy' 'bad' 'think' 'didnt' 'something' 'never' 'oh' 'wrong' 'see' 'get'
 'left' 'theres' 'love' 'got' 'post' 'live' 'though' 'gonna' 'doesnt'
 'said' 'stupid' 'go' 'taste' 'enough' 'look' 'start' 'still' 'awful'
 'wasnt' 'real' 'wanted' 'best' 'us' 'say' 'bit' 'lol' 'shes' 'nothing'
 'ohyuck' 'someone' 'much' 'anyone' 'getting' 'care' 'related' 'nah'
 'yeah' 'hate' 'game' 'worse' 'shit']
--------------------------------------------------
Top words for class 'confusion':
['name' 'like' 'dont' 'im' 'thats' 'know' 'get' 'youre' 'people' 'one'
 'would' 'never' 'oh' 'still' 'could' 'got' 'thing' 'think' 'well' 'right'
 'good' 'want' 'see' 'cant' 'go' 'way' 'bad' 'didnt' 'even' 'sure' 'also'
 'someone' 'thought' 'work' 'really' 'time' 'say' 'probably' 'actually'
 'much' 'make' 'back' 'fuck' 'whats' 'though' 'yes' 'look' 'yeah'
 'something' 'doesnt' 'theyre' 'guy' 'feel' 'life' 'money' 'let' 'tho'
 'said' 'maybe' 'many' 'looks' 'id' 'take' 'far' 'lot' 'going' 'getting'
 'every' 'day' 'wow' 'last' 'hes' 'sub' 'ive' 'love' 'gonna' 'long'
 'season' 'white' 'everyone' 'first' 'theres' 'may' 'weird' 'totally'
 'surprise' 'pretty' 'always' 'use' 'problem' 'talking' 'control' 'ever'
 'believe' 'read' 'stop' 'man' 'wrong' 'saying' 'huh']
--------------------------------------------------
Top words for class 'interest':
['name' 'like' 'would' 'get' 'think' 'wish' 'one' 'whats' 'im' 'see'
 'going' 'help' 'still' 'wonder' 'didnt' 'waiting' 'thats' 'actually'
 'youre' 'someone' 'mean' 'ive' 'watch' 'hope' 'anyone' 'friend' 'live'
 'believe' 'dont' 'cant' 'damn' 'much' 'id' 'thought' 'suspicious' 'year'
 'favorite' 'without' 'want' 'nice' 'love' 'really' 'water' 'something'
 'definitely' 'sounds' 'game' 'place' 'take' 'people' 'rough' 'yeah']
--------------------------------------------------
Top words for class 'sadness':
['im' 'name' 'sorry' 'sad' 'bad' 'like' 'people' 'feel' 'didnt' 'really'
 'know' 'thats' 'lost' 'dont' 'get' 'cant' 'see' 'even' 'go' 'good' 'yeah'
 'man' 'oh' 'wrong' 'problem' 'used' 'poor' 'pain' 'lol' 'would' 'well'
 'play' 'hate' 'back' 'think' 'youre' 'much' 'mean' 'sadly' 'makes' 'one'
 'make' 'face' 'family' 'hope' 'focus' 'thing' 'bamboozled' 'post' 'life'
 'miss' 'always' 'game' 'someone' 'ruined' 'destroyed' 'going' 'anymore'
 'loss' 'waiting' 'probably' 'isnt' 'different' 'exactly' 'got' 'look'
 'ago' 'wait' 'whole' 'could' 'enough' 'shitting' 'many']
--------------------------------------------------
Top words for class 'fear':
['im' 'name' 'scared' 'find' 'get' 'happen' 'creepy' 'worried' 'time'
 'problem' 'would' 'horrible' 'danger' 'comes' 'demonize' 'thats'
 'terrible' 'got' 'better' 'picture' 'lot' 'terrifying' 'anxious' 'bad']
--------------------------------------------------
Top words for class 'joy':
['name' 'happy' 'good' 'hope' 'im' 'one' 'well' 'going' 'would' 'dont'
 'thats' 'thought' 'know' 'didnt' 'like' 'see' 'even' 'luck' 'youre'
 'love' 'thanks' 'thank' 'right' 'think' 'ive' 'enjoy' 'help' 'day' 'got'
 'never' 'glad' 'us' 'time' 'still' 'cool' 'getting' 'really' 'id' 'true'
 'game' 'first' 'helpful' 'could' 'ill' 'guy' 'man' 'thing' 'friend'
 'someone' 'best' 'say' 'others' 'work' 'lmao' 'lol' 'wish' 'oh' 'better'
 'job' 'always' 'people' 'keep' 'lets' 'look' 'might' 'void' 'try'
 'already' 'years' 'home' 'finally' 'words' 'post' 'haha' 'something'
 'also' 'makes' 'posted' 'fun' 'play' 'need' 'much' 'come'
 'congratulations' 'seen' 'amazing' 'want' 'yeah' 'put' 'life' 'sure'
 'make' 'relationship' 'check' 'nice' 'probably' 'year' 'seems' 'feel'
 'incredible']
--------------------------------------------------
Top words for class 'love':
['love' 'thank' 'thanks' 'name' 'like' 'im' 'good' 'would' 'thats' 'one'
 'much' 'lot' 'see' 'really' 'ill' 'wow' 'way' 'dont' 'youre' 'got' 'know'
 'well' 'bro' 'back' 'hear' 'never' 'even' 'appreciate' 'always' 'feel'
 'try' 'trying' 'keep' 'best' 'baby' 'advice' 'lol' 'make' 'sir' 'get'
 'glad' 'think' 'people' 'post' 'please' 'loved' 'said' 'family' 'still'
 'though' 'looks' 'bless' 'sure' 'oh' 'thought' 'man' 'alone' 'sometimes'
 'help' 'hug' 'us' 'work' 'yes' 'friends' 'id' 'woman' 'didnt' 'makes'
 'clarification' 'means' 'something' 'sorry' 'social' 'question' 'dumb'
 'without' 'stronger' 'care' 'guy' 'making' 'welcome' 'come']
```

روش پنجم)

در **روش پنجم متعادل‌سازی**، از ترکیب **SMOTE (Synthetic Minority Over-sampling Technique)** و **TF-IDF (Term Frequency-Inverse Document Frequency)** برای ایجاد تعادل بین کلاس‌ها استفاده شد. این روش شامل دو بخش اصلی بود:

۱ .تبدیل متن‌ها به بردارهای عددی با TF-IDF

- برای اینکه بتوان از SMOTE روی داده‌های متنی استفاده کرد، ابتدا متن‌ها باید به بردارهای عددی تبدیل می‌شدند.
- برای این کار، از **TF-IDF** استفاده شد که وزن هر کلمه را بر اساس میزان اهمیت آن در متن محاسبه می‌کند.
- این روش نه تنها داده‌ها را به فرمت عددی قابل پردازش برای مدل‌های یادگیری ماشین تبدیل کرد، بلکه تأثیر کلمات پرتکرار و غیرمفید را کاهش داد.

۲ .استفاده از SMOTE برای افزایش نمونه‌های کلاس‌های کم‌تعداد

- SMOTE یکی از روش‌های محبوب **Oversampling** است که برخلاف روش‌های سنتی، به جای تکرار مستقیم نمونه‌های موجود، **داده‌های مصنوعی جدید** ایجاد می‌کند.
- در این روش، **نمونه‌های جدید در فضای برداری** بین نقاط داده‌ی واقعی ایجاد شدند، که باعث شد داده‌های افزوده‌شده شباهت بیشتری به داده‌های اصلی داشته باشند.
- این کار به مدل کمک کرد که روی کلاس‌های کم‌تعداد عملکرد بهتری داشته باشد، بدون اینکه داده‌های تکراری باعث بیش‌برازش (Overfitting) شوند.

با وجود استفاده از روش SMOTE و TF-IDF، تعادل بین کلاس‌ها به‌درستی ایجاد نشد.

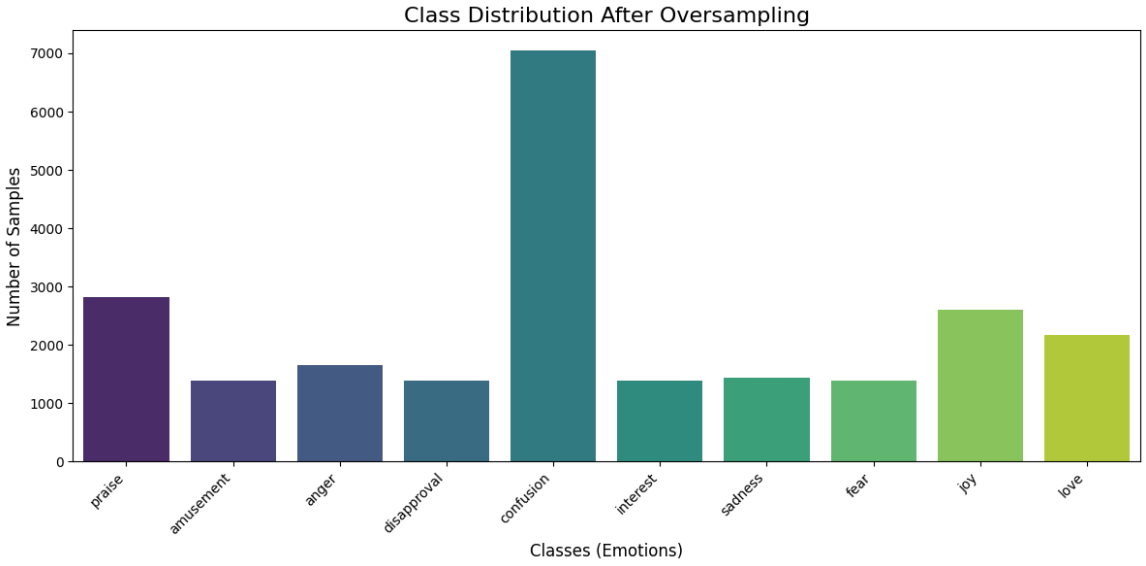
مشکلات اصلی این روش شامل موارد زیر بود:

SMOTE برای داده‌های متنی به‌خوبی عمل نمی‌کند، زیرا این روش در فضای برداری کار می‌کند و متن‌ها را به‌صورت عددی نمی‌فهمد. در نتیجه، نمونه‌های جدیدی که ایجاد شدند، ممکن است ارتباط معنایی درستی با داده‌های اصلی نداشته باشند.

TF-IDF اطلاعات معنایی عمیقی را حفظ نمی‌کند، بلکه تنها فراوانی کلمات را در نظر می‌گیرد. بنابراین، بردارهای تولیدشده ممکن است نتوانند تمام ویژگی‌های متن را به‌خوبی نمایان کنند.

عدم تعادل همچنان باقی ماند، زیرا SMOTE نتوانست به‌طور مؤثر کلاس‌های کم‌تعداد را تقویت کند. در نتیجه، مدل همچنان در تشخیص این کلاس‌ها ضعف داشت.

در مجموع، این روش نیز نتوانست مشکل نامتوازن بودن داده‌ها را حل کند و برای بهبود عملکرد مدل، نیاز به روش‌های متعادل‌سازی پیشرفته‌تر است.



روش ششم)

روش ششم: حذف نمونه‌های اضافی از کلاس‌های پرتکرار

در این روش، برای ایجاد تعادل بین کلاس‌ها، **تعداد نمونه‌های هر کلاس محدود شد** .به این صورت که:

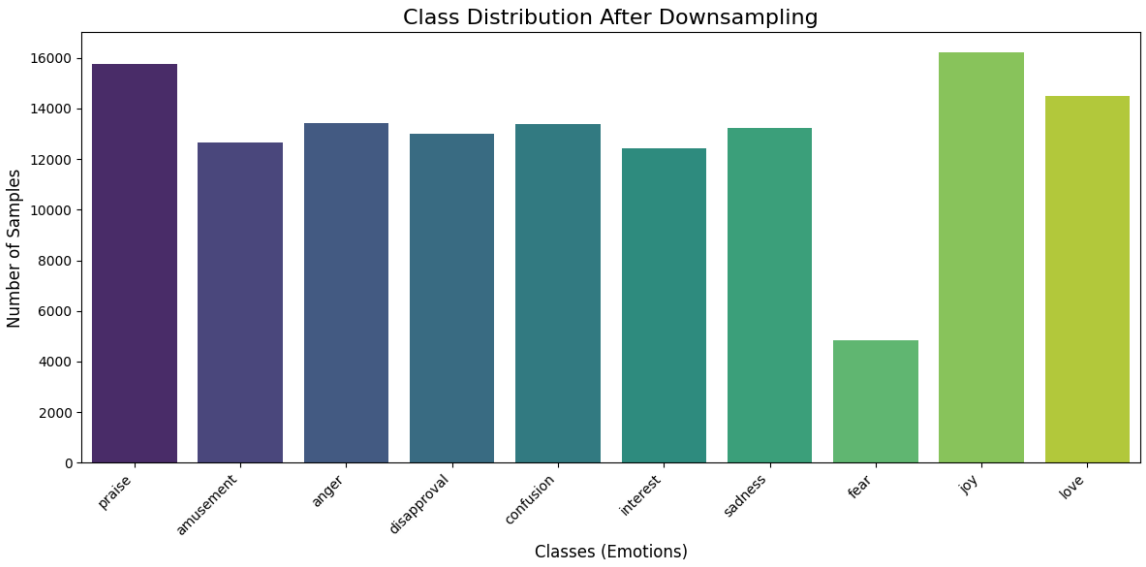
- اگر تعداد نمونه‌های یک کلاس **بیش از حد مشخصی بود، نمونه‌های اضافی حذف شدند.**
- این کار باعث شد که کلاس‌های پرتکرار بیش از حد غالب نباشند و مدل بتواند نمونه‌های کم‌تعداد را بهتر یاد بگیرد.

نتیجه و عملکرد

این روش به خوبی عمل کرد و چندین مزیت داشت:

- تعادل بهتری بین کلاس‌ها ایجاد شد و مدل توانست داده‌های کم‌تعداد را به درستی یاد بگیرد.
- **حجم کلی داده کاهش یافت**، که باعث افزایش سرعت پردازش و آموزش مدل شد.
- برخلاف روش‌هایOversampling ، هیچ داده‌ی تکراری یا مصنوعی ایجاد نشد، که احتمال **بیش‌برازش (Overfitting)**را کاهش داد.
- توزیع نهایی داده‌ها تقریباً متوازن شد و کلاس‌ها نزدیک به هم قرار گرفتند.

با توجه به نتایج، این روش نسبت به سایر روش‌های تست‌شده **بهترین عملکرد را داشت** و تعادل داده‌ها را با حفظ کیفیت آن‌ها برقرار کرد.



آموزش و تست مدل های مختلف

آموزش اول)

در اولین مرحله از آموزش، مدل DistilBERT برای طبقه‌بندی چندبرچسبی **با استفاده از روش‌های وزن‌دهی داخلی برای متعادل‌سازی کلاس‌ها** و **۱٪ از داده‌های خام** مورد استفاده قرار گرفت.

تنظیمات مدل شامل موارد زیر بود:

- **مدل پایه**distilbert-base-uncased :
- **تعداد برچسب‌ها :**برابر با تعداد کلاس‌های موجود در داده‌ها
- **نوع مسئله**) multi_label_classification :مدل می‌تواند بیش از یک برچسب را برای هر نمونه پیش‌بینی کند(
- **استفاده از وزن‌دهی داخلی :**مدل به‌طور خودکار کلاس‌های کم‌تعداد را بیشتر وزن‌دهی کرد تا تأثیر کلاس‌های پرتکرار کاهش یابد.

هدف از این آموزش اولیه:

- بررسی **عملکرد اولیه مدل** روی حجم کوچکی از داده‌ها قبل از استفاده از مجموعه داده کامل.
- تست کردن **تنظیمات مدل و پردازش داده‌ها** بدون نیاز به زمان طولانی برای آموزش.
- شناسایی **مشکلات احتمالی در داده‌ها یا مدل** قبل از اعمال روی مجموعه داده اصلی.

این مدل به دلیل استفاده از روش‌های وزن‌دهی داخلی توانست تا حدی تأثیر عدم تعادل کلاس‌ها را کاهش دهد، اما همچنان محدودیت‌هایی در عملکرد وجود داشت.

```
Classification Report (Optimized Thresholds):
              precision    recall  f1-score   support

      praise       0.42      0.44      0.43       314
   amusement       0.31      0.37      0.34       142
       anger       0.31      0.33      0.32       184
  disapproval       0.23      0.41      0.30       153
   confusion       0.45      0.73      0.56       671
    interest       0.40      0.41      0.40       128
     sadness       0.31      0.38      0.34       160
        fear       0.71      0.28      0.40        43
         joy       0.32      0.34      0.33       262
        love       0.70      0.68      0.69       240

   micro avg       0.41      0.52      0.46      2297
   macro avg       0.42      0.44      0.41      2297
weighted avg       0.42      0.52      0.45      2297
 samples avg       0.43      0.52      0.45      2297
```

```
Epoch 1, Loss: 0.2857, Accuracy: 5.22%
Epoch 2, Loss: 0.2453, Accuracy: 13.45%
Epoch 3, Loss: 0.2094, Accuracy: 27.89%
Epoch 4, Loss: 0.1683, Accuracy: 45.53%
Epoch 5, Loss: 0.1327, Accuracy: 59.31%
Epoch 6, Loss: 0.1075, Accuracy: 67.46%
Epoch 7, Loss: 0.0911, Accuracy: 71.88%
Epoch 8, Loss: 0.0782, Accuracy: 74.62%
Epoch 9, Loss: 0.0683, Accuracy: 76.48%
Epoch 10, Loss: 0.0607, Accuracy: 77.91%
Epoch 11, Loss: 0.0560, Accuracy: 78.64%
Epoch 12, Loss: 0.0518, Accuracy: 79.48%
Epoch 13, Loss: 0.0491, Accuracy: 80.29%
Epoch 14, Loss: 0.0468, Accuracy: 80.61%
Epoch 15, Loss: 0.0452, Accuracy: 81.06%
Epoch 16, Loss: 0.0433, Accuracy: 81.38%
Epoch 17, Loss: 0.0438, Accuracy: 81.19%
Epoch 18, Loss: 0.0420, Accuracy: 81.80%
Epoch 19, Loss: 0.0413, Accuracy: 81.94%
Epoch 20, Loss: 0.0397, Accuracy: 82.04%
```

آموزش دوم)

آموزش دوم مدل: بهینه‌سازی آستانه‌ها و استفاده از Early Stopping

در مرحله دوم آموزش مدل، دو تغییر کلیدی نسبت به آموزش اول اعمال شد:

۱. **استفاده از آستانه‌های بهینه برای هر کلاس** به جای مقدار ثابت **50٪**

۲. **استفاده از Early Stopping** برای جلوگیری از بیش‌برازش

۱ .تنظیم آستانه‌های بهینه برای هر کلاس

در این مرحله، به جای استفاده از مقدار پیش‌فرض **0.5**برای تصمیم‌گیری در پیش‌بینی کلاس‌ها، **آستانه بهینه برای هر برچسب به‌طور جداگانه محاسبه شد.**

روش کار:

- با استفاده از متریک **Precision-Recall Curve**برای هر کلاس، **بهترین مقدار آستانه** شناسایی شد.
- **آستانه‌ای انتخاب شد که بیشترین مقدار F1-Score را ارائه دهد.**
- این کار برای هر کلاس انجام شد تا تصمیم‌گیری برای پیش‌بینی بهتر شود.

مزیت این روش:

- استفاده از **آستانه‌های اختصاصی برای هر کلاس** باعث شد مدل بهتر تعادل بین Precision و Recall را برقرار کند.
- کلاس‌هایی که احتمال پیش‌بینی آن‌ها پایین بود، آستانه پایین‌تری دریافت کردند و برعکس.

۲ .اعمالEarly Stopping

هدف :جلوگیری از **بیش‌برازش (Overfitting)** و توقف آموزش مدل در بهترین نقطه ممکن.

نتایج آموزش دوم:

بهبود عملکرد مدل نسبت به آموزش اول
تعادل بهتر بین Precision و Recall برای هر کلاس
مدل سریع‌تر به نقطه بهینه رسید و از Overfitting جلوگیری شد

با این تغییرات، مدل عملکرد پایدارتری به دست آورد، اما همچنان نیاز به بهبودهای بیشتر برای بهینه‌سازی نهایی داشت.

```
Epoch 1, Loss: 0.2857, Accuracy: 5.22%
Epoch 2, Loss: 0.2453, Accuracy: 13.45%
Epoch 3, Loss: 0.2094, Accuracy: 27.89%
Epoch 4, Loss: 0.1683, Accuracy: 45.53%
Epoch 5, Loss: 0.1327, Accuracy: 59.31%
Epoch 6, Loss: 0.1075, Accuracy: 67.46%
Epoch 7, Loss: 0.0911, Accuracy: 71.88%
Epoch 8, Loss: 0.0782, Accuracy: 74.62%
Epoch 9, Loss: 0.0683, Accuracy: 76.48%
Epoch 10, Loss: 0.0607, Accuracy: 77.91%
Epoch 11, Loss: 0.0560, Accuracy: 78.64%
Epoch 12, Loss: 0.0518, Accuracy: 79.48%
Epoch 13, Loss: 0.0491, Accuracy: 80.29%
Epoch 14, Loss: 0.0468, Accuracy: 80.61%
Epoch 15, Loss: 0.0452, Accuracy: 81.06%
Epoch 16, Loss: 0.0433, Accuracy: 81.38%
Epoch 17, Loss: 0.0438, Accuracy: 81.19%
Epoch 18, Loss: 0.0420, Accuracy: 81.80%
Epoch 19, Loss: 0.0413, Accuracy: 81.94%
Epoch 20, Loss: 0.0397, Accuracy: 82.04%
```

آموزش سوم)

در آموزش سوم مدل از داده‌های متعادل‌شده که در مراحل بعدی پردازش داده به دست آمده بود استفاده شد اما در این مرحله از **روش‌های وزن‌دهی داخلی مدل** استفاده نشد و فقط داده‌های ورودی متوازن شدند و نتیجه تقریبا مشابه بود فقط داده ها کمتر بودند و سرعت آموزش بیشتر شد.

مدل روی مجموعه داده‌ای آموزش دید که کلاس‌ها در آن به تعداد نسبتاً برابری نمونه داشتند این داده‌ها با استفاده از روش‌هایی مانند حذف نمونه‌های اضافی از کلاس‌های پرتکرار ایجاد شده بودند اما هیچ گونه وزن‌دهی داخلی برای جبران عدم تعادل کلاس‌ها اعمال نشد

مزایای این روش شامل بهبود عملکرد مدل در تشخیص کلاس‌های کم‌تعداد کاهش تأثیر کلاس‌های پرتکرار بر خروجی مدل و بهبود تعادل بین precision و recall بود همچنین با کاهش تعداد داده‌ها زمان پردازش نیز بهینه‌تر شد

با این حال از آنجایی که مدل بدون استفاده از وزن‌دهی داخلی آموزش دیده بود ممکن است همچنان کلاس‌های نادر به‌خوبی یاد گرفته نشده باشند نتایج این آموزش نشان داد که استفاده از داده‌های متعادل‌شده تأثیر مثبتی بر عملکرد مدل دارد اما همچنان نیاز به بررسی و بهینه‌سازی بیشتر برای رسیدن به بهترین دقت ممکن وجود دارد

```
Average Loss on Test Set: 1.4560
Accuracy on Test Set (All Emotions Correct per Text): 10.69%

Classification Report (Using Pre-defined Thresholds):
              precision    recall  f1-score   support

      praise       0.43      0.52      0.47       431
   amusement       0.58      0.41      0.48       263
       anger       0.30      0.26      0.28       267
disapproval       0.52      0.70      0.60       345
   confusion       0.40      0.40      0.40       360
    interest       0.58      0.40      0.48       384
     sadness       0.41      0.40      0.41       344
        fear       0.72      0.57      0.64       202
         joy       0.47      0.37      0.41       388
        love       0.70      0.54      0.61       384

   micro avg       0.50      0.46      0.48      3368
   macro avg       0.51      0.46      0.48      3368
weighted avg       0.51      0.46      0.48      3368
 samples avg       0.48      0.45      0.44      3368
```

آموزش چهارم)

در آموزش چهارم، مدل با استفاده از **آستانه‌های جدید برای هر کلاس، داده‌های خام و روش‌های وزن‌دهی داخلی برای متعادل‌سازی کلاس‌ها** آموزش داده شد.

ویژگی‌های این مرحله:

- **استفاده از داده‌های خام :**در این مرحله داده‌ها بدون متعادل‌سازی قبلی استفاده شدند.
- **استفاده از آستانه‌های جدید :**برای هر کلاس، بهترین آستانه با استفاده از Precision-Recall Curve محاسبه شد تا تصمیم‌گیری مدل بهینه‌تر شود.
- **استفاده از وزن‌دهی داخلی مدل :**مدل از وزن‌دهی خودکار برای جبران عدم تعادل کلاس‌ها استفاده کرد.

نتایج:

- **عملکرد مدل بهبود پیدا نکرد .**علی‌رغم تنظیم آستانه‌های جدید، مدل همچنان در تشخیص کلاس‌های کم‌تعداد مشکل داشت.
- **روش وزن‌دهی داخلی تأثیر قابل توجهی نداشت .**مدل نتوانست تعادل مناسبی بین کلاس‌ها برقرار کند و نتایج همچنان نامتعادل باقی ماند.

- **مشکل اصلی :**داده‌های خام بدون متعادل‌سازی باعث شدند که مدل به کلاس‌های پرتکرار بیشتر متمایل شود و تنظیم آستانه‌های جدید نیز نتوانست این مشکل را به‌طور مؤثری برطرف کند.

در نتیجه، این روش نیز عملکرد مطلوبی نداشت و نیاز به تغییر در استراتژی داده‌پردازی و مدل‌سازی احساس شد.

```
Average Loss on Test Set: 0.7731
Accuracy on Test Set (All Emotions Correct per Text): 26.76%

Classification Report (Using Pre-defined Thresholds):
              precision    recall  f1-score   support

      praise       0.31      0.59      0.41       314
   amusement       0.41      0.28      0.33       142
       anger       0.34      0.34      0.34       184
disapproval       0.22      0.41      0.29       153
   confusion       0.50      0.55      0.52       671
    interest       0.36      0.40      0.38       128
     sadness       0.37      0.40      0.39       160
        fear       0.36      0.40      0.38        43
         joy       0.29      0.27      0.28       262
        love       0.67      0.61      0.64       240

   micro avg       0.39      0.46      0.42      2297
   macro avg       0.38      0.42      0.40      2297
weighted avg       0.41      0.46      0.43      2297
 samples avg       0.41      0.47      0.42      2297
```

آموزش پنجم)

در این مرحله، مدل با استفاده از **روش نمونه‌گیری بر اساس پرتکرارترین کلمات** آموزش داده شد تا مشکل عدم تعادل کلاس‌ها بهبود یابد.

این روش نتوانست پیش‌بینی‌های مدل را بهبود ببخشد .علی‌رغم تولید داده‌های جدید بر اساس ویژگی‌های پرتکرار هر کلاس، مدل همچنان در تشخیص برخی کلاس‌ها مشکل داشت.

کیفیت نمونه‌های تولیدشده کافی نبود .نمونه‌های ایجادشده با این روش ممکن است از نظر معنایی تنوع کافی نداشته باشند، که باعث شد مدل همچنان به سمت کلاس‌های پرتکرار متمایل بماند.

مشکل اصلی :در حالی که روش پیشنهادی باعث افزایش داده‌های کم‌تعداد شد، اما این داده‌ها به‌اندازه کافی نماینده کلاس‌های واقعی نبودند و مدل نتوانست الگوهای جدیدی را به‌خوبی یاد بگیرد.

```
Average Loss on Test Set: 0.7247
Accuracy on Test Set (All Emotions Correct per Text): 23.72%

Classification Report (Using Pre-defined Thresholds):
              precision    recall  f1-score   support

      praise       0.44      0.30      0.35       314
   amusement       0.31      0.49      0.38       142
       anger       0.30      0.26      0.28       184
 disapproval       0.16      0.31      0.22       153
   confusion       0.39      0.81      0.52       671
    interest       0.45      0.20      0.27       128
     sadness       0.29      0.38      0.33       160
        fear       0.80      0.19      0.30        43
         joy       0.29      0.32      0.30       262
        love       0.74      0.48      0.58       240

   micro avg       0.36      0.48      0.41      2297
   macro avg       0.42      0.37      0.35      2297
weighted avg       0.40      0.48      0.40      2297
 samples avg       0.39      0.49      0.42      2297
```

آموزش ششم)

در این مرحله، مدل با داده‌هایی که از طریق **SMOTE**متعادل شده بودند، آموزش دید. هدف، افزایش نمونه‌های کلاس‌های کم‌تعداد برای بهبود تعادل داده‌ها بود.

نتایج:

- **عملکرد مدل بدتر شد** .داده‌های مصنوعی ایجادشده کیفیت لازم را نداشتند.
- **تعادل برقرار نشد** .مدل همچنان به کلاس‌های پرتکرار متمایل بود.
- **پیش‌بینی‌ها نامعتبر شدند** SMOTE .برای داده‌های متنی مؤثر نبود و باعث ایجاد نویز در داده‌ها شد.

```
Average Loss on Test Set: 0.6310
Accuracy on Test Set (All Emotions Correct per Text): 17.64%

Classification Report (Using Pre-defined Thresholds):
              precision    recall  f1-score   support

      praise       0.46      0.33      0.39       314
   amusement       0.17      0.69      0.28       142
       anger       0.39      0.30      0.34       184
disapproval       0.20      0.36      0.26       153
   confusion       0.39      0.82      0.53       671
    interest       0.34      0.31      0.33       128
     sadness       0.23      0.53      0.32       160
        fear       0.50      0.02      0.04        43
         joy       0.30      0.31      0.30       262
        love       0.80      0.45      0.57       240

   micro avg       0.33      0.51      0.40      2297
   macro avg       0.38      0.41      0.34      2297
weighted avg       0.39      0.51      0.40      2297
 samples avg       0.36      0.52      0.41      2297
```

آموزش هفتم)

در این مرحله، مدل با **۵۰٪ از داده‌های خام** آموزش داده شد.

نتایج:

- **عملکرد مدل بهبود نیافت .**افزایش حجم داده بدون پردازش مناسب تأثیر مثبتی نداشت.
- **مشکل عدم تعادل کلاس‌ها باقی ماند .**مدل همچنان به سمت کلاس‌های پرتکرار متمایل بود.
- **زمان پردازش افزایش یافت بدون بهبود قابل‌توجهی در دقت.**

```
Test Loss: 0.3996

Classification Report:
              precision    recall  f1-score   support

      praise       0.41      0.46      0.44      1187
   amusement       0.28      0.70      0.40       512
       anger       0.40      0.20      0.27       675
  disapproval       0.18      0.65      0.28       566
   confusion       0.37      0.93      0.53      2462
    interest       0.38      0.24      0.29       450
     sadness       0.30      0.61      0.40       589
        fear       0.00      0.00      0.00       150
         joy       0.27      0.49      0.35       936
        love       0.82      0.41      0.54       870

   micro avg       0.34      0.59      0.43      8397
   macro avg       0.34      0.47      0.35      8397
weighted avg       0.39      0.59      0.42      8397
 samples avg       0.39      0.61      0.45      8397
```

آموزش هشتم)

در این مرحله، به جای **DistilBERT**از **مدل BERT** برای آموزش استفاده شد تا بررسی شود که آیا یک مدل پیچیده‌تر می‌تواند عملکرد بهتری ارائه دهد یا خیر.

نتایج:

- **افزایش زمان پردازش و نیاز به منابع بیشتر** .مدل BERT نسبت به DistilBERT سنگین‌تر است و زمان بیشتری برای آموزش نیاز دارد.
- **عملکرد بدتر**
- **مشکلات مشابه مدل‌های قبلی** .بدون پردازش مناسب داده‌ها، تغییر مدل پایه به‌تنهایی تفاوت قابل‌توجهی ایجاد نکرد.

Epoch	Training Loss	Validation Loss	Micro F1	Macro F1
1	0.324000	0.276831	0.316883	0.173840
2	0.261900	0.264784	0.373786	0.270173
3	0.234300	0.264414	0.412162	0.297415

```
Test Loss: 0.3872
Test Accuracy: 15.45%

Classification Report:
              precision    recall  f1-score   support

      praise       0.47      0.43      0.45       254
   amusement       0.27      0.53      0.35       109
       anger       0.46      0.26      0.34       148
disapproval       0.19      0.60      0.29       126
   confusion       0.34      0.99      0.51       545
    interest       0.41      0.41      0.41       110
     sadness       0.32      0.51      0.39       136
        fear       0.00      0.00      0.00        34
         joy       0.30      0.41      0.35       217
        love       0.75      0.53      0.62       188

   micro avg       0.34      0.60      0.44      1867
   macro avg       0.35      0.47      0.37      1867
weighted avg       0.39      0.60      0.43      1867
 samples avg       0.38      0.62      0.45      1867
```

آموزش نهم)

آموزش نهم: استفاده از ۱۰ مدل جداگانه BERT برای هر برچسب

در این مرحله، به جای استفاده از یک مدل برای طبقه‌بندی چندبرچسبی، **۱۰ مدل BERT جداگانه برای هر برچسب** آموزش داده شد تا ببینیم آیا مدل‌های اختصاصی برای هر کلاس عملکرد بهتری دارند یا خیر.

نتایج:

- **خطای حافظه .(Memory Error)** استفاده از ۱۰ مدل BERT به‌طور هم‌زمان منابع زیادی مصرف کرد و اجرای آن امکان‌پذیر نبود.
- **زمان پردازش بسیار زیاد .**حتی با مقدار داده کمتر، این روش به دلیل تعداد زیاد مدل‌ها بسیار کند بود.
- **عدم موفقیت در اجرا .**به دلیل محدودیت سخت‌افزاری، این روش عملیاتی نشد.

آموزش دهم)

در این مرحله، **۱۰ مدل DistilBERT جداگانه** برای هر کلاس آموزش داده شد تا بررسی شود که آیا این روش عملکرد بهتری نسبت به مدل چندبرچسبی دارد یا خیر.

نتایج:

- **عملکرد ضعیف :**مقدار **F1-score = 23٪**، که نشان‌دهنده کیفیت پایین پیش‌بینی‌ها بود.
- **زمان پردازش بالا :**اجرای ۱۰ مدل مجزا زمان‌بر و ناکارآمد بود.
- **عدم تعادل کلاس‌ها همچنان مشکل‌ساز بود .**مدل‌ها برای برخی کلاس‌ها عملکرد خوبی نداشتند.

آموزش یازدهم)

در این مرحله، به جای مدل‌های مبتنی بر شبکه‌های عصبی، از **ویژگی‌های TF-IDF همراه با رگرسیون لجستیک** برای طبقه‌بندی داده‌ها استفاده شد.

نتایج:

- **عملکرد نسبتاً قابل قبول :**دقت و F1-score بهتر از برخی مدل‌های قبلی بود.
- **سرعت پردازش بالا :**این روش نسبت به BERT و DistilBERT سریع‌تر اجرا شد.
- **BERTهمچنان عملکرد بهتری داشت :**اگرچه این روش کارآمد بود، اما مدل‌های مبتنی بر BERT نتایج بهتری ارائه دادند.

	precision	recall	f1-score	support
Class 0	0.35	0.44	0.39	169
Class 1	0.37	0.49	0.42	74
Class 2	0.30	0.44	0.36	98
Class 3	0.14	0.33	0.20	63
Class 4	0.45	0.64	0.53	319
Class 5	0.23	0.46	0.31	56
Class 6	0.30	0.45	0.36	91
Class 7	0.29	0.24	0.26	29
Class 8	0.33	0.44	0.38	137
Class 9	0.59	0.75	0.66	122
micro avg	0.36	0.52	0.43	1158
macro avg	0.34	0.47	0.39	1158
weighted avg	0.38	0.52	0.44	1158
samples avg	0.39	0.52	0.42	1158

مدل نهایی

پس از آزمایش مدل‌های مختلف، مشخص شد که **DistilBERTبهترین عملکرد را دارد .**حالا تغییراتی در این مدل اعمال می‌شود تا دقت و کارایی آن بیشتر بهینه شود.

توضیح :DistilBERTمعماری، نحوه آموزش و ویژگی‌ها

۱ DistilBERTیک نسخه سبک‌تر و سریع‌تر از **BERT (Bidirectional Encoder Representations from Transformers)**است که توسط شرکت Hugging Face توسعه یافته است. این مدل به طور خاص برای کاهش پیچیدگی محاسباتی و افزایش سرعت پردازش طراحی شده است، در حالی که **٪۷۰ از پارامترهای مدل BERT را حذف کرده و همچنان ٪۹۷ از دقت آن را حفظ می‌کند.**

۲ .معماری مدلDistilBERT

DistilBERTبه عنوان یک **مدل ترانسفورمر (Transformer)** طراحی شده است و بر اساس معماری **Encoder-Only** مانند BERT ساخته شده است. مهم‌ترین تفاوت‌ها و مشخصات معماری آن شامل موارد زیر است:

تعداد لایه‌های کمتر:

- BERT-Baseدارای **۱۲ لایه ترانسفورمر** است، اما **DistilBERTفقط ۶ لایه دارد** که باعث کاهش حجم محاسبات و افزایش سرعت پردازش می‌شود.

نگهداری ازEmbedding های ورودی و خروجی:

- DistilBERTهمچنان دارای ورودی‌های **Word Embedding، Position Embeddingو Token Type Embedding**است و مانند BERT این ویژگی‌ها را حفظ می‌کند.

کاهش تعداد هدهای:Attention

- BERT-Baseدارای **۱۲ هد توجه (Attention Heads)** در هر لایه است، اما DistilBERT فقط **۶ هد دارد.**

عدم استفاده از توکن‌های:(NSP (Next Sentence Prediction

- برخلاف BERT که از یک وظیفه **پیش‌بینی جمله بعدی (NSP)** برای آموزش استفاده می‌کند، DistilBERTاین بخش را حذف کرده است، زیرا مشخص شد که تأثیر چندانی بر عملکرد مدل ندارد.

استفاده از :Knowledge Distillation

- DistilBERTاز **دانش مدل BERT** به جای آموزش مستقیم از داده خام استفاده کرده است که باعث شده مدل سبک‌تر و سریع‌تر شود.

۳ .نحوه آموزشDistilBERT

مدل DistilBERT از تکنیک **Knowledge Distillation**برای آموزش استفاده کرده است. این روش شامل انتقال دانش از یک **مدل بزرگ‌تر و دقیق‌تر (Teacher Model)** به یک **مدل کوچک‌تر و سبک‌تر (Student Model)** است.

مدل معلم (Teacher Model): BERT-Base
مدل دانش‌آموز (Student Model): DistilBERT

فرایند آموزش به‌صورت زیر انجام شده است:

۱. **استفاده از خروجی‌های BERT به عنوان راهنما**

- به جای آموزش مستقیم از داده‌های متنی خام، DistilBERT خروجی‌های BERT را دریافت می‌کند تا بتواند همان رفتار را با تعداد پارامترهای کمتر یاد بگیرد.

۲. **استفاده از Loss ترکیبی:**

- در آموزش از ترکیب **Cross-Entropy Loss برای کلاس‌بندی** و **Kullback-Leibler Divergence برای تقلید از توزیع احتمالاتی BERT** استفاده شده است.

۳. **حفظ ویژگی‌های کلیدی BERT:**

- مدل همچنان به‌صورت **Bidirectional** عمل می‌کند و می‌تواند وابستگی‌های معنایی در متن را مانند BERT استخراج کند.

۴. ویژگی‌های کلیدی DistilBERT

۷۰٪ کاهش در تعداد پارامترها در مقایسه با BERT
۶۰٪ سرعت پردازش بالاتر نسبت به BERT
۹۷٪ دقت BERT را حفظ کرده است
بدون استفاده از Next Sentence Prediction (NSP)
نگه داشتن رفتار توجه (Self-Attention) دوطرفه مانند BERT

جزئیات معماری ترانسفورمر

یک ترانسفورمر از **لایه‌های متعدد Encoder و Decoder** تشکیل شده است. هر لایه شامل سه بخش اصلی است:

۱. مکانیزم Self-Attention

توجه به خود (Self-Attention) قلب اصلی ترانسفورمرهاست که کمک می‌کند مدل بتواند روابط بین کلمات را بهتر یاد بگیرد.

✓چگونه کار می‌کند؟

در یک جمله، هر کلمه ممکن است به سایر کلمات وابسته باشد. به جای پردازش ترتیبی، مدل می‌تواند **همزمان به تمام کلمات توجه کند** و میزان اهمیت هر کلمه را برای هر بخش از جمله محاسبه کند.

مثال:

برای جمله **"The cat sat on the mat."** کلمه **"cat"** احتمالاً بیشترین ارتباط را با **"sat"** دارد، درحالی‌که **"mat"** می‌تواند با **"on"** مرتبط باشد.

۲ .مکانیزم Multi-Head Attention

به جای استفاده از یک لایه Self-Attention ، چندین لایه **Attention موازی** در مدل وجود دارد که باعث افزایش توانایی مدل در یادگیری **وابستگی‌های پیچیده** بین کلمات می‌شود.

۳ .شبکه عصبی Feed-Forward

بعد از عملیات Self-Attention ، داده‌های پردازش‌شده از یک **شبکه عصبی چندلایه‌ای (MLP)** عبور می‌کنند تا اطلاعات پردازش شوند.

۴) Positional Encoding .رمزگذاری موقعیتی(

از آنجایی که ترانسفورمرها داده‌ها را **به‌صورت موازی** پردازش می‌کنند، برخلاف RNN ها نمی‌دانند که ترتیب کلمات چگونه است. برای حل این مشکل، از **رمزگذاری موقعیتی (Positional Encoding)** استفاده می‌شود تا مدل بتواند ترتیب کلمات را متوجه شود.

پیدا کردن hyper parameter های مناسب

جستجوی بهترین هایپرپارامترها برای یادگیری مدل

در این مرحله، **بهینه‌سازی هایپرپارامترها** شامل **نرخ یادگیری(learning rate) ، وزن‌زدایی(weight decay) ، و اندازه بچ (batch size)** انجام شد.

مراحل اصلی:

۱. **آماده‌سازی داده‌ها :** داده‌ها به فرمت Dataset تبدیل شده و با BERT Tokenizer پردازش شدند.

۲. **مدل** BERT-base-uncased **:** با مسئله **طبقه‌بندی چندبرچسبی** (multi_label_classification) تنظیم شد.

۳. **استفاده از وزن‌های کلاس :**برای مقابله با **عدم تعادل کلاس‌ها**، وزن‌های متناسب با هر کلاس محاسبه و در تابع **BCEWithLogitsLoss**استفاده شد.

۴. **Trainer سفارشی‌شده :**یک Trainerاختصاصی برای در نظر گرفتن **وزن کلاس‌ها در محاسبه‌ی خطا** ایجاد شد.

۵. **جستجوی بهینه‌سازی هایپرپارامترها با :Optuna**

- **نرخ یادگیری :**بین **1e-6**تا **۱ e-4**تنظیم شد.
- **وزن‌زدایی :**بین **0.0**تا **۰,۳** تست شد.
- **اندازه بچ :**بین **16**و **۳۲** تست شد.

نتایج:

بهترین هایپرپارامترها پس از **۱۰ آزمایش** یافت شدند:

- **نرخ یادگیری**4.72e-5 **:**
- **وزن‌زدایی**0.21 **:**
- **اندازه بچ**32 **:**

مدل با این هایپرپارامترهای بهینه دوباره آموزش داده شد.

نتیجه‌گیری:

این روش بهبودهایی را در **F1-score**مدل ایجاد کرد و با تنظیم دقیق هایپرپارامترها، کارایی مدل به حداکثر رسید.

پیدا کردن token length

در این مرحله، آزمایش‌هایی با طول توکن‌های ۲۰، ۶۴ و ۱۲۸ انجام شد تا تأثیر آن بر عملکرد مدل بررسی شود.

با طول ۲۰، سرعت پردازش افزایش یافت اما اطلاعات مهم در متون بلند حذف شد و دقت مدل کاهش یافت.

با طول ۶۴، تعادل بین سرعت و حفظ اطلاعات برقرار شد و مدل عملکرد بهتری داشت.

با طول ۱۲۸، بیشترین مقدار اطلاعات حفظ شد و دقت مدل بهبود یافت، اما پردازش کندتر شد و نیاز به حافظه بیشتری داشت.

نتیجه آزمایش‌ها نشان داد که مقدار 128 بهترین نتیجه را دارد.

20:

Epoch	Training Loss	Validation Loss	Micro F1	Macro F1
1	0.900100	0.894105	0.473355	0.456550
2	0.830300	0.913427	0.482528	0.458412
3	0.715800	0.941385	0.473584	0.450855
4	0.630800	1.088408	0.469542	0.440618

64:

Epoch	Training Loss	Validation Loss	Micro F1	Macro F1
1	0.847900	0.831379	0.487878	0.473631
2	0.785300	0.844576	0.496568	0.474039
3	0.678100	0.917831	0.483390	0.464929
4	0.603900	1.043733	0.485806	0.458422

128:

Epoch	Training Loss	Validation Loss	Micro F1	Macro F1
1	0.850700	0.831324	0.488121	0.473257
2	0.784900	0.847217	0.493674	0.473005
3	0.677400	0.912741	0.483391	0.460806

Dropoutبیشتر برای جلوگیری از بیش برازش

در این مرحله، مقدار **Dropout**در مدل افزایش یافت تا تأثیر آن بر عملکرد بررسی شود.

افزایش Dropout باعث کاهش **بیش‌برازش (Overfitting)** و بهبود تعمیم مدل شد. با این حال، مقدار بیش از حد بالای آن عملکرد مدل را کاهش داد، زیرا مدل اطلاعات مهم را از دست می‌داد.

نتایج نشان داد که مقدار متعادل Dropout به بهبود دقت مدل کمک می‌کند، اما افزایش بیش از حد آن منجر به کاهش عملکرد می‌شود.

Epoch	Training Loss	Validation Loss	Micro F1	Macro F1
1	0.851100	0.830310	0.493336	0.474952
2	0.780700	0.839827	0.501089	0.474966
3	0.674900	0.913473	0.487716	0.465558
4	0.596000	1.069025	0.486769	0.459494

اضافه کردن warm-up

در این مرحله، **Warm-up**به فرآیند آموزش اضافه شد تا تأثیر آن بررسی شود.

Warm-upبه مدل کمک کرد تا در ابتدا با نرخ یادگیری پایین‌تری آموزش را شروع کند و سپس به مقدار بهینه برسد. این کار باعث **پایداری بیشتر در مراحل اولیه آموزش** و جلوگیری از تغییرات ناگهانی در وزن‌ها شد.

نتایج نشان داد که اضافه کردن Warm-up باعث **بهبود پایداری و همگرایی بهتر مدل** شد، اما تأثیر آن در مقایسه با سایر بهینه‌سازی‌ها محدود بود.

Epoch	Training Loss	Validation Loss	Micro F1	Macro F1
1	0.854100	0.832790	0.487994	0.472462
2	0.787900	0.844212	0.494231	0.474584
3	0.681300	0.907317	0.489171	0.466776
4	0.599700	1.069377	0.480708	0.456188

استفاده از کل داده ها

در این مرحله، مدل به جای استفاده از بخشی از داده‌ها، **با کل مجموعه داده آموزش داده شد.**

استفاده از تمام داده‌ها باعث شد مدل الگوهای بیشتری را یاد بگیرد و عملکرد بهتری داشته باشد. اما این کار **زمان پردازش را افزایش داد** و نیاز به **حافظه بیشتری** داشت.

نتایج نشان داد که آموزش با کل داده‌ها باعث **بهبود دقت مدل** شد

Epoch	Training Loss	Validation Loss	Micro F1	Macro F1
1	0.838600	0.813878	0.493416	0.479370
2	0.783600	0.808976	0.499750	0.481428
3	0.735500	0.838605	0.512468	0.484693
4	0.659700	0.927543	0.485218	0.471573
5	0.607900	1.083281	0.493207	0.468959

استفاده از داده بالانس شده

در این مرحله، به جای استفاده از داده‌های نامتعادل، **از داده‌های متعادل‌شده** برای آموزش مدل استفاده شد.

متعادل‌سازی داده‌ها باعث شد که **زمان آموزش کاهش یابد**، زیرا تعداد نمونه‌ها در کلاس‌های پرتکرار کاهش یافت. در عین حال، **نتایج مدل مشابه حالت استفاده از کل داده‌ها باقی ماند** و دقت مدل تغییری نکرد.

این نشان داد که استفاده از داده‌های متعادل‌شده نه‌تنها **کارایی مدل را حفظ می‌کند**، بلکه باعث **افزایش سرعت آموزش** نیز می‌شود.

Epoch	Training Loss	Validation Loss	Micro F1	Macro F1
1	0.837600	0.816333	0.495662	0.479579
2	0.785000	0.810917	0.495602	0.479422
3	0.733000	0.836544	0.514986	0.487879
4	0.657800	0.914802	0.487060	0.473357
5	0.606200	1.039375	0.488518	0.466867

مقایسه مدل نهایی

مقایسه عملکرد مدل قبل و بعد از Fine-Tuning

۱ .عملکرد مدل قبل از Fine-Tuning (اولین تصویر)

- **Precision، Recall، F1-score برای تمام کلاس‌ها برابر ۰,۰۰ است.**
- مدل هیچ کدام از احساسات (praise ، amusement ، anger و غیره) را تشخیص نداده است.

- **Micro F1, Macro F1, Weighted F1 = 0.00**که نشان‌دهنده عملکرد کاملاً ناموفق مدل است.
- **Subset Accuracy = 0.0157 (~1.5%)**بسیار پایین است.
- **ROC AUC Score (Macro) = 0.49**که نشان می‌دهد عملکرد مدل تصادفی است.

نتیجه :مدل قبل از Fine-Tuning کاملاً ناموفق بوده و نتوانسته هیچ‌کدام از کلاس‌ها را به درستی تشخیص دهد.

۲ .عملکرد مدل بعد از) Fine-Tuning دومین تصویر(

- **Training Lossکاهش یافته است:**
  - **Epoch 1:** 0.833
  - **Epoch 2:** 0.762
  - **Epoch 3:** 0.682
  - نشان‌دهنده یادگیری تدریجی مدل است.
- **Validation Lossتقریباً ثابت مانده، اما کمی افزایش یافته است:**
  - **Epoch 1:** 0.8079
  - **Epoch 2:** 0.8034
  - **Epoch 3:** 0.8436
  - کمی افزایش در Validation Lossممکن است نشان‌دهنده شروع بیش‌برازش (Overfitting) باشد.
- **بهبود در:F1-score**
  - **Micro F1:**از ۰,۰۰ به حدود **۰,۵۰** افزایش یافته است.
  - **Macro F1:**از ۰,۰۰ به **۰,۴۸۹** افزایش یافته است.
  - مدل اکنون توانایی تشخیص احساسات را پیدا کرده است.

نتیجه :پس ازFine-Tuning ، مدل عملکرد بهتری نشان داده است F1-score .از ۰ به **۵۰٪** افزایش یافته که نشان‌دهنده یادگیری موفق‌تر است. با این حال، مقدار Validation Lossکمی افزایش یافته که باید بررسی شود تا از بیش‌برازش جلوگیری شود.

مدل اولیه بدون fine-tune

```
              precision    recall  f1-score   support

      praise       0.00      0.00      0.00      3155
   amusement       0.00      0.00      0.00      1428
       anger       0.00      0.00      0.00      1924
  disapproval       0.00      0.00      0.00      1456
   confusion       0.00      0.00      0.00      6763
    interest       0.00      0.00      0.00      1273
     sadness       0.00      0.00      0.00      1644
        fear       0.00      0.00      0.00       386
         joy       0.00      0.00      0.00      2525
        love       0.00      0.00      0.00      2309

   micro avg       0.00      0.00      0.00     22863
   macro avg       0.00      0.00      0.00     22863
weighted avg       0.00      0.00      0.00     22863
 samples avg       0.00      0.00      0.00     22863

Subset Accuracy: 0.01579707978272786
Hamming Loss: 0.11393332336672149
Jaccard Score (Samples Average): 0.0
ROC AUC Score (Macro): 0.4911902770432233
/usr/local/lib/python3.11/dist-packages/sklearn/metrics
  _warn_prf(average, modifier, f"{metric.capitalize()}
Average Precision Score (Macro): 0.11738129617691355
Label Ranking Loss: 0.4686187759102246
```

مدل نهایی)

توضیح دقیق‌تر از روش‌ها و تکنیک‌های استفاده‌شده در آموزش مدل

در این مدل، از تکنیک‌های مختلف برای بهینه‌سازی و بهبود عملکرد استفاده شده است. در ادامه، توضیحات بیشتری در مورد هر کدام از این تکنیک‌ها آورده شده است:

---

۱ .پردازش داده‌ها و آماده‌سازی مجموعه داده

۱,۱ .آماده‌سازی داده‌ها

در ابتدا، داده‌های موجود در DataFrame به طور خودکار استخراج و پردازش شدند. به طور خاص، برچسب‌های احساسات از ستون‌های مختلف جدا شده و به صورت **برداری** برای هر نمونه ذخیره شدند. به این معنی که برای هر نمونه متنی، مقادیر مختلفی از احساسات (مانند تحسین، خشم، ناراحتی، و غیره) به صورت **یک آرایه از اعداد** ذخیره می‌شوند.

۱,۲ .تبدیل به فرمت مناسبHugging Face

برای استفاده راحت‌تر در مدل‌های ترانسفورمر، داده‌ها به فرمت **Dataset**تبدیل شدند که یک ساختار داده‌ای خاص در کتابخانه

Hugging Faceاست. این مجموعه داده سپس به دو بخش اصلی تقسیم شد: بخش **آموزش** و بخش **اعتبارسنجی** .این تقسیم‌بندی برای ارزیابی مدل بعد از آموزش به کار می‌رود.

---

۲ .توکن‌سازی داده‌ها

۲,۱ .استفاده از توکن‌سازDistilBERT

در این مرحله، مدل **DistilBERT**برای تبدیل متن‌های خام به فرم عددی استفاده می‌شود. این مدل به طور خودکار کلمات را به **توکن‌ها** تبدیل می‌کند که برای پردازش توسط شبکه عصبی قابل استفاده باشند. توکن‌ساز **DistilBERT**از **نسخه کوچک‌تر BERT**استفاده می‌کند که سرعت بیشتری دارد و به همین دلیل برای کارهای بزرگ و پیچیده‌تر مناسب است.

۲,۲ .بهینه‌سازی سرعت با استفاده از پردازش موازی(Multiprocessing)

برای افزایش سرعت پردازش، از **پردازش موازی** استفاده شده است. این کار به توکن‌ساز کمک می‌کند تا از چندین پردازنده برای پردازش داده‌ها به طور همزمان استفاده کند که باعث تسریع فرآیند توکن‌سازی می‌شود. تنظیم num_proc=4به این معنی است که از چهار پردازنده برای این کار استفاده می‌شود.

---

۳ .تنظیمات معماری مدل

۳,۱ .انتخاب DistilBERT به عنوان مدل پایه

مدل **DistilBERT**به دلیل سبک بودن و سرعت بالاتر نسبت به **BERT**انتخاب شد. این مدل همچنان عملکرد مناسبی برای بیشتر وظایف پردازش زبان طبیعی دارد و می‌تواند ویژگی‌های مهمی را از داده‌های متنی استخراج کند.

۳,۲ .تنظیم Dropout برای جلوگیری از بیش‌برازش(Overfitting)

برای جلوگیری از **بیش‌برازش** (که وقتی مدل به یادگیری دقیق ویژگی‌های داده‌های آموزشی می‌پردازد و نمی‌تواند به درستی داده‌های جدید را پیش‌بینی کند)، مقدار **Dropout**در لایه‌های مختلف مدل تنظیم شد. این پارامتر از ٪۳۵ **به** ٪۴۰ افزایش پیدا کرد تا مدل نتواند به راحتی به ویژگی‌های خاص داده‌ها تکیه کند و بتواند ویژگی‌های عمومی‌تری یاد بگیرد.

---

۴ .استراتژی‌های بهینه‌سازی آموزش

۴,۱ .استفاده از Warm-up در آموزش مدل

در ابتدا، نرخ یادگیری مدل به تدریج افزایش می‌یابد (Warm-up) تا از **نوسانات زیاد** در مراحل ابتدایی آموزش جلوگیری کند. این کار باعث می‌شود مدل **به آرامی شروع به یادگیری کند** و از به‌روزرسانی‌های شدید جلوگیری شود. سپس پس از آن، نرخ یادگیری به سرعت افزایش می‌یابد تا مدل بهینه‌تر یاد بگیرد.

۴,۲ .تنظیم وزن‌زدایی(Weight Decay)

Weight decayبه معنای کاهش تدریجی وزن‌ها برای جلوگیری از بزرگ شدن بی‌رویه آنها است که باعث جلوگیری از بیش‌برازش می‌شود. در این مدل، مقدار weight decay از **۰,۲۱** به **۰,۲۵** افزایش داده شد تا مدل بتواند عملاً **تعادل بهتری میان دقت و تعمیم‌پذیری** برقرار کند.

۴,۳ .استفاده از پردازش عددی با دقت نصف(FP16)

در این مرحله، از **پردازش FP16** استفاده شد که به این معنی است که محاسبات مدل با دقت کمتری انجام می‌شود. این کار باعث کاهش مصرف حافظه و افزایش سرعت آموزش می‌شود بدون اینکه تأثیر زیادی بر دقت مدل داشته باشد.

---

۵ .استفاده از معیارهای ارزیابی

۵,۱ F1-score .برای ارزیابی مدل

معیار **F1-score**به طور خاص برای ارزیابی مدل‌های **چندبرچسبی** استفاده شده است. این معیار ترکیبی از **Precision**و **Recall**است که به مدل کمک می‌کند تا دقت و توانایی بازیابی صحیح برچسب‌ها را هم‌زمان اندازه‌گیری کند. برای هر برچسب، **آستانه تصمیم‌گیری متفاوتی** اعمال شده است تا دقت پیش‌بینی برای هر برچسب بهینه شود.

---

۶ .مقابله با داده‌های نامتعادل

۶,۱ .محاسبه وزن‌های کلاس‌ها

از آنجا که داده‌ها ممکن است نامتعادل باشند، وزن‌های **مثبت و منفی** برای هر کلاس محاسبه شده‌اند. این وزن‌ها به مدل کمک می‌کنند تا حساسیت بیشتری به کلاس‌هایی که داده‌های کمتری دارند، نشان دهد و از تاثیر بیشتر کلاس‌های پرتکرار جلوگیری کند.

۶,۲ .استفاده از تابع هزینه خاص برای وزن‌دهی

در این مدل، از **توابع هزینه خاص** استفاده شده که **وزن‌های کلاس‌ها را در محاسبات خطا لحاظ می‌کند** .این کار باعث می‌شود مدل توجه بیشتری به تشخیص کلاس‌های کمتر موجود در داده‌ها داشته باشد.

---

۷ .استفاده ازEarly Stopping

۷,۱ .جلوگیری از بیش‌برازش

برای جلوگیری از **بیش‌برازش**، از تکنیک **Early Stopping**استفاده شده است. در این روش، آموزش مدل متوقف می‌شود اگر مدل برای چندین دوره متوالی بهبود نیابد. این باعث می‌شود که آموزش در **نقطه بهینه** متوقف شود و از یادگیری بیش از حد از داده‌ها جلوگیری گردد.

---

۸ .جمع‌بندی

مدل با استفاده از **DistilBERT**و تکنیک‌های مختلفی از جمله **Warm-up، Dropout، تنظیم وزن‌دایی، FP16، و Early Stopping**آموزش داده شده است. این روش‌ها باعث بهبود دقت مدل، کاهش زمان آموزش و جلوگیری از بیش‌برازش شده‌اند. علاوه بر این، استفاده از **آستانه‌های بهینه برای هر برچسب** و **وزن‌دهی کلاس‌ها** برای مقابله با داده‌های نامتعادل به عملکرد بهتر مدل کمک کرده است.

Epoch	Training Loss	Validation Loss	Micro F1	Macro F1
1	0.833000	0.807996	0.497841	0.480447
2	0.762100	0.803457	0.503615	0.486530
3	0.682500	0.843657	0.508666	0.489816

```
TrainOutput(global_step=16932, training_loss=0.7693352197998105, metrics=
{'train_runtime': 1140.5052, 'train_samples_per_second': 475.045,
'train_steps_per_second': 14.846, 'total_flos': 5109194852826600.0, 'train_loss':
0.7693352197998105, 'epoch': 3.0})
```

آنالیز مدل نهایی)

Scores)

۱ .بررسی دقت(Precision) ، بازخوانی (Recall) و F1-score برای هر کلاس

در این نتایج، مدل توانسته است **احساسات مختلف را با دقت و بازخوانی مشخصی شناسایی کند** .به بررسی عملکرد در کلاس‌های مختلف می‌پردازیم:

```
Per-class Precision, Recall, F1-score, and Support:
Class 'praise': Precision=0.497, Recall=0.563, F1-score=0.528, Support=3155
Class 'amusement': Precision=0.496, Recall=0.595, F1-score=0.541, Support=1428
Class 'anger': Precision=0.319, Recall=0.652, F1-score=0.429, Support=1924
Class 'disapproval': Precision=0.294, Recall=0.472, F1-score=0.363, Support=1456
Class 'confusion': Precision=0.559, Recall=0.599, F1-score=0.578, Support=6763
Class 'interest': Precision=0.354, Recall=0.692, F1-score=0.468, Support=1273
Class 'sadness': Precision=0.527, Recall=0.499, F1-score=0.513, Support=1644
Class 'fear': Precision=0.297, Recall=0.593, F1-score=0.396, Support=386
Class 'joy': Precision=0.318, Recall=0.567, F1-score=0.407, Support=2525
Class 'love': Precision=0.696, Recall=0.674, F1-score=0.685, Support=2309

Subset Accuracy: 0.285
Hamming Loss: 0.130
Jaccard Score (Samples Average): 0.447
ROC AUC Score (Macro): 0.858
Average Precision Score (Macro): 0.482
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: U
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
Label Ranking Loss: 0.151
```

نتایج کلی:

- کلاس **"Love"** بالاترین مقدار **F1-score (0.685)** را دارد، که نشان‌دهنده تشخیص بهتر این احساس است.
- کلاس‌هایی مانند **"Confusion"** و **"Praise"** عملکرد نسبتاً خوبی دارند.
- برخی کلاس‌ها مانند **"Disapproval"** و **"Fear"** F1-score پایین‌تری دارند که نشان می‌دهد مدل در تشخیص آن‌ها ضعیف‌تر است.

۲ .متریک‌های کلی مدل

- **Subset Accuracy = 0.285:** یعنی مدل در ۲۸,۵٪ مواقع تمام برچسب‌های یک نمونه را به درستی پیش‌بینی کرده است.
- **Hamming Loss = 0.130:** مقدار پایین‌تر بهتر است، و نشان می‌دهد که نرخ خطای مدل در سطح برچسب‌ها نسبتاً مناسب است.
- **Jaccard Score = 0.447:** مقدار ۴۴,۷٪ نشان می‌دهد که مدل توانسته به طور نسبی مجموعه برچسب‌های صحیح را پیش‌بینی کند.
- **ROC AUC Score (Macro) = 0.858:** نشان‌دهنده تفکیک‌پذیری مناسب مدل بین کلاس‌های مختلف است.
- **Average Precision Score (Macro) = 0.482:** نشان می‌دهد که دقت میانگین مدل در دسته‌بندی چندبرچسبی تقریباً ۴۸,۲٪ است.

- **Label Ranking Loss = 0.151:** مقدار کمتر نشان‌دهنده رتبه‌بندی بهتر مدل در اولویت‌بندی احساسات مختلف برای یک نمونه است.

۳ .تحلیل کلی و نتیجه‌گیری

پیشرفت قابل توجه نسبت به مدل‌های اولیه:

- مدل دیگر کاملاً ناموفق نیست و توانسته است احساسات را تا حدودی تشخیص دهد.
- مقدار **F1-score** برای بیشتر کلاس‌ها به **بیش از ۰,۴** رسیده که نشان از بهبود مدل دارد.
- **ROC AUC برابر با ۰,۸۵۸** نشان می‌دهد که مدل عملکرد قابل قبولی دارد و از حالت تصادفی بسیار بهتر است.

چالش‌ها و بهینه‌سازی‌های پیشنهادی:

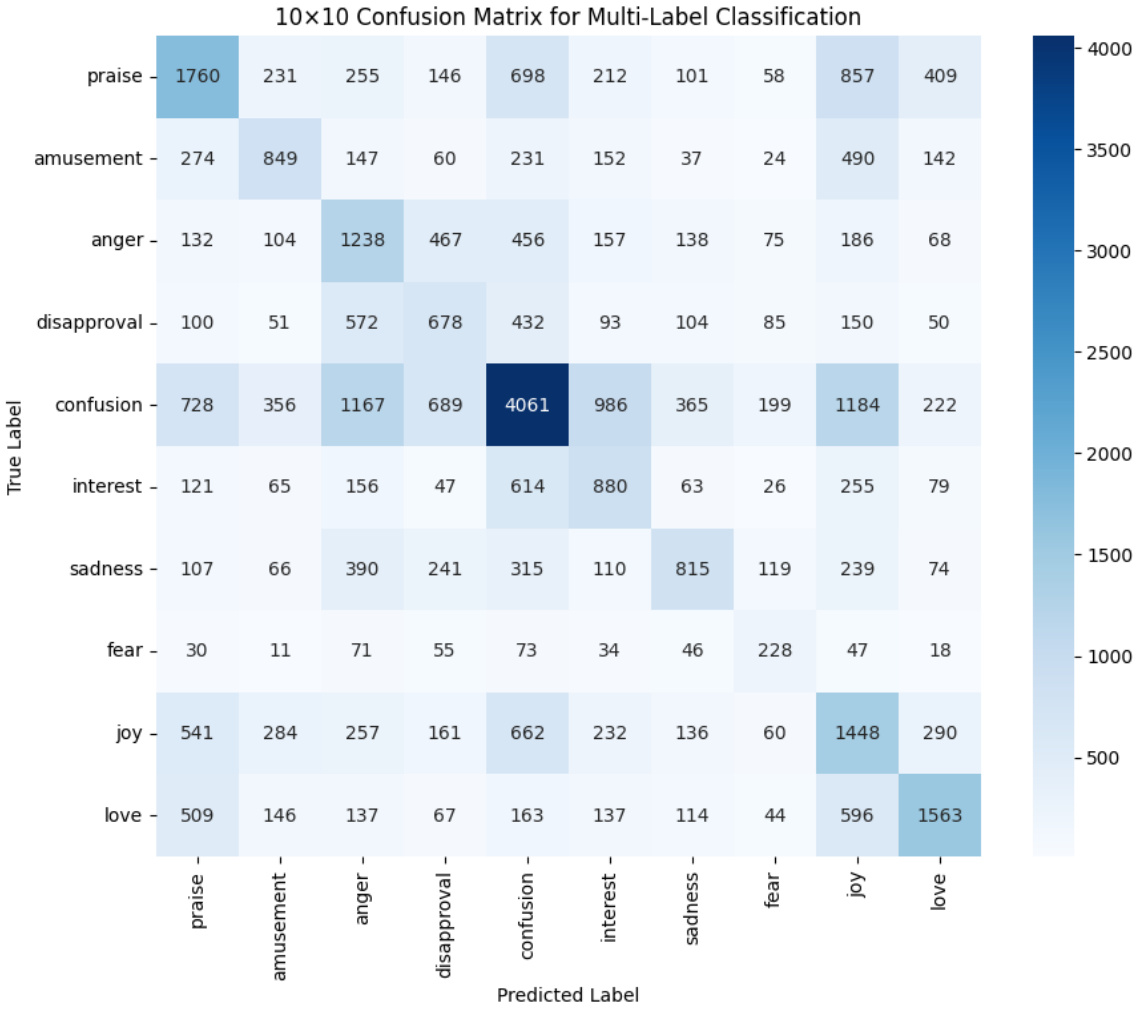
- برخی کلاس‌ها هنوز **F1-score پایینی** دارند، به ویژه **Fear و Disapproval.**
- **Recall معمولاً بالاتر از Precision است**، که نشان می‌دهد مدل تمایل دارد احساسات را بیشتر برچسب‌گذاری کند، اما همه پیش‌بینی‌ها دقیق نیستند.
- **امکان بهبود بیشتر با افزایش داده‌های متعادل، تنظیم آستانه‌های بهینه، و بهینه‌سازی وزن کلاس‌ها وجود دارد.**

Confusion matrix

مدل توانسته است احساسات مختلف را تشخیص دهد اما برخی از آن‌ها را با یکدیگر اشتباه می‌گیرد. احساساتی مانند ابهام، خشم و تحسین بیشترین میزان خطا را دارند. همچنین احساساتی مانند عشق و شادی به‌طور مکرر با یکدیگر اشتباه گرفته می‌شوند. مدل در تشخیص احساسات کم‌نمونه مانند ترس و علاقه عملکرد ضعیف‌تری دارد.

مشکلات اصلی مدل شامل عدم تمایز کافی بین احساسات مشابه، دشواری در تشخیص برخی برچسب‌های کم‌نمونه و همپوشانی بالا بین کلاس‌های نزدیک به هم است.

برای بهبود مدل، می‌توان داده‌های نامتوازن را متعادل کرد، آستانه‌های تصمیم‌گیری را تنظیم کرد، از مدل‌های پیشرفته‌تر استفاده کرد و ویژگی‌های ورودی بیشتری را در نظر گرفت. این روش‌ها می‌توانند دقت مدل را افزایش داده و پیش‌بینی‌های آن را بهبود ببخشند.



Samples

Sample Predictions:

Text: [NAME] goes to insults when you have nothing smart to say. goodbye

True Labels: [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

Predicted Labels: [0 0 1 0 0 0 0 0 0 0]

Text: lol, you dieded

True Labels: [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]

Predicted Labels: [0 1 0 0 0 0 0 0 0 0]

Text: Good for you, [NAME]! You deserved a nice drink to end a year where you made an impact in peoples lives! Cheers!

True Labels: [1. 0. 0. 0. 0. 0. 0. 0. 1. 0.]

Predicted Labels: [1 0 0 0 0 0 0 0 1 1]

Text: They come to your house and rob you at gun point. Be honest. What are some ways that this could be resolved?

True Labels: [0. 0. 0. 0. 1. 0. 0. 1. 0. 0.]

Predicted Labels: [0 0 1 0 1 1 0 0 0 0]

Text: It's to bad I live in a society that only wants my continual existence.

True Labels: [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]

Predicted Labels: [0 0 0 0 0 0 1 0 0 0]