در ابتدای کد، دو کتابخانه `emoji`و `contractions`نصب می‌شوند. سپس مجموعه‌ای از کتابخانه‌های مهم برای پردازش داده‌ها، مصورسازی، پردازش متن و مدل‌های یادگیری ماشین وارد کد می‌شوند.

- `pandas`و `numpy`برای پردازش داده‌های عددی و ساختاری.
- `json`و `pprint`برای نمایش خواناتر داده‌ها.
- `matplotlib`و `seaborn`برای مصورسازی داده‌ها.
- `emoji`، `re`، `contractions`برای پیش‌پردازش متن.
- `spacy`و `STOP_WORDS`برای پردازش زبان طبیعی و حذف کلمات پرتکرار.
- `sklearn`برای بردارسازی متن و مدل‌های یادگیری ماشین.

بارگذاری داده‌ها

داده‌های آموزشی و اعتبارسنجی از Google Drive بارگذاری می‌شوند. این داده‌ها شامل متونی با برچسب‌های احساسی هستند. دو فایل `EmoTrain.csv`و `EmoVal.csv`خوانده شده و اطلاعات اولیه آن‌ها نمایش داده می‌شود.

- تعداد نمونه‌های موجود در مجموعه داده‌ها محاسبه می‌شود.
- درصد هر مجموعه داده نسبت به کل داده‌ها محاسبه و نمایش داده می‌شود.

بارگذاری و نمایش دسته‌بندی احساسات

یک فایل متنی شامل دسته‌بندی‌های احساسی بارگذاری شده و لیست احساسات استخراج می‌شود. این دسته‌بندی‌ها در تحلیل داده‌ها استفاده می‌شوند.

ادغام داده‌های آموزشی و اعتبارسنجی

داده‌های `df_train`و `df_val`با یکدیگر ترکیب شده و به عنوان `df_all`ذخیره می‌شوند. سپس نمونه‌ای از این داده‌ها نمایش داده می‌شود.

تبدیل برچسب‌های احساسی به لیست‌های ایندکس

اگر ستون `GE_indices`در مجموعه داده وجود داشته باشد، مقادیر آن به لیست‌هایی از ایندکس‌ها تبدیل می‌شود. در غیر این صورت، پیامی جهت عدم وجود این ستون چاپ می‌شود.

نمایش نمونه‌ای از متن‌های داده‌شده

یک نمونه تصادفی از داده‌های متنی انتخاب شده و چاپ می‌شود. در صورتی که ستون مربوط به متن(`Text`) ، `text`یا (`Tweet`در داده‌ها موجود نباشد، پیامی جهت اطلاع‌رسانی نمایش داده می‌شود.

پیش‌پردازش متون

تابعی به نام `preprocess_corpus`تعریف می‌شود که چندین عملیات روی متون ورودی اعمال می‌کند:

۱. **افزودن فاصله بین کلمات و علائم نگارشی** برای بهبود تحلیل متن.
۲. **تبدیل ایموجی‌ها به متن** با استفاده از `emoji.demojize`.
۳. **گسترش اختصارات و مخففها** با استفاده از `contractions.fix`.
۴. **تبدیل متن به حروف کوچک** برای یکپارچگی پردازش.
۵. **تصحیح تایپوگرافی و کلمات پرکاربرد** مانند تغییر `ikr`به `i know right`.
۶. **حذف و جایگزینی تکرار حروف** برای کلماتی مانند `coooool`به `cool`.
۷. **جایگزینی ایموجی‌های متنی با معادل متنی آن‌ها** مانند تبدیل `:D`به `smiling_face`.
۸. **حذف نویسه‌های غیرحروفی و اعداد** برای بهبود پردازش متن.

این تابع روی کل داده‌های متنی اعمال شده و نتیجه در ستونی جدید با نام `Clean_text`ذخیره می‌شود.

تقسیم داده‌ها به مجموعه‌های آموزشی و اعتبارسنجی

تابعی برای تقسیم داده‌های پردازش‌شده به دو مجموعه `train_GE`و `val_GE`بر اساس اندازه اولیه مجموعه‌های آموزشی و اعتبارسنجی تعریف می‌شود. سپس شکل این مجموعه‌ها نمایش داده می‌شود.

بررسی تعداد برچسب‌های احساسی در هر نمونه

ستونی جدید با نام `Cardinality`ایجاد می‌شود که نشان می‌دهد هر نمونه از داده چند برچسب احساسی دارد. این توزیع نمایش داده شده و تعداد نمونه‌ها با تعداد برچسب‌های مختلف نمایش داده می‌شود.

مصورسازی توزیع برچسب‌های احساسی

۱. **نمودار تعداد احساسات در هر نمونه** با استفاده از `seaborn`نمایش داده می‌شود.
۲. **نمودار تعداد نمونه‌ها در هر احساس** برای نمایش توزیع احساسات در مجموعه داده.
۳. **نمایش درصد هر احساس در داده‌های آموزشی و اعتبارسنجی** با استفاده از نمودار ستونی.

تحلیل طول متون در هر دسته احساسی

تعداد کلمات موجود در متن‌های هر دسته احساسی محاسبه شده و میانگین آن‌ها در `df_length_GE`ذخیره می‌شود. سپس این داده‌ها مصورسازی شده و نموداری برای نمایش طول میانگین متون در هر دسته احساسی ترسیم می‌شود.

سپس ، داده‌های آموزشی و اعتبارسنجی که حاوی متونی با احساسات برچسب‌گذاری‌شده هستند، بارگذاری شده و پردازش‌های اولیه روی آن‌ها انجام می‌شود. مجموعه داده **train_GE** شامل داده‌های آموزشی و مجموعه **val_GE** شامل داده‌های اعتبارسنجی است. این دو مجموعه از فایل‌های ذخیره‌شده بارگیری شده و مجدداً به‌عنوان DataFrame های پانداس خوانده می‌شوند.

پس از خواندن داده‌ها، ستون **Clean_text** که شامل متون تمیزشده است، در کنار برچسب‌های مربوط به احساسات برای هر نمونه انتخاب می‌شود. سپس این داده‌ها در فایل‌های CSV جداگانه ذخیره می‌شوند تا در مراحل بعدی پردازش شوند.

پیش‌پردازش متن

پس از بارگذاری داده‌ها، نیاز است که متون موردنظر برای پردازش آماده شوند. در این مرحله، ستون **Clean_text** که شامل متن خام است، تمیز می‌شود. این فرآیند شامل حذف نویسه‌های غیرحروفی است، به این معنی که فقط کاراکترهای الفبایی و علامت زیرخط (_) در متن باقی می‌مانند و سایر نویسه‌ها حذف می‌شوند.

برای انجام این کار، ابتدا نوع داده‌های این ستون به رشته تبدیل شده و با استفاده از عبارات منظم(regex) ، تمام کاراکترهای غیرمجاز حذف می‌شوند. این مرحله باعث می‌شود که متن برای پردازش‌های بعدی مانند نشانه‌گذاری و ریشه‌یابی (lemmatization)آماده شود.

پس از آن، کتابخانه **spaCy** که یکی از قدرتمندترین ابزارهای پردازش زبان طبیعی (NLP) است، نصب و بارگذاری می‌شود. مدل زبانی **en_core_web_sm** از spaCy دانلود و مقداردهی اولیه می‌شود. این مدل شامل قابلیت‌هایی مانند نشانه‌گذاری (Tokenization)، شناسایی نام‌های خاص (Named Entity Recognition) و حذف کلمات توقف (Stop Words) است.

در این مرحله، مجموعه‌ای از ۱۰،۰۰۰ جمله نمونه از داده‌های آموزشی انتخاب شده و به کمک مدل زبانی spaCy پردازش می‌شوند. پردازش متن شامل شکستن جملات به کلمات (توکن‌ها)، حذف کلمات غیرضروری مانند "the" و"is" ، و استخراج ریشه‌ی کلمات (lemmatization) است. این کار باعث می‌شود که کلماتی با اشکال مختلف) مثلاً "running" و ("run" به یک فرم استاندارد تبدیل شوند.

در ادامه، پردازش برای ۲۰،۰۰۰ نمونه دیگر نیز تکرار می‌شود و پس از حذف کلمات توقف، کلمات باقی‌مانده در قالب یک رشته ذخیره می‌شوند. این فرآیند در نهایت برای اولین ۱۰،۰۰۰ نمونه داده نیز مجدداً اجرا شده و نتیجه در یک ستون جدید به نام **Clean_token** ذخیره می‌شود.

بردارسازی متن

پس از تمیز کردن متن و استخراج کلمات کلیدی، داده‌های متنی به قالب عددی تبدیل می‌شوند تا بتوان آن‌ها را به مدل یادگیری ماشین ارائه کرد. برای این منظور، از روش **TF-IDF (Term Frequency-Inverse Document Frequency)** استفاده می‌شود.

TF-IDFیکی از متداول‌ترین تکنیک‌های بردارسازی متن است که میزان اهمیت یک کلمه را در یک سند مشخص، نسبت به کل مجموعه اسناد تعیین می‌کند. در این مرحله، یک بردار TF-IDF با حداکثر ۱۰۰۰ ویژگی ایجاد شده و بر روی داده‌های متنی اعمال می‌شود.

ابتدا بردار TF-IDF روی مجموعه **train_GE**تنظیم شده و پس از تبدیل داده‌ها به بردارهای عددی، آن‌ها به آرایه‌های عددی تبدیل می‌شوند تا برای مدل‌سازی آماده شوند.

پردازش داده‌های برچسب‌گذاری‌شده

پس از تبدیل داده‌های متنی به بردارهای عددی، مجموعه‌ای از برچسب‌های مرتبط با احساسات استخراج می‌شود. این برچسب‌ها از فایل **emotions.txt**که شامل نام احساسات مختلف است، خوانده می‌شوند. سپس برای حذف هرگونه کاراکترهای اضافی مانند تب‌ها و فضای خالی، یک پردازش روی این لیست انجام می‌شود.

پس از آن، داده‌های **X_train**که شامل ویژگی‌های ورودی است، و **y_train**که شامل مقادیر برچسب‌های احساسات به‌صورت برداری است، استخراج می‌شوند. شکل این دو مجموعه بررسی شده تا از صحت داده‌ها اطمینان حاصل شود.

ایجاد پیش‌بینی‌های اولیه(Dummy Predictions)

برای ارزیابی اولیه مدل، یک مجموعه پیش‌بینی ساده ایجاد می‌شود که در آن تنها آخرین احساس (برچسب آخر) مقدار ۱ دارد و بقیه صفر هستند. این پیش‌بینی‌های اولیه به‌عنوان یک خط مبنا (Baseline) برای مقایسه عملکرد مدل‌های واقعی مورد استفاده قرار می‌گیرند.

ارزیابی مدل

تابعی به نام **model_eval**برای ارزیابی مدل تعریف می‌شود. این تابع معیارهای مختلفی از جمله **Precision**، **Recall**و **F1-score**را برای هر احساس محاسبه می‌کند.

این تابع برای هر برچسب موجود در مجموعه، عملکرد مدل را بررسی کرده و میزان دقت، بازخوانی و میانگین وزنی معیارها را محاسبه می‌کند. در انتها، میانگین کلی این مقادیر نیز محاسبه شده و در قالب یک DataFrame نمایش داده می‌شود.

آموزش مدل طبقه‌بندی RidgeClassifier

برای طبقه‌بندی احساسات در متن، از مدل **RidgeClassifier** که یک روش یادگیری ماشین بر اساس **رگرسیون ریدج** است، استفاده می‌شود. این مدل به دلیل استفاده از **تنظیم کلاس‌ها به‌صورت متوازن(class_weight='balanced')** ، برای مجموعه داده‌های نامتوازن مناسب است.

این مدل در قالب **MultiOutputClassifier** اجرا شده و روی ۱۰،۰۰۰ نمونه از داده‌های آموزشی آموزش داده می‌شود. پس از اتمام فرآیند آموزش، پیش‌بینی‌های مدل روی مجموعه آموزشی انجام شده و نتایج ارزیابی می‌شوند.

تابع پیش‌بینی احساسات برای متون جدید

در انتها، یک تابع به نام **predict_samples** تعریف می‌شود که امکان پردازش متون جدید و پیش‌بینی احساسات آن‌ها را فراهم می‌کند. این تابع شامل مراحل زیر است:

۱. دریافت یک نمونه متنی از کاربر.
۲. پیش‌پردازش متن با استفاده از توابع مشابهی که برای مجموعه داده‌های آموزشی انجام شد.
۳. تبدیل متن به بردار عددی.TF-IDF
۴. پیش‌بینی احساسات متن ورودی با استفاده از مدل آموزش‌دیده.
۵. تبدیل برچسب‌های پیش‌بینی‌شده به احساسات مربوطه.
۶. نمایش نتیجه نهایی.

به‌عنوان یک مثال، این تابع روی جمله **"Oh man, I forgot about eBay! I have some old textbooks I've been meaning to put on there too."** اجرا شده و احساسات مرتبط با این متن را برمی‌گرداند.

در این بخش، کتابخانه‌های موردنیاز نصب و در محیط برنامه بارگذاری می‌شوند. این کتابخانه‌ها شامل موارد زیر هستند:

transformers: برای استفاده از مدل BERT و توکنایزر آن.

emoji: برای پردازش ایموجی‌ها در متن (در این کد مستقیماً استفاده نشده است اما ممکن است در پیش‌پردازش داده مفید باشد).

contractions: برای بازنویسی و تبدیل اشکال مخفف) مثلاً "I'm" به ("I am" که به بهبود پردازش متن کمک می‌کند.

sklearn.metrics: شامل توابعی برای ارزیابی عملکرد مدل، مانند دقت (accuracy) و شاخص‌های precision, recall, f-score.

sklearn.utils.class_weight:برای محاسبه وزن کلاس‌ها در یادگیری مدل.

همچنین از tensorflow و keras برای پیاده‌سازی مدل شبکه عصبی استفاده می‌شود. این شامل:

تعریف مدل ورودی،لایه‌های دراپاوت (Dropout) برای جلوگیری از بیش‌برازش،بهینه‌ساز Adam برای تنظیم وزن‌ها،و مقداردهی اولیه با.TruncatedNormal

داده‌های متنی از دو مجموعه) train_GE آموزشی) و) val_GE اعتبارسنجی) جمع‌آوری شده و پردازش می‌شوند. ابتدا تمامی متن‌ها در یک مجموعه تجمیع شده و بیشترین طول جمله در این مجموعه محاسبه می‌شود:

این مقدار، طول حداکثری دنباله ورودی به مدل BERT را مشخص می‌کند، زیرا در مدل‌های پردازش زبان طبیعی، طول ورودی‌ها باید یکسان باشد.

در این مرحله، مدل BERT-base-uncased بارگیری شده و توکنایزر (Tokenizer) مربوطه برای تبدیل متن به عدد مقداردهی اولیه می‌شود:

BertConfigمشخصات مدل را تنظیم می‌کند.

BertTokenizerFastبرای تبدیل متن به بردارهای عددی استفاده می‌شود.

TFBertModelمدل اصلی BERT را بارگیری می‌کند.

در این مرحله، یک مدل سفارشی با استفاده از BERT ایجاد می‌شود ورودی‌ها شامل توکن‌های متن (input_ids) و ماسک توجه (attention_mask) هستند. یک لایه Lambda خروجی BERT را دریافت می‌کند.یک لایه Dropout برای جلوگیری از بیش‌برازش اضافه شده است. خروجی مدل یک لایه Dense با تابع سیگموید است که احتمال احساسات مختلف را تعیین می‌کند.

در این قسمت، داده‌های متنی با استفاده از توکنایزر BERT تبدیل به بردارهای عددی می‌شوند. همین فرآیند برای داده‌های اعتبارسنجی (val_GE) نیز انجام می‌شود.

سپس داده‌ها به TensorFlow Dataset تبدیل شده و در قالب batch پردازش می‌شوند. چون برخی احساسات ممکن است کمتر در داده‌ها مشاهده شوند، وزن کلاس‌ها به‌صورت متوازن محاسبه می‌شود.

یک تابع هزینه سفارشی بر اساس وزن کلاس‌ها تعریف شده و مدل با Adam optimizer مقداردهی می‌شود، مدل روی داده‌های آموزشی به مدت ۵ دوره (epochs) آموزش داده می‌شود

پس از آموزش، وزن‌های مدل ذخیره شده و در آینده می‌توان دوباره آن‌ها را بارگذاری کرد .مدل روی داده‌های آموزشی و اعتبارسنجی اجرا شده و نتایج پردازش می‌شوند سپس خروجی‌های احتمالی مدل به برچسب‌های نهایی تبدیل می‌شوند.

تابع `model_eval`برای ارزیابی کیفیت مدل استفاده می‌شود. هدف اصلی آن محاسبه معیارهای مختلف برای ارزیابی دقت پیش‌بینی مدل از جمله دقت(precision) ، بازخوانی (recall) و نمره F1 برای هر احساس (که در لیست `emotions` آورده شده است) است. در اینجا فرایند ارزیابی به تفصیل توضیح داده شده است:

ایجاد یک دیکشنری از احساسات :ابتدا یک دیکشنری به نام `idx2emotion`ساخته می‌شود که شامل ایندکس‌ها به عنوان کلید و نام احساسات به عنوان مقدار است. این دیکشنری برای دسترسی سریع به نام احساسات استفاده می‌شود.

محاسبه معیارهای دقت، بازخوانی و نمره F1 برای هر احساس :برای هر احساس، از تابع `precision_recall_fscore_support`برای محاسبه دقت، بازخوانی و نمره F1 استفاده می‌شود. این تابع برای هر احساس به طور جداگانه از برچسب‌های واقعی (`y_true`) و پیش‌بینی‌های مدل (`y_pred_labels`) محاسبات را انجام می‌دهد. این معیارها برای هر احساس ذخیره می‌شوند و سپس به دو رقم اعشاری گرد می‌شوند.

محاسبه میانگین‌های ماکرو و وزنی:

ماکرو :(Macro) این میانگین بدون در نظر گرفتن وزن کلاس‌ها (یعنی بدون توجه به تعداد نمونه‌ها در هر کلاس) محاسبه می‌شود.

وزنی :(Weighted) این میانگین با در نظر گرفتن تعداد نمونه‌ها در هر کلاس محاسبه می‌شود، بدین معنا که کلاس‌های پرنمونه وزن بیشتری خواهند داشت.

این میانگین‌ها نیز به همان ترتیب دقت، بازخوانی و نمره F1 محاسبه می‌شوند و به لیست‌های مربوطه اضافه می‌شوند.

ساخت یک :DataFrame پس از محاسبه تمامی این معیارها برای احساسات مختلف و میانگین‌ها، این نتایج در یک `DataFrame`از pandas ذخیره می‌شود که در آن ستون‌های"Precision" ، "Recall"و "F1" وجود دارد. ایندکس‌های این DataFrame نیز شامل نام احساسات و دو ردیف برای میانگین‌ها (ماکرو و وزنی) هستند.

در این بخش، هدف این است که بهترین آستانه (threshold) برای پیش‌بینی‌ها تعیین شود تا مدل بهترین عملکرد را در ارزیابی معیارهای مختلف داشته باشد. این کار به نحوی انجام می‌شود که نمره F1 ماکرو حداکثر شود.

محدوده آستانه‌ها :ابتدا یک محدوده از آستانه‌ها از ۰,۷ تا ۰,۹۹ به صورت گام به گام (با گام ۰,۰۱) تعریف می‌شود. این آستانه‌ها برای تعیین حد تصمیم‌گیری برای تبدیل احتمال‌های پیش‌بینی شده به برچسب‌های دودویی استفاده می‌شوند.

محاسبه پیش‌بینی‌ها و نمره F1 برای هر آستانه :برای هر آستانه در این محدوده، ابتدا احتمال پیش‌بینی شده (`y_pred_proba`)به برچسب‌های دودویی تبدیل می‌شود. سپس نمره F1 ماکرو با استفاده از تابع

precision_recall_fscore_supportمحاسبه می‌شود. این نمره به عنوان معیاری برای ارزیابی عملکرد آستانه در نظر گرفته می‌شود.

انتخاب بهترین آستانه :آستانه‌ای که بالاترین نمره F1 ماکرو را تولید کند به عنوان بهترین آستانه انتخاب می‌شود. علاوه بر این، پیش‌بینی‌ها بر اساس این آستانه جدید به‌روز می‌شوند.

بعد از تعیین بهترین آستانه، دو روش برای اصلاح پیش‌بینی‌های مدل پیاده‌سازی می‌شود:

اصلاح پیش‌بینی‌ها به‌گونه‌ای که حداقل یک پیش‌بینی برای هر نمونه وجود داشته باشد :اگر پیش‌بینی برای یک نمونه هیچ‌کدام از احساسات را برنگزید (یعنی تمام مقادیر پیش‌بینی شده برای آن نمونه صفر باشد)، این پیش‌بینی اصلاح می‌شود. در این اصلاح، برچسبی که بیشترین احتمال را دارد (از پیش‌بینی‌های احتمالی مدل) به آن نمونه اختصاص داده می‌شود. این روش برای جلوگیری از پیش‌بینی‌های بدون هیچ احساس خاصی استفاده می‌شود.

اصلاح پیش‌بینی‌ها با تخصیص برچسب "خنثی" به پیش‌بینی‌های خالی :اگر پیش‌بینی برای یک نمونه هیچ‌کدام از احساسات را شبیه‌سازی نکند، این پیش‌بینی به برچسب "خنثی" اختصاص داده می‌شود. این روش برای پیش‌بینی‌هایی است که نمی‌توانند احساس خاصی را تشخیص دهند و به نوعی آنها را در دسته "خنثی" قرار می‌دهد.

پس از اصلاح پیش‌بینی‌ها با استفاده از دو روش توضیح داده شده، مدل دوباره ارزیابی می‌شود تا تأثیر این اصلاحات بر نتایج بررسی شود. ارزیابی دوباره با استفاده از تابع model_evalانجام می‌شود.

در این بخش از کد، مدل پس از آموزش و اعمال اصلاحات، وزن‌های خود را ذخیره می‌کند. این کار با استفاده از model.save_weightsانجام می‌شود که فایل وزن‌های مدل را در مسیر خاصی ذخیره می‌کند. این فایل را می‌توان بعداً برای بارگذاری مجدد مدل و استفاده از آن برای پیش‌بینی‌ها استفاده کرد.

در این مرحله، برچسب‌های احساسات مدل که بر اساس دسته‌بندی GoEmotions است، به دسته‌بندی Ekman تبدیل می‌شوند. این تبدیل از آن جهت انجام می‌شود که دسته‌بندی Ekman شامل شش احساس اصلی است: "خشم"، "ترس"، "لذت"، "غم"، "تعجب" و "خنثی."

منطق تبدیل :در اینجا قوانینی برای تبدیل احساسات مختلف از GoEmotions به Ekman تعریف شده است:

اگر پیش‌بینی شامل احساسات "خشم" یا "ناامیدی" باشد، آن نمونه به احساس "خشم" تخصیص داده می‌شود.

اگر پیش‌بینی شامل احساس "ترس" باشد، آن نمونه به احساس "ترس" تخصیص داده می‌شود.

به همین ترتیب، مجموعه‌ای از احساسات از GoEmotions به احساسات مشابه در Ekman تبدیل می‌شوند.

تابع predict_samplesبرای پیش‌بینی احساسات جملات جدید طراحی شده است. در این تابع:

پیش‌پردازش متن :ابتدا جملات ورودی پیش‌پردازش می‌شوند. این مرحله شامل حذف کلمات زائد، اصلاح اشتباهات تایپی و سایر مراحل تمیز کردن داده‌ها است. سپس جملات تمیز شده به توکن‌های مدل تبدیل می‌شوند. این توکن‌ها به مدل داده می‌شوند تا پیش‌بینی‌های احتمال (proba) برای احساسات تولید شوند.

پیش‌بینی برچسب‌ها :پیش‌بینی‌های احتمال به برچسب‌های دودویی تبدیل می‌شوند با استفاده از آستانه بهینه که قبلاً انتخاب شده است.

ساخت DataFrame از نتایج :نتایج پیش‌بینی‌ها در قالب یک `DataFrame` نمایش داده می‌شوند که در آن هر جمله ورودی با احساسات پیش‌بینی شده‌اش نمایش داده می‌شود. این اطلاعات می‌تواند برای نمایش به کاربر یا پردازش‌های بعدی استفاده شود.

تمامی این مراحل به گونه‌ای طراحی شده‌اند که دقت مدل را بهبود بخشیده و پیش‌بینی‌های آن را دقیق‌تر و منطقی‌تر سازند. از انتخاب آستانه بهینه گرفته تا اصلاح پیش‌بینی‌ها و تبدیل دسته‌بندی‌ها، هر کدام از این بخش‌ها باعث می‌شوند که مدل نه تنها عملکرد بهتری داشته باشد، بلکه نتایج آن به صورت معناداری به واقعیت نزدیک‌تر شود.