توضیحات داده و نتایج:

مجموعه داده شامل ۳بخش train که برای آموزش و dev که برای تنظیم hyper parameterها و test که برای ارزیابی مدل هست.

مجموعه آموزشی دارای ۸۷۸۴ داده است که ۵۵۴۷ داده در دسته منفی و ۱۷۹۰ داده در دسته خنثی و ۱۴۴۷ داده در دسته مثبت قرار دارد که مجموعه داده نامتوازن هست اما با آموزش مدل SVM روی ویژگی‌های بدست آمده از chi-squared می‌توان دقت ۷۷.۷۳ درصد گرفت که precision و recall کلاس‌ها به شرح زیر است:

	precision	recall
1-	0.7958236658932715	0.93921139101862
0	0.6954436450839329	0.4559748427672956
1	0.7612359550561798	0.58154506437768 24

که پایین بودن میزان recall در کلاس‌های مثبت و خنثی تا حدی به علت نامتوازن بودن داده است و دیگر معیارها هم در فایل evaluation_result_3class.json در فایل‌های پروژه در دسترس است.

بعد از این مرحله با توجه به مقدار confidence کلاس‌ها را از -۱ تا ۱ به بازه -۳ تا ۳ بسط دادم به این شکل که اگر این مقدار بین ۰ تا ۰.۶ بود مقدار یک و اگر بین ۰.۶ تا ۱ بود مقدار ۲ و اگر دقیقا برابر با یک بود مقدار ۳ را درنظر می‌گرفتم و با ضرب این مقادیر در کلاس sentiment این مقدار به بازه -۳ تا ۳ تبدیل می‌شد و دلیل این بازه‌ها هم به خاطر چگال بودن confidenceها در این بازه‌ها است و بین بازه‌ها یک فاصله جدا کننده قرار داشت و در گام بعدی با استفاده از تعداد retweet تغییراتی را در این کلاس‌ها ایجاد کردم به این شکل که توییت‌هایی که تعداد retweetهای آن‌ها بیشتر یا مساوی ۲ بود یه درجه بالاتر بردم چون احتمالا وقتی دیگر افراد هم به نوعی با نظر این شخص موافق هستند می‌توان گفت اطمینان بیشتری نسبت به مثبت یا منفی بودن آن توییت داریم و توییت‌هایی هم که retweet نداشتند را یه درجه پایین‌تر آوردم با این کار داده‌ها مقداری نسبت به حالتی که فقط بازه را به -۳ تا ۳ تبدیل کرده بودیم متوازن‌تر می‌شوند اما همچنان داده‌ها نامتوازن هستند و با آموزش یک مدل SVM روی این داده‌ها می‌توان به دقت ۶۲.۸۴ درصد رسید که precision و recall کلاس‌ها به شرح زیر است:

	precision	recall
-3	0.0	0.0
-2	0.6424581005586593	0.9260614934114202
-1	0.5	0.006042296072507553
0	0.6208	0.610062893081761
1	0.5	0.011904761904761904
2	0.5613496932515337	0.6443661971830986
3	0.0	0.0

و پایین بودن precision یا recall در بعضی از کلاس‌ها به علت نامتوازن بودن داده در میان دسته‌های متفاوت مثبت یا دسته‌های متفاوت منفی است که به خیلی از داده‌ها برچسب دسته‌ای زده شده که در دسته‌های مثبت یا منفی داده بیشتری داشتند اما اگر از این دسته‌بند استفاده کنیم و بعد از دسته‌بندی ارزیابی را با توجه به برچسب‌های زده شده بر مبنای سه دسته مثبت و منفی و خنثی انجام بدهیم به دقت ۷۷.۷۶ درصد می‌رسیم که تقریبا معادل با دسته‌های مثبت و منفی و خنثی هست و دیگر معیارها هم در فایل evaluation_result_7class.json در فایل‌های پروژه در دسترس است.

توضیحات کد:

- فایل tokenizer: این فایل وظیفه ایجاد توکن‌ها را از روی یک متن دارد که برگرفته شده از لینک اما با تغییرات و شخصی‌سازی شده است.
  - تابع tokenize: این تابع یک متن را به عنوان ورودی دریافت می‌کند و توکن‌های بدست آمده از متن را به عنوان خروجی برمی‌گرداند.
- فایل classifier: این فایل شامل کلاس SentimentAnalyzer هست که مراحل خواندن و تمیزسازی داده تا آموزش و ارزیابی در این کلاس انجام می‌شود.
  - تابع load_dataset: این تابع آدرس یک مجموعه داده یا به عبارتی یک فایل csv را به عنوان ورودی دریافت می‌کند و بعد از خواندن داده‌ها sentiment هر کدام از داده‌ها را به نحوی به یک عدد تبدیل می‌کند و یک لیست به عنوان خروجی برمی‌گرداند که شامل متن توییت‌ها و عدد بدست آمده است.
  - تابع preprocess_tweet_text: این تابع متن یک توییت را به عنوان ورودی دریافت می‌کند و بعد از تمیزسازی متن را به عنوان خروجی برمی‌گرداند.
  - تابع get_vocabulary: این تابع متن تمیزشده توییت‌ها و sentiment مربوط به آن‌ها و یک متغیر که اندازه vocabulary نهایی را مشخص می‌کند به عنوان ورودی دریافت می‌کند و بعد محاسبه معیار chi-squared کلمات با بیشترین ارزش را در قالب یک لیست به عنوان vocabulary برمی‌گرداند.
  - تابع train: این تابع وظیفه آموزش مدل را دارد که برای این کار در ابتدا لازم است تا با استفاده از تابع get_vocabulary و مجموعه داده dev اندازه مناسبی را برای vocabulary بیابیم که این کار را توسط الگوریتم ژنتیک انجام می‌دهیم که در هر مرحله یک مجموعه عدد را به عنوان جمعیت درنظر می‌گیریم و روی کلمات بدست آمده مدلی را آموزش می‌دهیم و درنهایت با توجه به معیار accuracy جمعیت را بروز می‌کنیم و بعد از چند مرحله تولیدمثل بهترین نمونه را به عنوان اندازه vocabulary نهایی درنظر می‌گیریم.

- تابع **evaluate**: این تابع وظیفه دارد تا با استفاده مجموعه داده **test** مدل آموزش دیده در تابع **train** را ارزیابی کند که یک دیکشنری که حاوی معیارهای مختلف می‌شود را به عنوان خروجی برمی‌گرداند.
- تابع **predict**: این تابع وظیفه بدست آوردن دسته متناظر با یک توییت است که می‌توان از آن برای پاسخگویی به داده‌های جدید استفاده کرد و متن یک توییت را به عنوان ورودی دریافت می‌کند و در خروجی کلاس حدس زده شده را برمی‌گرداند.