توضیحات و نتایج:

دسته‌بند KNN وقتی K برابر با ۵ باشد و روی تمامی کلمات آموزش دیده باشد به دقت ۹۶ درصد می‌رسد و مقادیر دیگر معیارها در جدول زیر ذکر شده است:

	precision	recall	f1-score
ham	0.9554	0.9650	0.9602
spam	0.9646	0.9550	0.9598

وقتی K برابر با ۳ باشد دقت به ۹۵.۵۰ درصد می‌رسد و معیارهای دیگر به صورت زیر است:

	precision	recall	f1-score
ham	0.9550	0.9550	0.9550
spam	0.9646	0.9550	0.9550

وقتی K برابر با ۹ باشد دقت برابر با ۹۵.۲۵ درصد می‌شود و معیارهای دیگر به صورت زیر است:

	precision	recall	f1-score
ham	0.9415	0.9650	0.9531
spam	0.9641	0.9400	0.9519

و اگر K برابر با ۵ باشد و دسته‌بند را روی ۲۰۰تا کلمه با بیشترین امتیاز بر مبنای chi-squared باشد دقت به ۹۲.۷۵ درصد می‌رسد و معیارهای دیگر به صورت زیر است:

	precision	recall	f1-score
ham	0.9254	0.9300	0.9277
spam	0.9296	0.9250	0.9273

اگر K برابر با ۵ باشد و دسته‌بند را روی ۵۰۰تا کلمه با بیشترین امتیاز بر مبنای chi-squared باشد دقت به ۹۴.۵۰ درصد می‌رسد و معیارهای دیگر به صورت زیر است:

	precision	recall	f1-score
ham	0.9159	0.9800	0.9469
spam	0.9785	0.9100	0.9430

اگر K برابر با ۵ باشد و دسته‌بند را روی ۱۰۰۰تا کلمه با بیشترین امتیاز بر مبنای chi-squared باشد دقت به ۹۵.۲۵ درصد می‌رسد و معیارهای دیگر به صورت زیر است:

	precision	recall	f1-score
ham	0.9249	0.9850	0.9540
spam	0.9840	0.9200	0.9509

و اگر به جای دسته‌بند KNN از دسته‌بند NB روی تمامی کلمات استفاده کنیم به دقت ۹۶.۵۰ درصد می‌رسیم و معیارهای دیگر به صورت زیر است:

	precision	recall	f1-score
ham	0.9387	0.9950	0.9660
spam	0.9947	0.9350	0.9639

و درنهایت برای این مسئله با توجه به نتایج بالا می‌توان بر مبنای میزان اهمیت‌های متفاوت یکی را انتخاب کرد. مثلا در صورتی که دقت کلی مورد اهمیت باشد دسته‌بند NB بهترین نتیجه را دارد اما اگر بالا بودن هم مقدار precision و recall اهمیت بیشتری داشته باشد باید به سمت دسته‌بند KNN رفت.

توضیحات کد:

- فایل main.ipynb: این فایل حاوی روند کلی اجرا برنامه است که در ابتدا مجموعه داده را آماده و تمیزسازی می‌کند و در گام بعدی دسته‌بند را اجرا کرده و معیارهای ارزیابی متفاوت را نمایش می‌دهد.
- فایل data_cleaner.py: این فایل شامل کلاس TextCleaner است که وظیفه تمیزسازی و آماده کردن مجموعه داده را دارد که شامل توابع متنوعی است.
- فایل vectorizer.py: این فایل شامل کلاس TfidfVectorizer است که وظیفه تبدیل متون به بردارهای tf-idf را دارد.
  - تابع fit: این تابع با ورودی گرفتن مجموعه‌ای اسناد پارامترهای موردنیاز برای محاسبه بردارهای tf-idf هست را بدست آورد که در واقع باید مقادیر document frequency را به ازای کلمات مختلف محاسبه کند.
  - تابع transform: این تابع مجموعه‌ای از اسناد را به عنوان ورودی می‌گیرد و بعد از محاسبه term frequency کلمات حاضر در این اسناد و با استفاده از document frequencyهای بدست آمده در مرحله fit بردارهای tf-idf این اسناد را بسازد.
  - تابع fit_transform: این تابع مجموعه‌ای از اسناد را به عنوان ورودی می‌گیرد و ابتدا تابع fit و سپس تابع transform را صدا می‌زند و خروجی حاصل را برمی‌گرداند.
- فایل classifier: این فایل شامل کلاس KNNClassifier است که وظیفه دسته‌بندی اسناد را دارد و به عنوان ورودی مقدار K و تعداد ویژگی‌ها برای محاسبه بهترین کلمات بر مبنای chi-squared را می‌گیرد.
  - تابع fit: این تابع متن اسناد آموزشی و کلاس متناظر آن‌ها را به عنوان ورودی می‌گیرد و متون را به حالت برداری درمی‌آورد.
  - تابع predict: این تابع مجموعه‌ای از متن اسناد را به عنوان ورودی می‌گیرد و بعد محاسبه نزدیک‌ترین همسایه‌های هر کدام از متون کلاس با بیشترین وقوع را به آن سند اختصاص می‌دهد.