به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

پردازش زبان طبیعی

تمرین کامپیوتری شماره 2

نام و نام خانوادگی

امیرحسین دبیری اقدم

شماره دانشجویی

810197502

فروردین ماه 1401

فهرست

** تمام کدهای مربوط به به هرکدام از پیاده‌سازی‌های عادی (بدون tf-idf) و با tf-idf و توابع ذکر شده در گزارش، در فایل نوت‌بوک ضمیمه شده آمده است. توضیحات و نتایج کدها در این گزارش ارائه شده است.*

** نوت‌بوک‌ها از کولب گوگل استخراج شده و ممکن است برخی دستورات آن (مثل دستورات unzip و...) روی ویندوز قابل اجرا نباشد و نیاز است که روی کولب اجرا شود. البته بقیه موارد قابلیت اجرای مجدد روی ویندوز را دارد به شرطی که به صورت دستی فایل‌های دیتاست‌ها آنزیپ شده و در کنار نوت‌بوک قرار بگیرد.*

بخش 1 – Spam or Ham SMS Detection

پیش پردازش

پس از خواندن دیتاست از فایل csv، قبل از انجام هر پیش‌پردازشی ابتدا توزیع کلاس‌های مختلف را رسم می‌کنیم که در شکل 1 آمده است.

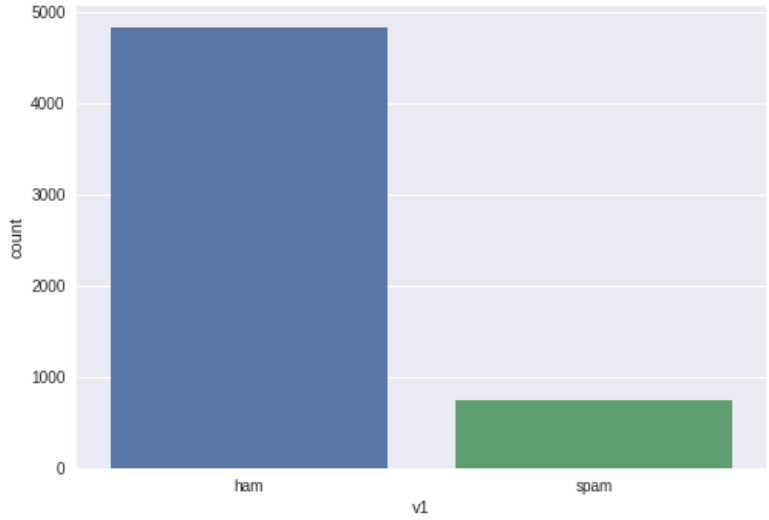


Figure 1

مشاهده می‌شود که دیتاست imbalance بوده و کلاس ham اکثریت نمونه‌ها را به خود اختصاص داده است که این موضوع ممکن است بر عملکرد طبقه‌بند تاثیر سو بگذارد. همچنین پس از بررسی دیتاست مشاهده شد که ستون‌های :Unnamed که حاوی اطلاعات نبودند وجود دارد که آن‌ها را هم دراپ کردیم.

در ادامه با کمک کتابخانه پرقدرت NLTK روش‌های مختلف پیش پردازش را پیاده کردیم.

حذف Stop Wordها:

تابع remove_stopwords از مجموعه stop wordهای زبان انگلیسی موجود در کتابخانه NLTK استفاده کرده و کلمات پرتکرار جمله ورودی که information زیادی هم نمی‌توان از آن‌ها کسب کرد را حذف می‌کند.

Stemming:

این تابع با استفاده از دو stemmer معروف زبان انگلیسی (porter و snowball) کلمات عبارت ورودی را stem می‌کند.

Lemmatizing:

این تابع ابتدا part of speech هر کلمه را با استفاده از تابع pos_tag کتابخانه NLTK بدست آورده و بعد با استفاده از wordnet_lemmatizer کلمات را با بن‌واژه (Lemma) آن‌ها جایگزین می‌کند.

در قسمت مربوط به ارزیابی، استفاده از ترکیب مختلف این توابع (استفاده یا عدم استفاده از آن‌ها) و تاثیرشان بر عملکرد طبقه‌بند Naïve Bayes بررسی شده است.

همانطورکه در صورت پروژه گفته شده بود، 20٪ داده برای تست و بقیه را برای آموزش در نظر می‌گیریم. (برای اینکه نتایج قابل بازتولید باشد random_state تابع train_test_split را برابر با شماره دانشجویی تنظیم کرده‌ام.)

استخراج ویژگی‌ها

برای بازنمایی متن به صورت عدد و استخراج ویژگی، از کلاس Countvectorizer موجود در کتابخانه قدرتمند Sklearn استفاده می‌کنیم و با استفاده از آن متن را به Bag Of Word تبدیل می‌کنیم. در حقیقت آنچه در نهایت حاصل می‌شود همان ماتریس document-term است که سطرهای آن عبارات موجود در داده آموزش و ستون‌های آن کلمات داده آموزش است و تعداد تکرار هرکلمه را در هر عبارت داده آموزش پیدا می‌کند و به صورت یک ماتریس sparse خروجی می‌دهد. به بیان دیگر هر سطر این ماتریس یک بردار به طول |V| (اندازه vocabulary که در اینجا برابر با 7277 است) است که یک بازنمایی برداری از هر کدام از جمله یا عبارات دیتاست است که به عنوان بردار ویژگی برای مدل Naïve Bayes استفاده خواهد شد. با Countvectorizer قبل از تولید بردارها، جملات را tokenize و tokenها را به lowercase تبدیل می‌کنیم. همچنین چون دیتاست مربوط به SMS است تعدادی کاراکتر دارای accent مثل å و... است که با دادن پارامتر 'trip_accents='ascii این موضوع را هم هندل می‌کند و همچنین punctuationها (علامت ! و...) را هم حذف می‌کنیم تا متن تمیزتر و نرمال‌تری در نهایت داشته باشیم.

با بررسی بیشتر دیتاست و نمونه‌های Spam مشاهده می‌شود که یکی از ویژگی‌های این نمونه‌ها وجود علامت ! و نیز استفاده زیاد از حروف uppercase است که با توجه به اینکه گفته شد Countvectorizer این ویژگی‌ها را از متن حذف می‌کند بنابراین تعداد علامت ! ها در متن و نیز تعداد حروف uppercase در متن را به صورت جدا به بردار ویژگی‌ها اضافه می‌کنیم. همچنین به طور متوسط طول جملات نمونه‌های spam طولانی‌تر است بنابراین یک ویژگی دیگر (Length) را هم به اینصورت در نظر می‌گیریم که اگر

تعداد کلمات یک عبارت از میانگین بیشتر بود مقدار این ویژگی یک و در غیراینصورت صفر در نظر گرفته می‌شود. پس این ویژگی را هم به بردار ویژگی‌های ایجاد شده اضافه می‌کنیم. (به صورت ستون به ماتریس document-term)

فرکانس 15 کلمه پر تکرار (بدون حذف stop wordها) در دیتاست UCIML در شکل زیر آمده است:

	Frequency
you	1811
to	1783
the	1085
and	795
it	794
in	727
is	712
my	600
me	597
your	565
for	555
that	507
call	501

2 **Figure**

برای بررسی بیشتر به جای استفاده مستقیم از فرکانس کلمات در بردارها، یکبار هم از روش tf-idf برای هر کلمه که به صورت زیر تعریف می‌شود استفاده می‌کنیم (به کمک TfidfTransformer موجود در کتابخانه Sklearn). تاثیر استفاده از این متد در عملکرد مدل Naïve Bayes در قسمت ارزیابی آمده است.

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t,d) & \text{if } \mathrm{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{idf}_i = \log\left(\frac{N}{\mathrm{df}_i}\right)$$

Total # of docs in collection

# of docs that have word i

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

استفاده از روش tf-idf باعث می‌شود که کلمات پرتکرار که در همه نمونه‌ها موجود هستند اما بار اطلاعاتی آنچنانی ندارند وزن کمتری در بردار ویژگی‌ها داشته باشند. این موضوع در شکل 3 مشاهده می‌شود که کلمات با فرکانس زیاد مقدار tf-idf آن‌ها کم است و برعکس.

	idf_weight	Frequency	tfidf
to	2.198306	1783.0	9.345332
you	2.243942	1811.0	9.554520
the	2.666675	1085.0	10.761182
and	2.925483	795.0	11.410458
in	2.927023	727.0	11.302799
it	2.958324	794.0	11.536934
is	3.007194	712.0	11.585154
me	3.164131	597.0	11.947677
my	3.208050	600.0	12.120499
your	3.208050	565.0	12.036759
for	3.218307	555.0	12.050283
call	3.279963	501.0	12.135330
have	3.295433	493.0	12.169528
that	3.315681	507.0	12.284624
of	3.317956	498.0	12.267245

3 **Figure**

برای بررسی بیشتر داده‌ها از نظر آماری می‌توان ماکسیمم، مینیمم و میانگین وزن‌های idf کلمات را بررسی کرد.

```
max = 8.7093
min = 2.1983
mean = 8.1302
```

از نزدیک بودن میانگین و ماکسیمم idf می‌توان نتیجه گرفت که پراکندگی اکثر کلمات در دیتاست یکسان است و اکثراً هم در تعداد کمی از نمونه‌ها ظاهر شده‌اند.

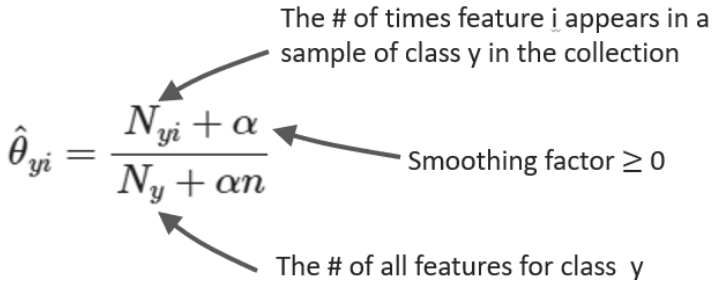
آموزش و ارزیابی طبقه‌بند Naïve Bayes

Naïve Bayes یکی از پرکاربردترین طبقه‌بندها در زمینه NLP هستند که براساس قانون بیز و فرض استقلال ویژگی‌ها (که لزوماً فرض صحیحی نیست) کار می‌کنند. در عمل دیده شده است که این طبقه‌بند

در برخی تسک‌های NLP عملکرد خوبی دارد و فرض استقلال باعث می‌شود که آموزش و همچنین طبقه‌بندی با استفاده از این طبقه‌بند سرعت بالایی داشته باشد.

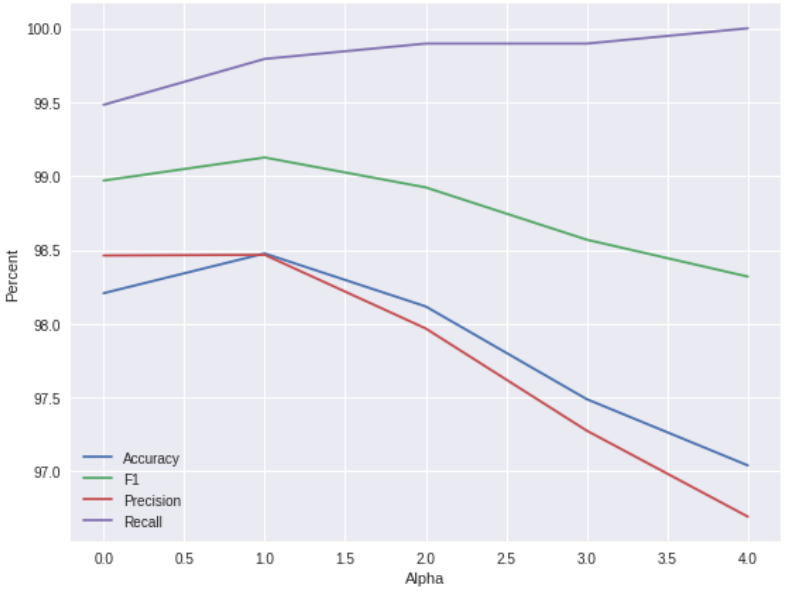
$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

در Naïve Bayes از رابطه فوق برای تخمین احتمال تعلق به کلاس y به شرط ویژگی‌های x_1 تا x_n استفاده می‌شود. در NLP x_1 تا x_n کلمات هر جمله هستند که به صورت BOW در نظر گرفته می‌شوند. همچنین برای تخمین توزیع این کلمات ($P(x_i|y)$) فرض می‌شود که این کلمات توزیع Multinomial دارند و با استفاده از عبارات موجود در داده آموزش این احتمالات را براساس MLE تخمین می‌زنیم. همچنین برای جلوگیری از وقوع احتمال صفر که باعث صفر شدن کل عبارت می‌شود از additive smoothing استفاده می‌شود.



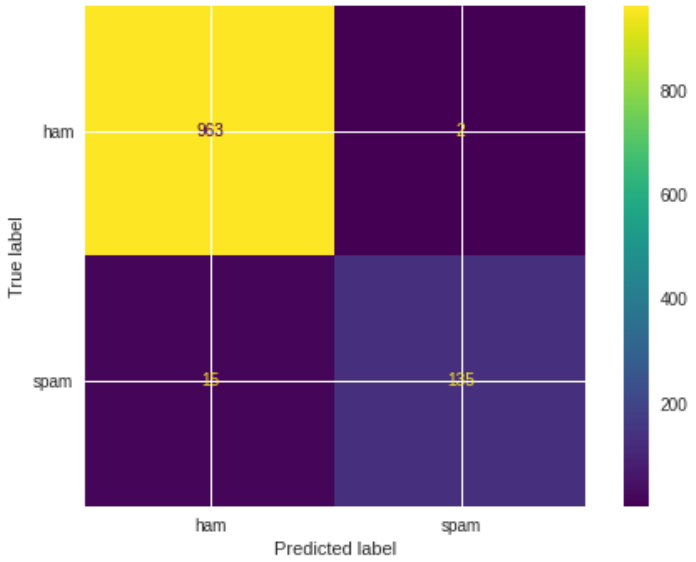
در ادامه با استفاده از بردارهایی که در قسمت پیش‌پردازش بدست آوردیم، با بررسی دقت طبقه‌بند در هنگام اعمال ترکیب مختلف پیش‌پردازش‌های مطرح شده، بهترین عملکرد مدل Naïve Bayes وقتی بود که از porter stemmer استفاده می‌شد (بدون حذف stop wordها) زیرا این stemming اندکی دقت مدل را افزایش می‌دا. استفاده از lemmatization در اینجا به دلیل اینکه متن مربوط به SMS بود و لزوماً فرم درست کلمات استفاده نشده بود، به اندازهporter stemmer نمی‌توانست دقت طبقه‌بند را افزایش دهد. همچنین چون جملات نسبتاً کوتاه بودند، حذف stop wordهای ارائه شده در NLTK نه تنها کمکی نمی‌کرد بلکه اندکی باعث کاهش دقت طبقه‌بند می‌شدند.

در ادامه طبقه‌بند را به ازای مقادیر α برابر با 0 تا 4 آموزش داده‌ایم (در ابتدا بدون استفاده از tf-idf) که نتایج آن در شکل زیر آمده است:



4 **Figure**

دیده می‌شود که مقدار $\alpha = 1$ مناسب‌ترین مقدار است پس به ازای این مقدار ماتریس کانفیوژن و... را رسم می‌کنیم:



	precision	recall	f1-score	support
ham	0.9847	0.9979	0.9913	965
spam	0.9854	0.9000	0.9408	150
accuracy			0.9848	1115
macro avg	0.9850	0.9490	0.9660	1115
weighted avg	0.9848	0.9848	0.9845	1115

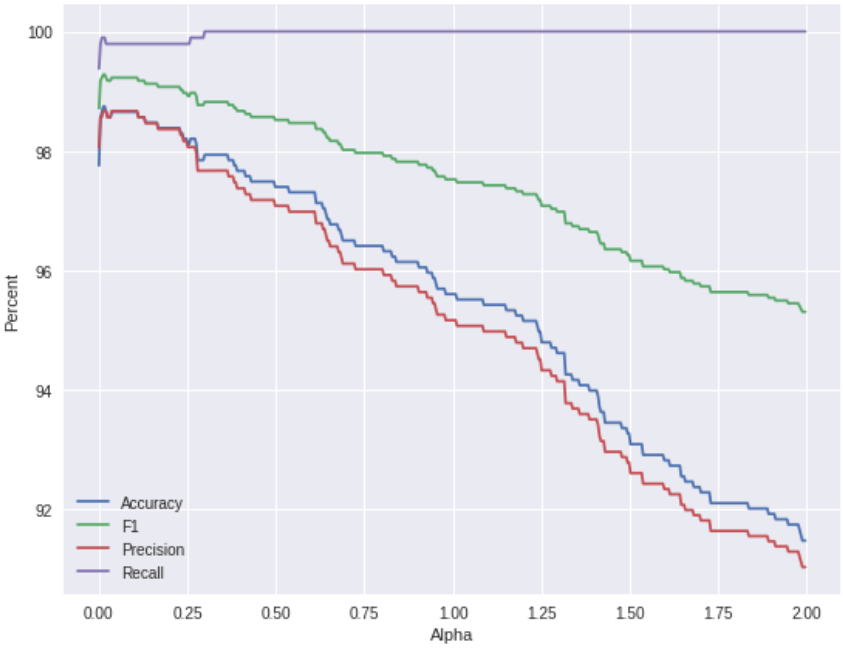
5 **Figure**

دقت طبقه‌بند در حدود 98.5 درصد و مقادیر precision،recall و f1 نیز 90٪ به بالا بوده که مقدار کاملاً قابل قبولی است. در ادامه برخی از مواردی که مدل در تخمین کلاس آن دچار خطا شده است آمده است:

#	Sentence	Predicted Class	Actual Class
1	u will switch your fone on dammit!!	spam	ham
2	gettin rdi to ship comp	spam	ham
3	you have 1 new message. pleas call 08718738034.	ham	spam
4	miss call alert. these number call but left no message. 07008009200	ham	spam
5	hi babe it jordan, how r u? im home from abroad and lonely, text me back if u wanna chat xxsp visionsms.com text stop to stopcost 150p 08712400603	ham	spam

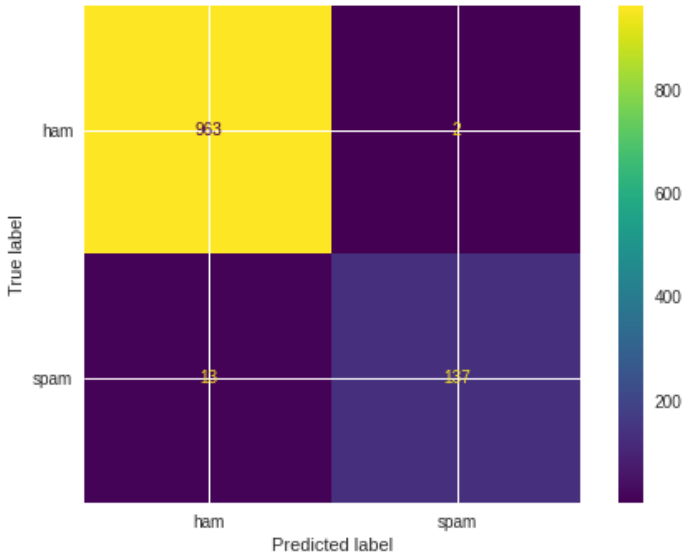
برای مثال به نظر می‌رسد طبقه‌بند به دلیل وجود دو علامت تعجب در عبارت 1 آن را به اشتباه spam تشخیص داده است. همچنین مواردی نظیر 3 و 4 نیز حتی برای یک انسان ممکن است ساده نباشد تا تشخیص دهد که spam است زیرا جمله‌بندی و کلمات مورد استفاده آن تا حد زیادی شبیه یک SMS معمول از طرف اپراتور تلفن است. اما به صورت کلی بدون در نظر گرفتن معدود موارد فوق الذکر عملکرد طبقه‌بند بسیار مناسب بوده و با توجه به ماتریس کانفیوژن می‌توان دید که از میان 150 جمله spam موجود در داده تست تنها 15 تا اشتباه طبقه‌بندی شده‌اند و همچنین از میان 965 نمونه غیر spam تنها 2 مورد اشتباه طبقه‌بندی شده‌اند.

حال یکبار هم طبقه‌بند را این بار با استفاده از بردارهایی که در آن‌ها به جای frequency از مقدار tf-idf استفاده شده است، آموزش می‌دهیم که عملکرد آن به ازای مقادیر α از 0 تا 2 در شکل زیر آمده است. (در اینجا چون مقادیر tf-idf اعشاری بودند مقدار α را هم به صورت اعشاری از 0 تا 2 با step=0.004 تغییر دادیم برعکس حالت قبل که مقادیر آن به صورت گسسته از 0 تا 4 در نظر گرفته بودیم زیرا در آنجا با تعداد که اعداد صحیح و گسسته بودند طرف بودیم و بنابراین این کار منطقی‌تر از اعشاری در نظر گرفتن α بود)



6 **Figure**

مشاهده می‌شود که مقدار $\alpha \approx 0.02$ مناسب‌ترین مقدار است پس به ازای این مقدار ماتریس کانفیوژن و... را رسم می‌کنیم:



	precision	recall	f1-score	support
ham	0.9867	0.9979	0.9923	965
spam	0.9856	0.9133	0.9481	150
accuracy			0.9865	1115
macro avg	0.9861	0.9556	0.9702	1115
weighted avg	0.9865	0.9865	0.9863	1115

7 **Figure**

با استفاده از tf-idf دقت طبقه‌بند در حدود 98.6 درصد و مقادیر recall،precision و f1 نیز 91٪ به بالا بوده که مقدار کاملاً قابل قبولی است و نسبت به حالت قبل نیز اندکی بهبود یافته است. در ادامه برخی از مواردی که مدل در تخمین کلاس آن دچار خطا شده است آمده است:

#	Sentence	Predicted Class	Actual Class
1	u will switch your fone on dammit!!	spam	ham
2	babe !!! i miiiiiiissssssssss you ! i need you !!! i crave you !!! :-(... geeee ... i'm so sad without you babe ... i love you ...	spam	ham
3	you have 1 new message. pleas call 08718738034.	ham	spam
4	miss call alert. these number call but left no message. 07008009200	ham	spam

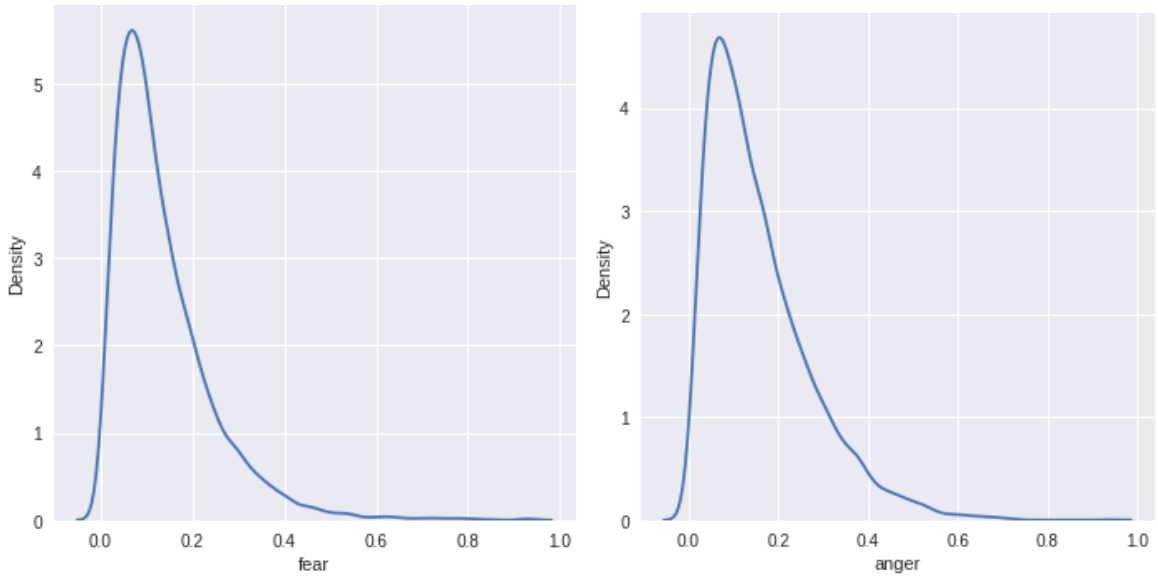
مشاهده می‌شود که برای مثال در اینجا هم احتمالاً طبقه‌بند به دلیل وجود علامت تعجب در عبارات 1 و 2 آن را به اشتباه spam تشخیص داده است. همچنین مواردی نظیر 3 و 4 هم مشابه قبل هستند و طبقه‌بند در این حالت هم این موارد را اشتباه کرده زیرا حتی برای یک انسان ممکن است ساده نباشد تا تشخیص دهد که spam هستند این جملات. مشابه حالت قبل وقتی از tf-idf استفاده شد به صورت کلی بدون در نظر گرفتن موارد فوق الذکر عملکرد طبقه‌بند بسیار مناسب بوده و با توجه به ماتریس کانفیوژن می‌توان دید که از میان 150 جمله spam موجود در داده تست تنها 13 تا اشتباه طبقه‌بندی شده‌اند (2 مورد کمتر از حالت بدون استفاده از tf-idf) و همچنین از میان 965 نمونه غیر spam تنها 2 مورد اشتباه طبقه‌بندی شده‌اند (مشابه حالت بدون استفاده از tf-idf).

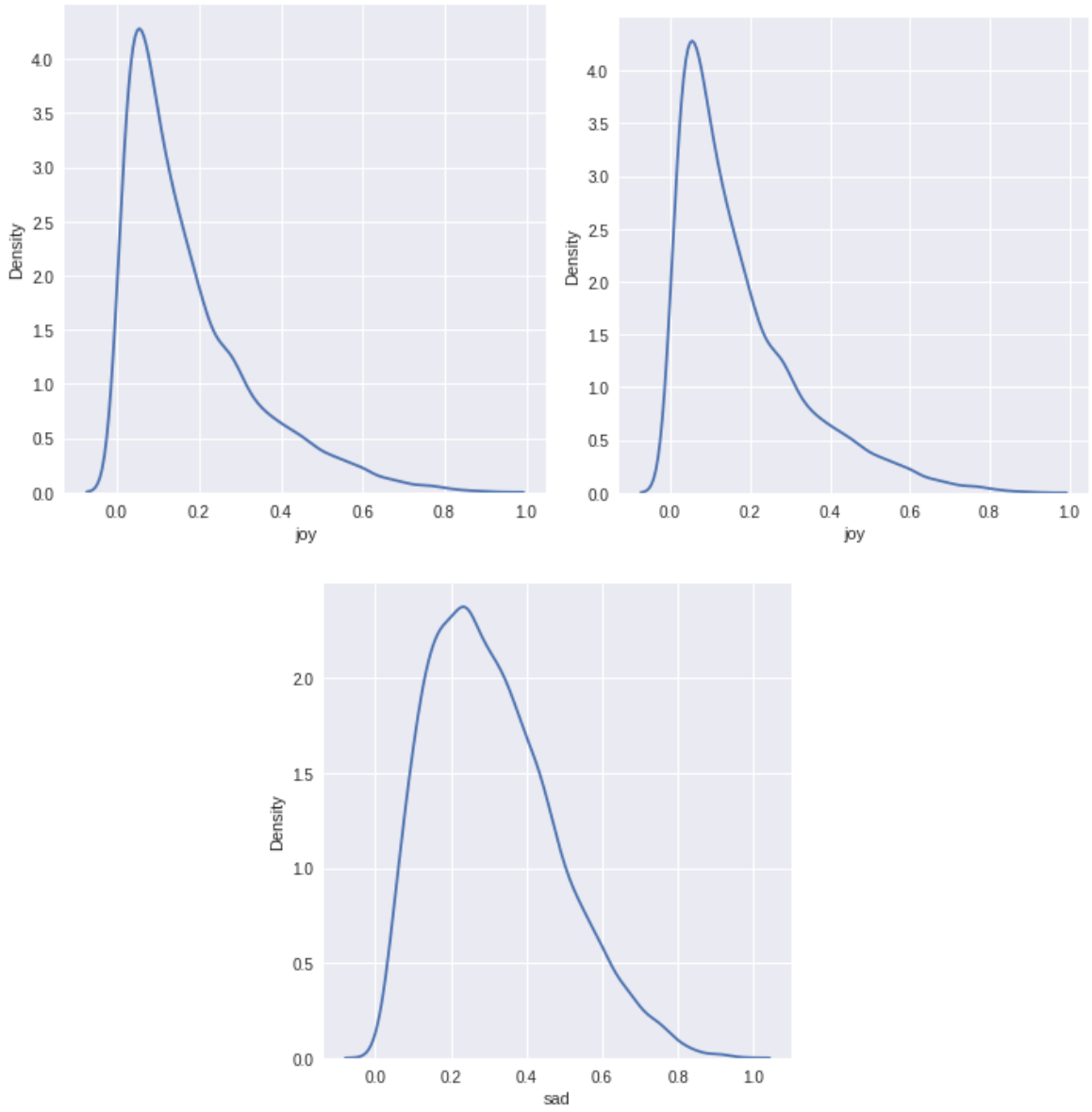
به طور خلاصه می‌توان ادعا کرد برای این دیتاست (UCIML) و این تسک (Spam Detection) طبقه‌بند Naïve Bayes به همراه BOW (با یا بدون tf-idf) عملکرد و دقت خیلی خوبی دارد.

بخش 2 – (Sentimental LIAR) Liar Detector

پیش‌پردازش‌ها

در این بخش چون علاوه بر ویژگی‌های متنی، ویژگی‌های دیگری نظیر fear،anger،sentiment و... نیز قابل استفاده بود لازم بود این ویژگی‌ها را ابتدا به نوعی پیش‌پردازش کنیم تا در طبقه‌بند Multinomial Naïve Bayes قابل استفاده باشند یعنی برای مثال مقادیر آن‌ها را به صورت صفر و یک (binary) دربیاوریم. ویژگی sentiment را که مقادیر Negative،Positive یا NAN داشتند را به این صورت در نظر می‌گیریم که اگر Positive بود آنگاه مقدار این ویژگی 1 و در غیر اینصورت 0 باشد. دیگر ویژگی‌ها (anger,fear ...) چون ماهیت پیوسته بین 0 تا 1 دارند، یک threshold می‌توان در نظر گرفت و اگر مقدار بیشتر از threshold بود آن ویژگی را یک و اگر کمتر بود صفر در نظر می‌گیریم. برای تعیین threshold با اندکی سعی و خطا و با توجه به pdf تخمین زده شده برای هر کدام از ویژگی‌ها (شکل‌های زیر) در نهایت مقدار 0.375 را در نظر گرفتم.





8 **Figure**

همچنین همانطور که در صورت پروژه گفته شده بود برای سادگی labelهای barely-true و pants fire را هم false در نظر گرفتیم و به طور مشابه labelهای half-true و mostly-true را هم true در نظر گرفتیم و البته بقیه ویژگی‌ها (ستون‌ها) را هم دراپ کردیم.

تمام موارد فوق الذکر در تابع preprocess_df پیاده‌سازی شده‌اند. بعد از اعمال این تابع روی هرکدام از dataframeهای آموزش و تست بررسی شد که سطر NAN نداشته باشیم.

در شکل زیر توزیع کلاس‌های دیتافریم آموزش آمده است که دیده می‌شود تا حد قابل قبولی balance است.

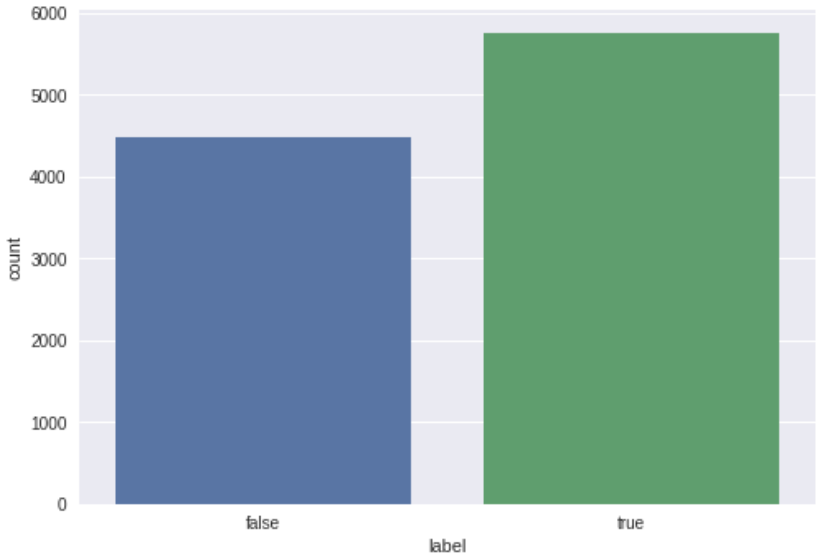


Figure 9

همچنین مشابه قبل در بخش مربوط به ارزیابی از ترکیب مختلف توابع preprocess متن (remove_stopwords,stemming,lemmatizing) و تاثیرشان بر عملکرد طبقه‌بند بررسی شده است.

استخراج ویژگی‌ها

مثل قسمت قبل برای بازنمایی متن به صورت عددی و استخراج ویژگی از کلاس Countvectorizer موجود در کتابخانه Sklearn استفاده می‌کنیم و با استفاده از آن متن را به Bag Of Word تبدیل می‌کنیم. (تولید ماتریس document-term است که سطرهای آن عبارات موجود در داده آموزش و ستون‌های آن کلمات داده آموزش است و تعداد تکرار هرکلمه را در هر عبارت داده آموزش پیدا می‌کند و به صورت یک ماتریس sparse خروجی می‌دهد). به بیان دیگر هر سطر این ماتریس یک بردار با طول |V| (اندازه vocabulary که در اینجا برابر با 10540 است) است که یک بازنمایی برداری از هر کدام از جمله یا عبارات دیتاست است که به عنوان بردار ویژگی برای مدل Naïve Bayes استفاده خواهد شد (همانطور که گفتیم در Countvectorizer قبل از تولید بردارها، جملات را tokenize و tokenها را به lowercase تبدیل می‌کنیم). همچنین ویژگی‌های sentiment و... را که به صورت باینری درآورده بودیم را هم به عنوان 5 ستون انتهایی ماتریس document-term اضافه می‌کنیم.

فرکانس 15 کلمه پر تکرار (بدون حذف stop wordها) در دیتاست Sentimental LIAR در زیر آمده است:

	Frequency
the	9777
in	5146
of	4827
to	4501
and	2851
say	2488
for	2126
that	1963
is	1787
on	1523
have	1314
ha	1290
state	1240
percent	1190
are	1134

Figure10

همچنین برای بررسی بیشتر به جای استفاده مستقیم از فرکانس کلمات در بردارها، یکبار هم از روش tf-idf برای هر کلمه استفاده می‌کنیم که نحوه، علت و اثر این کار در قسمت قبل گفته شد که در اینجا از ذکر مجدد آن پرهیز می‌کنیم. تاثیر استفاده از این متد در عملکرد مدل Naïve Bayes روی این دیتاست در قسمت ارزیابی آمده است. در ادامه مقادیر tf-idf محاسبه شده آمده است:

	idf_weight	Frequency	tfidf
the	1.517228	9777.0	15.457199
in	1.929269	5146.0	18.416754
of	1.998403	4827.0	18.948820
to	2.063861	4501.0	19.425175
and	2.431555	2851.0	21.775605
say	2.436062	2488.0	21.484201
for	2.692611	2126.0	23.323394
that	2.798916	1963.0	24.020939
is	2.836202	1787.0	24.074518
on	2.995267	1523.0	24.945894
ha	3.121436	1290.0	25.478405
have	3.137043	1314.0	25.663615
state	3.201140	1240.0	26.002430
than	3.278171	1103.0	26.244348
are	3.313092	1134.0	26.615749

11 **Figure**

برای بررسی بیشتر داده‌ها از نظر آماری می‌توان ماکسیمم، مینیمم و میانگین وزن‌های idf کلمات را بررسی کرد.

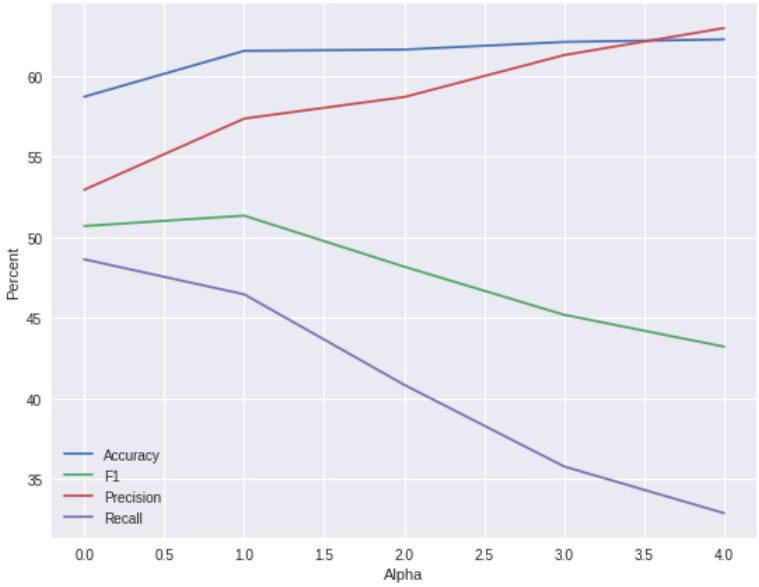
```
max = 9.5406
min = 1.5172
mean = 8.7355
```

در این دیتاست هم مشابه قبل از نزدیک بودن میانگین و ماکسیمم idf می‌توان نتیجه گرفت که پراکندگی اکثر کلمات در دیتاست یکسان است و اکثراً هم در تعداد کمی از نمونه‌ها ظاهر شده‌اند.

آموزش و ارزیابی طبقه‌بند

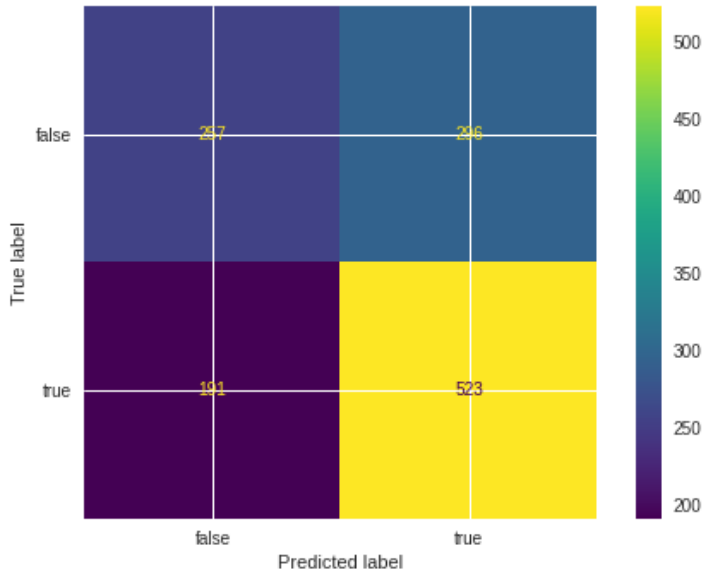
در ادامه با استفاده از بردارهایی که در قسمت پیش‌پردازش بدست آوردیم، با بررسی دقت طبقه‌بند در هنگام اعمال ترکیب مختلف پیش‌پردازش‌های مطرح شده، بهترین عملکرد مدل Naïve Bayes وقتی بود که از porter stemmer استفاده می‌شد (بدون حذف stop wordها) و زیرا اعمال آن اندکی باعث افزایش دقت مدل می‌شد. در اینجا هم به دلیل کوتاهی جملات و ساختار آن‌ها، استفاده از lemmatization و یا حذف stop wordهای ارائه شده در NLTK نه تنها بهبود ایجاد نمی‌کرد بلکه اندکی هم باعث کاهش دقت طبقه‌بند می‌شد.

در ادامه طبقه‌بند را به ازای مقادیر α برابر با 0 تا 4 آموزش داده‌ایم (در ابتدا بدون استفاده از tf-idf) که نتایج آن در شکل زیر آمده است:



12 **Figure**

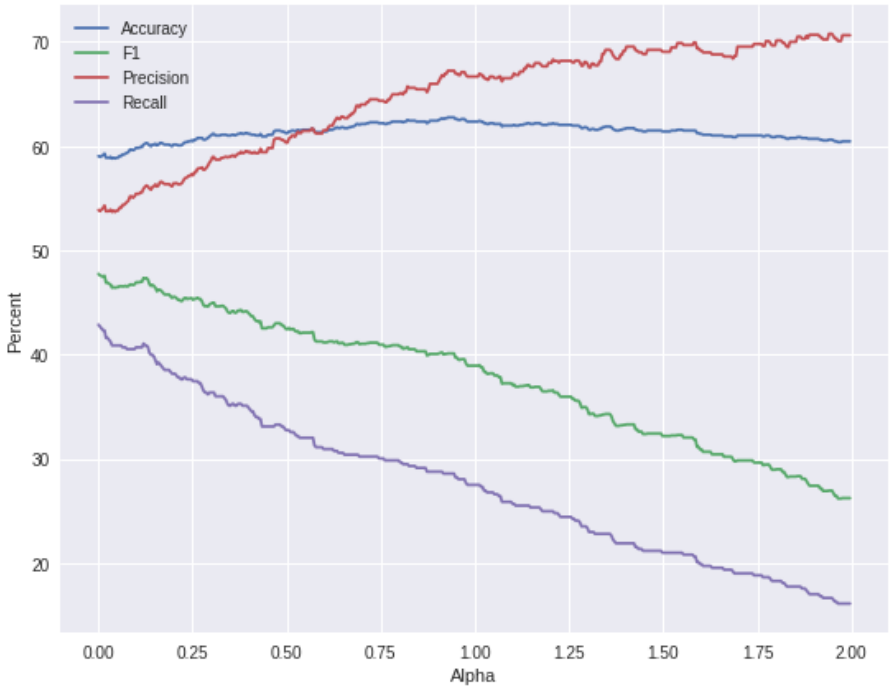
دیده می‌شود که مقدار $\alpha = 1$ مناسب‌ترین مقدار است پس به ازای این مقدار ماتریس کانفیوژن و... را رسم می‌کنیم:



	precision	recall	f1-score	support
false	0.5737	0.4647	0.5135	553
true	0.6386	0.7325	0.6823	714
accuracy			0.6156	1267
macro avg	0.6061	0.5986	0.5979	1267
weighted avg	0.6102	0.6156	0.6086	1267

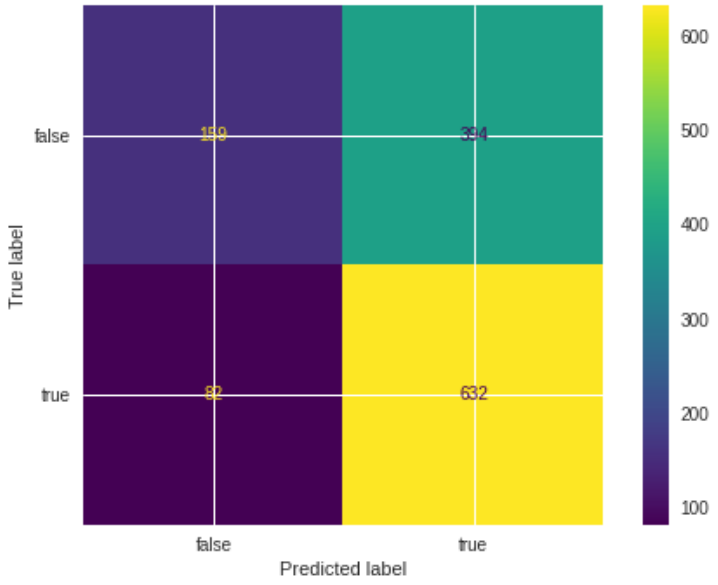
13 **Figure**

دقت طبقه‌بند در حدود 61.5 درصد و مقادیر recall،precision و f1 نیز حدود 46-73٪ بدست آمد که مقدار چندان خوبی نیست. در ادامه طبقه‌بند را با استفاده از بردارهایی که در آن‌ها به جای frequency از مقدار tf-idf استفاده شده است، آموزش می‌دهیم تا ببینیم عملکرد آن بهتر می‌شود یا خیر که به ازای مقادیر α از 0 تا 2 که در شکل زیر آمده است. (گفتیم که چون مقادیر tf-idf اعشاری بودند مقدار α را هم به صورت اعشاری از 0 تا 2 با step=0.004 تغییر دادیم)



14 **Figure**

مقدار $\alpha \approx 0.9$ در نظر می‌گیریم که نزدیک مقدار حالت قبل باشد تا بتوان نتیجه استفاده و عدم استفاده از tf-idf را بررسی کرد پس به ازای این مقدار ماتریس کانفیوژن و... را رسم می‌کنیم:



	precision	recall	f1-score	support
false	0.6598	0.2875	0.4005	553
true	0.6160	0.8852	0.7264	714
accuracy			0.6243	1267
macro avg	0.6379	0.5863	0.5635	1267
weighted avg	0.6351	0.6243	0.5842	1267

15 **Figure**

مشاهده می‌شود که در این حالت دقت اندکی بالاتر می‌رود اما مقدار recall مربوط به عبارات کلاس false حدود 20 درصد کاهش می‌یابد که اصلاً مناسب نیست. به عبارت دیگر استفاده از tf-idf در شرایط مشابه (α نزدیک به هم) با اینکه دقت مدل حدود 1 درصد افزایش می‌یابد اما در واقع مدل در این حالت بایاس پیدا می‌کند به سمت کلاس true یعنی بیشتر نمونه‌ها را متعلق به این کلاس تشخیص می‌دهد. (394 نمونه از کلاس false که 553 نمونه دارد، را به اشتباه از کلاس true تشخیص داده یعنی بیشتر از نیمی را اشتباه طبقه‌بندی کرده) از طرفی چون تعداد نمونه‌های کلاس true اندکی بیشتر از کلاس false است باعث می‌شود تا مدلی که بایاس به سمت کلاس true دارد دقتش اندکی بیشتر شود اما در عمل چون هدف شناسایی کلاس false است (جملات دروغ) چنین مدلی اصلاً مناسب نیست زیرا حتی نسبت به مدلی که رندوم طبقه‌بندی می‌کند عملکردش بدتر است. (مدل رندوم در حالت 2 کلاس به طور متوسط نیمی از نمونه‌ها را به اشتباه طبقه‌بندی می‌کند که یعنی حدود 275 نمونه را اشتباه طبقه‌بندی می‌کند که نسبت به 394 نمونه بهتر است پس مدل رندوم مدل بهتری است!)

بنابراین به صورت کلی نتیجه‌ای که می‌توان گرفت این است که برای این دیتاست و این تسک (شناسایی جملات دروغ) استفاده از Naïve Bayes و BOW ابداً مناسب نمی‌باشد (همچنین اعمال tf-idf هم تاثیر

سو روی عملکرد مدل دارد و به اصطلاح آن را دچار بایاس به سمت کلاس true می‌کند). اگرچه به طور کلی چنین تسکی برای انسان‌ها هم بسیار دشوار بوده و همچنین مقاله[1]ای که دیتاست Sentimental LIAR در آن معرفی شده است نیز با بهره گیری از مدل‌های دیپ لرنینگ بسیار پیچیده نظیر Bert و... توانسته به عنوان نتیجهٔ State of The Art، Accuracy در حدود 70٪ بگیرد که کمتر از 10٪ از نتیجه Naïve Bayes ما بهتر است و میانگین macro f1 در حدود 63٪ که اندکی از 59٪ بدست آمده در این تمرین بیشتر است؛ اما در کل دقت قابل قبول ولی نه چندان مناسبی است که دلیل آن این است که چنین تسکی بسیار دشوار می‌باشد و بنابراین نمی‌توان انتظار داشت که با استفاده از طبقه‌بند نه چندان پیچیده Naïve Bayes بتواند پیچیدگی این مسئله را به خوبی هندل کند.

[1] Sentimental LIAR: Extended Corpus and Deep Learning Models for Fake Claim Classification