Assignment 2

NLP

PREPARED BY

Mohsen Fayyaz - 810100524

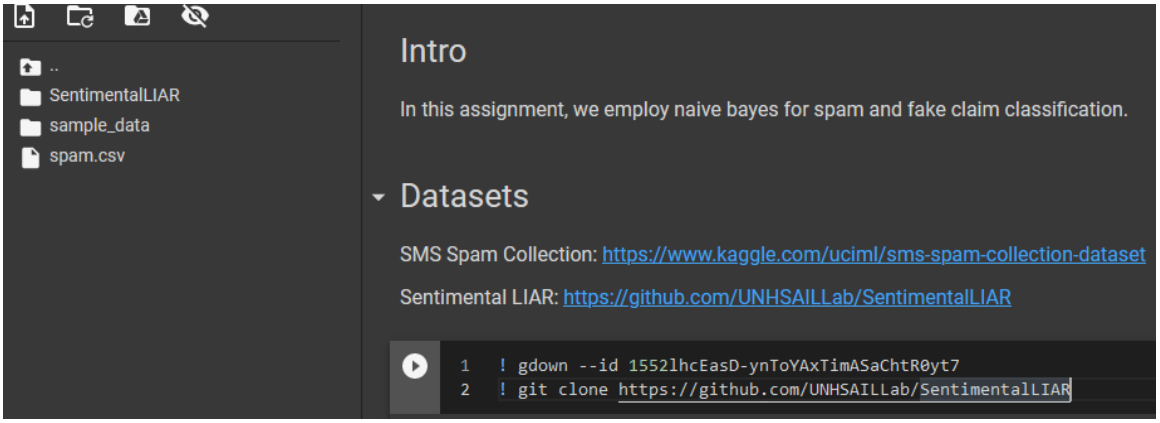
Spring 2022

فهرست مطالب

مقدمه

در انجام این تمرین کامپیوتری، با استفاده ازNaïve Bayes تلاش خواهیم کرد تا Classification را بر روی دو دیتاست انجام دهیم. در ابتدا بر روی یک دیتاست SMS های Spam و غیر Spam از هم تفکیک می کنیم و در ادامه در یک دیتاست دیگر یک ابزار طبقه بندی برای تشخیص دروغ و حقیقت آموزش می دهیم. هدف از انجام این تمرین آشنایی بیشتر با این نوع طبقه‌بند، تمرین استخراج ویژگی های مناسب و آشنایی با ابزارهای موجود است.

دیتاستها



پیش‌پردازش‌ها

برای این منظور یک کلاس جامع نوشته شد که انواع مختلف پیش‌پردازش در آن پیاده سازی شد.

```
from spellchecker import SpellChecker
import re


class Preprocessing:
    def __init__(self, techniques=["lowercase", "spellcheck"]):
        nltk.download('stopwords', quiet=True)
        nltk.download('punkt', quiet=True)
        nltk.download('words', quiet=True)
        nltk.download('wordnet', quiet=True)
        self.techniques = techniques
        self.techniques_map = {
            "lowercase": self.lowercase,
            "spellcheck": self.spellcheck,
            "stopwords": self.remove_stopwords,
            "lemmatization": self.lemmatization,
```

```
            "stemming": self.stemming,
            "alphanumeric": self.alphanumeric
        }

    def preprocess(self, text, verbose=True):
        words = nltk.tokenize.word_tokenize(text)
        for technique in self.techniques:
            words = self.techniques_map[technique](words)
            if verbose:
                print(technique.ljust(15), words)
        return words

    def lowercase(self, words):
        return [w.lower() for w in words]

    def spellcheck(self, words):
        spell = SpellChecker()
        return [spell.correction(w) for w in words]

    def remove_stopwords(self, words):
        return [w for w in words if w not in nltk.corpus.stopwords.words('english')]

    def lemmatization(self, words):
        lemmatizer = nltk.stem.WordNetLemmatizer()
        return [lemmatizer.lemmatize(w) for w in words]

    def stemming(self, words):
        stemmer_ss = nltk.stem.SnowballStemmer("english")
        return [stemmer_ss.stem(w) for w in words]

    def alphanumeric(self, words):
        return [re.sub(r'\W+', '', w) for w in words if re.sub(r'\W+', '', w) != '']


for t in Preprocessing().techniques_map.keys():
    Preprocessing(techniques=[t]).preprocess("Hello wrold to everybody running corpora.!")

lowercase       ['hello', 'wrold', 'to', 'everybody', 'running', 'corpora', '.', '!']
spellcheck      ['Hello', 'world', 'to', 'everybody', 'running', 'corporal', '.', '!']
stopwords       ['Hello', 'wrold', 'everybody', 'running', 'corpora', '.', '!']
lemmatization   ['Hello', 'wrold', 'to', 'everybody', 'running', 'corpus', '.', '!']
stemming        ['hello', 'wrold', 'to', 'everybodi', 'run', 'corpora', '.', '!']
alphanumeric    ['Hello', 'wrold', 'to', 'everybody', 'running', 'corpora']
```

این کد به شکلی زده شده که لیستی از تکنیکهای پیش‌پردازش می‌گیرد و به ترتیب آن‌ها را اجرا می‌کند. در بخش‌های بعدی از هر کدام استفاده می‌کنیم و دقت نهایی مدل را می‌سنجیم تا متوجه شویم در این مسئله خاص چه پیش پردازش هایی بهتر عمل می‌کنند.

استخراج ویژگی‌ها

در این بخش مهم‌ترین کار این است که دیتاست را بهتر بشناسیم. برای اینکار چند سطر از دیتاست ها را بررسی می‌کنیم.

ابتدا دیتاست اسپم را می‌بینیم.

index	v1	v2
0	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives around here though
5	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, â£1.50 to rcv
6	ham	Even my brother is not like to speak with me. They treat me like aids patent.
7	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
8	spam	WINNER!! As a valued network customer you have been selected to receivea â£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
9	spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
10	ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.
11	spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
12	spam	URGENT! You have won a 1 week FREE membership in our â£100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18
13	ham	I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing at all times.
14	ham	I HAVE A DATE ON SUNDAY WITH WILL!!

طبق بررسی این موارد می‌توان گفت اسپم‌ها اکثرا نکات متنی مثل وجود کلماتی مانند FREE و win و cache دارند که مربوط به موارد مالی می‌شود. مواردی مثل uppercase بودن هم می‌توانست در اسپم بیشتر باشد اما می‌بینیم که در ham نیز متنهای با uppercase داریم و این گزینه مناسبی نیست. به همین دلیل بهترین کار همان ویژگی‌های متنی است که از آن استفاده می‌کنیم.

سپس به سراغ دیتاست liar می‌رویم.

	label	statement	barely_true_counts	false_counts	half_true_counts	mostly_true_counts	pants_on_fire_counts	sentiment	sentiment_score	sentiment_magnitude	anger	fear	joy	disgust	sad
0	0	Says the Annies List political group supports ...	0.0	1.0	0.0	0.0	0.0	0	-0.5	0.5	0.121137	0.008926	0.026096	0.263479	0.531887
1	1	When did the decline of coal start? It started...	0.0	0.0	1.0	1.0	0.0	0	-0.4	0.8	0.095352	0.124566	0.191357	0.016999	0.102045
2	1	Hillary Clinton agrees with John McCain "by vo...	70.0	71.0	160.0	163.0	9.0	0	-0.3	0.3	0.039559	0.024162	0.500384	0.454228	0.052453
3	0	Health care reform legislation is likely to ma...	7.0	19.0	3.0	5.0	44.0	0	-0.3	0.3	0.004804	0.194674	0.375055	0.022509	0.383403
4	1	The economic turnaround started at the end of ...	15.0	9.0	20.0	19.0	2.0	-1	0.0	0.0	0.044237	0.215996	0.222402	0.045672	0.274343

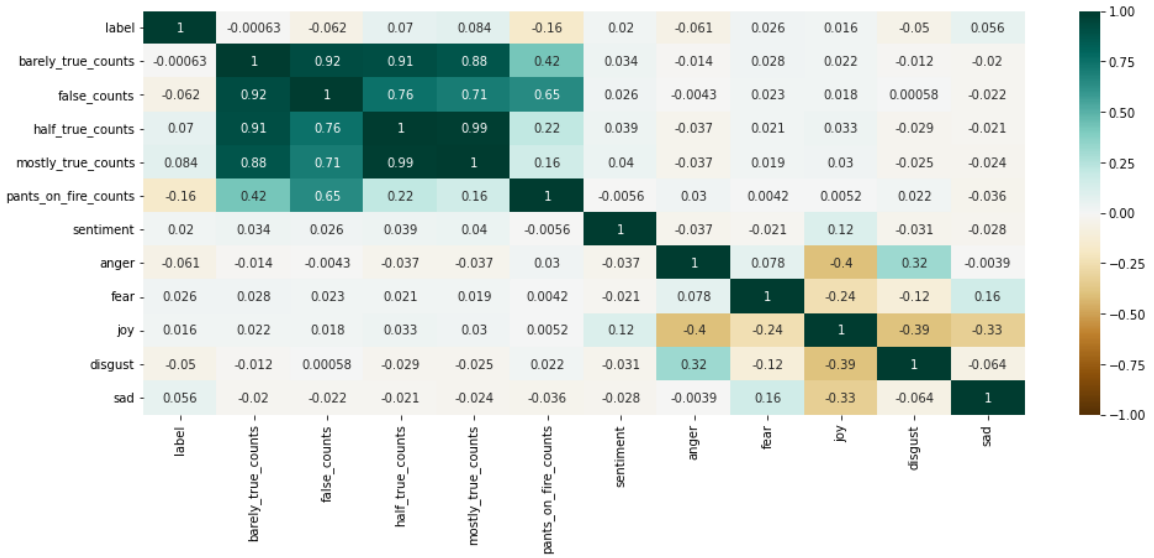
ستون label همانطور که گفته شده بود باینری شده و برای سادگی label های pants fire،barely true FALSE را دروغ و label هایTRUE ،mostly- true half-true را به عنوان حقیقت در نظر می‌گیریم.

(البته قابل ذکر است که در توضیحات[1] خود دیتاست آمده است که

In our dataset, the multi-class labeling of LIAR is converted to a binary annotation by changing half-true, false, barely-true and pants-fire labels to False, and the remaining labels to True.

اما طبق نوشته صورت پروژه ما half-true را True در نظر می‌گیریم.)

طبق چیزی که دیده می‌شود ستونهای قابل استفاده عددی هستند. برای اینکه بفهمیم چقدر واقعا این ویژگی‌ها پتانسیل کمک به مسئله را دارند ماتریس کوریلیشن را رسم می‌کنیم.



بین sentiment و ستونهای احساسات دیده می‌شود که anger و sad و disgust اندازه کوریلیشن بزرگتری با لیبل دارند و پس از آن fear و sentiment و joy قرار دارند. در کل این مقادیر بزرگ نیستند و نباید انتظار تغییر شگرفی داشت ولی در ادامه استفاده می‌کنیم و تاثیر آن را می‌بینیم.

[1] https://github.com/UNHSAILLab/SentimentalLIAR

برای مدیریت دیتاست‌ها و ویژگی‌هایشان کد زیر به صورت ارث بری پیاده سازی شد تا کمترین duplication در کد باشد و خوانایی افزایش یابد.

```
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn import naive_bayes, metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize


class DatasetHandler:
    def __init__(self, vectorizer, preprocessing_techniques):
        self.preprocessing = Preprocessing(preprocessing_techniques)
        if vectorizer == "tfidf":
            self.vectorizer = TfidfVectorizer(lowercase=False)
        elif vectorizer == "count":
            self.vectorizer = CountVectorizer(lowercase=False)
        else:
            raise f"Unknown vectorizer: {vectorizer}"
        self.label_encoder = LabelEncoder()

    def prepare_xy(self):
        self.x_train = self.preprocessing.preprocess_corpus(self.x_train)
        self.x_test = self.preprocessing.preprocess_corpus(self.x_test)
        self.x_train = self.vectorizer.fit_transform(self.x_train)
        self.x_test = self.vectorizer.transform(self.x_test)
        self.y_train = self.label_encoder.fit_transform(self.y_train)
        self.y_test = self.label_encoder.transform(self.y_test)
        print(self.label_encoder.classes_)

    def get_dataset(self):
        return {
            "x_train": self.x_train,
            "y_train": self.y_train,
            "x_test": self.x_test,
            "y_test": self.y_test
        }

class SpamDatasetHandler(DatasetHandler):
    def __init__(self, vectorizer="tfidf", preprocessing_techniques=[]):
        super().__init__(vectorizer, preprocessing_techniques)
        self.df = pd.read_csv("spam.csv", encoding='latin-1')
        self.df.dropna(axis=1, inplace=True)
        x = self.df["v2"].values
        y = self.df["v1"].values
        self.x_train, self.x_test, self.y_train, self.y_test = train_test_split(x, y, test_size=0.2)
        self.prepare_xy()

class LiarDatasetHandler(DatasetHandler):
    def __init__(self, vectorizer="tfidf", preprocessing_techniques=[], extra_cols=[]):
        super().__init__(vectorizer, preprocessing_techniques)
        def read_df(path):
            df = pd.read_csv(path)
            df.replace(['barely-true', 'false', 'pants-fire'], 'false', inplace=True)
            df.replace(['half-true', 'mostly-true', 'true'], 'true', inplace=True)
            x, y = df["statement"].values, df["label"].values
            return x, y

        def add_extra_cols(x, path):
            df = pd.read_csv(path)
            df["sentiment"] = df["sentiment"].factorize()[0] + 1
            # d = df[["anger", "fear", "joy", "disgust", "sad"]].values
            if len(extra_cols) > 0:
                d = df[extra_cols].values
                x = x.toarray()
                x = np.concatenate((x, d), axis=-1)
                x = normalize(x, axis=1)
            return x

        self.x_train, self.y_train = read_df("SentimentalLIAR/train_final.csv")
        self.x_test, self.y_test = read_df("SentimentalLIAR/test_final.csv")
        self.prepare_xy()
        self.x_train = add_extra_cols(self.x_train, "SentimentalLIAR/train_final.csv")
        self.x_test = add_extra_cols(self.x_test, "SentimentalLIAR/test_final.csv")
```

آموزش طبقه‌بندها

برای آموزش مدل‌ها، یک کلاس Trainer پیاده سازی شد.

```
class Trainer:
    def __init__(self, dataset_handler):
        self.dataset = dataset_handler.get_dataset()
        self.dataset_handler = dataset_handler

    def train(self):
        self.clf = naive_bayes.MultinomialNB()
        self.clf.fit(self.dataset["x_train"], self.dataset["y_train"])

    def eval(self):
        predictions = clf.predict(self.dataset["x_test"])
        return metrics.classification_report(self.dataset['y_test'],
                                             predictions,
                                             output_dict=True,
```

همانطور که در سوال خواسته شده، از یک مدل naive bayes برای آموزش استفاده می‌کنیم و در آخر کیفیت آن روی بخش تست داده را می‌سنجیم. تقسیم دیتاستها به آموزش و تست در بخش استخراج ویژگی‌ها که کلاس دیتاست زده شد انجام شده و اینجا کافیست یکی از آن کلاس به این مدل داده شود.

ارزیابی

برای ارزیابی همانطور که در کد آمده است از classification report استفاده می‌کنیم و برای گزارش مشابه مقاله[2] از accuracy و macro f1 استفاده می‌کنیم که در بخش‌ها بعدی نمایش داده می‌شود.

[2] https://arxiv.org/pdf/2009.01047.pdf

گزارش نهایی

1) تاثیر پیش‌پردازش‌ها

نتیجه اجرای پیش‌پردازش‌های مختلف روی دیتاست اسپم در ادامه آمده است.

SPAM Dataset

	change	accuracy	macro_f1
0		0.962332	0.908571
1	lowercase	0.963229	0.904631
2	stopwords	0.972197	0.932741
3	lemmatization	0.967713	0.922151
4	stemming	0.958744	0.908295
5	alphanumeric	0.961435	0.907926

همانطور که دیده می‌شود لزوما هر پیش‌پردازشی به دقت کمک نمی‌کند. مثلا اینکه فقط alphanumeric را نگه داریم و علائم نگارشی حذف شود قطعا اسپم هایی که نشانه خاصشان داشتن علائم زیاد مانند ! است ویژگی مهمشان را از دست می‌دهند. یا مثلا stemming چون لزوما همیشه درست کار نمی‌کند و خیلی وقتها ممکن است کلمات را اشتباه تغییر دهد (مخصوصا در دیتاست هایی که غلط املایی هم ممکن است) برای طبقه‌بندی مفید نخواهد بود. اما حذف stopwords به خوبی دیده می‌شود که تاثیر مثبتی دارد. این کلمات بار معنایی خاصی ندارند و حذفشان باعث می‌شود bayes به اشتباه از آنها به عنوان فیچر استفاده نکند و به این جهت generalization بهتری پیدا می‌کند. همچنین lemmatization مفیده بوده است که چون این کار با استفاده از دیکشنری انجام می‌شود فکر شده است و تغییرهای خوبی مانند مثالی که در بخشهای قبل آورده شده بود corpora به corpus تبدیل می‌شود که باعث یکریختی بیشتر کلمات و در نتیجه تجمیع تعداد در یک فیچر می‌شود که از sparse بودن فیچرها کمی می‌کاهد و بهبود تعمیم پذیری مدل را داریم. بنابراین از این دو پیش‌پردازش با هم استفاده می‌کنیم که نتیجه در ادامه آمده است.

SPAM Dataset

	change	accuracy	macro_f1
0		0.962332	0.908571
1	lowercase	0.963229	0.904631
2	stopwords	0.972197	0.932741
3	lemmatization	0.967713	0.922151
4	stemming	0.958744	0.908295
5	alphanumeric	0.961435	0.907926
6	stopwords + lemmatization	0.965919	0.923793
7	lemmatization + stopwords	0.968610	0.919755

اما نکته قابل توجه این است که این که دو پیش پردازش بهبود داشته اند، لزومی ندارد ترکیب آن دو نتیجه ای بهتر از آن دو بدهد. چیزی که اینجا هم دیده می‌شود. یکی از دلایل می‌تواند این باشد حذف stopwords و اعمال lemmatization بخش مشترکی هم داشته‌اند که اجرای هر دو آنها روی یکدیگر تاثیر می‌گذارد و این عدم استقلال باعث می‌شود نتوان از ترکیب این دو بهترین نتیجه را گرفت. البته که تمام این موارد مخصوص این دیتاست خاص است و روی هر دیتاست دیگر ممکن است نتیجه چیز دیگری باشد و تحلیل جامعی نمیتوان برای تمام دیتاست ها ارائه داد.

همین تحلیل را برای دیتاست liar نیز انجام می‌دهیم. نتایج به شکل زیر است.

Liar Dataset

	change	accuracy	macro_f1
0		0.599053	0.527516
1	lowercase	0.604578	0.535034
2	stopwords	0.599053	0.536018
3	lemmatization	0.602210	0.533251
4	stemming	0.610103	0.548195
5	alphanumeric	0.597474	0.524273

در این مورد می‌بینیم که stemming بهترین نتیجه را داشته است. علت این مورد این است که این دیتاست متنهای رسمی و با تغییرات کمتری دارد و همین شکل فرمال باعث می‌شود که stemming بتواند با rule های خود نتایج خوبی بگیرد و کلمات مشابه را یکی کند تا فیچرهای dense تری داشته باشیم. همچنین

در اینجا هم می‌بینیم که حذف علائم نگارشی اثر منفی دارد و فیچر مهمی برای تشخیص این دیتاست هستند. ضمنا باید توجه داشت که در اینجا lowercase کردن نیز تاثیر مثبتی داشته. در کل می‌توان گفت به علت فرمال بودن متنهای این دیتاست، اکثر پیش پردازش ها اثر مثبتی میتوانند داشته باشند چون دچار اشتباه خاصی نمیشوند. (به جز مواردی که توضیح داده شد).

در ادامه ترکیب پیش پردازش ها را هم تست می‌کنیم.

Liar Dataset

	change	accuracy	macro_f1
0		0.599053	0.527516
1	lowercase	0.604578	0.535034
2	stopwords	0.599053	0.536018
3	lemmatization	0.602210	0.533251
4	stemming	0.610103	0.548195
5	alphanumeric	0.597474	0.524273
6	all	0.607735	0.551109
7	all - alphanumeric	0.606946	0.551636

خوشبختانه در این مورد ترکیب توانست نسبتا بهتر از بقیه عمل کند و به همین دلیل برای این دیتاست از تمامی پیش پردازش ها استفاده می‌کنیم.

2) ویژگی‌های انتخاب شده و علت

ابتدا باید ذکر کنیم که برای ویژگی‌های متنی از tfidf استفاده شده است که پس از انجام، یک ماتریس اسپارس به ما می‌دهد. این مورد را طبق چیزی که در درس خواندیم بر شمردن خالی ترجیح دادیم چون تقسیم بر فرکانس داکیومنت هم انجام می‌شود و کلمات پر تکرار کم ارزش تر می‌شوند. مثلا کلمه ای مثل the که در خیلی از داکیومنت ها می‌آید، تاثیر خاصی در نتیجه طبقه‌بندی نخواهد داشت. (البته این مورد خاص در پیش پردازش حذف stopwords هم پاک می‌شود)

$$\text{tfidf}_{i,j} = \text{tf}_{i,j} \times \log\left(\frac{\text{N}}{\text{df}_i}\right)$$

$\text{tf}_{i,j}$ = total number of occurences of i in j
df_i = total number of documents (speeches) containing i
N = total number of documents (speeches)

در ابتدا بهترین نتیجه پیش پردازش دو دیتاست با فقط در نظر گرفتن ویژگی‌های متن را گزارش می‌کنیم.

نتیجه اسپم

	precision	recall	f1-score	support
ham	0.96	1.00	0.98	960
spam	1.00	0.72	0.83	155
accuracy			0.96	1115
macro avg	0.98	0.86	0.91	1115
weighted avg	0.96	0.96	0.96	1115

برای تغییر ویژگی در این دیتاست، به جای tfidf از count هم استفاده می‌کنیم که نتیجه به شکل زیر است.

	precision	recall	f1-score	support
ham	0.99	1.00	0.99	974
spam	0.98	0.90	0.94	141
accuracy			0.98	1115
macro avg	0.98	0.95	0.96	1115
weighted avg	0.98	0.98	0.98	1115

دیده می‌شود که با این تغییر توانستیم دقت بهتری بگیریم. اما باید توجه داشت که مدل قبلی precision برای spam برابر 1 بود. یعنی هیچ چیزی را به اشتباه spam تشخیص نداده بود. بنابراین طبق مسئله که خطای نوع اول یا دوم برای ما مهمتر است، حتی ممکن است مدل قبلی که دقت کلی کمتری دارد بهتر باشد و بسته به نیاز نهایی است. اما در کل مدل دوم بهتر شد.

همچنین باید توجه داشت که با توجه با unbalance بودن دیتاست و نسبت 974 و 141، اگر مدل تماما ham تشخیص می‌داد دقت حدود 0.87 می‌گرفت. بنابراین دقت 96 و 98 آموزش موفق بوده و مدل خوبی آموزش داده‌ایم.

سپس به سراغ دیتاست Liar می‌رویم و ابتدا بهترین مدل با پیش پردازش را گزارش می‌کنیم.

	precision	recall	f1-score	support
false	0.61	0.29	0.39	553
true	0.61	0.85	0.71	714
accuracy			0.61	1267
macro avg	0.61	0.57	0.55	1267
weighted avg	0.61	0.61	0.57	1267

و اگر از count استفاده کنیم نتیجه بهتری خواهیم داشت.

	precision	recall	f1-score	support
false	0.58	0.48	0.53	553
true	0.65	0.73	0.69	714
accuracy			0.62	1267
macro avg	0.61	0.61	0.61	1267
weighted avg	0.62	0.62	0.62	1267

همچنین اینها با Multinomial Naive Bayes انجام شده بود. در ادامه Complement Naive Bayes استفاده می‌کنیم که گفته شده برای دیتاست هایی که بالانس نیستند، مثل مسائل ما مفیدترند.

	precision	recall	f1-score	support
false	0.58	0.45	0.51	553
true	0.64	0.74	0.69	714
accuracy			0.62	1267
macro avg	0.61	0.60	0.60	1267
weighted avg	0.61	0.62	0.61	1267

در این مورد tfidf بهتر از count دیگر عمل می‌کند.

در ادامه به آزمودن ستونهای دیگر دیتا می‌پردازیم. همانطور که در بخشهای قبل ماتریس کوریلیشن نمایش داده شده بود سعی می‌کنیم رابطه آن و نتیجه این بخش را مقایسه کنیم. برای اضافه کردن این فیچرها از KBinsDiscretizer[3] استفاده می‌کنیم تا مقادیر پیوسته را به گسسته تبدیل کند تا برای مدل راحت تر شود. همانطور که در مقاله هم نوشته شده است، انتظار داریم این اطلاعات اضافه به تشخیص مدل کمک کند. زیرا انتظار داریم ادعاهای جعلی را بتوان در متن کوتاه با توجه به عبارات اغراق آمیز و احساسات قوی نشان داده شده در متن تشخیص داد.

	change	accuracy	macro_f1
2	anger	0.625099	0.615321
5	disgust	0.616417	0.606514
3	fear	0.617206	0.605747
0		0.615627	0.604340
1	sentiment	0.615627	0.604340
6	sad	0.615627	0.603678
4	joy	0.614838	0.603418

[3] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.KBinsDiscretizer.html

همانطور که دیده می‌شود، بیشتر از همه anger تاثیر مثبت داشته است. پس از آن به ترتیب disgust و fear و سپس sad و joy و sentiment می‌آیند. جالب توجه است که این ترتیب تا حدودی مشابه ماتریس کوریلیشن که در بخشهای قبل آورده شد است که anger و sad و disgust اندازه کوریلیشن بزرگتری با لیبل دارند و پس از آن fear و sentiment و joy قرار دارند. یعنی مثلا anger هم بیشترین کوریلیشن را داشت که می‌توانستیم حدس بزنیم نتیجه مدل را نیز بیشتر از بقیه بهبود دهد که همینطور شد.

در آخر ترکیب چند تا را هم امتحان می‌کنیم و

	change	accuracy	macro_f1
2	anger	0.625099	0.615321
7	all	0.620363	0.610259
8	anger+fear+disgust	0.619574	0.610149
5	disgust	0.616417	0.606514
3	fear	0.617206	0.605747
0		0.615627	0.604340
1	sentiment	0.615627	0.604340
6	sad	0.615627	0.603678
4	joy	0.614838	0.603418

و می‌بینیم که باز هم استفاده از anger می‌تواند نتیجه بهتری داشته باشد. باید توجه داشت که مدل naive bayes مدلی محدود است و روشی که در مقاله طی شده برای encode کردن متن با BERT و سپس استفاده از CNN و LR برای ترکیب احساسات بسیار پیچیده تر است و لزوما نباید انتظار نتایج یکسان داشت. اما در بهش بعدی بهترین نتیجه را مقایسه می‌کنیم.

3) نتایج و تحلیل

ابتدا بهترین نتیجه دیتاست اسپم از بخشهای قبل را می‌آوریم.

	precision	recall	f1-score	support
ham	0.99	0.99	0.99	979
spam	0.93	0.93	0.93	136
accuracy			0.98	1115
macro avg	0.96	0.96	0.96	1115
weighted avg	0.98	0.98	0.98	1115

و در ادامه بهترین نتیجه دیتاست Liar

	precision	recall	f1-score	support
false	0.57	0.55	0.56	553
true	0.66	0.68	0.67	714
accuracy			0.62	1267
macro avg	0.61	0.61	0.61	1267
weighted avg	0.62	0.62	0.62	1267

در مورد دیتاست اول همانطور که قبلا هم توضیح داده شد، نتیجه بسیار بهتر از مقدار تئوری است و توانستیم مدل بسیار خوبی آموزش دهیم. از مواردی که می‌توان در نتیجه با آن اشاره کرد این است که ham ها بهتر از spam ها شناسایی می‌شوند که نشان می‌دهد بعضی spam های هوشمندانه تر که توزیع مشابه تری به ham داشته اند از دست مدل رفته است. قطعا مدلهای پیچیده تر و بافتاری می‌توانند تا حدی این موارد را نیز تشخیص دهند.

در مورد دیتاست دوم نیز توضیحات مفصل داده شد در بخشهای قبل. فقط خوب است اشاره کنیم که بیشترین دقت مقاله 0.70 و بیشترین macro f1 آن برابر 0.65 گزارش شده است. با این توضیح که مدل آن مقاله مدل contextual یعنی BERT بوده و همچنین از لایه های CNN و Logistic Regression هم استفاده می‌کرده است درحالیکه ما از یک naive bayes استفاده کردیم که فرض توزیع خاص روی داده دارد و فرض استقلال و محدودیت های خیلی بیشتر نسبت به BERT دارد، باید گفت که دقت

0.62 و macro f1 برابر 0.61 واقعا مقادیر خوبی برای این مدل است و ویژگی ها به خوبی شناخته شدند و مدل به خوبی آموزش دیده است.