detect_youtube_spam

September 25, 2019

In [3]:
```
import pandas as pd
d = pd.read_csv("youtube-spam/Youtube01-Psy.csv")
```

In [4]:
```
d.tail()
```

Out[4]:

	COMMENT_ID	AUTHOR	DATE	CONTENT	CLASS
345	z13th1q4yzihf1bll23qxzpjeujterydj	Carmen Racasanu	2014-11-14T13:27:52	How can this have 2 billion views when there's...	0
346	z13fcn1wfpb5e51xe04chdxakpzgchyaxzo0k	diego mogrovejo	2014-11-14T13:28:08	I don't now why I'm watching this in 2014	0
347	z130zd5b3titudkoe04ccbeohojxuzppvbg	BlueYetiPlayz -Call Of Duty and More	2015-05-23T13:04:32	subscribe to me for call of duty vids and give...	1
348	z12he50arvrkivl5u04cctawgxzkjfsjcc4	Photo Editor	2015-06-05T14:14:48	hi guys please my android photo editor downloa...	1
349	z13vhvu54u3ewpp5h04ccb4zuoardrmjlyk0k	Ray Benich	2015-06-05T18:05:16	The first billion viewed this because they tho...	0

In [5]:
```
len(d.query('CLASS == 1'))
```

Out[5]: 175

In [6]:
```
len(d.query('CLASS == 0'))
```

Out[6]: 175

In [7]:
```
len(d)
```

Out[7]: 350

In [8]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
```

```
In [9]: dvec = vectorizer.fit_transform(d['CONTENT'])

In [11]: dvec

Out[11]: <350x1418 sparse matrix of type '<class 'numpy.int64'>'
                 with 4354 stored elements in Compressed Sparse Row format>

In [13]: print(d['CONTENT'][349])

The first billion viewed this because they thought it was really cool, the  other billion and a


In [16]: vectorizer.get_feature_names()

Out[16]: ['00',
          '000',
          '02',
          '034',
          '05',
          '08',
          '10',
          '100',
          '100000415527985',
          '10200253113705769',
          '1030',
          '1073741828',
          '11',
          '1111',
          '112720997191206369631',
          '12',
          '123',
          '124',
          '124923004',
          '126',
          '127',
          '13017194',
          '131338190916',
          '1340488',
          '1340489',
          '1340490',
          '1340491',
          '1340492',
          '1340493',
          '1340494',
          '1340499',
          '1340500',
          '1340502',
          '1340503',
          '1340504',
```

```
'1340517',
'1340518',
'1340519',
'1340520',
'1340521',
'1340522',
'1340523',
'1340524',
'134470083389909',
'1415297812',
'1495323920744243',
'1496241863981208',
'1496273723978022',
'1498561870415874',
'161620527267482',
'171183229277',
'19',
'19924',
'1firo',
'1m',
'20',
'2009',
'2012',
'2012bitches',
'2013',
'2014',
'201470069872822',
'2015',
'2017',
'210',
'23',
'24',
'24398',
'243a',
'279',
'29',
'2b',
'2billion',
'2x10',
'300',
'3000',
'313327',
'315',
'322',
'33',
'33gxrf',
'39',
'390875584405933',
```

```
'391725794320912',
'40beuutvu2zkxk4utgpz8k',
'4436607',
'4604617',
'48051',
'484',
'492',
'4shared',
'4snjqp',
'4th',
'50',
'521',
'5242575',
'5277478',
'5287',
'57',
'58',
'5800',
'5af506e1',
'5c2f',
'5million',
'5s',
'616375350',
'636',
'6381501',
'694',
'700',
'733634264',
'733949243353321',
'734237113324534',
'74',
'750',
'783',
'79',
'821',
'884',
'898',
'8bit',
'9082175',
'9107',
'9277547',
'950',
'969',
'9bzkp7q19f0',
'_0f9fa8aa',
'__killuminati94',
'_bzszz',
'_chris_cz',
```

```
'_fphgk5zllsvdqv0zuf0mb',
'_gibu',
'_o3h',
'_ry6f57sprnd2xv',
'_thqbeum69aqup1ih',
'_trksid',
'_vlczzrg8vgctlpsd9ongewhj8',
'aaaaaaa',
'able',
'about',
'above',
'absolutely',
'access',
'accessories',
'account',
'accounts',
'acn2g',
'acting',
'active',
'actor',
'actually',
'add',
'adding',
'admit',
'adsense',
'advice',
'affiliateid',
'after',
'again',
'ago',
'ahhh',
'al',
'alive',
'all',
'allot',
'allow',
'allways',
'almond',
'almost',
'alone',
'alot',
'also',
'am',
'amazing',
'amazon',
'america',
'amount',
'amp',
```

```
'an',
'ana',
'and',
'andrijamatf',
'android',
'angel',
'angry',
'animations',
'annoying',
'another',
'any',
'anybody',
'anymore',
'anyone',
'anyway',
'apocalypse',
'app',
'apparel',
'apparently',
'appreciate',
'apps',
'are',
'around',
'art',
'as',
'aseris',
'asia',
'asian',
'aspx',
'ass',
'at',
'attention',
'auburn',
'audiojungle',
'auditioning',
'avaaz',
'avoid',
'aw',
'away',
'aways',
'awesome',
'awesomeness',
'b00ecvf93g',
'baby',
'back',
'bad',
'band',
'bass',
```

```
'bd3721315',
'be',
'beautiful',
'because',
'been',
'before',
'behind',
'behold',
'beibs',
'being',
'believe',
'best',
'between',
'bf4',
'bieber',
'big',
'bighit',
'bilion',
'billion',
'billions',
'billon',
'binbox',
'bing',
'bit',
'bitch',
'blanc',
'block',
'blue',
'bomb',
'book',
'bother',
'bots',
'bottom',
'bowl',
'boyfriend',
'boys',
'bps',
'br',
'brain',
'brand',
'brew',
'bringing',
'brother',
'brotherhood',
'brothers',
'brt0u5',
'bs',
'bubblews',
```

```
'bucket',
'burned',
'but',
'butalabs',
'buy',
'by',
'c349',
'call',
'called',
'came',
'camera',
'can',
'cant',
'capitalized',
'car',
'card',
'cards',
'care',
'caroline',
'cash',
'cats',
'cd92db3f4',
'censor',
'certain',
'chacking',
'chainise',
'challenges',
'chance',
'chanel',
'change',
'chanicka',
'channel',
'channels',
'check',
'checked',
'checking',
'chhanel',
'chick',
'child',
'china',
'chinese',
'ching',
'chiptunes',
'christmas',
'chubbz',
'chuck',
'cirus',
'citizen',
```

```
'cjfoftxeba',
'cleaning',
'click',
'clicked',
'clip',
'close',
'clothes',
'clothing',
'co',
'code',
'codes',
'codytolleson',
'college',
'com',
'come',
'comentars',
'comes',
'coming',
'comment',
'comment_id',
'commenting',
'comments',
'commercial',
'company',
'complaining',
'completely',
'concerts',
'condition',
'confidence',
'confirmed',
'connected',
'constructive',
'content',
'cool',
'could',
'count',
'countries',
'cover',
'covers',
'craft',
'crap',
'crazy',
'crdits',
'crea',
'credit',
'crew',
'criticism',
'crop',
```

```
'cross',
'crowd',
'cs',
'csgo',
'curled',
'cute',
'cvhmklt',
'cxpzpgb',
'czfcxsn0jnq',
'da',
'daaaaaaaaaaannng',
'dad',
'dafuq',
'daily',
'dance',
'dances',
'dancing',
'day',
'dealing',
'dear',
'december',
'decent',
'dedicated',
'defuse',
'deserve',
'designs',
'details',
'dick',
'did',
'diddle',
'didn',
'die',
'difference',
'difficult',
'dinero',
'ding',
'direction',
'disappointed',
'discover',
'dislike',
'dislikes',
'dislikesssssssssssssssssssssssssssssss',
'divertenti',
'diys',
'dizzy',
'do',
'does',
'doesn',
```

```
'doesnt',
'doing',
'dolacz',
'dominate',
'don',
'donate',
'done',
'dont',
'download',
'downloading',
'dresses',
'dressprettyonce',
'driving',
'drone',
'drones',
'drop',
'drugs',
'drunk',
'dubstep',
'dumb',
'dunno',
'during',
'duty',
'dylan',
'earn',
'earth',
'easily',
'easy',
'easypromosapp',
'ebay',
'ede05ea397ca',
'editor',
'edm',
'eeccon',
'effects',
'effort',
'ehi',
'ejw9kvkoxdamqm808h5z',
'elevator',
'eliminate',
'emerson_zanol',
'end',
'english',
'enimen',
'enjoy',
'enough',
'enter',
'entertaining',
```

```
'entire',
'epic',
'equals',
'ermail',
'esteem',
'etc',
'eugenekalinin',
'eve',
'even',
'event',
'ever',
'every',
'everyday',
'everyone',
'everyones',
'everything',
'ex',
'excuse',
'expectations',
'expensive',
'experiments',
'explore',
'exposure',
'eyebrows',
'eyes',
'f7',
'fablife',
'face',
'facebook',
'facts',
'fail',
'fake',
'family',
'fanboys',
'fantastic',
'far',
'fashionable',
'fb',
'featured',
'feedback',
'feel',
'festival',
'few',
'fighting',
'figure',
'find',
'fire',
'fireball',
```

```
'first',
'fish',
'flipagram',
'flute',
'fly',
'follow',
'follower',
'football',
'for',
'forget',
'foto',
'found',
'four',
'freddy',
'free',
'freemyapps',
'french',
'friend',
'friends',
'frigea',
'from',
'fruity',
'fuck',
'fucked',
'fucken',
'fucking',
'fudairyqueen',
'fuego',
'fun',
'funnier',
'funny',
'funnytortspics',
'future',
'gabriel',
'game',
'games',
'gamestop',
'gaming',
'ganga',
'gangam',
'gangman',
'gangnam',
'gangnamstyle',
'gatti',
'gay',
'gbphotographygb',
'gcmforex',
'get',
```

```
'gets',
'getting',
'ghost',
'gift',
'girl',
'girls',
'give',
'giveaways',
'giver',
'glasses',
'go',
'goal',
'goes',
'gofundme',
'going',
'gonna',
'good',
'goodbye',
'google',
'gook',
'got',
'gotta',
'gotten',
'government',
'gp',
'grateful',
'great',
'greetings',
'group',
'grow',
'grwmps',
'gt',
'gta5',
'guardalo',
'gun',
'guy',
'guys',
'gvr7xg',
'gwar',
'hacked',
'hackers',
'hackfbaccountlive',
'had',
'haha',
'hahah',
'hahahahah',
'hair',
'half',
```

```
'halftime',
'halp',
'happy',
'hard',
'has',
'hassle',
'hate',
'haters',
'hav',
'have',
'having',
'he',
'head',
'headbutt',
'hear',
'heart',
'heck',
'hello',
'help',
'her',
'here',
'hey',
'hi',
'high',
'him',
'hip',
'his',
'history',
'hit',
'hits',
'hl',
'hole',
'holy',
'hop',
'hope',
'hoppa',
'hour',
'how',
'html',
'http',
'https',
'huge',
'huh',
'humanity',
'hunger',
'hw',
'hwang',
'hyperurl',
```

```
'hyuna',
'ice',
'id',
'idea',
'ie',
'if',
'ig',
'il',
'ill',
'illuminati',
'im',
'image2you',
'imagine',
'improve',
'in',
'including',
'incmedia',
'indiegogo',
'inspired',
'instagram',
'instagraml',
'internet',
'into',
'inviting',
'io',
'iphone',
'iq2',
'irl',
'is',
'isn',
'isnt',
'it',
'ithat',
'itm',
'its',
'itunes',
'itz',
'jackal',
'jackson',
'jae',
'james',
'jap',
'jaroadc',
'jb',
'jelly',
'jellyfish',
'jenny',
'job',
```

```
'join',
'joint2',
'joke',
'joking',
'jr',
'jtcjnmho',
'juice',
'julie',
'julien',
'juno',
'just',
'justin',
'juyk',
'jyp',
'k6a5xt',
'kamal',
'keep',
'keeps',
'keyword',
'kickstarter',
'kids',
'kidsmediausa',
'kidz',
'kind',
'kinda',
'king',
'knew',
'know',
'knows',
'kobyoshi02',
'kodysman',
'koean',
'kollektivet',
'korea',
'korean',
'koreans',
'kyle',
'l2649',
'l551h',
'la',
'lacked',
'lada',
'language',
'last',
'later',
'laugh',
'launchpad',
'lazy',
```

```
'leader',
'league',
'leah',
'learn',
'least',
'leave',
'left',
'let',
'lets',
'lexis',
'like',
'liked',
'likes',
'liking',
'limit',
'ling',
'link',
'linkbucks',
'listen',
'listening',
'listing',
'lists',
'little',
'littlebrother',
'live',
'll',
'lnuj',
'lol',
'long',
'look',
'looking',
'looks',
'lool',
'loool',
'loops',
'lordviperas',
'lot',
'lots',
'love',
'loved',
'loving',
'low',
'lt',
'lucas',
'lucks',
'luka1qmrhf',
'luther',
'lyrics',
```

```
'm1555',
'mabey',
'made',
'make',
'making',
'man',
'management',
'many',
'mario',
'marius',
'market',
'marketglory',
'markusmairhofer',
'martin',
'mathster',
'may',
'me',
'mean',
'mee',
'meets',
'members',
'memories',
'men',
'mercury',
'meselx',
'michael',
'miley',
'milions',
'million',
'millions',
'millioon',
'millisecond',
'millon',
'min',
'mind',
'mine',
'minecraft',
'minoo',
'minutes',
'miss',
'mix',
'model',
'mom',
'mon',
'money',
'monkey',
'monkeys',
'montages',
```

```
'month',
'months',
'more',
'morgage',
'moroccan',
'most',
'mother',
'moved',
'movie',
'mp3s',
'ms',
'mscalifornia95',
'msmarilynmiles',
'much',
'multiple',
'murdev',
'music',
'must',
'mute',
'my',
'myself',
'nail',
'nails',
'name',
'national',
'necks',
'need',
'neon',
'net',
'network',
'never',
'new',
'newest',
'news',
'next',
'nice',
'nicushorbboy',
'night',
'nike',
'ninja',
'no',
'non',
'norrus',
'not',
'notch',
'now',
'number',
'numbers',
```

```
'obsessed',
'odowd',
'of',
'off',
'offer',
'officialpsy',
'offset',
'offcal',
'often',
'old',
'older',
'olds',
'oldspice',
'olp_tab_refurbished',
'omg',
'on',
'once',
'oncueapparel',
'one',
'only',
'oppa',
'or',
'org',
'other',
'our',
'out',
'outfit',
'ovbiously',
'over',
'own',
'p3984',
'page',
'pages',
'paid',
'pal',
'pan',
'pants',
'part',
'partners',
'party',
'passed',
'pause',
'pay',
'pazzi',
'pdf',
'pe',
'peace',
'penis',
```

```
'people',
'peoples',
'perfect',
'perform',
'permpage',
'person',
'petition',
'petitions',
'phenomena',
'photo',
'photos',
'php',
'pictures',
'piece',
'pivot',
'pl',
'place',
'plan',
'planet',
'platform',
'play',
'please',
'plizz',
'pls',
'plus',
'plz',
'pnref',
'po',
'point',
'pop',
'popaegis',
'popular',
'population',
'populatoin',
'portfolio',
'post',
'posting',
'posts',
'pouring',
'pplease',
'pray',
'praying',
'prehistoric',
'premium',
'pretty',
'preview',
'pride',
'problem',
```

```
          'prod',
          'producer',
          'project',
          'projects',
          'promise',
          ...]

In [17]: dshuf = d.sample(frac=1)

In [18]: d_train = dshuf[:300]
         d_test = dshuf[300:]
         d_train_att = vectorizer.fit_transform(d_train['CONTENT']) #fit bag-of-words on train
         d_test_att = vectorizer.transform(d_test['CONTENT']) #reuse on testing set
         d_train_label = d_train['CLASS']
         d_test_label = d_test['CLASS']

In [19]: d_train_att

Out[19]: <300x1289 sparse matrix of type '<class 'numpy.int64'>'
                with 3725 stored elements in Compressed Sparse Row format>

In [20]: d_test_att

Out[20]: <50x1289 sparse matrix of type '<class 'numpy.int64'>'
                with 498 stored elements in Compressed Sparse Row format>

In [21]: from sklearn.ensemble import RandomForestClassifier
         clf = RandomForestClassifier(n_estimators=80)

In [22]: clf.fit(d_train_att, d_train_label)

Out[22]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=80, n_jobs=None,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False)

In [23]: clf.score(d_test_att, d_test_label)

Out[23]: 0.96

In [24]: from sklearn.metrics import confusion_matrix
         pred_labels = clf.predict(d_test_att)
         confusion_matrix(d_test_label, pred_labels)

Out[24]: array([[19,  0],
                [ 2, 29]], dtype=int64)
```

```
In [25]: from sklearn.model_selection import cross_val_score
         scores = cross_val_score(clf, d_train_att, d_train_label, cv=5)
         print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()*2))

Accuracy: 0.97 (+/- 0.06)


In [27]: d = pd.concat([pd.read_csv("youtube-spam/Youtube01-Psy.csv"),
                        pd.read_csv("youtube-spam/Youtube02-KatyPerry.csv"),
                        pd.read_csv("youtube-spam/Youtube03-LMFAO.csv"),
                        pd.read_csv("youtube-spam/Youtube04-Eminem.csv"),
                        pd.read_csv("youtube-spam/Youtube05-Shakira.csv")])

In [28]: len(d)

Out[28]: 1956

In [30]: len(d.query('CLASS == 1'))

Out[30]: 1005

In [31]: len(d.query('CLASS == 0'))

Out[31]: 951

In [32]: dshuf=d.sample(frac=1)
         d_content = dshuf['CONTENT']
         d_label = dshuf['CLASS']

In [33]: # set up a pipeline
         from sklearn.pipeline import Pipeline, make_pipeline
         pipeline = Pipeline([
             ('bag-of-words', CountVectorizer()),
             ('random forest', RandomForestClassifier()),
         ])
         pipeline

Out[33]: Pipeline(memory=None,
              steps=[('bag-of-words', CountVectorizer(analyzer='word', binary=False, decode_er
                 dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                 lowercase=True, max_df=1.0, max_features=None, min_df=1,
                 ngram_range=(1, 1), preprocessor=None, stop_words=None,
              ...obs=None,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False))])

In [34]: # We need to perform a couple of steps here with CountVecotizer followed by
         # the random forest. For this, we will use a feature in scikit-learn called a Pipeli
         # Pipeline is really convenient and will bring together two or more steps so that all
         # the steps are treated as on. So, we will build a pipeline with the bag of words, an
         # then use countVectorizer followed by the random forest clasifier. Then we will
         # print the pipeline, and it the steps requried
```

```
In [36]: make_pipeline(CountVectorizer(), RandomForestClassifier())

Out[36]: Pipeline(memory=None,
              steps=[('countvectorizer', CountVectorizer(analyzer='word', binary=False, decode
                 dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                 lowercase=True, max_df=1.0, max_features=None, min_df=1,
                 ngram_range=(1, 1), preprocessor=None, stop_words=None,
            ...obs=None,
                       oob_score=False, random_state=None, verbose=0,
                       warm_start=False))])

In [37]: pipeline.fit(d_content[:1500],d_label[:1500])

d:\dev\python\python36\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The de
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


Out[37]: Pipeline(memory=None,
              steps=[('bag-of-words', CountVectorizer(analyzer='word', binary=False, decode_er
                 dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                 lowercase=True, max_df=1.0, max_features=None, min_df=1,
                 ngram_range=(1, 1), preprocessor=None, stop_words=None,
              ...obs=None,
                       oob_score=False, random_state=None, verbose=0,
                       warm_start=False))])

In [38]: pipeline.score(d_content[1500:], d_label[1500:])

Out[38]: 0.9385964912280702

In [39]: pipeline.predict(["what a neat video!"])

Out[39]: array([0], dtype=int64)

In [40]: pipeline.predict(["plz subscribe to my channel"])

Out[40]: array([1], dtype=int64)

In [42]: scores = cross_val_score(pipeline, d_content, d_label, cv=5)
         print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()*2))

Accuracy: 0.94 (+/- 0.04)


In [45]: # Let's add TF-IDF
         from sklearn.feature_extraction.text import TfidfTransformer
         pipeline2 = make_pipeline(CountVectorizer(),
                                   TfidfTransformer(norm=None),
                                   RandomForestClassifier())
```

```
In [47]: scores = cross_val_score(pipeline2, d_content, d_label, cv=5)
         print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()*2))

d:\dev\python\python36\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The de
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
d:\dev\python\python36\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The de
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
d:\dev\python\python36\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The de
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


Accuracy: 0.94 (+/- 0.03)


d:\dev\python\python36\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The de
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
d:\dev\python\python36\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The de
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


In [48]: pipeline2.steps

Out[48]: [('countvectorizer',
           CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                   dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                   lowercase=True, max_df=1.0, max_features=None, min_df=1,
                   ngram_range=(1, 1), preprocessor=None, stop_words=None,
                   strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                   tokenizer=None, vocabulary=None)),
          ('tfidftransformer',
           TfidfTransformer(norm=None, smooth_idf=True, sublinear_tf=False, use_idf=True)),
          ('randomforestclassifier',
           RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
                       oob_score=False, random_state=None, verbose=0,
                       warm_start=False))]

In [56]: parameters = {
             'countvectorizer__max_features':(None, 1000, 2000),
             'countvectorizer__ngram_range':((1,1),(1,2)),
             'countvectorizer__stop_words':('english', None),
             'tfidftransformer__use_idf': (True, False),
             'randomforestclassifier__n_estimators':(20,50,100)
         }
         from sklearn.model_selection import GridSearchCV
         grid_search = GridSearchCV(pipeline2, parameters, n_jobs=-1, verbose=1)
```

```
In [57]: grid_search.fit(d_content, d_label)

d:\dev\python\python36\lib\site-packages\sklearn\model_selection\_split.py:2053: FutureWarning
  warnings.warn(CV_WARNING, FutureWarning)
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.


Fitting 3 folds for each of 72 candidates, totalling 216 fits


[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    5.9s
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed:   15.2s
[Parallel(n_jobs=-1)]: Done 216 out of 216 | elapsed:   17.3s finished


Out[57]: GridSearchCV(cv='warn', error_score='raise-deprecating',
               estimator=Pipeline(memory=None,
              steps=[('countvectorizer', CountVectorizer(analyzer='word', binary=False, decode
                 dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                 lowercase=True, max_df=1.0, max_features=None, min_df=1,
                 ngram_range=(1, 1), preprocessor=None, stop_words=None,
            ...obs=None,
                      oob_score=False, random_state=None, verbose=0,
                      warm_start=False))]),
               fit_params=None, iid='warn', n_jobs=-1,
               param_grid={'countvectorizer__max_features': (None, 1000, 2000), 'countvectoriz
               pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
               scoring=None, verbose=1)

In [59]: print("Best score: %0.3f" % grid_search.best_score_)
         print("Best parameters set:")
         best_parameters = grid_search.best_estimator_.get_params()
         for param_name in sorted(parameters.keys()):
             print("\t%s: %r" % (param_name, best_parameters[param_name]))

Best score: 0.962
Best parameters set:
        countvectorizer__max_features: 2000
        countvectorizer__ngram_range: (1, 2)
        countvectorizer__stop_words: 'english'
        randomforestclassifier__n_estimators: 50
        tfidftransformer__use_idf: False


In [ ]:
```