# answers

September 12, 2023

## 0.1 Read data

[21]:
```
import pandas as pd

file_path = './amazon_reviews_us_Office_Products_v1_00.tsv'
full_df = pd.read_csv(file_path, delimiter='\t', on_bad_lines='skip')
full_df.dropna()

print(full_df.head(100))
```

```
/var/folders/4k/8jk4g7fx3xl9p5mzpl0rkqj40000gp/T/ipykernel_32723/2109664044.py:4
: DtypeWarning: Columns (7) have mixed types. Specify dtype option on import or
set low_memory=False.
  full_df = pd.read_csv(file_path, delimiter='\t', on_bad_lines='skip')

   marketplace  customer_id       review_id  product_id  product_parent  \
0           US     43081963  R18RVCKGH1SSI9  B001BM2MAC       307809868
1           US     10951564  R3L4L6LW1PUOFY  B00DZYEXPQ        75004341
2           US     21143145  R2J8AWXWTDX2TF  B00RTMUHDW       529689027
3           US     52782374  R1PR37BR7G3M6A  B00D7H8XB6       868449945
4           US     24045652  R3BDDDZMZBZDPU  B001XCWP34        33521401
..         …           …             …           …               …
95          US     43069257  R2Y8H6IMJICNHE  B00E7W6SIU       649134050
96          US      4219837  R3BOZ2S3XKQQDQ  B005XBFYY8       292062400
97          US     10021573  R2EWS2YM55KC99  B007AJ92T4       172016162
98          US     24270459  R3SW4W88I5NUWB  B000E25X92        19288137
99          US     15354510  R15QQMDRBSSDGY  B00N1Q70GM       137395659

                                        product_title product_category  \
0      Scotch Cushion Wrap 7961, 12 Inches x 100 Feet  Office Products
1           Dust-Off Compressed Gas Duster, Pack of 4  Office Products
2   Amram Tagger Standard Tag Attaching Tagging Gu…  Office Products
3   AmazonBasics 12-Sheet High-Security Micro-Cut …  Office Products
4   Derwent Colored Pencils, Inktense Ink Pencils,…  Office Products
..                                              …               …
95                        CHART EL ALFABETO - T-38505  Office Products
96          Crystal Desk Calculator (Assorted Colors)  Office Products
97            Custom For Deposit Only Stamp (3 Lines)  Office Products
98  Bankers Box Stor/Drawer Steel Plus Storage Dra…  Office Products
```

```
99  2 x FLOUERON 2.4V 800mAh Reachargeable Battery…  Office Products

   star_rating  helpful_votes  total_votes vine verified_purchase  \
0            5            0.0          0.0    N                 Y
1            5            0.0          1.0    N                 Y
2            5            0.0          0.0    N                 Y
3            1            2.0          3.0    N                 Y
4            4            0.0          0.0    N                 Y
..         …            …          …  …               …
95           3            0.0          0.0    N                 Y
96           1            0.0          0.0    N                 Y
97           5            0.0          0.0    N                 Y
98           5            0.0          0.0    N                 Y
99           5            0.0          0.0    N                 Y

                                    review_headline  \
0                                         Five Stars
1   Phfffffffft, Phfffffft. Lots of air, and it's C…
2                       but I am sure I will like it.
3   and the shredder was dirty and the bin was par…
4                                         Four Stars
..                                               …
95  It is ok, but considering the price plus shipp…
96                                            Bling
97                              Totally satisfied.
98                            Perfect Bankers Box!
99                                     I am happy.

                                       review_body review_date
0                                    Great product.  2015-08-31
1   What's to say about this commodity item except…  2015-08-31
2     Haven't used yet, but I am sure I will like it.  2015-08-31
3   Although this was labeled as &#34;new&#34; the…  2015-08-31
4                     Gorgeous colors and easy to use  2015-08-31
..                                                …         …
95  It is ok, but considering the price plus shipp…  2015-08-31
96  It has nothing to do with the performance of t…  2015-08-31
97           Exactly as promised. Totally satisfied.  2015-08-31
98  Excellent storage!  Once I found the instructi…  2015-08-31
99  Have had them for about one year now. They are…  2015-08-31

[100 rows x 15 columns]
```

### 0.2 Keep reviews and ratings

[22]:
```
cols = ['review_body', 'star_rating']
reviews_ratings_df = full_df[cols]

print(reviews_ratings_df)
```

```
                                              review_body star_rating
0                                           Great product.           5
1        What's to say about this commodity item except…           5
2          Haven't used yet, but I am sure I will like it.           5
3        Although this was labeled as &#34;new&#34; the…           1
4                           Gorgeous colors and easy to use           4
…                                                      …         …
2640249  I can't live anymore whithout my Palm III. But…           4
2640250  Although the Palm Pilot is thin and compact it…           4
2640251  This book had a lot of great content without b…           4
2640252  I am teaching a course in Excel and am using t…           5
2640253  A very comprehensive layout of exactly how Vis…           5

[2640254 rows x 2 columns]
```

## Form two classes and select 50000 reviews randomly from each class

[23]:
```
class_1 = reviews_ratings_df[reviews_ratings_df['star_rating'].isin([1, 2, 3])].
 ↪copy()
class_2 = reviews_ratings_df[reviews_ratings_df['star_rating'].isin([4, 5])].
 ↪copy()

sample_size = 50_000
class_1 = class_1.sample(n=min(len(class_1), sample_size))
class_2 = class_2.sample(n=min(len(class_2), sample_size))
classified_df = pd.concat([class_1, class_2])

classified_df.loc[:, 'class'] = classified_df['star_rating'].apply(lambda x: 1␣
 ↪if x in [1,2,3] else 2)

print(classified_df.tail(50))
```

```
                                              review_body star_rating  class
1427149  It's difficult to get a telephone/fax/answerin…           5      2
2333726  First of all, I ordered this thing at 3:30pm o…         5.0      2
43704                                I couldn't be happier.           5      2
2284893  This is by far the coolest note pad I have eve…         4.0      2
2632318  I stand by my decision to buy it. The case loo…           4      2
1052648  This product does a good job. No problems in u…         4.0      2
523414             Just as described like the style of it           4      2
394123   Very easy to use magnetic sheet. You can write…           5      2
```

| | | | |
|---|---|---|---|
| 971798 | Easy to use | 5 | 2 |
| 1267915 | meet expectation.  delivery was timely, produc… | 4 | 2 |
| 410864 | Seems to be of good quality and coordinates wi… | 5 | 2 |
| 1930978 | i wanted this phones for an older person that … | 4 | 2 |
| 1161143 | Very serviceable. | 5 | 2 |
| 1849119 | This thing will write on anything, just tried … | 5 | 2 |
| 2314012 | I love these Scotch Bubble Mailers, I use them… | 5.0 | 2 |
| 2025068 | If you've ever shopped for ink, you'll know ho… | 5 | 2 |
| 1759085 | This stamp is very easy to use and makes a ver… | 5 | 2 |
| 1896572 | Love this bag. I had it over a year now. It ta… | 5 | 2 |
| 2063460 | Love it, plenty of room to write in the square… | 5 | 2 |
| 358717 | I love these - they are colorful - have a zipp… | 5 | 2 |
| 1183534 | Great laser printer for the price.  Only compl… | 4 | 2 |
| 5397 | does the job, great price | 5 | 2 |
| 2031000 | I was having some trouble with the toner not f… | 5 | 2 |
| 1626819 | I looked at a number of printers before select… | 4 | 2 |
| 1679898 | Love this Bible tote! Very attractive and I lo… | 5 | 2 |
| 1097747 | This is a very good printer!  The only shortco… | 4.0 | 2 |
| 1044753 | Exactly what I needed, small, sticky and lots … | 5 | 2 |
| 1738589 | I love this cash box… it keep bills organize… | 5 | 2 |
| 866205 | Perfect for my needs, used it to wrap extensio… | 5 | 2 |
| 113384 | Quite simply the GOLD STANDARD by which all ot… | 5 | 2 |
| 376320 | I would buy another works fine. | 5 | 2 |
| 2064509 | came in the mail within 1 week of ordering. it… | 5 | 2 |
| 1932869 | I bought it because I needed a place a place f… | 5 | 2 |
| 1471820 | After having carpal tunnel and shoulder and ne… | 5 | 2 |
| 2308073 | I am very happy with this phone.  I like the w… | 5.0 | 2 |
| 2259204 | Last forever compaired to my old system, doese… | 5.0 | 2 |
| 2039221 | did just what i wanted and love averys online … | 4 | 2 |
| 937760 | My husband loves these | 5 | 2 |
| 1686897 | Love these paint pens. I saw them on a tutoria… | 5 | 2 |
| 2058881 | This pen writes really smooth and has a nice w… | 5 | 2 |
| 1723215 | I purchased this W52P cordless model in late A… | 5 | 2 |
| 2124618 | Worked great for my need an was the best price… | 5 | 2 |
| 926853 | I bought these to be used in my Cannon MG6320 … | 5 | 2 |
| 1981015 | I bought this scanner for scanning 35M film. I… | 4 | 2 |
| 1847159 | Great headset for the house. My kids are getti… | 5 | 2 |
| 1313431 | a very good unit | 5 | 2 |
| 1207699 | Just received yesterday.  Meets my needs but I… | 4 | 2 |
| 53121 | I gave this journal as a gift to a little girl… | 5 | 2 |
| 45356 | Arrived in perfect condition…just put in ou… | 5 | 2 |
| 591625 | Hard to fine these, They work great | 5 | 2 |

### 0.3 Data cleaning

[24]:
```
import re
from contractions import fix

cleaned_df = classified_df.dropna(subset=['review_body'])
cleaned_df.loc[:, 'review_body'] = cleaned_df['review_body'].astype(str)
cleaned_df.loc[:, 'review_body'] = cleaned_df['review_body'].str.lower()
cleaned_df.loc[:, 'review_body'] = cleaned_df['review_body'].apply(lambda x: re.
 ↪sub(r'<.*?>', '', x))
cleaned_df.loc[:, 'review_body'] = cleaned_df['review_body'].apply(lambda x: re.
 ↪sub(r'http\S+', '', x))
cleaned_df.loc[:, 'review_body'] = cleaned_df['review_body'].apply(lambda x: re.
 ↪sub(r'[^a-z\s]', '', x))
cleaned_df.loc[:, 'review_body'] = cleaned_df['review_body'].apply(lambda x: '␣
 ↪'.join(x.split()))
cleaned_df.loc[:, 'review_body'] = cleaned_df['review_body'].apply(lambda x:␣
 ↪fix(x))

ave_len_bef = classified_df['review_body'].str.len().mean()
ave_len_aft = cleaned_df['review_body'].str.len().mean()

print(f'Average length of reviews before vs after: {ave_len_bef:.1f},␣
 ↪{ave_len_aft:.1f}')
```

```
Average length of reviews before vs after: 321.5, 305.1
```

### 0.4 Preprocessing

[25]:
```
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/scottsus/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/scottsus/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /Users/scottsus/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[25]:
```
True
```

### 0.4.1 Remove stop words

[26]:
```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

def remove_stop_words(text):
    tokens = nltk.word_tokenize(text)
    tokens = [token for token in tokens if token not in stop_words]
    return ' '.join(tokens)

no_stopwords_df = cleaned_df
no_stopwords_df.loc[:, 'review_body'] = no_stopwords_df['review_body'].
 ↪apply(remove_stop_words)

print(no_stopwords_df)
```

```
                                                review_body star_rating  class
939996   soon put mfcdw printer signaled jam inside eve…           1      1
1698116  dealt machine years hated every minute yes eat…           1      1
1965940  let first saythis biggest junk ever purchased …           1      1
2600383  like reviewers also replace mine unit would re…           2      1
2281202  printers last spews ink crimps page corners cr…         1.0      1
…                                                      …          …    …
1313431                                        good unit            5      2
1207699  received yesterday meets needs speak hold long…           4      2
53121    gave journal gift little girl loves writing st…           5      2
45356    arrived perfect conditionjust put printer comm…           5      2
591625                             hard fine work great             5      2

[99993 rows x 3 columns]
```

### 0.4.2 Perform lemmatization

[27]:
```
from nltk.stem import WordNetLemmatizer

def lemmatize(text):
    lemmatizer = WordNetLemmatizer()
    tokens = nltk.word_tokenize(text)
    tokens = [lemmatizer.lemmatize(token) for token in tokens]
    return ' '.join(tokens)

lemmatized_df = no_stopwords_df
lemmatized_df.loc[:, 'review_body'] = lemmatized_df['review_body'].
 ↪apply(lemmatize)

print(lemmatized_df)
```

```
                                                review_body star_rating  class
```

```
939996   soon put mfcdw printer signaled jam inside eve…          1      1
1698116  dealt machine year hated every minute yes eat …          1      1
1965940  let first saythis biggest junk ever purchased …          1      1
2600383  like reviewer also replace mine unit would rec…          2      1
2281202  printer last spews ink crimp page corner crump…        1.0      1
…                                                        …       …    …
1313431                                        good unit            5      2
1207699  received yesterday meet need speak hold long t…          4      2
53121    gave journal gift little girl love writing sto…          5      2
45356    arrived perfect conditionjust put printer comm…          5      2
591625                          hard fine work great             5      2

[99993 rows x 3 columns]
```

### 0.4.3 Print results

[28]:
```
ave_len_bef = cleaned_df['review_body'].str.len().mean()
ave_len_aft = classified_df['review_body'].str.len().mean()

print(f'Average length of reviews before vs after data preprocessing:␣
 ↪{ave_len_bef:.1f}, {ave_len_aft:.1f}')
```

```
Average length of reviews before vs after data preprocessing: 189.4, 321.5
```

# 1 TF-IDF and BoW Feature Extraction

[29]:
```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.model_selection import train_test_split

processed_df = lemmatized_df
max_features = 5000

bow_vectorizer = CountVectorizer(max_features=max_features)
X_bow = bow_vectorizer.fit_transform(processed_df['review_body'])

tfidf_vectorizer = TfidfVectorizer(max_features=max_features)
X_tfidf = tfidf_vectorizer.fit_transform(processed_df['review_body'])

y = processed_df['class']
X_train_bow, X_test_bow, y_train_bow, y_test_bow = train_test_split(X_bow, y,␣
 ↪test_size=0.20)
X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfidf =␣
 ↪train_test_split(X_tfidf, y, test_size=0.20)

print(X_train_bow, X_test_bow, y_train_bow, y_test_bow)
print(X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfidf)
```

```
(0, 2814)     1
(0, 1246)     1
(0, 4263)     1
(0, 4482)     1
(0, 2975)     1
(0, 1757)     1
(1, 3305)     1
(1, 4927)     1
(1, 1869)     1
(2, 3305)     1
(2, 3311)     2
(2, 3302)     1
(2, 2188)     1
(2, 3045)     1
(2, 2582)     1
(2, 2783)     1
(2, 1906)     1
(2, 2490)     1
(2, 3394)     1
(2, 4349)     1
(2, 2535)     1
(2, 3893)     1
(2, 784)      1
(2, 3150)     1
(2, 574)      1
:     :
(79992, 2089) 1
(79993, 3305) 1
(79993, 602)  1
(79993, 1463) 1
(79993, 4500) 2
(79993, 3890) 2
(79993, 2200) 2
(79993, 3333) 1
(79993, 2202) 1
(79993, 4528) 1
(79993, 558)  1
(79993, 1815) 1
(79993, 4523) 1
(79993, 557)  1
(79993, 169)  1
(79993, 3326) 1
(79993, 2972) 1
(79993, 4608) 1
(79993, 1399) 1
(79993, 1444) 1
(79993, 3666) 1
(79993, 3629) 1
```

```
(79993, 3884) 1
(79993, 772)  1
(79993, 2934) 1   (0, 4722)   1
(0, 2077)     1
(0, 2841)     1
(0, 2310)     1
(0, 270)      1
(0, 2636)     1
(0, 4077)     1
(0, 2754)     1
(0, 4272)     1
(0, 2368)     1
(0, 263)      1
(1, 3305)     1
(1, 3302)     1
(1, 602)      2
(1, 1640)     1
(1, 4927)     1
(1, 2582)     1
(1, 2200)     1
(1, 3152)     1
(1, 1869)     1
(1, 4950)     1
(1, 4528)     1
(1, 3294)     1
(1, 3177)     1
(1, 3408)     1
:     :
(19997, 4402) 1
(19997, 4939) 1
(19998, 3844) 1
(19998, 1655) 1
(19998, 1438) 1
(19998, 1998) 1
(19998, 4868) 1
(19998, 307)  1
(19998, 220)  1
(19998, 1335) 1
(19998, 458)  1
(19998, 473)  1
(19998, 4473) 1
(19998, 3400) 1
(19998, 1416) 1
(19998, 2512) 1
(19998, 1966) 1
(19998, 1042) 1
(19998, 2320) 1
(19998, 2597) 1
```

```
  (19998, 1277) 1
  (19998, 3887) 1
  (19998, 4502) 1
  (19998, 1944) 1
  (19998, 4012) 1 1925547    2
843190     2
535063     2
1369260    2
226906     2
          ..
2197983    2
853686     1
1889918    2
259358     1
162774     1
Name: class, Length: 79994, dtype: int64 759199     2
2377014    1
2268781    1
1657103    2
1382480    1
          ..
469406     1
1315743    1
2629940    1
2044469    2
1706523    2
Name: class, Length: 19999, dtype: int64
  (0, 1085)     0.30216686918891206
  (0, 3344)     0.30556570704736863
  (0, 1900)     0.3273702360359684
  (0, 4487)     0.2904565218742197
  (0, 2657)     0.2430145169426823
  (0, 3068)     0.2745147623198679
  (0, 1767)     0.21686692078166545
  (0, 4338)     0.27806138563075805
  (0, 1219)     0.2178803885486463
  (0, 294)      0.27571275060970674
  (0, 550)      0.23662243598596383
  (0, 4911)     0.16876641313201604
  (0, 307)      0.17678274682136624
  (0, 3487)     0.17193911358904465
  (0, 1863)     0.15295558702300235
  (0, 4722)     0.1100435530696343
  (0, 1869)     0.11195888404975853
  (0, 3844)     0.16657942208513832
  (0, 948)      0.14303622104097619
  (1, 1107)     0.15118517900585887
  (1, 4053)     0.1261355260234855
```

```
(1, 2379)     0.1331744746469727
(1, 1105)     0.1413862877096064
(1, 413)      0.14459241944591947
(1, 2363)     0.1567053768945572
:     :
(79992, 558)  0.07245652341910126
(79992, 145)  0.056415544323589716
(79992, 3333) 0.07913596955828489
(79992, 4465) 0.05768995671314646
(79992, 2936) 0.31684611936143164
(79992, 2608) 0.0836732699832487
(79992, 3609) 0.0632180461341658
(79992, 2517) 0.05166143305007873
(79992, 4731) 0.05057673675087526
(79992, 4722) 0.03759363673966073
(79992, 135)  0.04862122448090043
(79992, 584)  0.30326724178023484
(79992, 4448) 0.08701869998322961
(79992, 3395) 0.05384304476001583
(79992, 1166) 0.05967857993487565
(79992, 4927) 0.10211694893631033
(79992, 1463) 0.05738055326805145
(79992, 4979) 0.048483237072622774
(79992, 2832) 0.30276429071734545
(79992, 4930) 0.052667066397388475
(79992, 2940) 0.03518307628347529
(79992, 2935) 0.11294656654634452
(79992, 2080) 0.05644117572197056
(79993, 1869) 0.553904247234137
(79993, 1573) 0.8325803774387095   (0, 3352)  0.48580535924366347
(0, 3937)     0.4724084751500792
(0, 1481)     0.6364932396916703
(0, 3408)     0.2209306910135614
(0, 4868)     0.21523273624616898
(0, 3333)     0.2014056612401231
(1, 4341)     0.31131937756425376
(1, 1473)     0.2116066758084749
(1, 133)      0.2205337320245255
(1, 3304)     0.20426792283226528
(1, 1388)     0.7282610882176584
(1, 535)      0.14699845567842942
(1, 3785)     0.2642278746413375
(1, 3333)     0.12883496804913525
(1, 2772)     0.15503038001332461
(1, 4950)     0.12141917820656134
(1, 3930)     0.21556506422055635
(1, 3045)     0.15449464291917608
(1, 602)      0.14297148025861312
```

```
  (2, 3417)     0.04796336001573255
  (2, 3415)     0.04667439941042788
  (2, 2203)     0.14525685616701664
  (2, 2443)     0.13777906153533542
  (2, 1262)     0.09456085314544435
  (2, 4867)     0.0457304063403224
  :     :
  (19998, 3099) 0.2535479830772926
  (19998, 719)  0.11056104983868319
  (19998, 4724) 0.07373699245721714
  (19998, 3106) 0.17899929818337293
  (19998, 4911) 0.1804987554461824
  (19998, 1962) 0.13832741376794663
  (19998, 1742) 0.09341084707909941
  (19998, 3408) 0.13588351243204436
  (19998, 2535) 0.07736906394407185
  (19998, 2530) 0.08559470719831909
  (19998, 3108) 0.10315336561439598
  (19998, 3294) 0.06977640415210738
  (19998, 2971) 0.09185698796700663
  (19998, 1998) 0.10972001130500947
  (19998, 4950) 0.05837218183066907
  (19998, 2980) 0.10247841577910136
  (19998, 4500) 0.06301386383142916
  (19998, 4385) 0.08895036662291647
  (19998, 4979) 0.07589271125989246
  (19998, 2832) 0.07898813515277323
  (19998, 2940) 0.05507344828516656
  (19998, 2935) 0.0883998437424699
  (19998, 179)  0.0979871155653915
  (19998, 1834) 0.06388800710440762
  (19998, 1456) 0.07386597096285923 2203498    1
1911974    1
635480     2
2411092    2
2632706    1
          ..
418867     1
1428090    2
229070     2
1730070    1
782229     2
Name: class, Length: 79994, dtype: int64 810696     2
404055     1
2286479    1
841322     1
218835     1
          ..
```

```
738371     2
1211626    2
2349667    1
2523399    2
949104     2
Name: class, Length: 19999, dtype: int64
```

## 2 Perceptron Using Both Features

[30]:
```
from sklearn.linear_model import Perceptron
from sklearn.metrics import precision_score, recall_score, f1_score

max_iters = 10_000

perc_bow = Perceptron(max_iter=max_iters)
perc_bow.fit(X_train_bow, y_train_bow)
y_pred_bow = perc_bow.predict(X_test_bow)

precision_bow = precision_score(y_test_bow, y_pred_bow)
recall_bow = recall_score(y_test_bow, y_pred_bow)
f1_bow = f1_score(y_test_bow, y_pred_bow)

print('Bag of Words Perceptron')
print(f'Precision: {precision_bow:.2f}, Recall: {recall_bow:.2f}, F1: {f1_bow:.
 ↪2f}')

perc_tfidf = Perceptron(max_iter=max_iters)
perc_tfidf.fit(X_train_tfidf, y_train_tfidf)
y_pred_tfidf = perc_tfidf.predict(X_test_tfidf)

precision_tfidf = precision_score(y_test_tfidf, y_pred_tfidf)
recall_tfidf = recall_score(y_test_tfidf, y_pred_tfidf)
f1_tfidf = f1_score(y_test_tfidf, y_pred_tfidf)

print('TF-IDF Perceptron')
print(f'Precision: {precision_bow:.2}, Recall: {recall_tfidf:.2}, F1: {f1_tfidf:
 ↪.2}')
```

```
Bag of Words Perceptron
Precision: 0.76, Recall: 0.81, F1: 0.79
TF-IDF Perceptron
Precision: 0.76, Recall: 0.79, F1: 0.79
```

## 3 SVM Using Both Features

[31]:
```
from sklearn.svm import LinearSVC

svm_bow = LinearSVC(dual=False, max_iter=max_iters)
svm_bow.fit(X_train_bow, y_train_bow)
y_pred_bow = svm_bow.predict(X_test_bow)

precision_bow = precision_score(y_test_bow, y_pred_bow)
recall_bow = recall_score(y_test_bow, y_pred_bow)
f1_bow = f1_score(y_test_bow, y_pred_bow)

print('Bag of Words SVM')
print(f'Precision: {precision_bow:.2}, Recall: {recall_bow:.2}, F1: {f1_bow:.
 ↪2}')

svm_tfidf = LinearSVC(dual=False, max_iter=max_iters)
svm_tfidf.fit(X_train_tfidf, y_train_tfidf)
y_pred_tfidf = svm_tfidf.predict(X_test_tfidf)

precision_tfidf = precision_score(y_test_tfidf, y_pred_tfidf)
recall_tfidf = recall_score(y_test_tfidf, y_pred_tfidf)
f1_tfidf = f1_score(y_test_tfidf, y_pred_tfidf)

print('TF-IDF SVM')
print(f'Precision: {precision_tfidf:.2}, Recall: {recall_tfidf:.2}, F1:␣
 ↪{f1_tfidf:.2}')
```

```
Bag of Words SVM
Precision: 0.84, Recall: 0.81, F1: 0.82
TF-IDF SVM
Precision: 0.84, Recall: 0.84, F1: 0.84
```

## 4 Logistic Regression Using Both Features

[32]:
```
from sklearn.linear_model import LogisticRegression

logreg_bow = LogisticRegression(max_iter=max_iters)
logreg_bow.fit(X_train_bow, y_train_bow)
y_pred_bow = logreg_bow.predict(X_test_bow)

precision_bow = precision_score(y_test_bow, y_pred_bow)
recall_bow = recall_score(y_test_bow, y_pred_bow)
f1_bow = f1_score(y_test_bow, y_pred_bow)

print('Bag of Words Logistic Regression')
```

```
print(f'Precision: {precision_bow:.2}, Recall: {recall_bow:.2}, F1: {f1_bow:.
 ↪2}')

logreg_tfidf = LogisticRegression(max_iter=max_iters)
logreg_tfidf.fit(X_train_tfidf, y_train_tfidf)
y_pred_tfidf = logreg_tfidf.predict(X_test_tfidf)

precision_tfidf = precision_score(y_test_tfidf, y_pred_tfidf)
recall_bow = recall_score(y_test_tfidf, y_pred_tfidf)
f1_bow = f1_score(y_test_tfidf, y_pred_tfidf)

print('TF-IDF Logistic Regression')
print(f'Precision: {precision_tfidf:.2}, Recall: {recall_tfidf:.2}, F1:␣
 ↪{f1_tfidf:.2}')
```

```
Bag of Words Logistic Regression
Precision: 0.84, Recall: 0.82, F1: 0.83
TF-IDF Logistic Regression
Precision: 0.84, Recall: 0.84, F1: 0.84
```

## 5 Naive Bayes Using Both Features

[33]:
```
from sklearn.naive_bayes import MultinomialNB

nb_bow = MultinomialNB()
nb_bow.fit(X_train_bow, y_train_bow)
y_pred_bow = nb_bow.predict(X_test_bow)

precision_bow = precision_score(y_test_bow, y_pred_bow)
recall_bow = recall_score(y_test_bow, y_pred_bow)
f1_bow = f1_score(y_test_bow, y_pred_bow)

print('Bag of Words Multinomial Naive Bayes')
print(f'Precision: {precision_bow:.2}, Recall: {recall_bow:.2}, F1: {f1_bow:.
 ↪2}')

nb_tfidf = MultinomialNB()
nb_tfidf.fit(X_train_tfidf, y_train_tfidf)
y_pred_tfidf = nb_tfidf.predict(X_test_tfidf)

precision_tfidf = precision_score(y_test_tfidf, y_pred_tfidf)
recall_bow = recall_score(y_test_tfidf, y_pred_tfidf)
f1_bow = f1_score(y_test_tfidf, y_pred_tfidf)

print('TF-IDF Multinomial Naive Bayes')
```

```
print(f'Precision: {precision_tfidf:.2}, Recall: {recall_tfidf:.2}, F1:␣
  ↪{f1_tfidf:.2}')
```

```
Bag of Words Multinomial Naive Bayes
Precision: 0.83, Recall: 0.74, F1: 0.78
TF-IDF Multinomial Naive Bayes
Precision: 0.82, Recall: 0.84, F1: 0.84
```