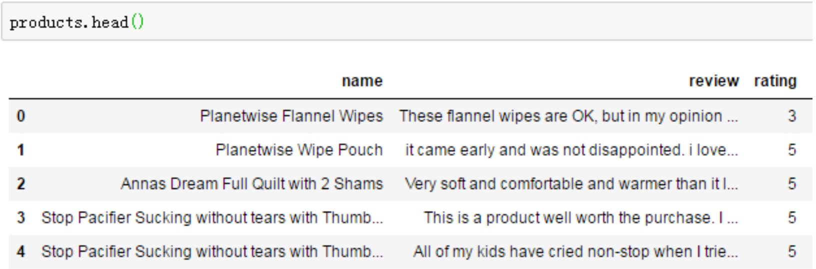
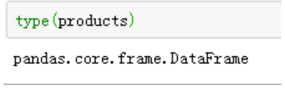


```
products.head()
```

	name	review	rating
0	Planetwise Flannel Wipes	These flannel wipes are OK, but in my opinion ...	3
1	Planetwise Wipe Pouch	it came early and was not disappointed. i love...	5
2	Annas Dream Full Quilt with 2 Shams	Very soft and comfortable and warmer than it l...	5
3	Stop Pacifier Sucking without tears with Thumb...	This is a product well worth the purchase. I ...	5
4	Stop Pacifier Sucking without tears with Thumb...	All of my kids have cried non-stop when I trie...	5



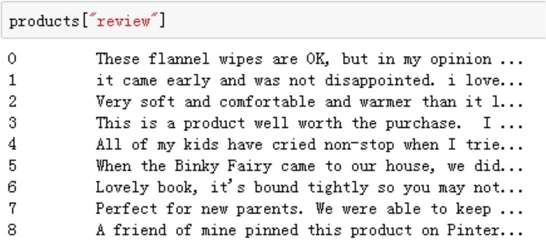
```
type(products)
```

```
pandas.core.frame.DataFrame
```

When *products* is DataFrame, *products[“review”]* is pandas.Series, *products[[“review”]]* is DataFrame, *products[“review”][0]* is String

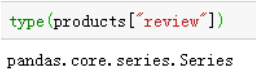
replace function only works for String.

Object *String* does not have attribute *apply*.



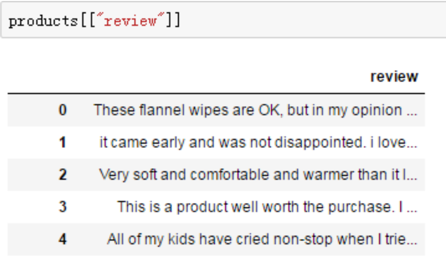
```
products["review"]
```

```
0        These flannel wipes are OK, but in my opinion ...
1        it came early and was not disappointed. i love...
2        Very soft and comfortable and warmer than it l...
3        This is a product well worth the purchase.  I ...
4        All of my kids have cried non-stop when I trie...
5        When the Binky Fairy came to our house, we did...
6        Lovely book, it's bound tightly so you may not...
7        Perfect for new parents. We were able to keep ...
8        A friend of mine pinned this product on Pinter...
```



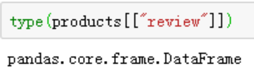
```
type(products["review"])
```

```
pandas.core.series.Series
```



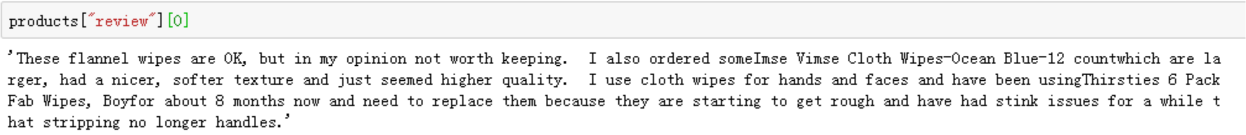
```
products[["review"]]
```

	review
0	These flannel wipes are OK, but in my opinion ...
1	it came early and was not disappointed. i love...
2	Very soft and comfortable and warmer than it l...
3	This is a product well worth the purchase. I ...
4	All of my kids have cried non-stop when I trie...



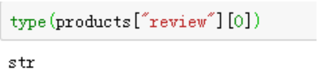
```
type(products[["review"]])
```

```
pandas.core.frame.DataFrame
```



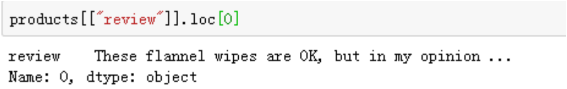
```
products["review"][0]
```

```
'These flannel wipes are OK, but in my opinion not worth keeping.  I also ordered someImse Vimse Cloth Wipes-Ocean Blue-12 countwhich are la
rger, had a nicer, softer texture and just seemed higher quality.  I use cloth wipes for hands and faces and have been usingThirsties 6 Pack
Fab Wipes, Boyfor about 8 months now and need to replace them because they are starting to get rough and have had stink issues for a while t
hat stripping no longer handles.'
```



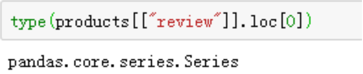
```
type(products["review"][0])
```

```
str
```



```
products[["review"]].loc[0]
```

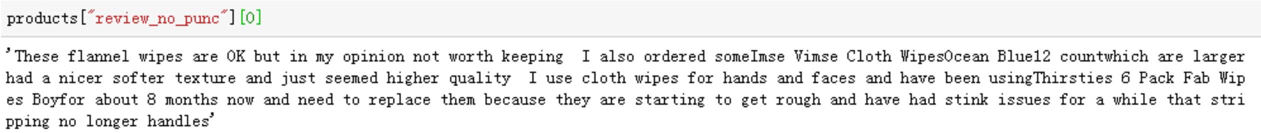
```
review    These flannel wipes are OK, but in my opinion ...
Name: 0, dtype: object
```



```
type(products[["review"]].loc[0])
```

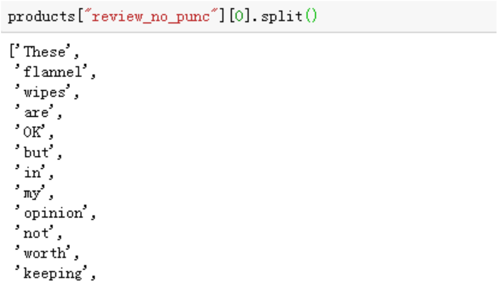
```
pandas.core.series.Series
```

The function *text.split()* splits words in String, and output List.



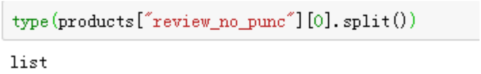
```
products["review_no_punc"][0]
```

```
'These flannel wipes are OK but in my opinion not worth keeping  I also ordered someImse Vimse Cloth WipesOcean Blue12 countwhich are larger
had a nicer softer texture and just seemed higher quality  I use cloth wipes for hands and faces and have been usingThirsties 6 Pack Fab Wip
es Boyfor about 8 months now and need to replace them because they are starting to get rough and have had stink issues for a while that stri
pping no longer handles'
```



```
products["review_no_punc"][0].split()
```

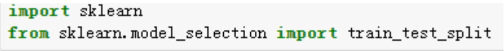
```
['These',
 'flannel',
 'wipes',
 'are',
 'OK',
 'but',
 'in',
 'my',
 'opinion',
 'not',
 'worth',
 'keeping',
```



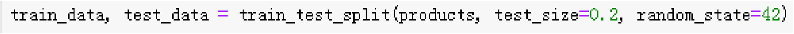
```
type(products["review_no_punc"][0].split())
```

```
list
```

The function *train_test_split* splits data into training and test sets.



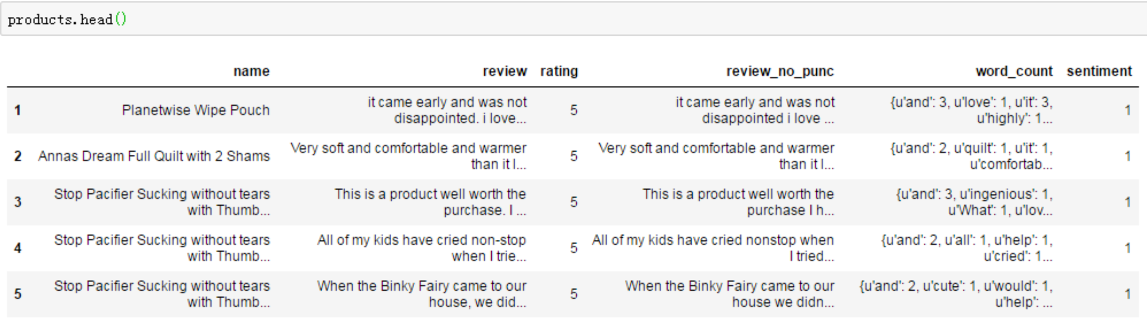
```
import sklearn
from sklearn.model_selection import train_test_split
```



```
train_data, test_data = train_test_split(products, test_size=0.2, random_state=42)
```

The function *df.reset_index(drop=True)* reset the DataFrame's index.

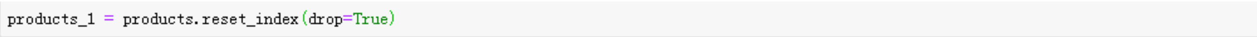
Before the reset:



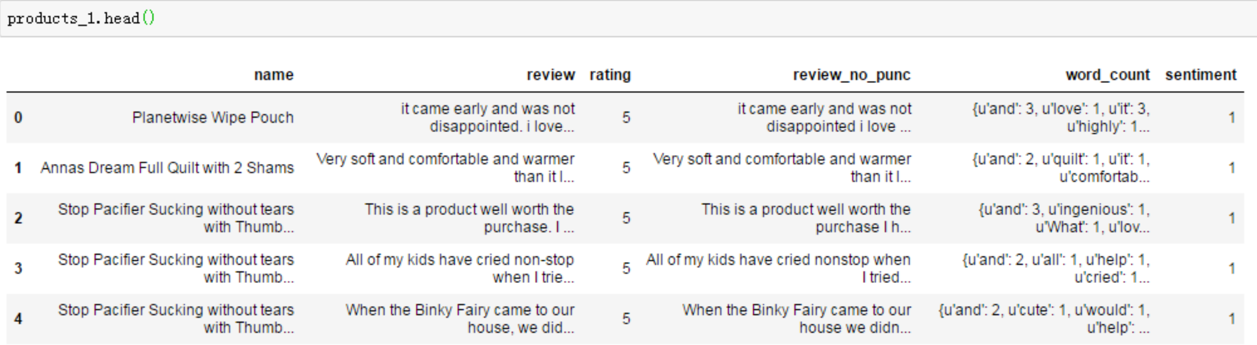
```
products.head()
```

	name	review	rating	review_no_punc	word_count	sentiment
1	Planetwise Wipe Pouch	it came early and was not disappointed. i love...	5	it came early and was not disappointed i love ...	{u'and': 3, u'love': 1, u'it': 3, u'highly': 1...	1
2	Annas Dream Full Quilt with 2 Shams	Very soft and comfortable and warmer than it l...	5	Very soft and comfortable and warmer than it l...	{u'and': 2, u'quilt': 1, u'it': 1, u'comfortab...	1
3	Stop Pacifier Sucking without tears with Thumb...	This is a product well worth the purchase. I ...	5	This is a product well worth the purchase I h...	{u'and': 3, u'ingenious': 1, u'What': 1, u'lov...	1
4	Stop Pacifier Sucking without tears with Thumb...	All of my kids have cried non-stop when I trie...	5	All of my kids have cried nonstop when I tried...	{u'and': 2, u'all': 1, u'help': 1, u'cried': 1...	1
5	Stop Pacifier Sucking without tears with Thumb...	When the Binky Fairy came to our house, we did...	5	When the Binky Fairy came to our house we didn...	{u'and': 2, u'cute': 1, u'would': 1, u'help': ...	1

After the reset:



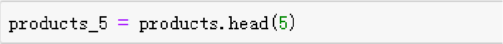
```
products_1 = products.reset_index(drop=True)
```



```
products_1.head()
```

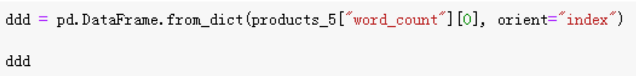
	name	review	rating	review_no_punc	word_count	sentiment
0	Planetwise Wipe Pouch	it came early and was not disappointed. i love...	5	it came early and was not disappointed i love ...	{u'and': 3, u'love': 1, u'it': 3, u'highly': 1...	1
1	Annas Dream Full Quilt with 2 Shams	Very soft and comfortable and warmer than it l...	5	Very soft and comfortable and warmer than it l...	{u'and': 2, u'quilt': 1, u'it': 1, u'comfortab...	1
2	Stop Pacifier Sucking without tears with Thumb...	This is a product well worth the purchase. I ...	5	This is a product well worth the purchase I h...	{u'and': 3, u'ingenious': 1, u'What': 1, u'lov...	1
3	Stop Pacifier Sucking without tears with Thumb...	All of my kids have cried non-stop when I trie...	5	All of my kids have cried nonstop when I tried...	{u'and': 2, u'all': 1, u'help': 1, u'cried': 1...	1
4	Stop Pacifier Sucking without tears with Thumb...	When the Binky Fairy came to our house, we did...	5	When the Binky Fairy came to our house we didn...	{u'and': 2, u'cute': 1, u'would': 1, u'help': ...	1

The code below transforms bag of words [“word_count”] to sparse matrix.

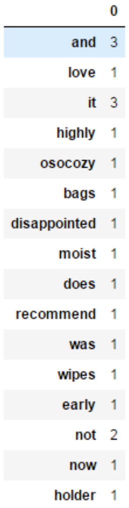


```
products_5 = products.head(5)
```

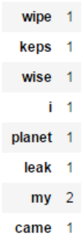
The function *from_dict* converts Dictionary to DataFrame, and *keys()* is the index.



```
ddd = pd.DataFrame.from_dict(products_5["word_count"][0], orient="index")

ddd
```

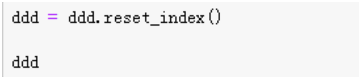


	0
and	3
love	1
it	3
highly	1
osocozy	1
bags	1
disappointed	1
moist	1
does	1
recommend	1
was	1
wipes	1
early	1
not	2
now	1
holder	1

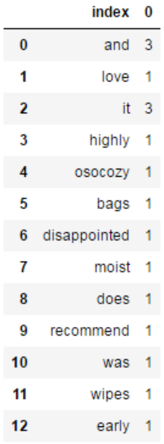


wipe	1
keps	1
wise	1
i	1
planet	1
leak	1
my	2
came	1

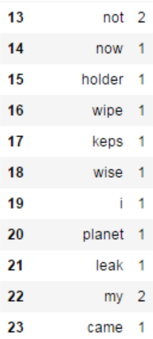
The function *reset_index()* resets the index as 0, 1, …, N, and the old index column turns into “feature”.



```
ddd = ddd.reset_index()

ddd
```

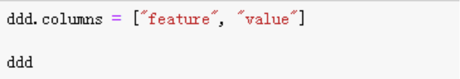


	index	0
0	and	3
1	love	1
2	it	3
3	highly	1
4	osocozy	1
5	bags	1
6	disappointed	1
7	moist	1
8	does	1
9	recommend	1
10	was	1
11	wipes	1
12	early	1

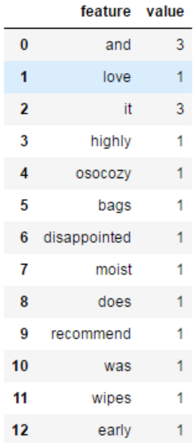


13	not	2
14	now	1
15	holder	1
16	wipe	1
17	keps	1
18	wise	1
19	i	1
20	planet	1
21	leak	1
22	my	2
23	came	1

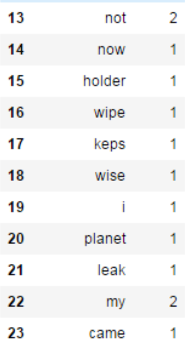
The function *.columns=[“”, “”]* renames the columns.



```
ddd.columns = ["feature", "value"]

ddd
```

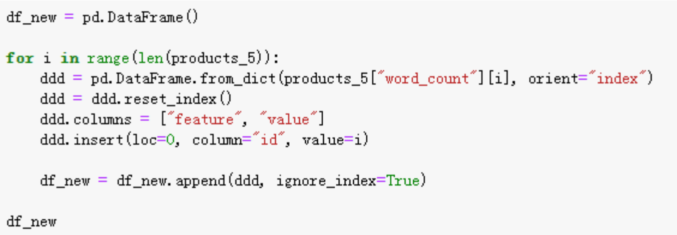


	feature	value
0	and	3
1	love	1
2	it	3
3	highly	1
4	osocozy	1
5	bags	1
6	disappointed	1
7	moist	1
8	does	1
9	recommend	1
10	was	1
11	wipes	1
12	early	1

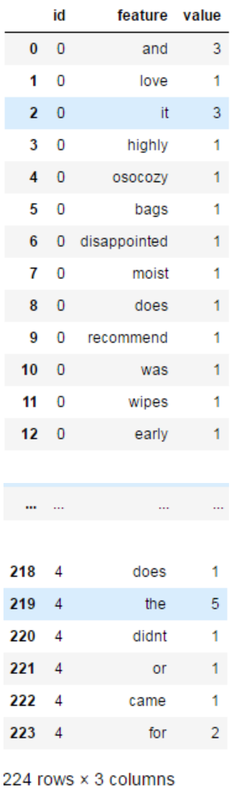


13	not	2
14	now	1
15	holder	1
16	wipe	1
17	keps	1
18	wise	1
19	i	1
20	planet	1
21	leak	1
22	my	2
23	came	1

Combined:



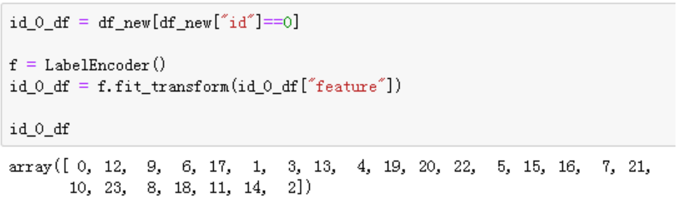
```
df_new = pd.DataFrame()

for i in range(len(products_5)):
    ddd = pd.DataFrame.from_dict(products_5["word_count"][i], orient="index")
    ddd = ddd.reset_index()
    ddd.columns = ["feature", "value"]
    ddd.insert(loc=0, column="id", value=i)

    df_new = df_new.append(ddd, ignore_index=True)

df_new
```



	id	feature	value
0	0	and	3
1	0	love	1
2	0	it	3
3	0	highly	1
4	0	osocozy	1
5	0	bags	1
6	0	disappointed	1
7	0	moist	1
8	0	does	1
9	0	recommend	1
10	0	was	1
11	0	wipes	1
12	0	early	1
...	...	...	...
218	4	does	1
219	4	the	5
220	4	didnt	1
221	4	or	1
222	4	came	1
223	4	for	2

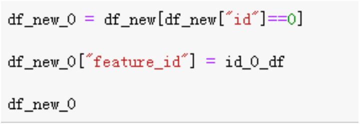
224 rows × 3 columns

Label encoding:

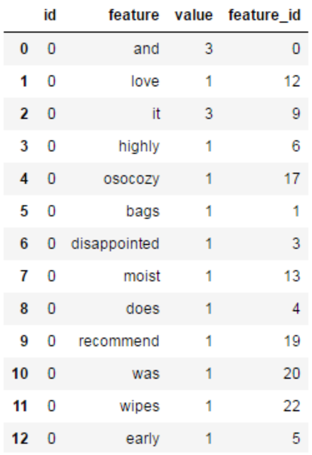


```
id_0_df = df_new[df_new["id"]==0]

f = LabelEncoder()
id_0_df = f.fit_transform(id_0_df["feature"])

id_0_df
```

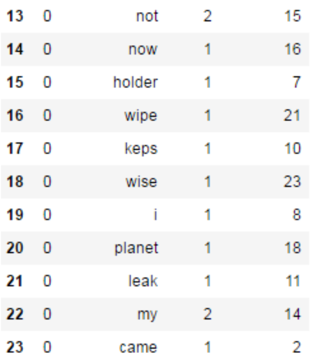
```
array([ 0, 12,  9,  6, 17,  1,  3, 13,  4, 19, 20, 22,  5, 15, 16,  7, 21,
       10, 23,  8, 18, 11, 14,  2])
```



```
df_new_0 = df_new[df_new["id"]==0]

df_new_0["feature_id"] = id_0_df

df_new_0
```

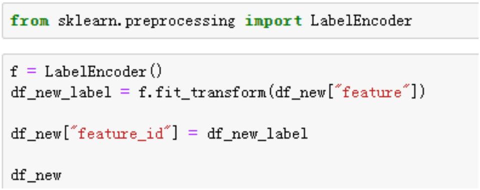


	id	feature	value	feature_id
0	0	and	3	0
1	0	love	1	12
2	0	it	3	9
3	0	highly	1	6
4	0	osocozy	1	17
5	0	bags	1	1
6	0	disappointed	1	3
7	0	moist	1	13
8	0	does	1	4
9	0	recommend	1	19
10	0	was	1	20
11	0	wipes	1	22
12	0	early	1	5



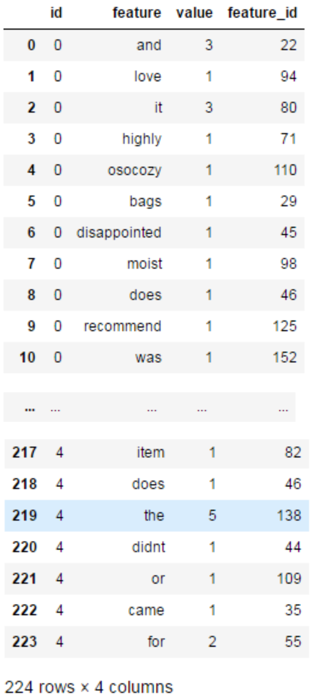
13	0	not	2	15
14	0	now	1	16
15	0	holder	1	7
16	0	wipe	1	21
17	0	keps	1	10
18	0	wise	1	23
19	0	i	1	8
20	0	planet	1	18
21	0	leak	1	11
22	0	my	2	14
23	0	came	1	2

Combined for the DataFrame *products_5*:



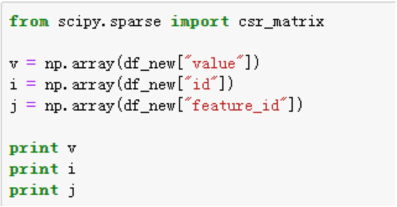
```
from sklearn.preprocessing import LabelEncoder
```

```
f = LabelEncoder()
df_new_label = f.fit_transform(df_new["feature"])

df_new["feature_id"] = df_new_label

df_new
```

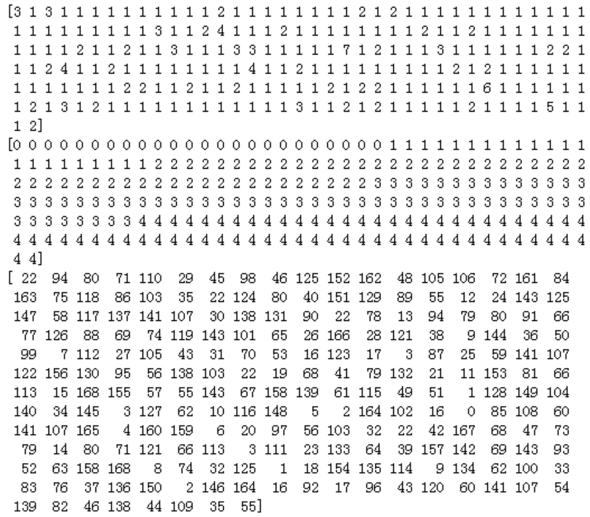


	id	feature	value	feature_id
0	0	and	3	22
1	0	love	1	94
2	0	it	3	80
3	0	highly	1	71
4	0	osocozy	1	110
5	0	bags	1	29
6	0	disappointed	1	45
7	0	moist	1	98
8	0	does	1	46
9	0	recommend	1	125
10	0	was	1	152
...	...	...	...	...
217	4	item	1	82
218	4	does	1	46
219	4	the	5	138
220	4	didnt	1	44
221	4	or	1	109
222	4	came	1	35
223	4	for	2	55

224 rows × 4 columns

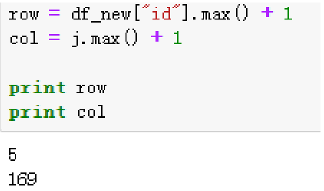


```
from scipy.sparse import csr_matrix

v = np.array(df_new["value"])
i = np.array(df_new["id"])
j = np.array(df_new["feature_id"])

print v
print i
print j
```



```
[3 1 3 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 3 1 1 2 4 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1
 1 1 1 1 2 1 1 2 1 1 3 1 1 1 3 3 1 1 1 1 1 7 1 2 1 1 1 3 1 1 1 1 1 1 2 2 1
 1 1 2 4 1 1 2 1 1 1 1 1 1 1 1 4 1 1 2 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1
 1 1 1 1 1 1 1 2 2 1 1 2 1 1 2 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 6 1 1 1 1 1 1
 1 2 1 3 1 2 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 2 1 2 1 1 1 1 1 2 1 1 1 1 5 1 1
 1 2]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4]
[ 22  94  80  71 110  29  45  98  46 125 152 162  48 105 106  72 161  84
 163  75 118  86 103  35  22 124  80  40 151 129  89  55  12  24 143 125
 147  58 117 137 141 107  30 138 131  90  22  78  13  94  79  80  91  66
  77 126  88  69  74 119 143 101  65  26 166  28 121  38   9 144  36  50
  99   7 112  27 105  43  31  70  53  16 123  17   3  87  25  59 141 107
 122 156 130  95  56 138 103  22  19  68  41  79 132  21  11 153  81  66
 113  15 168 155  57  55 143  67 158 139  61 115  49  51   1 128 149 104
 140  34 145   3 127  62  10 116 148   5   2 164 102  16   0  85 108  60
 141 107 165   4 160 159   6  20  97  56 103  32  22  42 167  68  47  73
  79  14  80  71 121  66 113   3 111  23 133  64  39 157 142  69 143  93
  52  63 158 168   8  74  32 125   1  18 154 135 114   9 134  62 100  33
  83  76  37 136 150   2 146 164  16  92  17  96  43 120  60 141 107  54
 139  82  46 138  44 109  35  55]
```

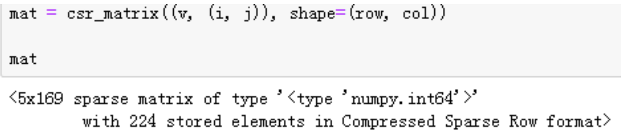
Array1 is the value, array2 is the new row index, array3 is the new column index.



```
row = df_new["id"].max() + 1
col = j.max() + 1

print row
print col
```

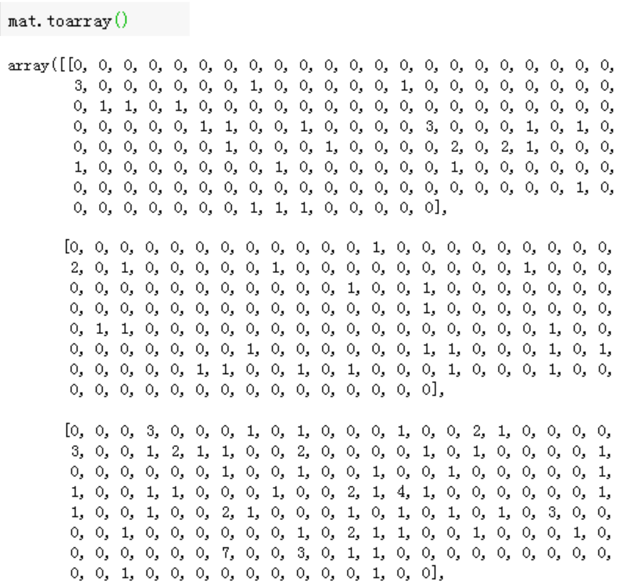
```
5
169
```

row is the total number of new rows, *col* is the total number of new columns.



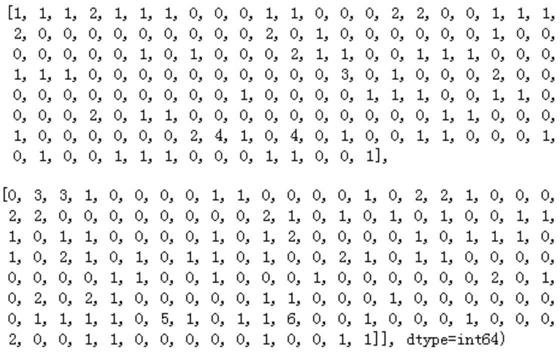
```
mat = csr_matrix((v, (i, j)), shape=(row, col))

mat
```

```
<5x169 sparse matrix of type '<type 'numpy.int64'>'
	with 224 stored elements in Compressed Sparse Row format>
```



```
mat.toarray()
```

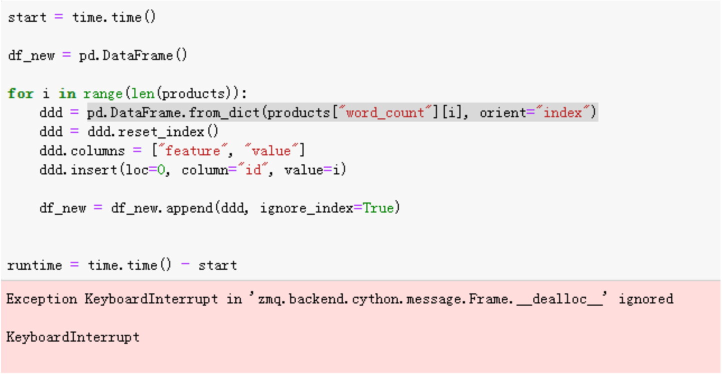
```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        3, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 3, 0, 0, 0, 1, 0, 1, 0,
        0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 2, 1, 0, 0, 0,
        1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
        0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0],

       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        2, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
        0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1,
        0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

       [0, 0, 0, 3, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 0,
        3, 0, 0, 1, 2, 1, 1, 0, 0, 2, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
        0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
        1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 2, 1, 4, 1, 0, 0, 0, 0, 0, 0, 1,
        1, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 3, 0, 0,
        0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 2, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
        0, 0, 0, 0, 0, 0, 7, 0, 0, 3, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
```



```
 [1, 1, 1, 2, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 2, 2, 0, 0, 1, 1, 1,
  2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
  0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 2, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0,
  1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 1, 0, 0, 0, 2, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0,
  0, 0, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
  1, 0, 0, 0, 0, 0, 0, 2, 4, 1, 0, 4, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1,
  0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1],

[0, 3, 3, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 2, 2, 1, 0, 0, 0,
 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1,
 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 2, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
 1, 0, 2, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 2, 1, 0, 1, 1, 0, 0, 0, 0,
 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 0, 1,
 0, 2, 0, 2, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
 0, 1, 1, 1, 1, 0, 5, 1, 0, 1, 1, 6, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
 2, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1]], dtype=int64)
```

Trying to do the same work on the whole data:

Step 1: stack *products[“word_count”]*

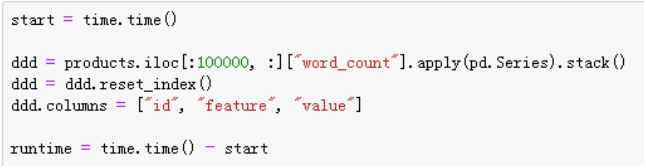


```
start = time.time()

df_new = pd.DataFrame()

for i in range(len(products)):
    ddd = pd.DataFrame.from_dict(products["word_count"][i], orient="index")
    ddd = ddd.reset_index()
    ddd.columns = ["feature", "value"]
    ddd.insert(loc=0, column="id", value=i)

    df_new = df_new.append(ddd, ignore_index=True)


runtime = time.time() - start
```

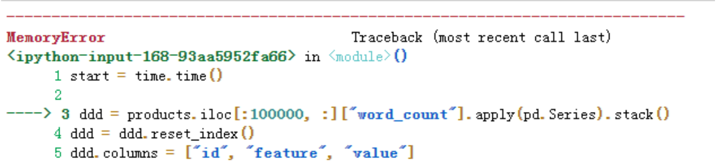
```
Exception KeyboardInterrupt in 'zmq.backend.cython.message.Frame.__dealloc__' ignored

KeyboardInterrupt
```

Ran a whole day!!!!

There is another way:

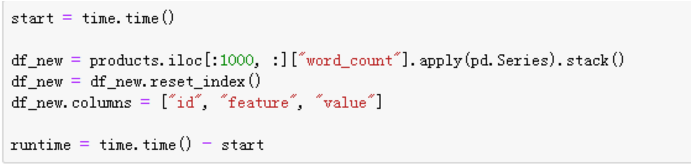


```
start = time.time()

ddd = products.iloc[:100000, :]["word_count"].apply(pd.Series).stack()
ddd = ddd.reset_index()
ddd.columns = ["id", "feature", "value"]

runtime = time.time() - start
```

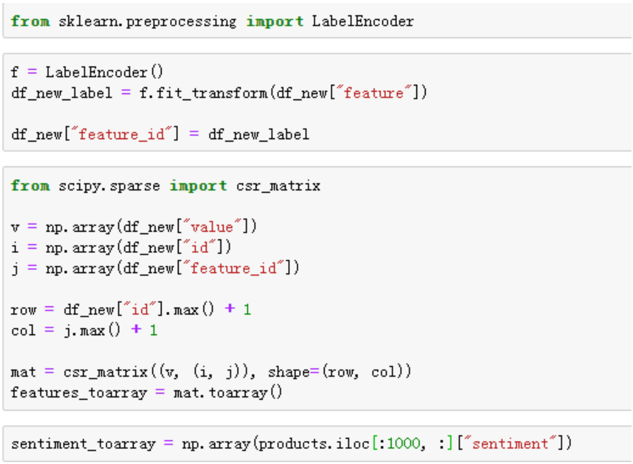


```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
<ipython-input-168-93aa5952fa66> in <module>()
      1 start = time.time()
      2
----> 3 ddd = products.iloc[:100000, :]["word_count"].apply(pd.Series).stack()
      4 ddd = ddd.reset_index()
      5 ddd.columns = ["id", "feature", "value"]
```

So, to save time, I only use the first 1000 observations.



```
start = time.time()

df_new = products.iloc[:1000, :]["word_count"].apply(pd.Series).stack()
df_new = df_new.reset_index()
df_new.columns = ["id", "feature", "value"]

runtime = time.time() - start
```



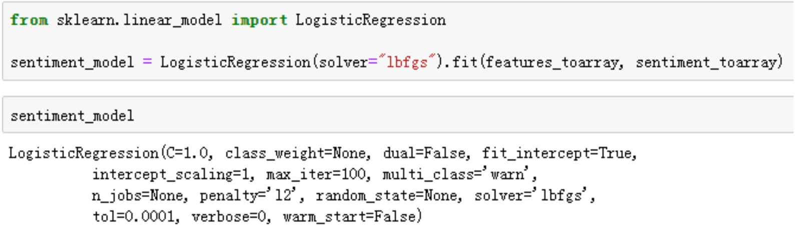
```python
from sklearn.preprocessing import LabelEncoder
```

```python
f = LabelEncoder()
df_new_label = f.fit_transform(df_new["feature"])

df_new["feature_id"] = df_new_label
```

```python
from scipy.sparse import csr_matrix

v = np.array(df_new["value"])
i = np.array(df_new["id"])
j = np.array(df_new["feature_id"])

row = df_new["id"].max() + 1
col = j.max() + 1

mat = csr_matrix((v, (i, j)), shape=(row, col))
features_toarray = mat.toarray()
```

```python
sentiment_toarray = np.array(products.iloc[:1000, :]["sentiment"])
```

The functions below train an logistic regression model:

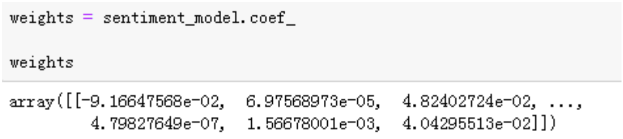


```python
from sklearn.linear_model import LogisticRegression

sentiment_model = LogisticRegression(solver="lbfgs").fit(features_toarray, sentiment_toarray)
```

```python
sentiment_model
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='lbfgs',
          tol=0.0001, verbose=0, warm_start=False)
```

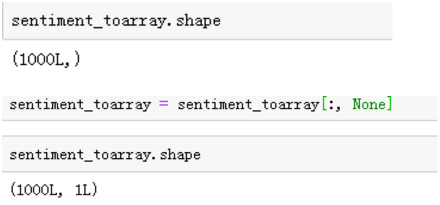
The function *model.coef_* outputs the coefficient for each feature:



```python
weights = sentiment_model.coef_

weights
```

```
array([[-9.16647568e-02,  6.97568973e-05,  4.82402724e-02, ...,
         4.79827649e-07,  1.56678001e-03,  4.04295513e-02]])
```

(1000,) and *(1000, 1)* are not the same shape. The function *[:, None]* converts *(1000,)* to the shape *(1000, 1).*



```python
sentiment_toarray.shape
```

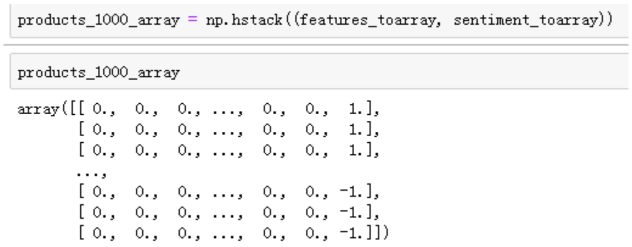
```
(1000L,)
```

```python
sentiment_toarray = sentiment_toarray[:, None]
```

```python
sentiment_toarray.shape
```

```
(1000L, 1L)
```

The function *np.hstack((a, b))* stacks two arrays with same number of rows horizontally.

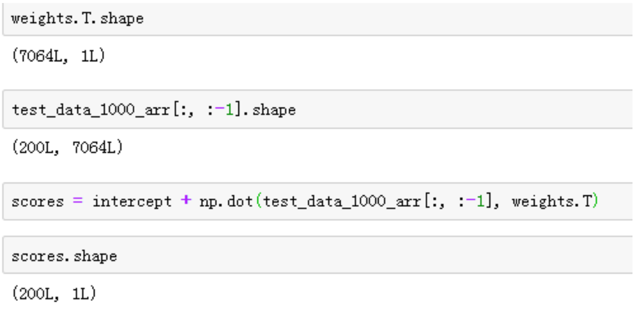


```
products_1000_array = np.hstack((features_toarray, sentiment_toarray))
```

```
products_1000_array
```

```
array([[ 0.,   0.,   0., ...,  0.,   0.,   1.],
       [ 0.,   0.,   0., ...,  0.,   0.,   1.],
       [ 0.,   0.,   0., ...,  0.,   0.,   1.],
       ...,
       [ 0.,   0.,   0., ...,  0.,   0.,  -1.],
       [ 0.,   0.,   0., ...,  0.,   0.,  -1.],
       [ 0.,   0.,   0., ...,  0.,   0.,  -1.]])
```

The function *np.dot(A, B)* performs the matrix multiplication.



```
weights.T.shape
```

```
(7064L, 1L)
```

```
test_data_1000_arr[:, :-1].shape
```

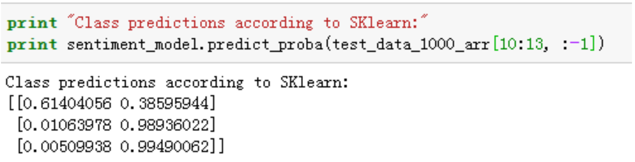
```
(200L, 7064L)
```

```
scores = intercept + np.dot(test_data_1000_arr[:, :-1], weights.T)
```

```
scores.shape
```

```
(200L, 1L)
```

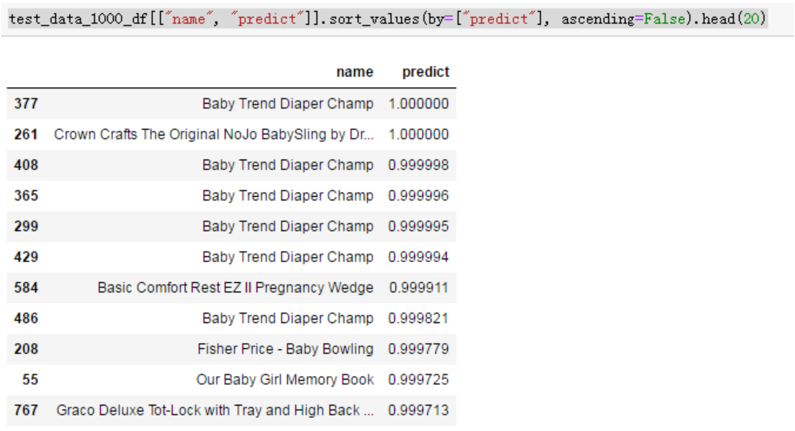
The function *model.predict_proba(array-like)* predicts the probability.



```
print "Class predictions according to SKlearn:"
print sentiment_model.predict_proba(test_data_1000_arr[10:13, :-1])
```

```
Class predictions according to SKlearn:
[[0.61404056 0.38595944]
 [0.01063978 0.98936022]
 [0.00509938 0.99490062]]
```

The function *df.sort_values(by=["column"])* sort the dataframe by a column.



```
test_data_1000_df[["name", "predict"]].sort_values(by=["predict"], ascending=False).head(20)
```

	name	predict
377	Baby Trend Diaper Champ	1.000000
261	Crown Crafts The Original NoJo BabySling by Dr...	1.000000
408	Baby Trend Diaper Champ	0.999998
365	Baby Trend Diaper Champ	0.999996
299	Baby Trend Diaper Champ	0.999995
429	Baby Trend Diaper Champ	0.999994
584	Basic Comfort Rest EZ II Pregnancy Wedge	0.999911
486	Baby Trend Diaper Champ	0.999821
208	Fisher Price - Baby Bowling	0.999779
55	Our Baby Girl Memory Book	0.999725
767	Graco Deluxe Tot-Lock with Tray and High Back ...	0.999713