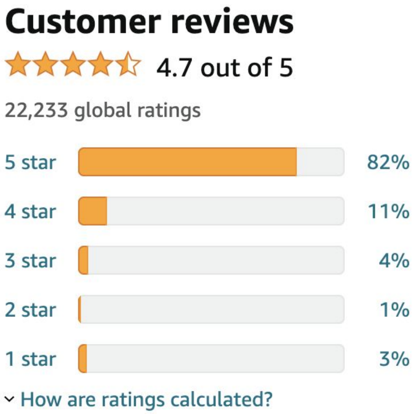
Data handling: Text data (Lab)

ECE30008 Intro to AI

한동대학교
HANDONG GLOBAL UNIVERSITY

Exercise data



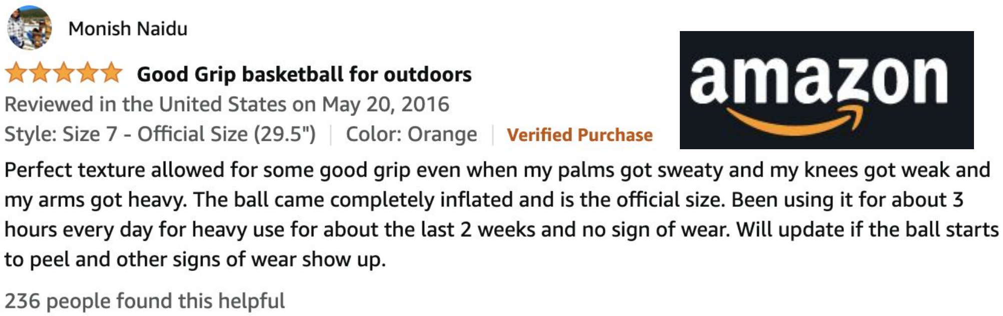
Customer reviews

4.7 out of 5

22,233 global ratings

5 star	82%
4 star	11%
3 star	4%
2 star	1%
1 star	3%

How are ratings calculated?



Monish Naidu

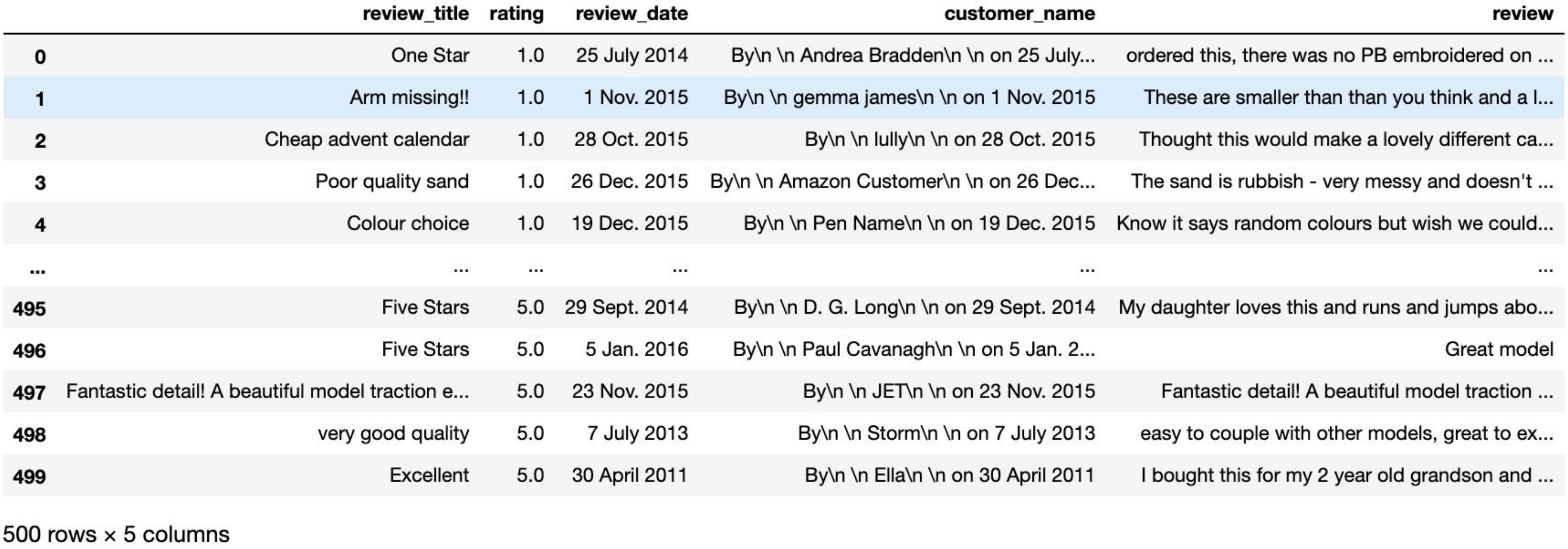
Good Grip basketball for outdoors

Reviewed in the United States on May 20, 2016

Style: Size 7 - Official Size (29.5") | Color: Orange | Verified Purchase

Perfect texture allowed for some good grip even when my palms got sweaty and my knees got weak and my arms got heavy. The ball came completely inflated and is the official size. Been using it for about 3 hours every day for heavy use for about the last 2 weeks and no sign of wear. Will update if the ball starts to peel and other signs of wear show up.

236 people found this helpful

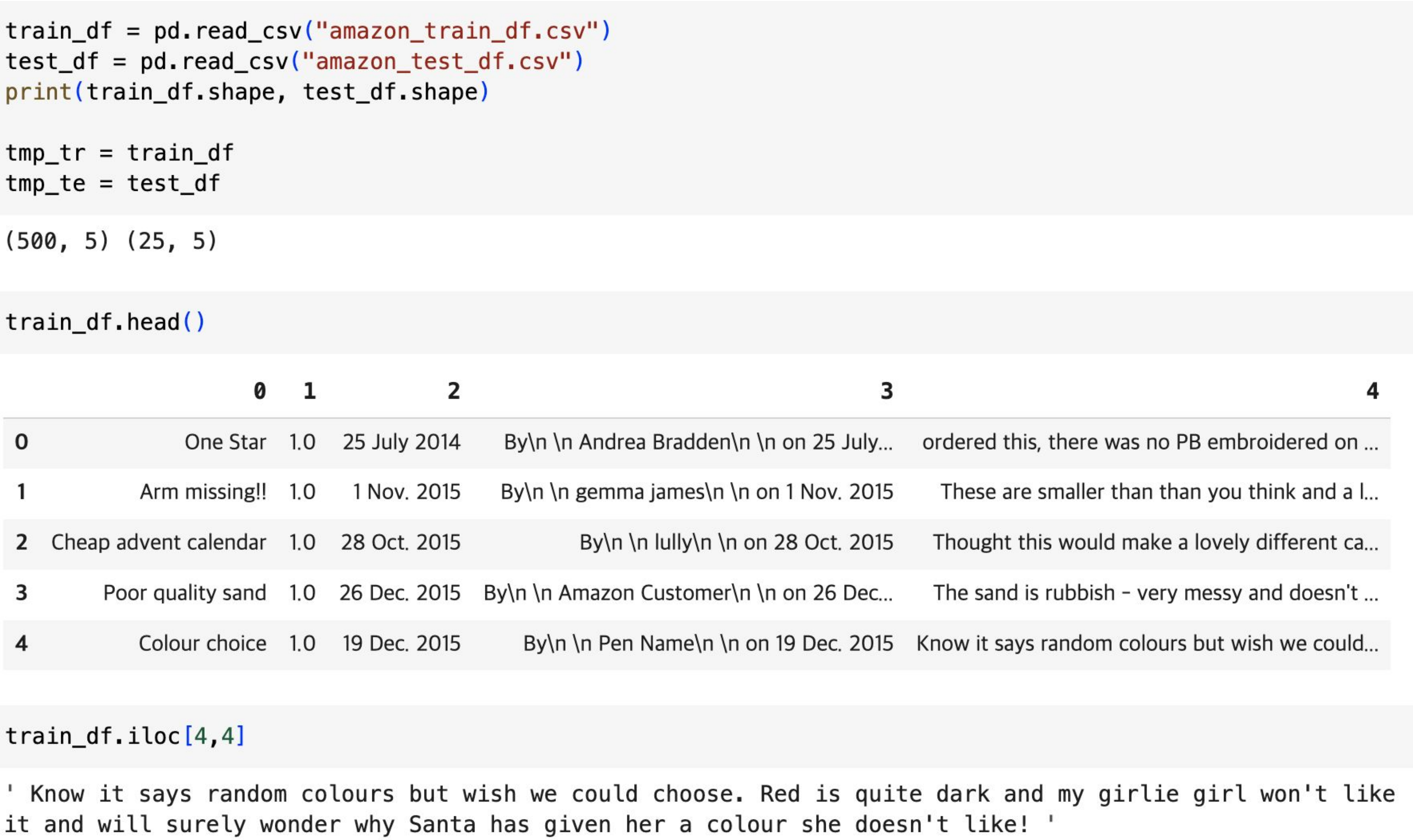


	review_title	rating	review_date	customer_name	review
0	One Star	1.0	25 July 2014	By\n \n Andrea Bradden\n \n on 25 July...	ordered this, there was no PB embroidered on ...
1	Arm missing!!	1.0	1 Nov. 2015	By\n \n gemma james\n \n on 1 Nov. 2015	These are smaller than than you think and a l...
2	Cheap advent calendar	1.0	28 Oct. 2015	By\n \n lully\n \n on 28 Oct. 2015	Thought this would make a lovely different ca...
3	Poor quality sand	1.0	26 Dec. 2015	By\n \n Amazon Customer\n \n on 26 Dec...	The sand is rubbish - very messy and doesn't ...
4	Colour choice	1.0	19 Dec. 2015	By\n \n Pen Name\n \n on 19 Dec. 2015	Know it says random colours but wish we could...
...	...	...	...	...	...
495	Five Stars	5.0	29 Sept. 2014	By\n \n D. G. Long\n \n on 29 Sept. 2014	My daughter loves this and runs and jumps abo...
496	Five Stars	5.0	5 Jan. 2016	By\n \n Paul Cavanagh\n \n on 5 Jan. 2...	Great model
497	Fantastic detail! A beautiful model traction e...	5.0	23 Nov. 2015	By\n \n JET\n \n on 23 Nov. 2015	Fantastic detail! A beautiful model traction ...
498	very good quality	5.0	7 July 2013	By\n \n Storm\n \n on 7 July 2013	easy to couple with other models, great to ex...
499	Excellent	5.0	30 April 2011	By\n \n Ella\n \n on 30 April 2011	I bought this for my 2 year old grandson and ...

500 rows × 5 columns

Exercise(1)

- Read in csv file, create Dataframe and check the shape.



```
train_df = pd.read_csv("amazon_train_df.csv")
test_df = pd.read_csv("amazon_test_df.csv")
print(train_df.shape, test_df.shape)

tmp_tr = train_df
tmp_te = test_df
```

```
(500, 5) (25, 5)
```

```
train_df.head()
```

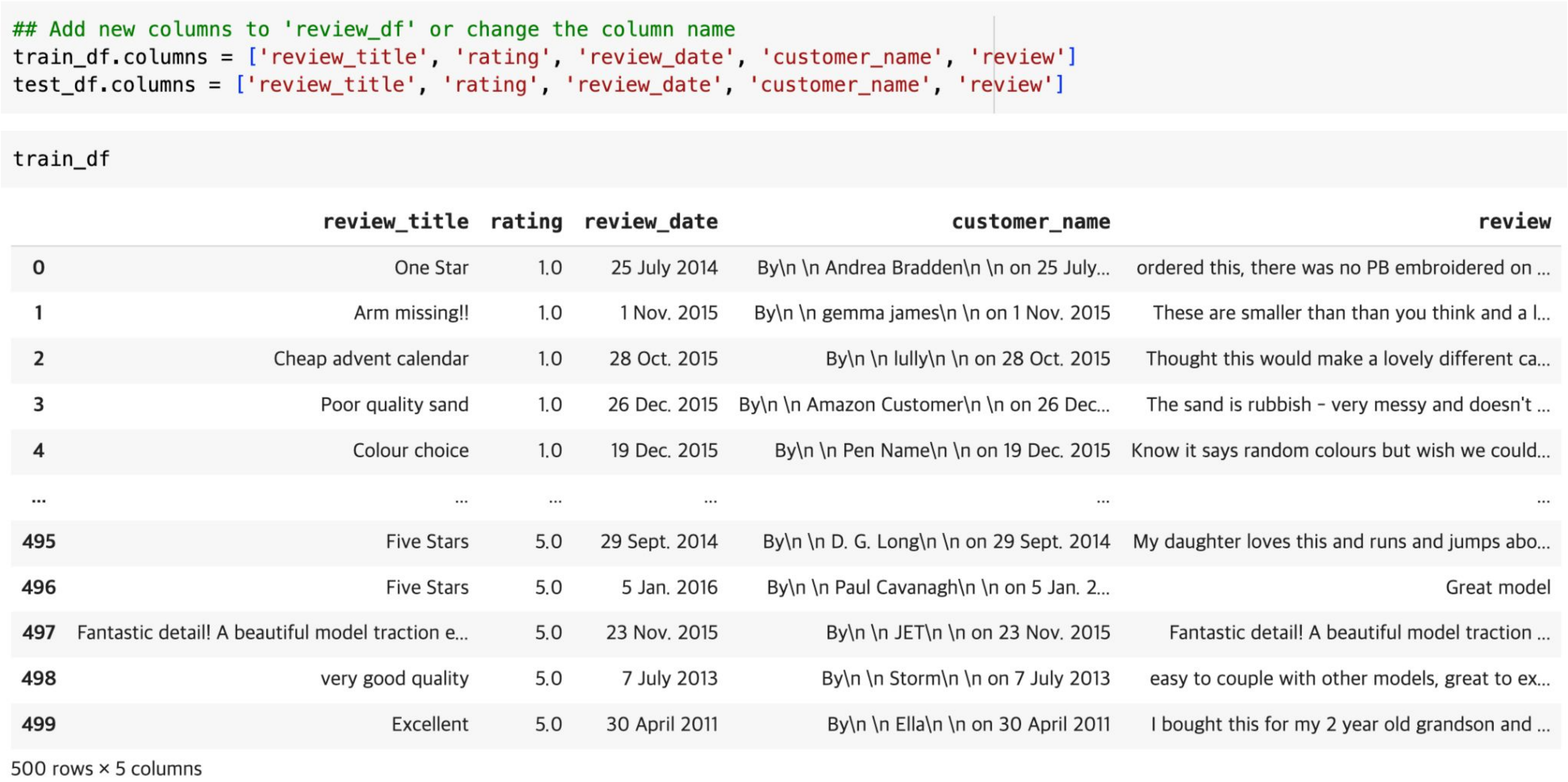
	0	1	2	3	4
0	One Star	1.0	25 July 2014	By\n \n Andrea Bradden\n \n on 25 July...	ordered this, there was no PB embroidered on ...
1	Arm missing!!	1.0	1 Nov. 2015	By\n \n gemma james\n \n on 1 Nov. 2015	These are smaller than than you think and a l...
2	Cheap advent calendar	1.0	28 Oct. 2015	By\n \n lully\n \n on 28 Oct. 2015	Thought this would make a lovely different ca...
3	Poor quality sand	1.0	26 Dec. 2015	By\n \n Amazon Customer\n \n on 26 Dec...	The sand is rubbish - very messy and doesn't ...
4	Colour choice	1.0	19 Dec. 2015	By\n \n Pen Name\n \n on 19 Dec. 2015	Know it says random colours but wish we could...

```
train_df.iloc[4,4]
```

```
' Know it says random colours but wish we could choose. Red is quite dark and my girlie girl won't like
it and will surely wonder why Santa has given her a colour she doesn't like! '
```

Exercise(2) - Cleaning

- Add new columns to review_df or change the column name



```
## Add new columns to 'review_df' or change the column name
train_df.columns = ['review_title', 'rating', 'review_date', 'customer_name', 'review']
test_df.columns = ['review_title', 'rating', 'review_date', 'customer_name', 'review']
```

```
train_df
```

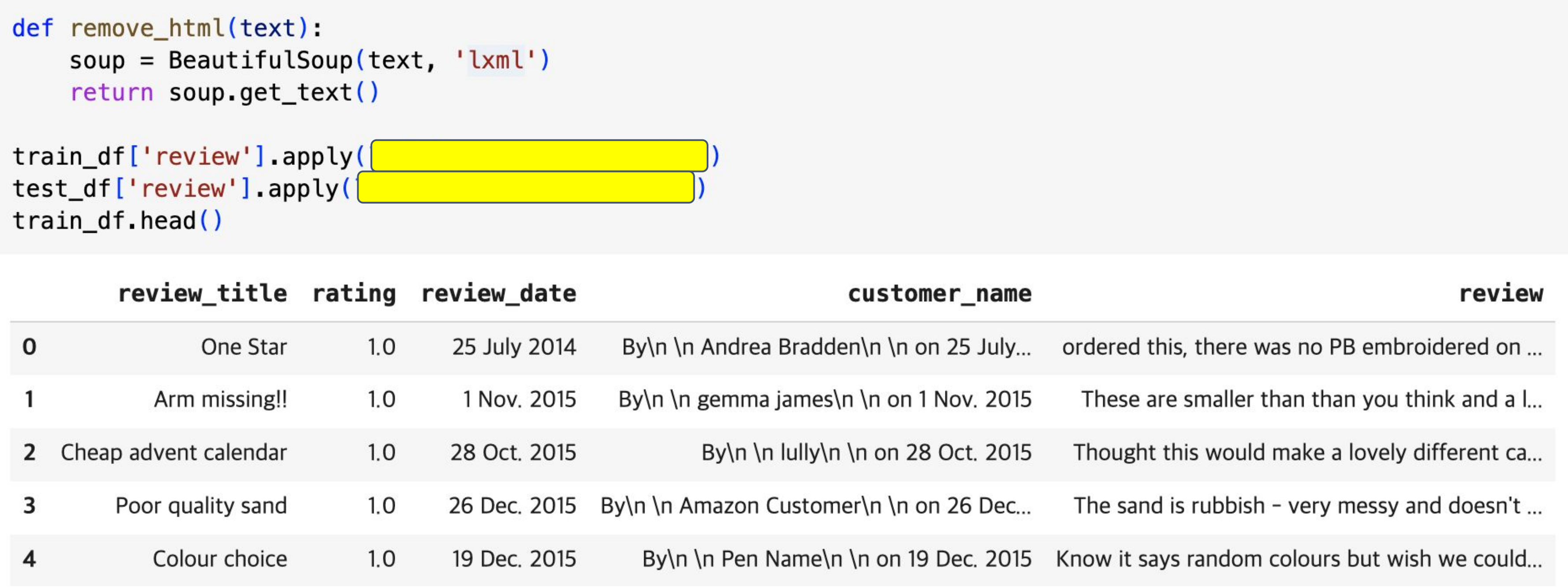
	review_title	rating	review_date	customer_name	review
0	One Star	1.0	25 July 2014	By\n \n Andrea Bradden\n \n on 25 July...	ordered this, there was no PB embroidered on ...
1	Arm missing!!	1.0	1 Nov. 2015	By\n \n gemma james\n \n on 1 Nov. 2015	These are smaller than than you think and a l...
2	Cheap advent calendar	1.0	28 Oct. 2015	By\n \n lully\n \n on 28 Oct. 2015	Thought this would make a lovely different ca...
3	Poor quality sand	1.0	26 Dec. 2015	By\n \n Amazon Customer\n \n on 26 Dec...	The sand is rubbish - very messy and doesn't ...
4	Colour choice	1.0	19 Dec. 2015	By\n \n Pen Name\n \n on 19 Dec. 2015	Know it says random colours but wish we could...
...	...	...	...	...	...
495	Five Stars	5.0	29 Sept. 2014	By\n \n D. G. Long\n \n on 29 Sept. 2014	My daughter loves this and runs and jumps abo...
496	Five Stars	5.0	5 Jan. 2016	By\n \n Paul Cavanagh\n \n on 5 Jan. 2...	Great model
497	Fantastic detail! A beautiful model traction e...	5.0	23 Nov. 2015	By\n \n JET\n \n on 23 Nov. 2015	Fantastic detail! A beautiful model traction ...
498	very good quality	5.0	7 July 2013	By\n \n Storm\n \n on 7 July 2013	easy to couple with other models, great to ex...
499	Excellent	5.0	30 April 2011	By\n \n Ella\n \n on 30 April 2011	I bought this for my 2 year old grandson and ...

500 rows × 5 columns

Exercise(2) - Cleaning

- Removes HTML tags from a text and extracts plain text only.

  use lambda.

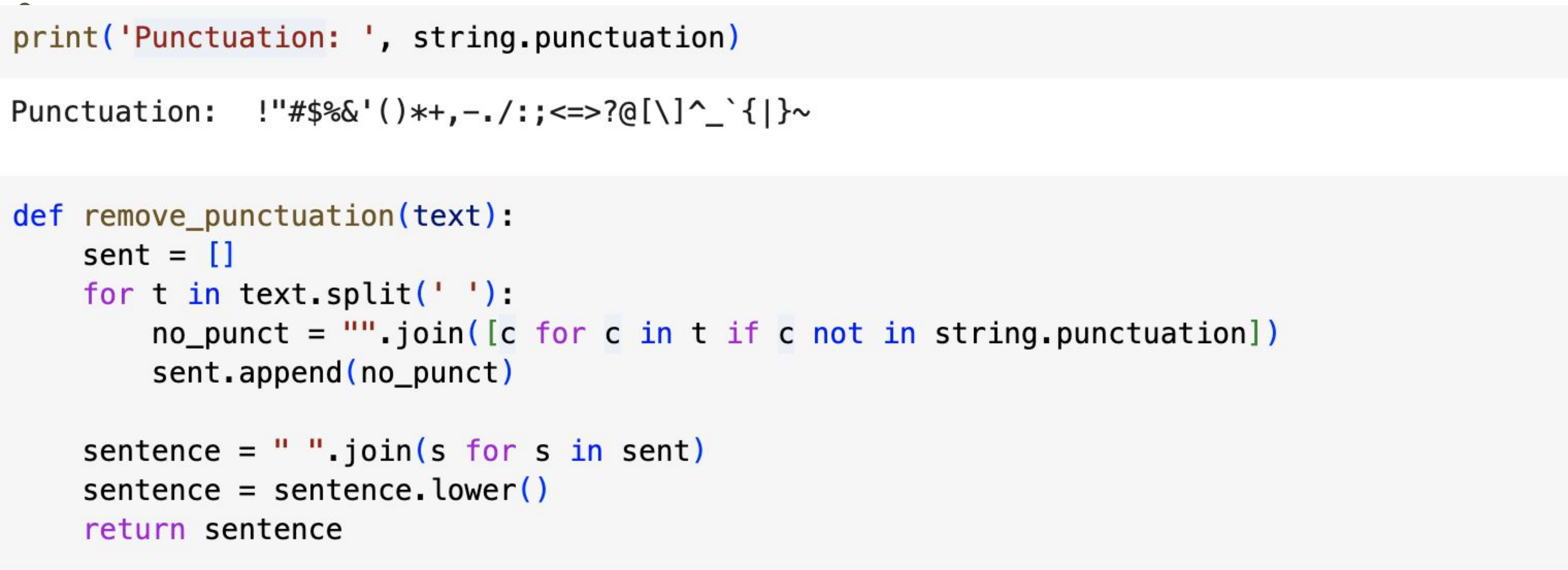


```
def remove_html(text):
    soup = BeautifulSoup(text, 'lxml')
    return soup.get_text()

train_df['review'].apply(                        )
test_df['review'].apply(                        )
train_df.head()
```

	review_title	rating	review_date	customer_name	review
0	One Star	1.0	25 July 2014	By\n \n Andrea Bradden\n \n on 25 July...	ordered this, there was no PB embroidered on ...
1	Arm missing!!	1.0	1 Nov. 2015	By\n \n gemma james\n \n on 1 Nov. 2015	These are smaller than than you think and a l...
2	Cheap advent calendar	1.0	28 Oct. 2015	By\n \n lully\n \n on 28 Oct. 2015	Thought this would make a lovely different ca...
3	Poor quality sand	1.0	26 Dec. 2015	By\n \n Amazon Customer\n \n on 26 Dec...	The sand is rubbish - very messy and doesn't ...
4	Colour choice	1.0	19 Dec. 2015	By\n \n Pen Name\n \n on 19 Dec. 2015	Know it says random colours but wish we could...

Exercise(3) - Remove punctuation & lower case

- Removes all punctuation from the sentence and converts it to lowercase.



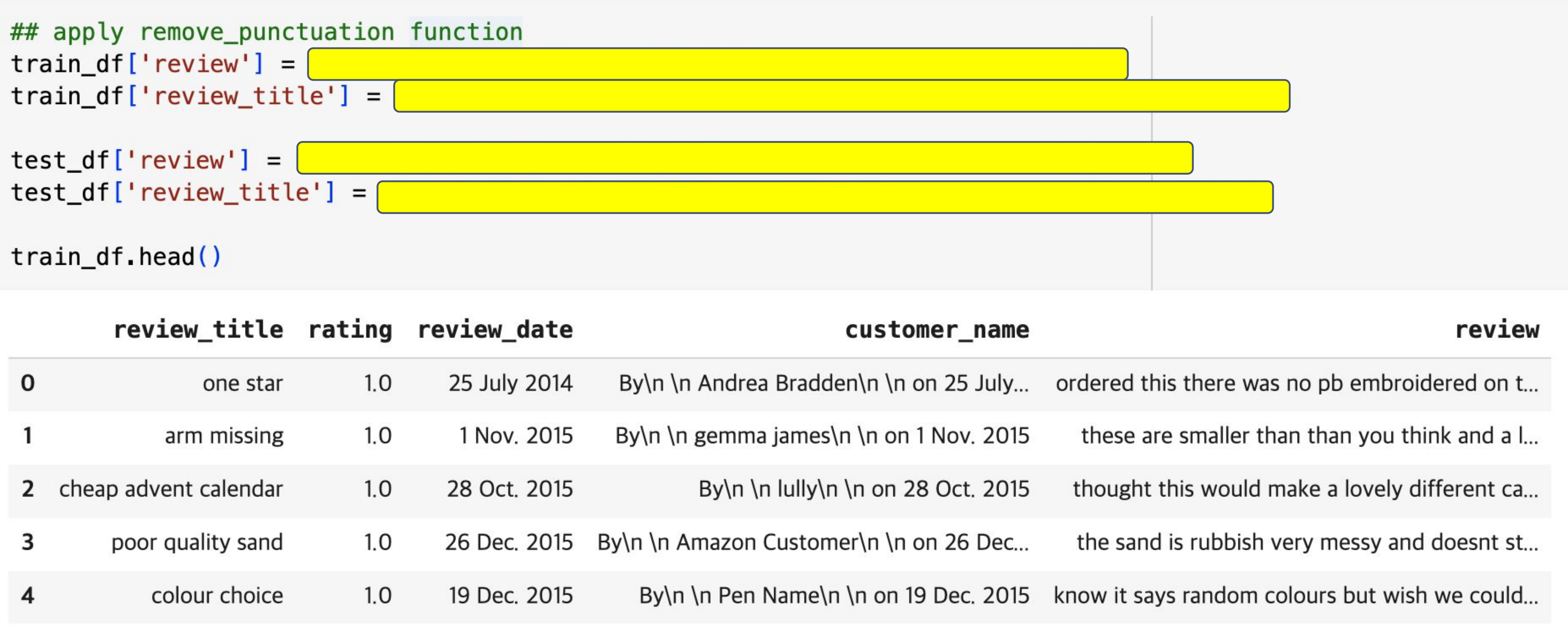
```
print('Punctuation: ', string.punctuation)
```

```
Punctuation:  !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

```
def remove_punctuation(text):
    sent = []
    for t in text.split(' '):
        no_punct = "".join([c for c in t if c not in string.punctuation])
        sent.append(no_punct)

    sentence = " ".join(s for s in sent)
    sentence = sentence.lower()
    return sentence
```

Exercise(3) - Remove punctuation & lower case

- Removes all punctuation from the sentence and converts it to lowercase.

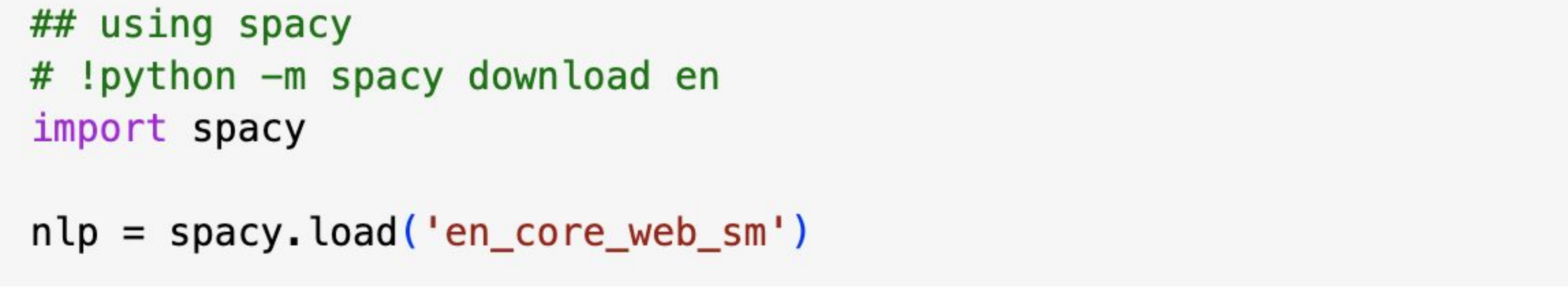


```
## apply remove_punctuation function
train_df['review'] = [redacted]
train_df['review_title'] = [redacted]

test_df['review'] = [redacted]
test_df['review_title'] = [redacted]

train_df.head()
```

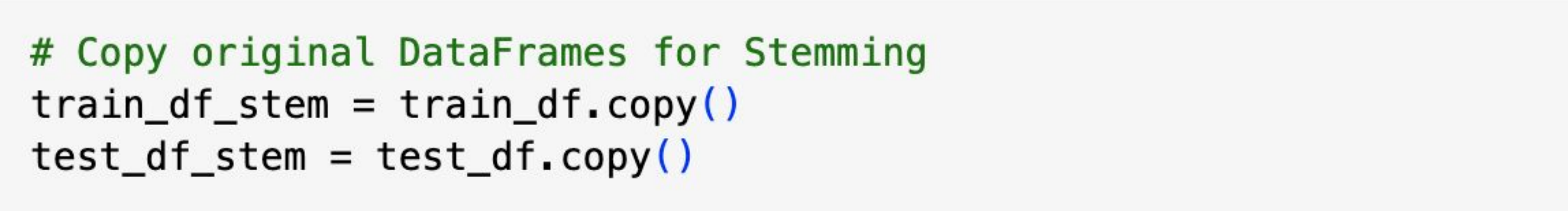
	review_title	rating	review_date	customer_name	review
0	one star	1.0	25 July 2014	By\n \n Andrea Bradden\n \n on 25 July...	ordered this there was no pb embroidered on t...
1	arm missing	1.0	1 Nov. 2015	By\n \n gemma james\n \n on 1 Nov. 2015	these are smaller than than you think and a l...
2	cheap advent calendar	1.0	28 Oct. 2015	By\n \n lully\n \n on 28 Oct. 2015	thought this would make a lovely different ca...
3	poor quality sand	1.0	26 Dec. 2015	By\n \n Amazon Customer\n \n on 26 Dec...	the sand is rubbish very messy and doesnt st...
4	colour choice	1.0	19 Dec. 2015	By\n \n Pen Name\n \n on 19 Dec. 2015	know it says random colours but wish we could...

Exercise(4) - Lemmatization or Stemming

- Sets up spaCy for English text processing and creates copies of the original DataFrames for preprocessing



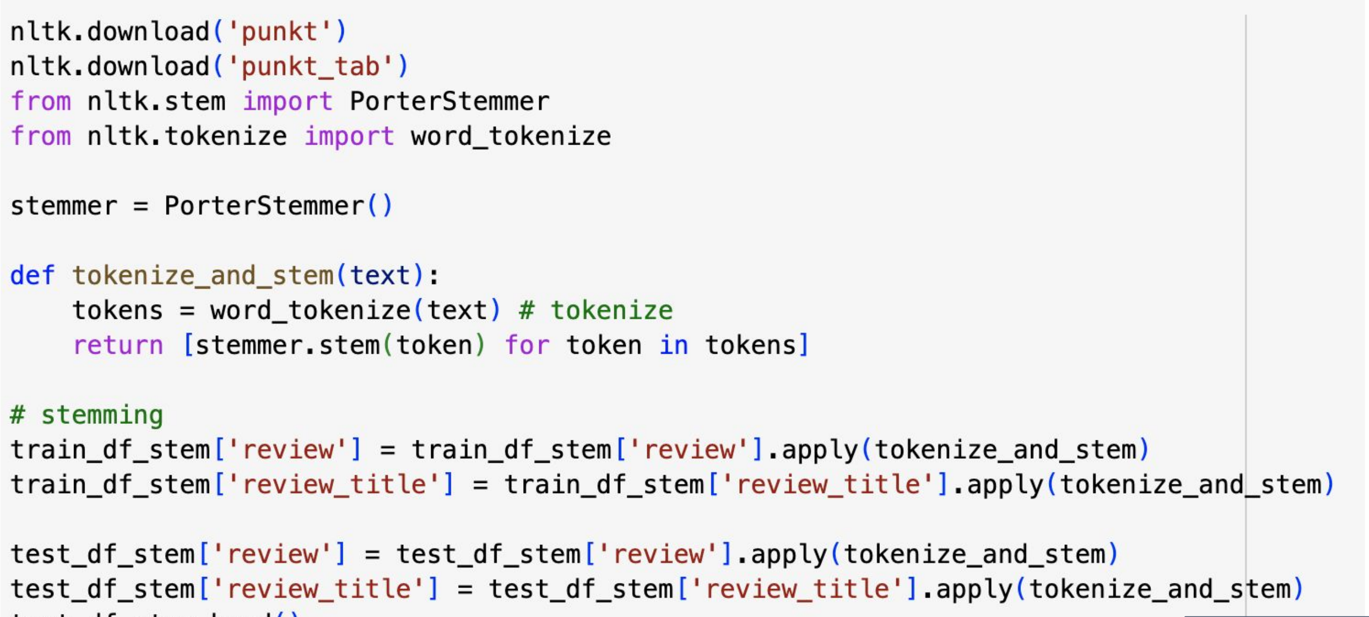
```
## using spacy
# !python -m spacy download en
import spacy

nlp = spacy.load('en_core_web_sm')
```



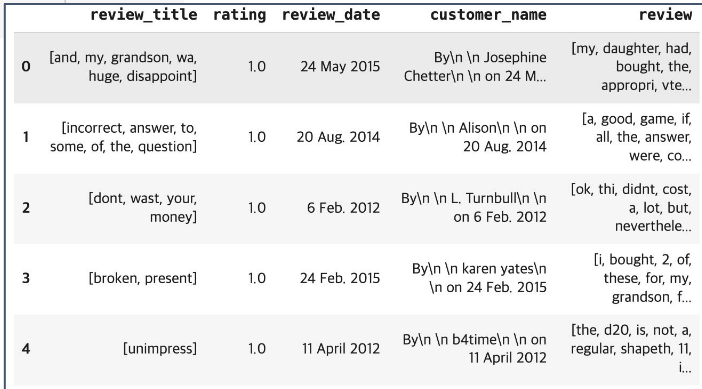
```
# Copy original DataFrames for Stemming
train_df_stem = train_df.copy()
test_df_stem = test_df.copy()
```

Exercise(4) - Lemmatization or Stemming

- Use NLTK to tokenize and stem the text in the 'review' and 'review_title' columns of both the training and test datasets



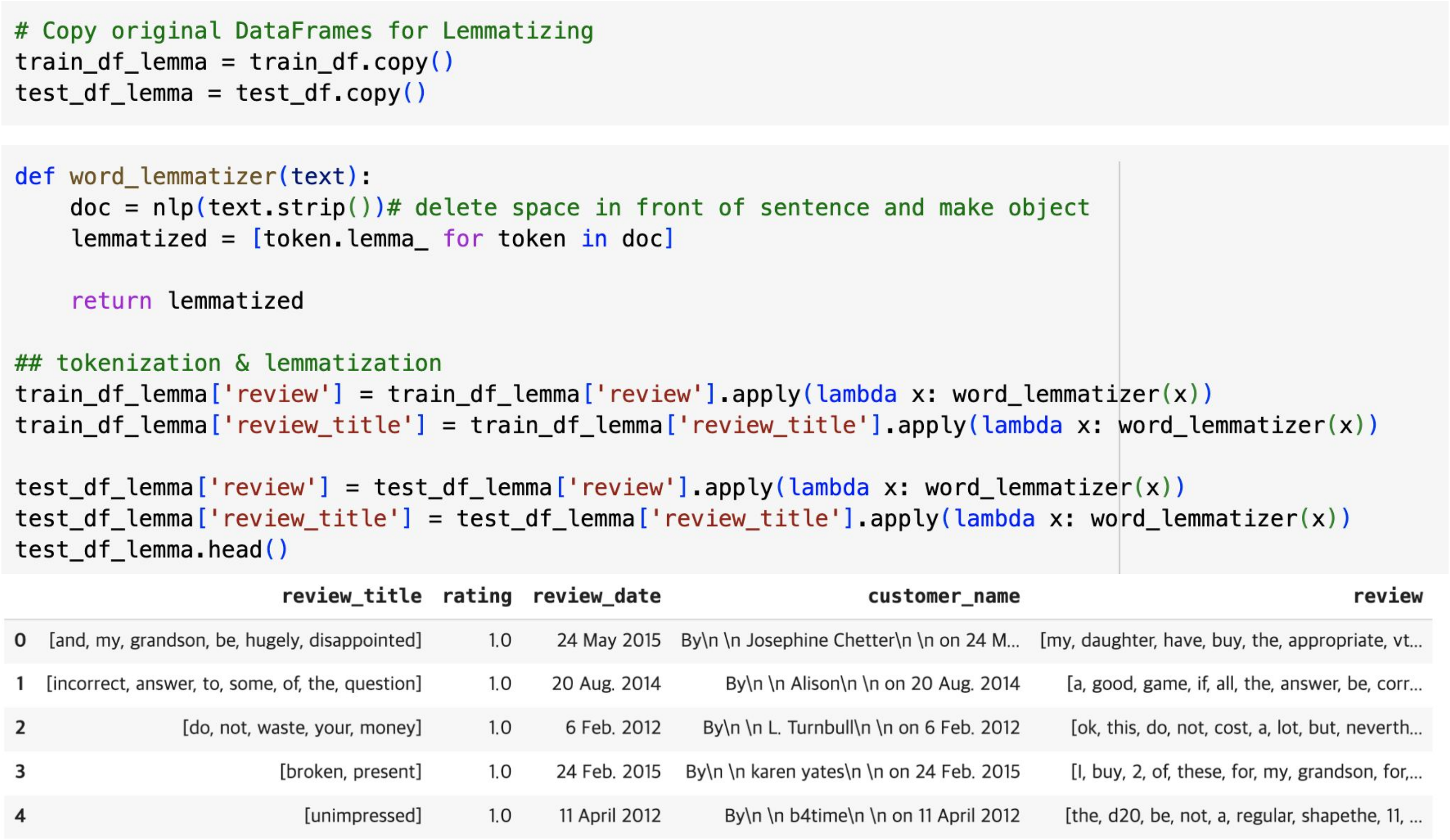
```
nltk.download('punkt')
nltk.download('punkt_tab')
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

stemmer = PorterStemmer()

def tokenize_and_stem(text):
    tokens = word_tokenize(text) # tokenize
    return [stemmer.stem(token) for token in tokens]

# stemming
train_df_stem['review'] = train_df_stem['review'].apply(tokenize_and_stem)
train_df_stem['review_title'] = train_df_stem['review_title'].apply(tokenize_and_stem)

test_df_stem['review'] = test_df_stem['review'].apply(tokenize_and_stem)
test_df_stem['review_title'] = test_df_stem['review_title'].apply(tokenize_and_stem)
test_df_stem.head()
```



	review_title	rating	review_date	customer_name	review
0	[and, my, grandson, wa, huge, disappoint]	1.0	24 May 2015	By\n \n Josephine Chetter\n \n on 24 M...	[my, daughter, had, bought, the, appropri, vte...
1	[incorrect, answer, to, some, of, the, question]	1.0	20 Aug. 2014	By\n \n Alison\n \n on 20 Aug. 2014	[a, good, game, if, all, the, answer, were, co...
2	[dont, wast, your, money]	1.0	6 Feb. 2012	By\n \n L. Turnbull\n \n on 6 Feb. 2012	[ok, thi, didnt, cost, a, lot, but, neverthele...
3	[broken, present]	1.0	24 Feb. 2015	By\n \n karen yates\n \n on 24 Feb. 2015	[i, bought, 2, of, these, for, my, grandson, f...
4	[unimpress]	1.0	11 April 2012	By\n \n b4time\n \n on 11 April 2012	[the, d20, is, not, a, regular, shapeth, 11, i...

Exercise(4) - Lemmatization or Stemming

- Use NLTK to tokenize and lemmatize the text in the 'review' and 'review_title' columns.



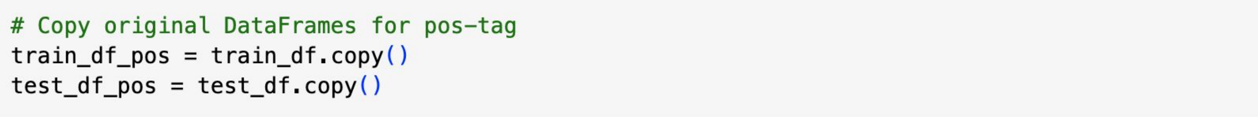
```
# Copy original DataFrames for Lemmatizing
train_df_lemma = train_df.copy()
test_df_lemma = test_df.copy()
```

```
def word_lemmatizer(text):
    doc = nlp(text.strip())# delete space in front of sentence and make object
    lemmatized = [token.lemma_ for token in doc]

    return lemmatized

## tokenization & lemmatization
train_df_lemma['review'] = train_df_lemma['review'].apply(lambda x: word_lemmatizer(x))
train_df_lemma['review_title'] = train_df_lemma['review_title'].apply(lambda x: word_lemmatizer(x))

test_df_lemma['review'] = test_df_lemma['review'].apply(lambda x: word_lemmatizer(x))
test_df_lemma['review_title'] = test_df_lemma['review_title'].apply(lambda x: word_lemmatizer(x))
test_df_lemma.head()
```

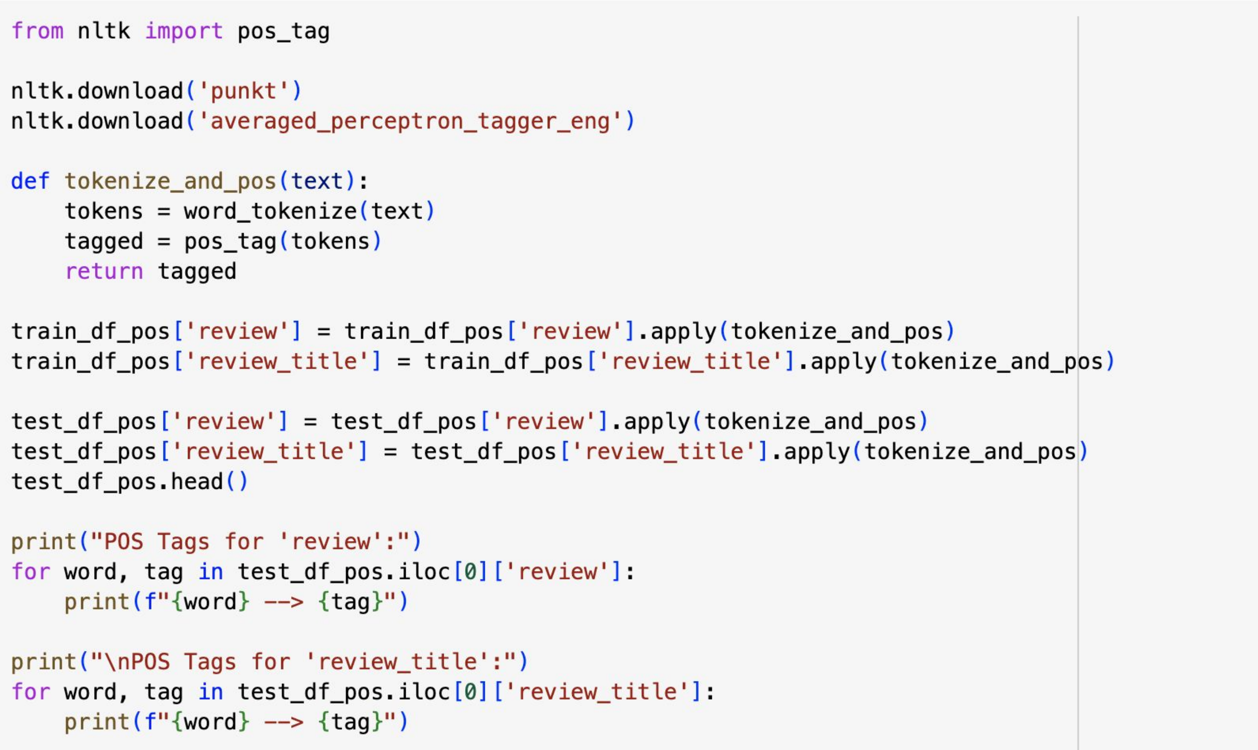
	review_title	rating	review_date	customer_name	review
0	[and, my, grandson, be, hugely, disappointed]	1.0	24 May 2015	By\n \n Josephine Chetter\n \n on 24 M...	[my, daughter, have, buy, the, appropriate, vt...
1	[incorrect, answer, to, some, of, the, question]	1.0	20 Aug. 2014	By\n \n Alison\n \n on 20 Aug. 2014	[a, good, game, if, all, the, answer, be, corr...
2	[do, not, waste, your, money]	1.0	6 Feb. 2012	By\n \n L. Turnbull\n \n on 6 Feb. 2012	[ok, this, do, not, cost, a, lot, but, neverth...
3	[broken, present]	1.0	24 Feb. 2015	By\n \n karen yates\n \n on 24 Feb. 2015	[I, buy, 2, of, these, for, my, grandson, for,...
4	[unimpressed]	1.0	11 April 2012	By\n \n b4time\n \n on 11 April 2012	[the, d20, be, not, a, regular, shapethe, 11, ...

Exercise(4) - Lemmatization or Stemming

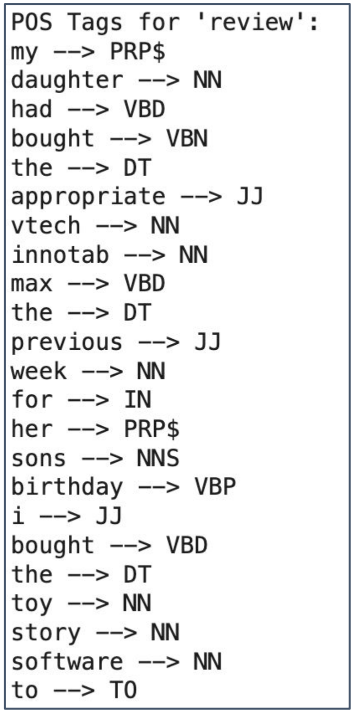
- Use NLTK to tokenize and assign POS tags to the text in the 'review' and 'review_title' columns.



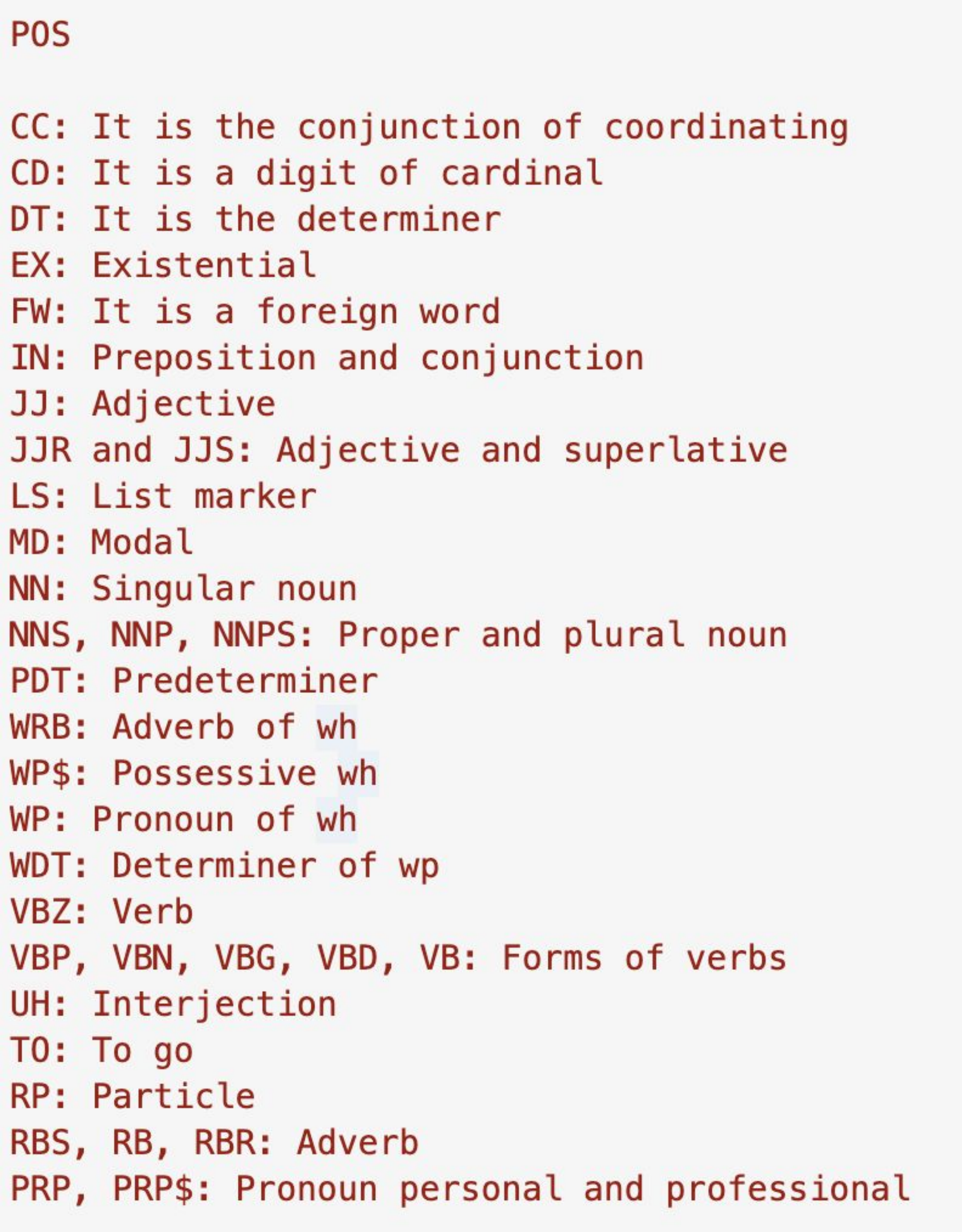
```
# Copy original DataFrames for pos-tag
train_df_pos = train_df.copy()
test_df_pos = test_df.copy()
```



```
from nltk import pos_tag

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger_eng')

def tokenize_and_pos(text):
    tokens = word_tokenize(text)
    tagged = pos_tag(tokens)
    return tagged

train_df_pos['review'] = train_df_pos['review'].apply(tokenize_and_pos)
train_df_pos['review_title'] = train_df_pos['review_title'].apply(tokenize_and_pos)

test_df_pos['review'] = test_df_pos['review'].apply(tokenize_and_pos)
test_df_pos['review_title'] = test_df_pos['review_title'].apply(tokenize_and_pos)
test_df_pos.head()

print("POS Tags for 'review':")
for word, tag in test_df_pos.iloc[0]['review']:
    print(f"{word} --> {tag}")

print("\nPOS Tags for 'review_title':")
for word, tag in test_df_pos.iloc[0]['review_title']:
    print(f"{word} --> {tag}")
```



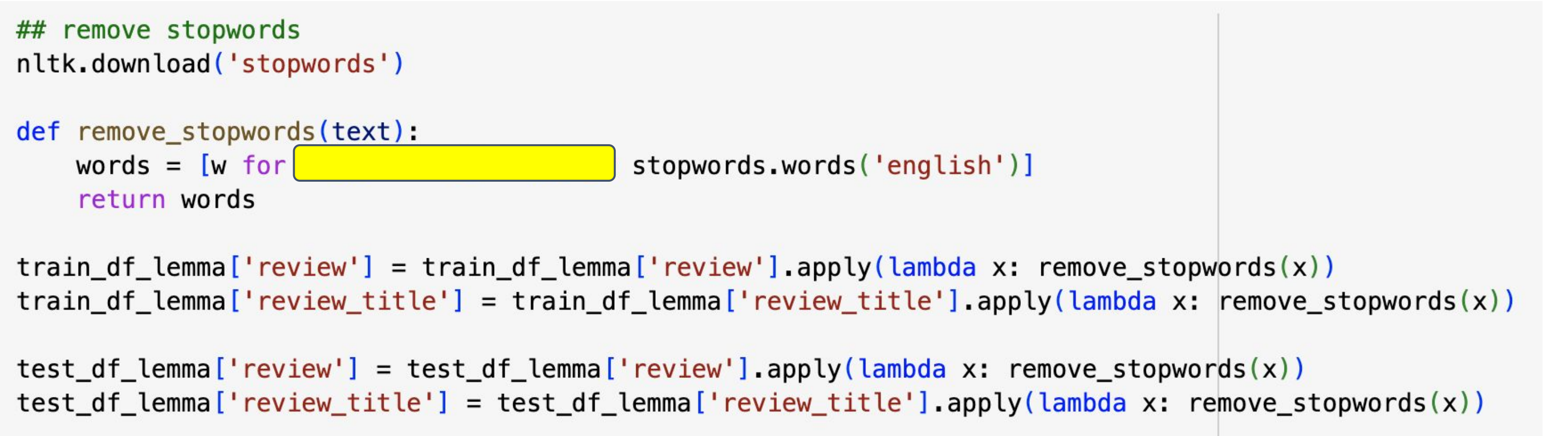
```
POS Tags for 'review':
my --> PRP$
daughter --> NN
had --> VBD
bought --> VBN
the --> DT
appropriate --> JJ
vtech --> NN
innotab --> NN
max --> VBD
the --> DT
previous --> JJ
week --> NN
for --> IN
her --> PRP$
sons --> NNS
birthday --> VBP
i --> JJ
bought --> VBD
the --> DT
toy --> NN
story --> NN
software --> NN
to --> TO
```



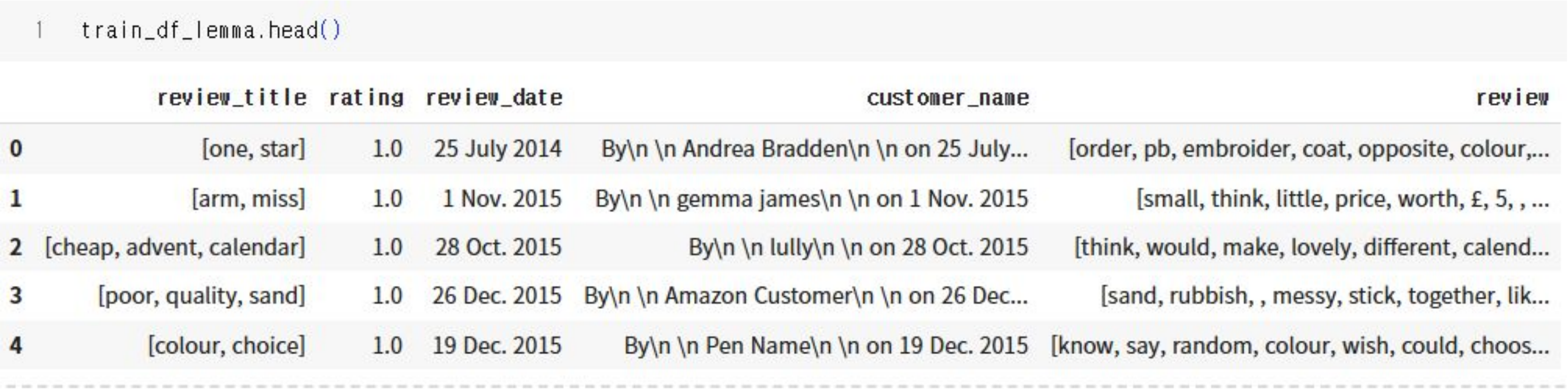
```
POS

CC: It is the conjunction of coordinating
CD: It is a digit of cardinal
DT: It is the determiner
EX: Existential
FW: It is a foreign word
IN: Preposition and conjunction
JJ: Adjective
JJR and JJS: Adjective and superlative
LS: List marker
MD: Modal
NN: Singular noun
NNS, NNP, NNPS: Proper and plural noun
PDT: Predeterminer
WRB: Adverb of wh
WP$: Possessive wh
WP: Pronoun of wh
WDT: Determiner of wp
VBZ: Verb
VBP, VBN, VBG, VBD, VB: Forms of verbs
UH: Interjection
TO: To go
RP: Particle
RBS, RB, RBR: Adverb
PRP, PRP$: Pronoun personal and professional
```

Exercise(5) - Removing stop words

- Remove English stop words from the 'review' and 'review_title' columns.



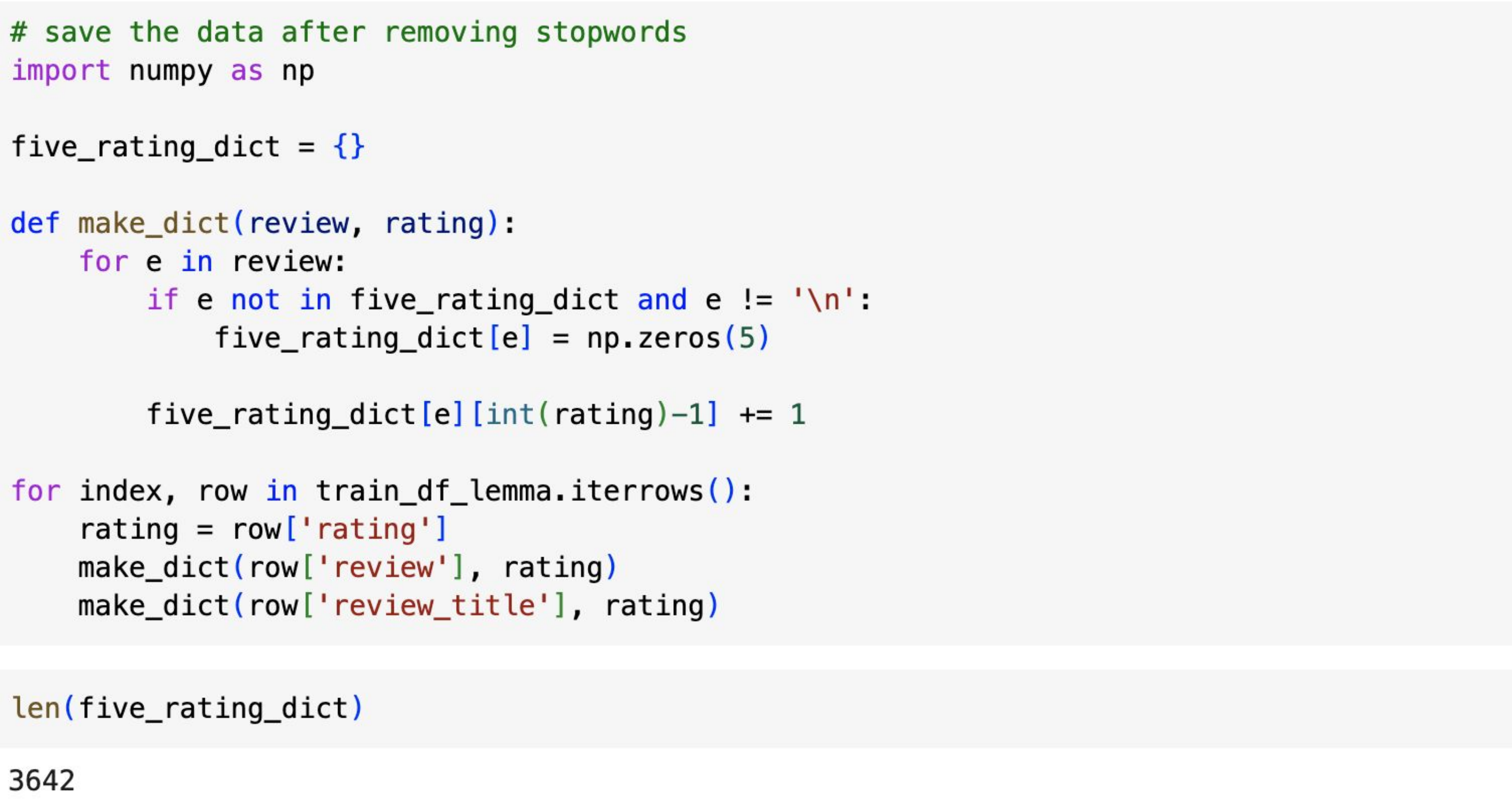
```
## remove stopwords
nltk.download('stopwords')

def remove_stopwords(text):
    words = [w for                          stopwords.words('english')]
    return words

train_df_lemma['review'] = train_df_lemma['review'].apply(lambda x: remove_stopwords(x))
train_df_lemma['review_title'] = train_df_lemma['review_title'].apply(lambda x: remove_stopwords(x))

test_df_lemma['review'] = test_df_lemma['review'].apply(lambda x: remove_stopwords(x))
test_df_lemma['review_title'] = test_df_lemma['review_title'].apply(lambda x: remove_stopwords(x))
```



```
train_df_lemma.head()
```

	review_title	rating	review_date	customer_name	review
0	[one, star]	1.0	25 July 2014	By\n \n Andrea Bradden\n \n on 25 July...	[order, pb, embroider, coat, opposite, colour,...
1	[arm, miss]	1.0	1 Nov. 2015	By\n \n gemma james\n \n on 1 Nov. 2015	[small, think, little, price, worth, £, 5, , ...
2	[cheap, advent, calendar]	1.0	28 Oct. 2015	By\n \n lully\n \n on 28 Oct. 2015	[think, would, make, lovely, different, calend...
3	[poor, quality, sand]	1.0	26 Dec. 2015	By\n \n Amazon Customer\n \n on 26 Dec...	[sand, rubbish, , messy, stick, together, lik...
4	[colour, choice]	1.0	19 Dec. 2015	By\n \n Pen Name\n \n on 19 Dec. 2015	[know, say, random, colour, wish, could, choos...

Exercise(6) - Making Dictionary

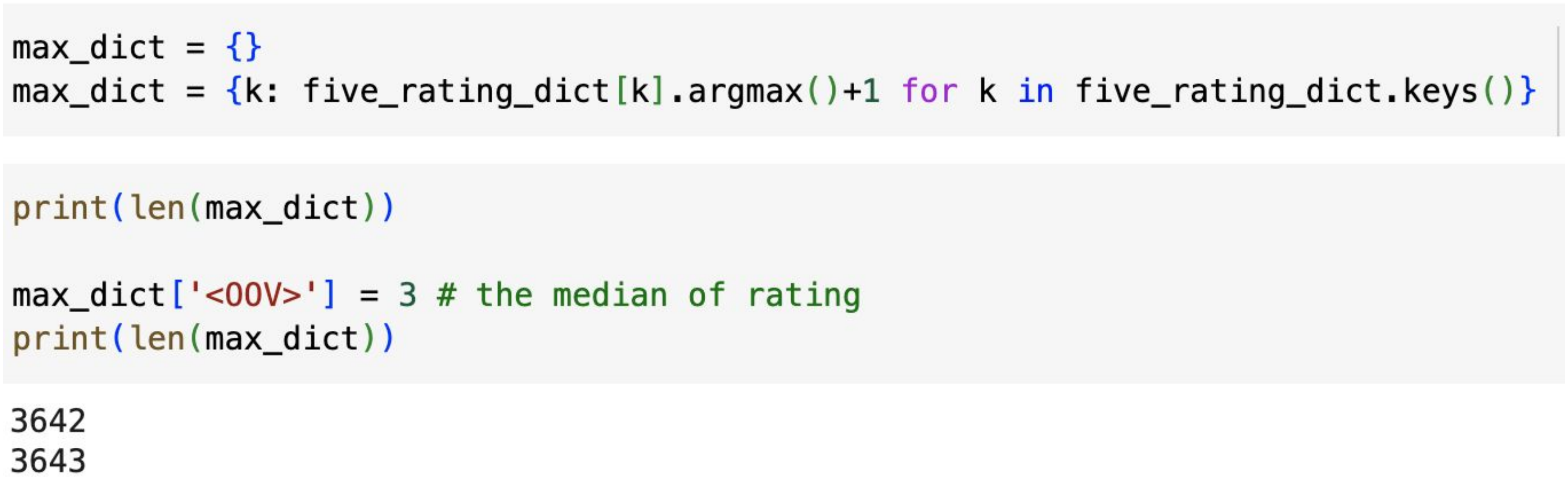


```
# save the data after removing stopwords
import numpy as np

five_rating_dict = {}

def make_dict(review, rating):
    for e in review:
        if e not in five_rating_dict and e != '\n':
            five_rating_dict[e] = np.zeros(5)

        five_rating_dict[e][int(rating)-1] += 1

for index, row in train_df_lemma.iterrows():
    rating = row['rating']
    make_dict(row['review'], rating)
    make_dict(row['review_title'], rating)
```

```
len(five_rating_dict)
```

```
3642
```

Exercise(6) - Making Dictionary

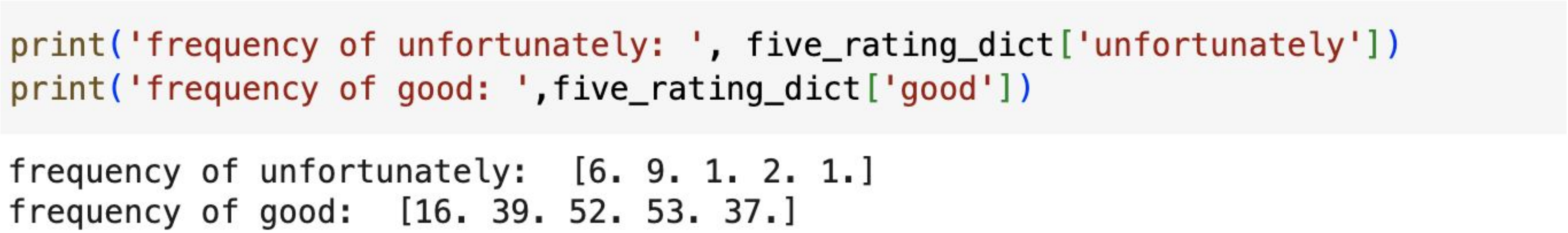


```
max_dict = {}
max_dict = {k: five_rating_dict[k].argmax()+1 for k in five_rating_dict.keys()}
```

```
print(len(max_dict))

max_dict['<OOV>'] = 3 # the median of rating
print(len(max_dict))
```

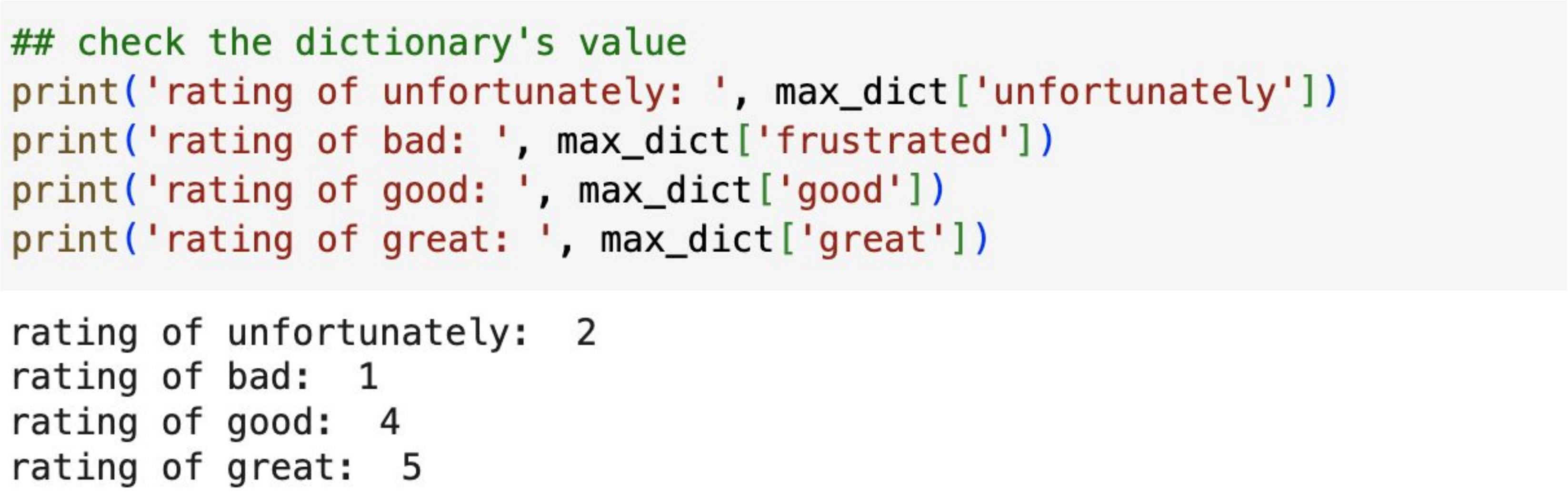
```
3642
3643
```

Exercise(6) - Making Dictionary



```
print('frequency of unfortunately: ', five_rating_dict['unfortunately'])
print('frequency of good: ',five_rating_dict['good'])
```

```
frequency of unfortunately:  [6. 9. 1. 2. 1.]
frequency of good:  [16. 39. 52. 53. 37.]
```

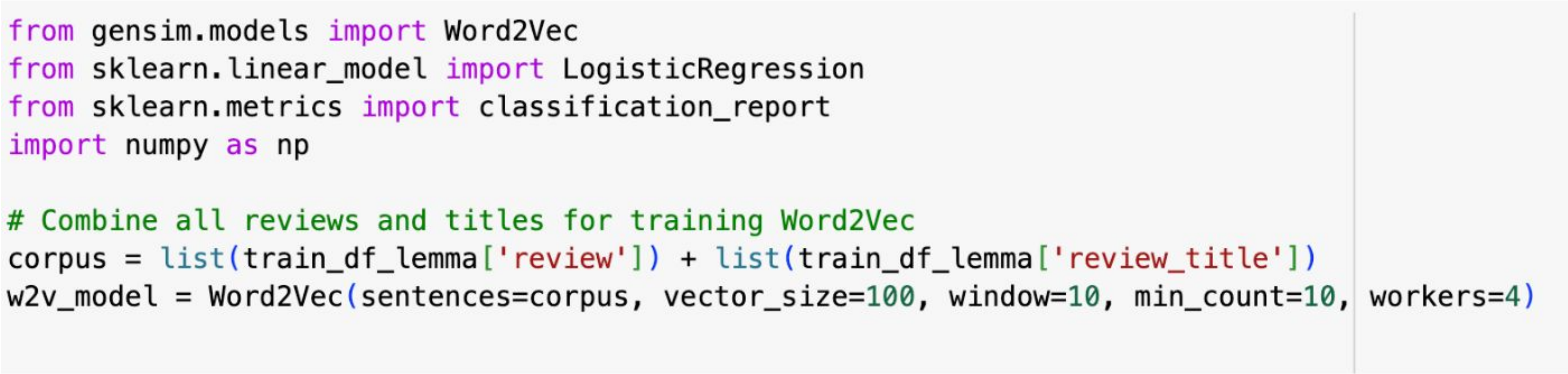


```
## check the dictionary's value
print('rating of unfortunately: ', max_dict['unfortunately'])
print('rating of bad: ', max_dict['frustrated'])
print('rating of good: ', max_dict['good'])
print('rating of great: ', max_dict['great'])
```

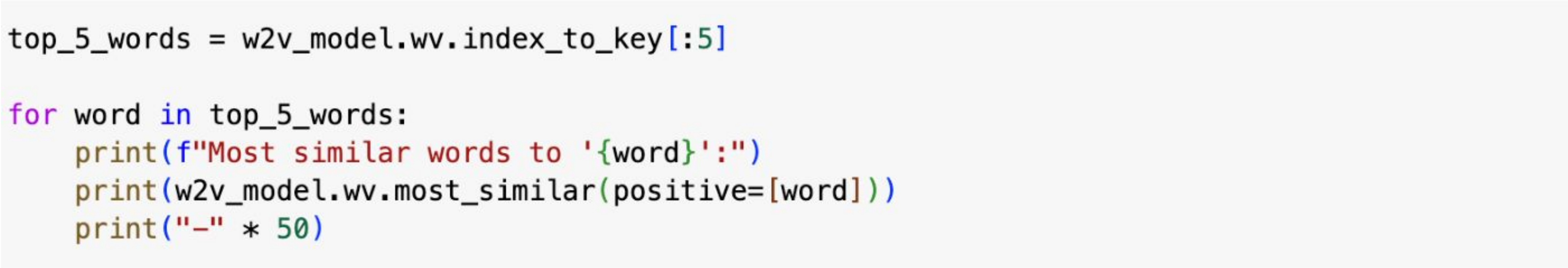
```
rating of unfortunately:  2
rating of bad:  1
rating of good:  4
rating of great:  5
```

Exercise(7)

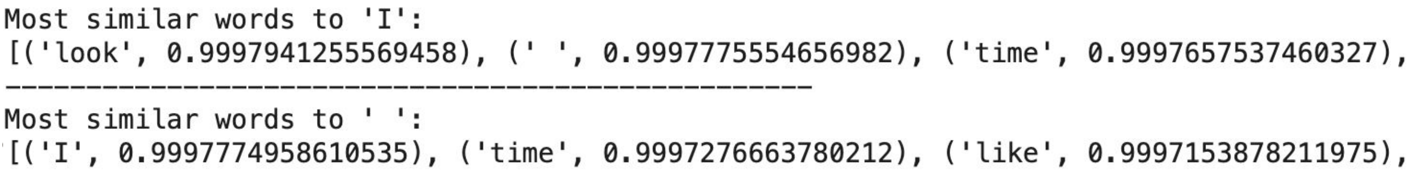
- Train a Word2Vec model on the review and title text, then print the most similar words for the top 5 frequent words in the vocabulary.



```python
from gensim.models import Word2Vec
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
import numpy as np

# Combine all reviews and titles for training Word2Vec
corpus = list(train_df_lemma['review']) + list(train_df_lemma['review_title'])
w2v_model = Word2Vec(sentences=corpus, vector_size=100, window=10, min_count=10, workers=4)
```



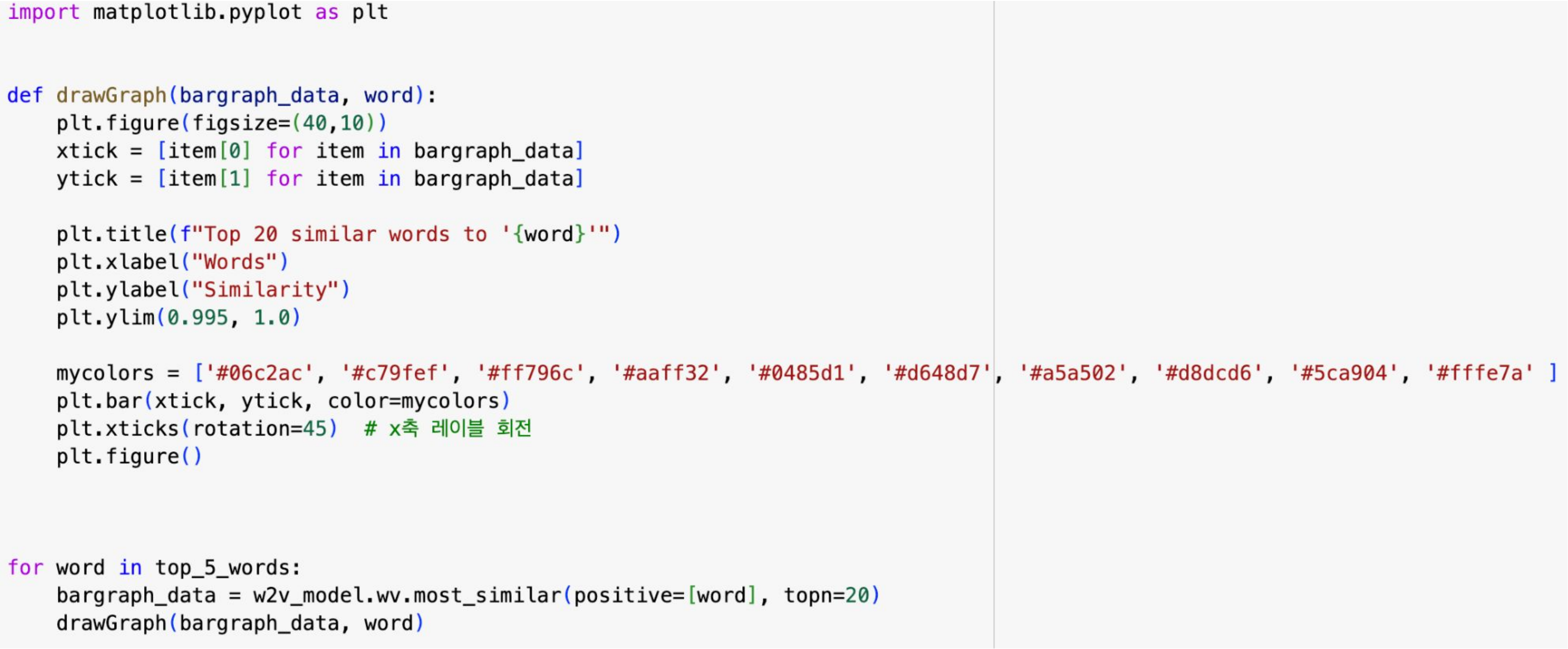
```python
top_5_words = w2v_model.wv.index_to_key[:5]

for word in top_5_words:
    print(f"Most similar words to '{word}':")
    print(w2v_model.wv.most_similar(positive=[word]))
    print("-" * 50)
```



```
Most similar words to 'I':
[('look', 0.9997941255569458), (' ', 0.9997775554656982), ('time', 0.9997657537460327),
--------------------------------------------------
Most similar words to ' ':
[('I', 0.9997774958610535), ('time', 0.9997276663780212), ('like', 0.9997153878211975),
```

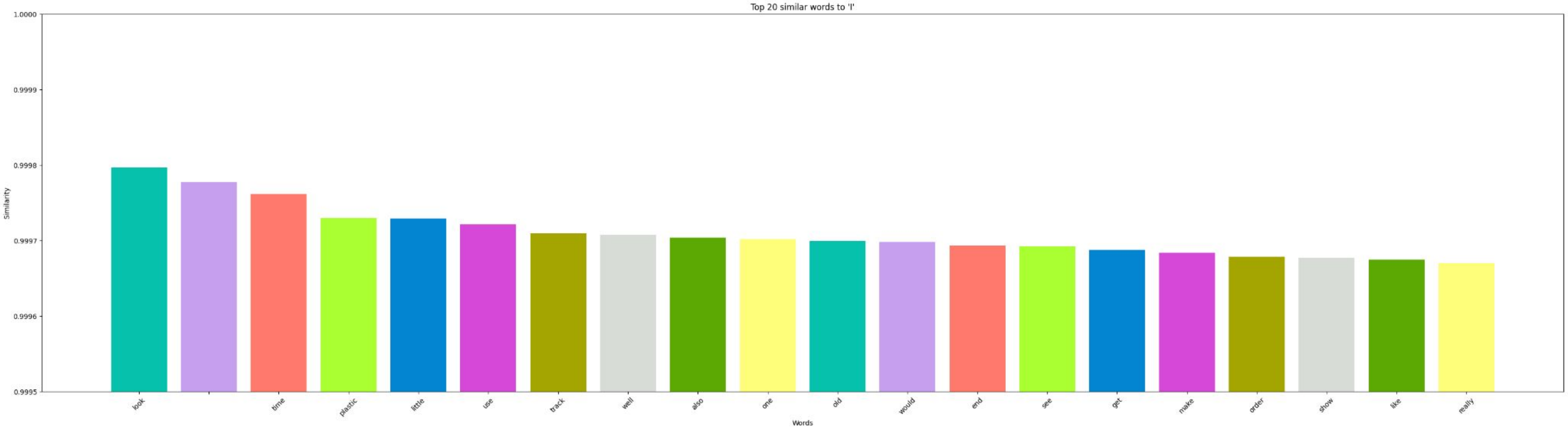
Exercise(7)

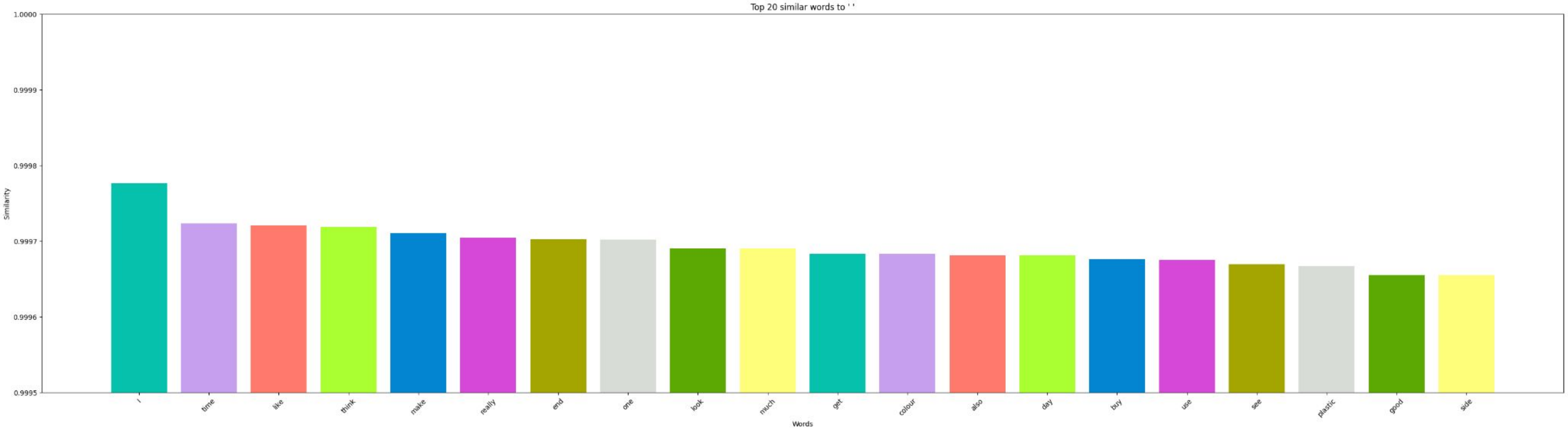
- Visualize the top 20 most similar words to each of the top 5 frequent words using bar charts based on Word2Vec similarity scores.



```
import matplotlib.pyplot as plt


def drawGraph(bargraph_data, word):
    plt.figure(figsize=(40,10))
    xtick = [item[0] for item in bargraph_data]
    ytick = [item[1] for item in bargraph_data]

    plt.title(f"Top 20 similar words to '{word}'")
    plt.xlabel("Words")
    plt.ylabel("Similarity")
    plt.ylim(0.995, 1.0)

    mycolors = ['#06c2ac', '#c79fef', '#ff796c', '#aaff32', '#0485d1', '#d648d7', '#a5a502', '#d8dcd6', '#5ca904', '#fffe7a' ]
    plt.bar(xtick, ytick, color=mycolors)
    plt.xticks(rotation=45)  # x축 레이블 회전
    plt.figure()



for word in top_5_words:
    bargraph_data = w2v_model.wv.most_similar(positive=[word], topn=20)
    drawGraph(bargraph_data, word)
```

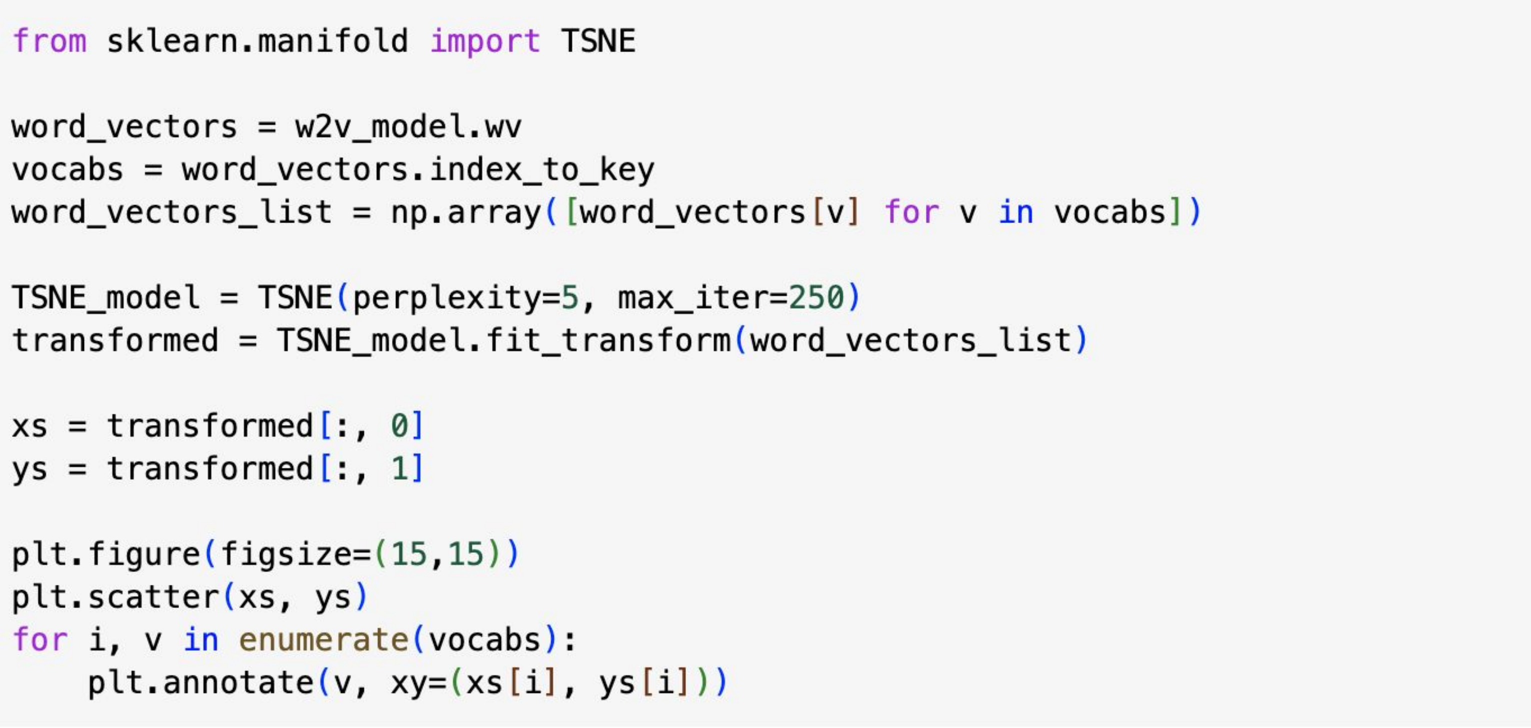
Exercise(7)


Top 20 similar words to 'I'
Similarity
1.0000
0.9999
0.9998
0.9997
0.9996
0.9995
look
time
plastic
little
use
track
well
also
one
old
would
end
see
get
make
order
show
like
really
Words


Top 20 similar words to ' '
Similarity
1.0000
0.9999
0.9998
0.9997
0.9996
0.9995
i
time
like
think
make
really
end
one
look
much
get
colour
also
day
buy
use
see
plastic
good
side
Words

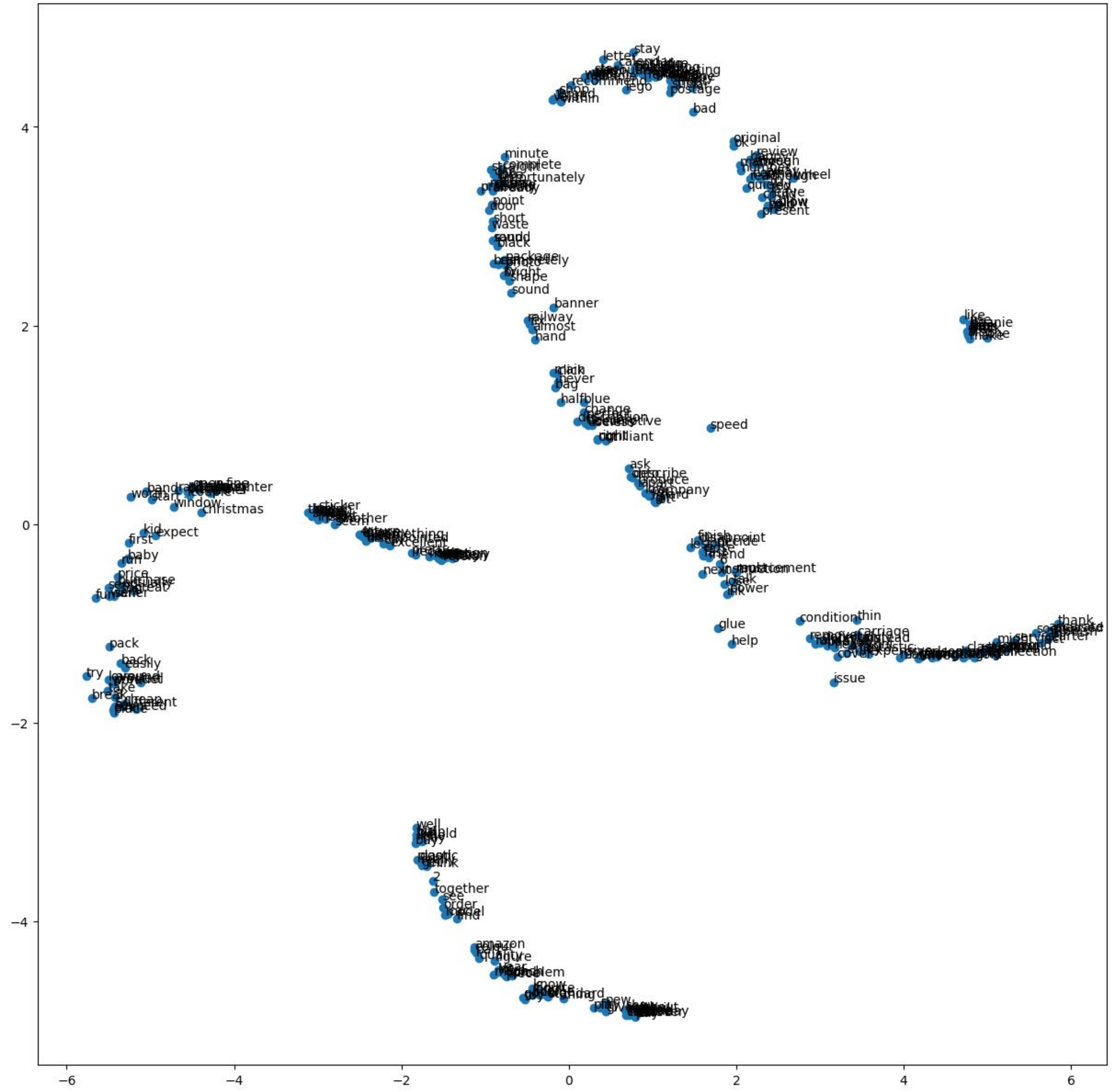
Exercise(8)

- Use t-SNE to reduce the dimensionality of Word2Vec word vectors and visualize them in a 2D scatter plot with word labels.



```
from sklearn.manifold import TSNE

word_vectors = w2v_model.wv
vocabs = word_vectors.index_to_key
word_vectors_list = np.array([word_vectors[v] for v in vocabs])

TSNE_model = TSNE(perplexity=5, max_iter=250)
transformed = TSNE_model.fit_transform(word_vectors_list)

xs = transformed[:, 0]
ys = transformed[:, 1]

plt.figure(figsize=(15,15))
plt.scatter(xs, ys)
for i, v in enumerate(vocabs):
    plt.annotate(v, xy=(xs[i], ys[i]))
```

Exercise(8)

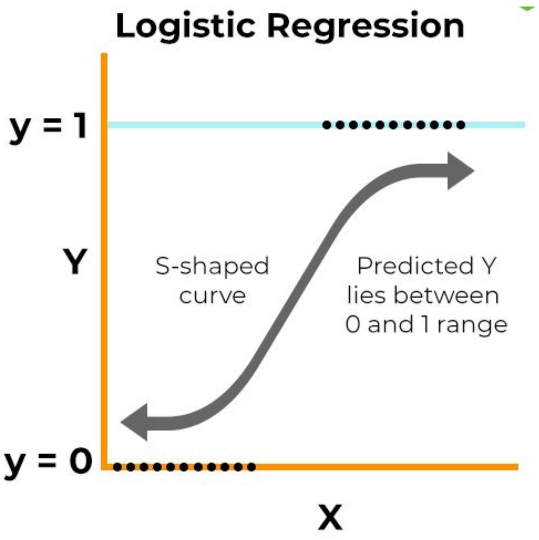

stay
lette
recommen
lego
postage
within
bad
original
ok
review
wheel
present
minute
complete
unfortunately
point
door
short
waste
black
package
bright
shape
sound
banner
railway
almost
hand
never
bag
half
blue
change
brilliant
speed
like
make
ask
describe
company
finish
point
friend
next
statement
look
power
glue
help
condition
thin
carriage
thank
starter
collection
issue
window
christmas
kid
expect
first
baby
price
funnel
pack
back
easily
try
break
sticker
other
excellent
well
together
see
order
find
amazon
picture
problem
standard
new
way
4
2
0
−2
−4
−6
−4
−2
0
2
4
6

Logistic Regression

- **Logistic Regression** is a popular machine learning algorithm used for classification tasks.

- It takes input features and calculates the **probability that the input belongs to a certain class**.

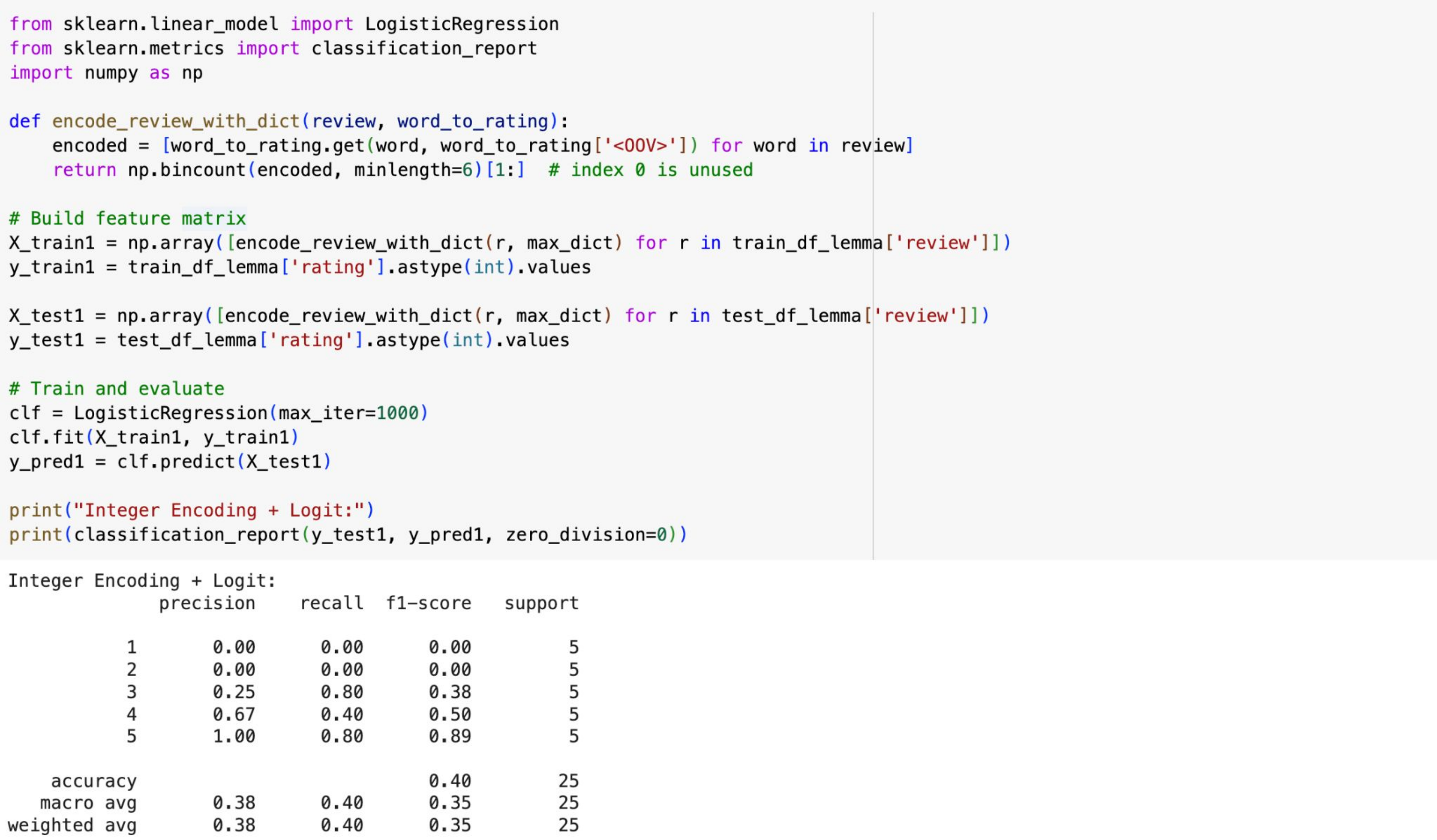
- For example, it can be used to determine whether an email is spam or not, or whether a review is positive or negative.



https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/

Logistic Regression

- Encode reviews using integer mappings, train a logistic regression model.

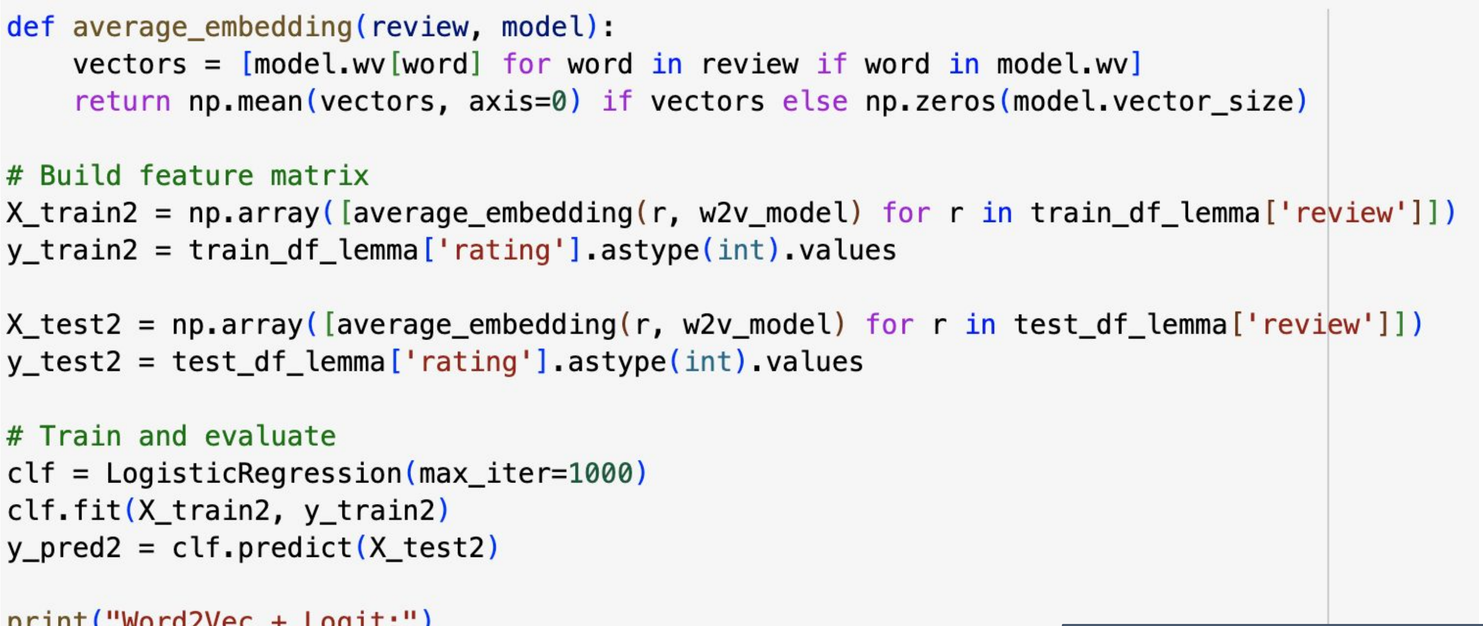


```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
import numpy as np

def encode_review_with_dict(review, word_to_rating):
    encoded = [word_to_rating.get(word, word_to_rating['<OOV>']) for word in review]
    return np.bincount(encoded, minlength=6)[1:]  # index 0 is unused

# Build feature matrix
X_train1 = np.array([encode_review_with_dict(r, max_dict) for r in train_df_lemma['review']])
y_train1 = train_df_lemma['rating'].astype(int).values

X_test1 = np.array([encode_review_with_dict(r, max_dict) for r in test_df_lemma['review']])
y_test1 = test_df_lemma['rating'].astype(int).values

# Train and evaluate
clf = LogisticRegression(max_iter=1000)
clf.fit(X_train1, y_train1)
y_pred1 = clf.predict(X_test1)

print("Integer Encoding + Logit:")
print(classification_report(y_test1, y_pred1, zero_division=0))
```

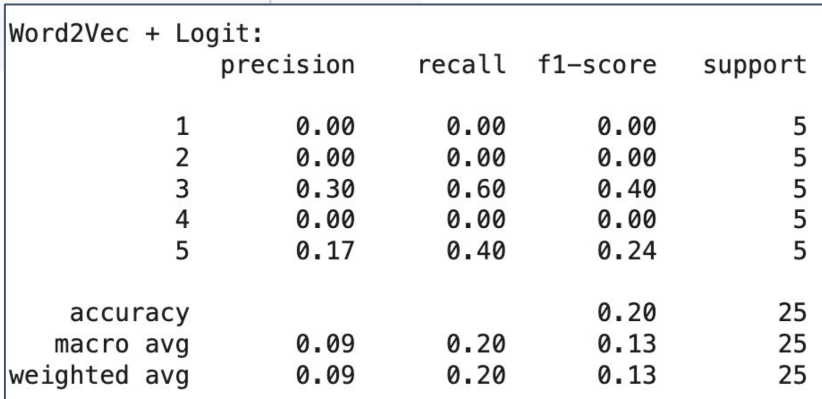
```
Integer Encoding + Logit:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         5
           2       0.00      0.00      0.00         5
           3       0.25      0.80      0.38         5
           4       0.67      0.40      0.50         5
           5       1.00      0.80      0.89         5

    accuracy                           0.40        25
   macro avg       0.38      0.40      0.35        25
weighted avg       0.38      0.40      0.35        25
```

Logistic Regression

- Use the average of Word2Vec embeddings to represent reviews, train a logistic regression model.



```
def average_embedding(review, model):
    vectors = [model.wv[word] for word in review if word in model.wv]
    return np.mean(vectors, axis=0) if vectors else np.zeros(model.vector_size)

# Build feature matrix
X_train2 = np.array([average_embedding(r, w2v_model) for r in train_df_lemma['review']])
y_train2 = train_df_lemma['rating'].astype(int).values

X_test2 = np.array([average_embedding(r, w2v_model) for r in test_df_lemma['review']])
y_test2 = test_df_lemma['rating'].astype(int).values

# Train and evaluate
clf = LogisticRegression(max_iter=1000)
clf.fit(X_train2, y_train2)
y_pred2 = clf.predict(X_test2)

print("Word2Vec + Logit:")
print(classification_report(y_test2, y_pred2, zero_division=0))
```

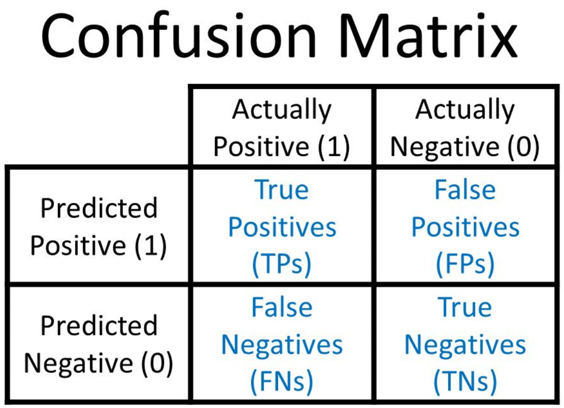


```
Word2Vec + Logit:
             precision    recall  f1-score   support

           1       0.00      0.00      0.00         5
           2       0.00      0.00      0.00         5
           3       0.30      0.60      0.40         5
           4       0.00      0.00      0.00         5
           5       0.17      0.40      0.24         5

    accuracy                           0.20        25
   macro avg       0.09      0.20      0.13        25
weighted avg       0.09      0.20      0.13        25
```

Confusion Matrix

- A **confusion matrix** is a tool used to evaluate the performance of a classification model.

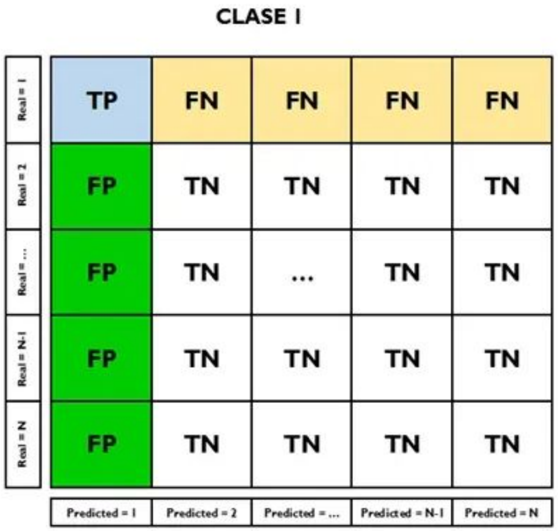
- It shows how well the model's predictions match the actual labels by displaying the counts of true positives, true negatives, false positives, and false negatives.



Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

https://glassboxmedicine.com/2019/02/17/measuring-performance-the-confusion-matrix/

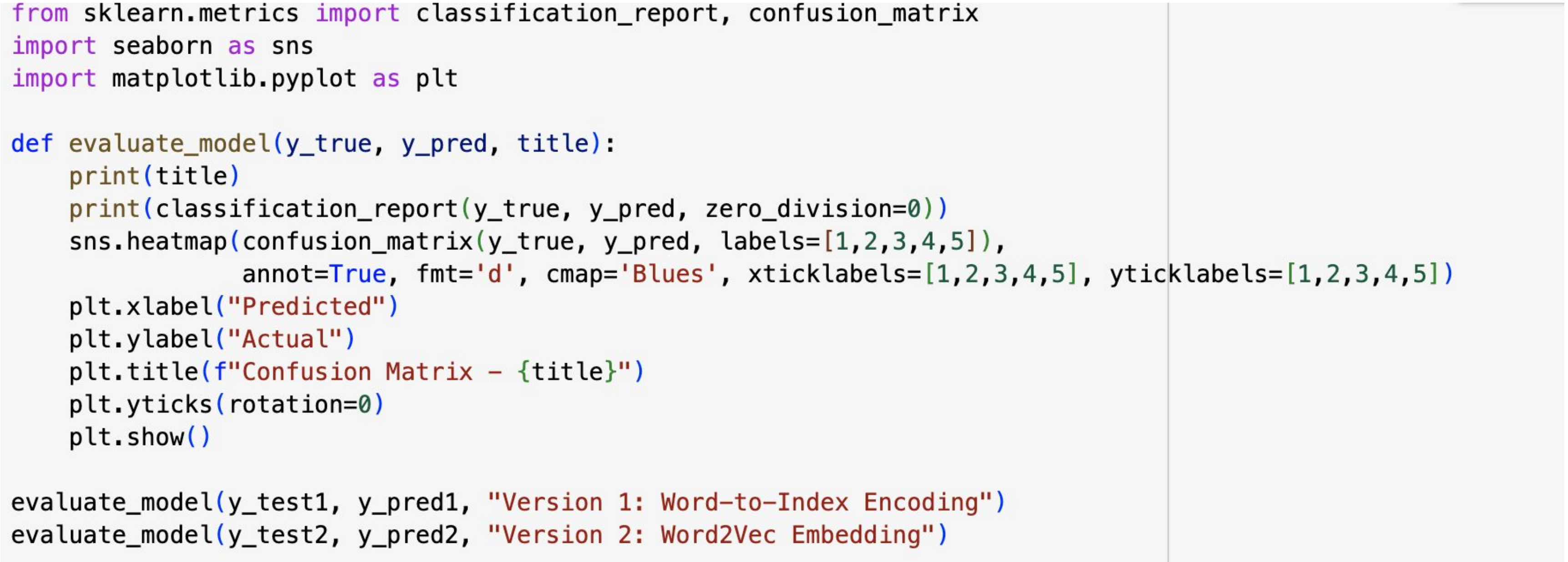


CLASE I

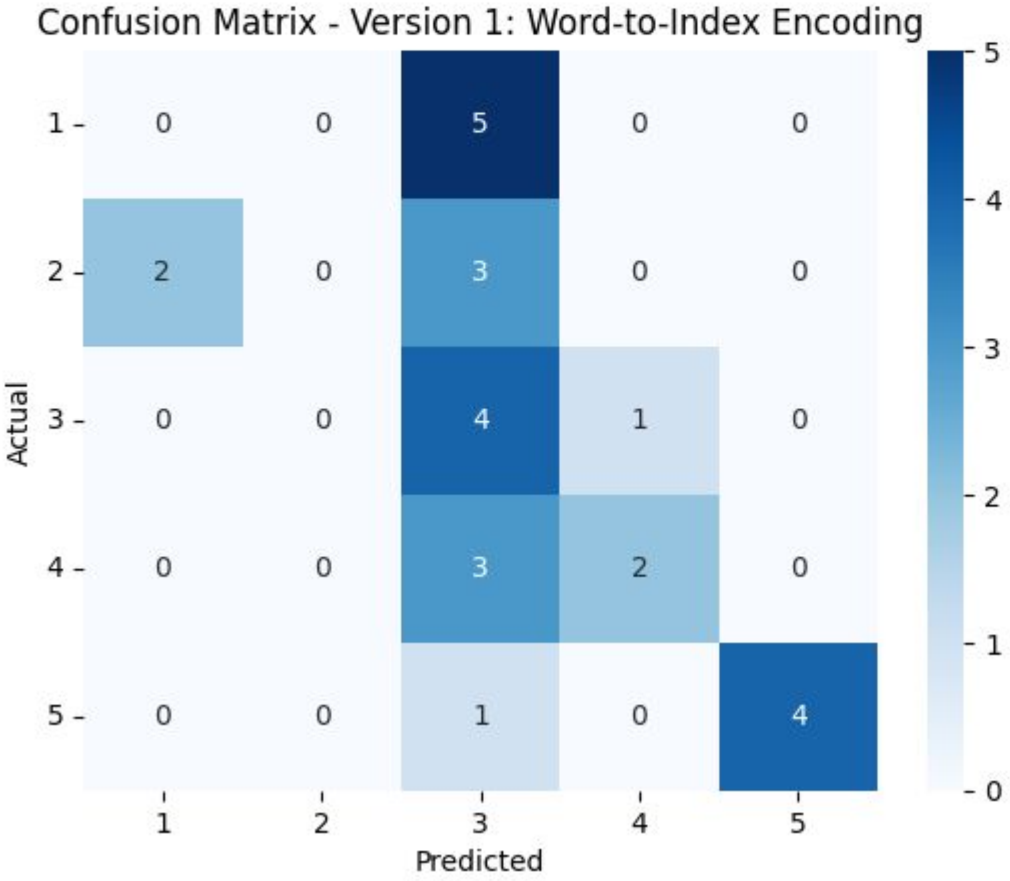
	Predicted = I	Predicted = 2	Predicted = ...	Predicted = N-I	Predicted = N
Real = I	TP	FN	FN	FN	FN
Real = 2	FP	TN	TN	TN	TN
Real = ...	FP	TN	...	TN	TN
Real = N-I	FP	TN	TN	TN	TN
Real = N	FP	TN	TN	TN	TN

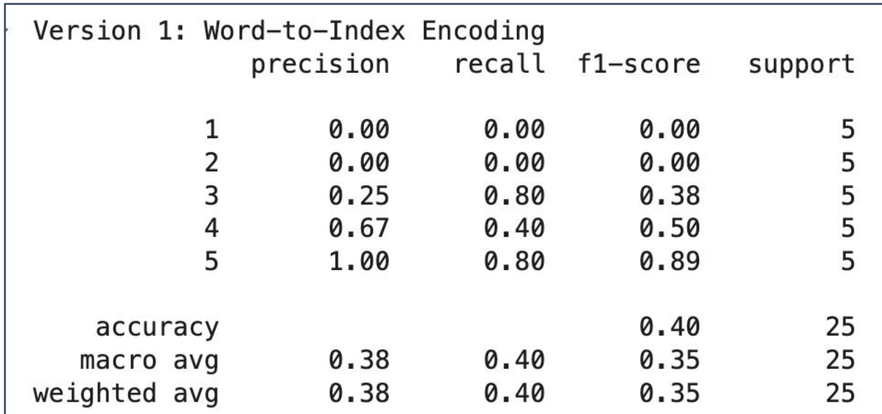
https://glassboxmedicine.com/2019/02/17/measuring-performance-the-confusion-matrix/

- Evaluate the model using classification reports and visualize the results with confusion matrices for different input representations.

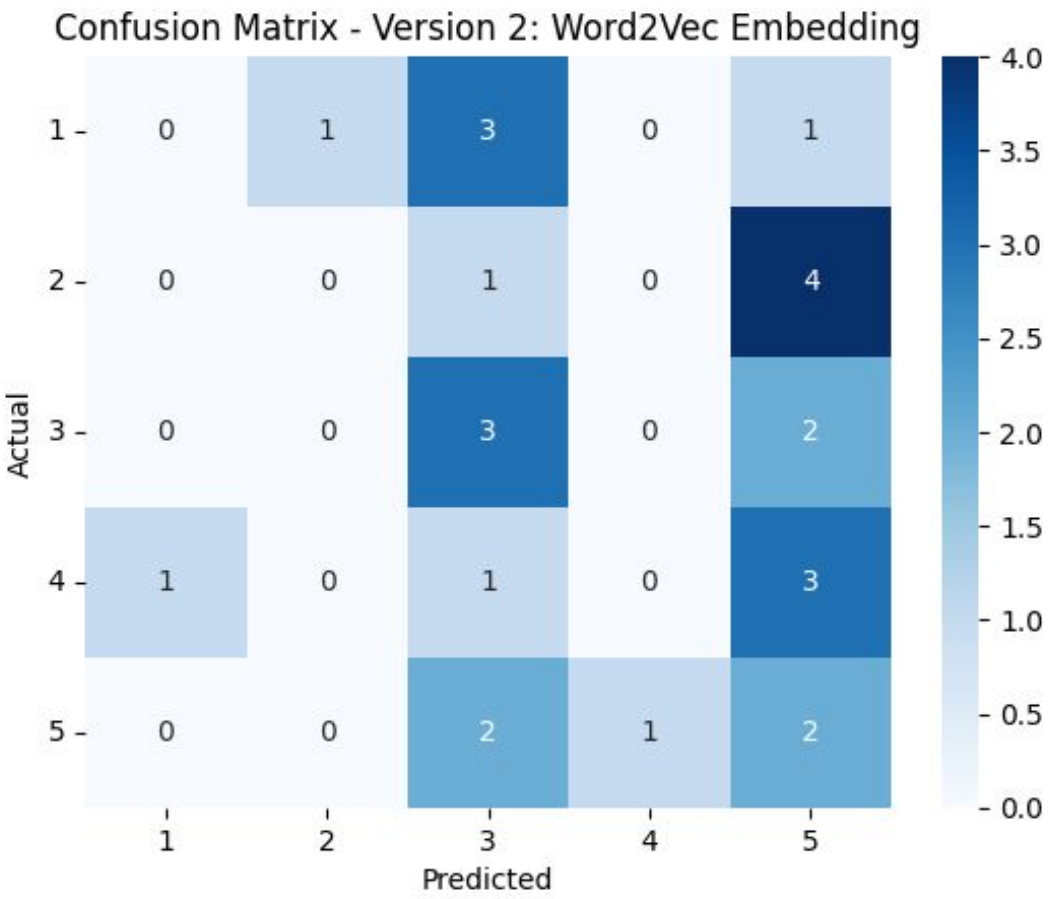


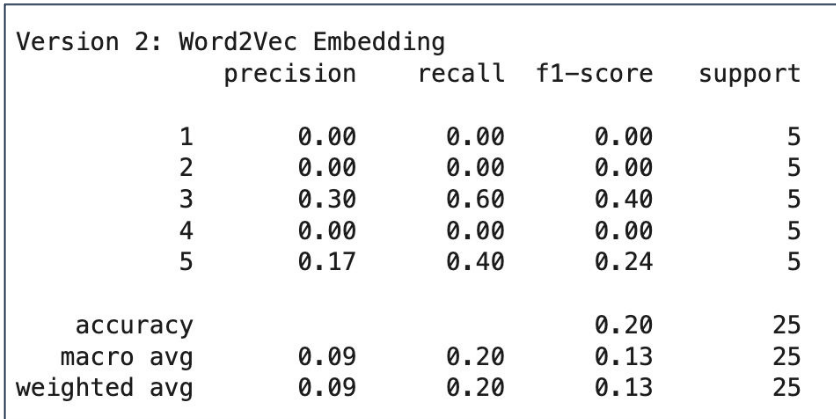
```python
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

def evaluate_model(y_true, y_pred, title):
    print(title)
    print(classification_report(y_true, y_pred, zero_division=0))
    sns.heatmap(confusion_matrix(y_true, y_pred, labels=[1,2,3,4,5]),
                annot=True, fmt='d', cmap='Blues', xticklabels=[1,2,3,4,5], yticklabels=[1,2,3,4,5])
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title(f"Confusion Matrix - {title}")
    plt.yticks(rotation=0)
    plt.show()

evaluate_model(y_test1, y_pred1, "Version 1: Word-to-Index Encoding")
evaluate_model(y_test2, y_pred2, "Version 2: Word2Vec Embedding")
```





```
Version 1: Word-to-Index Encoding
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         5
           2       0.00      0.00      0.00         5
           3       0.25      0.80      0.38         5
           4       0.67      0.40      0.50         5
           5       1.00      0.80      0.89         5

    accuracy                           0.40        25
   macro avg       0.38      0.40      0.35        25
weighted avg       0.38      0.40      0.35        25
```





```
Version 2: Word2Vec Embedding
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         5
           2       0.00      0.00      0.00         5
           3       0.30      0.60      0.40         5
           4       0.00      0.00      0.00         5
           5       0.17      0.40      0.24         5

    accuracy                           0.20        25
   macro avg       0.09      0.20      0.13        25
weighted avg       0.09      0.20      0.13        25
```

Announcement

- Make sure to submit today's assignment by 11:59 PM on April 5th.
- If you get stuck while working on the assignment, try to solve it on your own first. If you're still unsure, feel free to email the TA for help.
- For Absence for unavoidable Reasons(공인결석), please use the Google Form available on the LMS.