Assignment 3

You are a manager of Xstation company, which is planning to launch a video game console soon. You are planning a marketing campaign, and you want to learn more about the strengths and weaknesses of the products from your main rivals – PlayStation 4 (asin "B00BGA9WK2") and Xbox One (asin "B00KAI3KW2"). The first market to launch your product is the USA, so you will use the customer data from this market.

In order to learn more about the competitive product, you will have to:

- **a)** Based on the ratings of Xbox One and PlayStation 4 products report the sentiment (number of positive and negative reviews) and show it using the bar chart. **(7 points)**
- **b)** Develop and update (if necessary) word clouds to examine positive and negative keywords for both PlayStation 4 and Xbox One. Report where you should put the focus to put your product in a winning position from positive and negative keywords perspective **(8 points)**
- **c)** Create a Naive Bayes Classifier for each product in order to prepare for the work with your own reviews in the future. Calculate, report and explain following scores: Accuracy, Precision, Recall and F1. Explain the difference in scores between products. Report the confusion matrix as well. **(10 points)** *NB! Use random_state = 42 when making a train-split. Points will be reduced for a different random_state.*

Make report and save it as a pdf file, attach your .ipynb code as pdf as well. You will need to submit one pdf file overall. You can write your report within ipynb file as well.

Dataset description is available here: https://jmcauley.ucsd.edu/data/amazon/

```
# Load necessary datasets
# DO NOT MODIFY THIS CELL

# Load the dataset with reviews from the server
!wget http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Video_Games.

# Load the dataset with metadata from the server
!wget http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/meta_Video_Games.jso
```

```
--2022-11-16 20:33:18--  http://snap.stanford.edu/data/amazon/productGraph/categoryFi
Resolving snap.stanford.edu (snap.stanford.edu)... 171.64.75.80
Connecting to snap.stanford.edu (snap.stanford.edu)|171.64.75.80|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 386419180 (369M) [application/x-gzip]
Saving to: ‘reviews_Video_Games.json.gz’

reviews_Video_Games 100%[===================>] 368.52M  38.4MB/s    in 25s

2022-11-16 20:33:44 (14.6 MB/s) - ‘reviews_Video_Games.json.gz’ saved [386419180/3864
```

```
--2022-11-16 20:33:44--  http://snap.stanford.edu/data/amazon/productGraph/categoryF
Resolving snap.stanford.edu (snap.stanford.edu)... 171.64.75.80
Connecting to snap.stanford.edu (snap.stanford.edu)|171.64.75.80|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23557720 (22M) [application/x-gzip]
Saving to: ‘meta_Video_Games.json.gz’

meta_Video_Games.js 100%[===================>]  22.47M  6.22MB/s    in 4.2s

2022-11-16 20:33:48 (5.31 MB/s) - ‘meta_Video_Games.json.gz’ saved [23557720/2355772
```

```
# Import dependencies
# DO NOT MODIFY THIS CELL
import pandas as pd
import gzip
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

import nltk
from nltk import word_tokenize
from nltk.corpus import stopwords
nltk.download('punkt')
nltk.download('stopwords')
import re

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransfo
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score, acc
from collections import Counter

from wordcloud import WordCloud
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
# Functions to process file with reviews
# DO NOT MODIFY THIS CELL

# Parse all the parts of the review file
def parse(path):
  g = gzip.open(path, 'rb')
  for l in g:
    yield eval(l)

# Read json to dictionary and turn it into the Data Frame
def getDF(path):
```

```
  i = 0
  df = {}
  for d in parse(path):
    df[i] = d
    i += 1
  return pd.DataFrame.from_dict(df, orient='index')


# Fix the given products asin (Amazon Standart Identification Number)
# DO NOT MODIFY THIS CELL
asin_playstation = "B00BGA9WK2"
asin_xbox = "B00KAI3KW2"


# Create DataFrame from JSON-file with reviews
# df = ...

# YOUR CODE HERE
df = getDF("reviews_Video_Games.json.gz")


# Create DataFrame from JSON-file with meta data
# df_meta = ...

# YOUR CODE HERE
df_meta = getDF('meta_Video_Games.json.gz')


# See the meta description of the products
# df_meta.loc[...]

# YOUR CODE HERE
df_meta.loc[df_meta['asin'].isin([asin_playstation,asin_xbox])]
```

	asin	description	price	imUrl	relat
45933	B00BGA9WK2		28.12	http://ecx.images-amazon.com/images/I/41omR-LT...	{'also_boug ['B00BGA9X9 'B00CXCCI8
50837	B00KAI3KW2		399.00	http://ecx.images-amazon.com/images/I/31BUxym4	{'also_boug ['B00CMQTUS 'B00EM5IY4

```
# Create product-related data frames
# df_playstation = ...
# df_xbox = ...

# YOUR CODE HERE
df_playstation = df[df.asin.isin([asin_playstation])]
df_xbox = df[df.asin.isin([asin_xbox])]


# Check the distribution of ratings for PlayStation 4
```

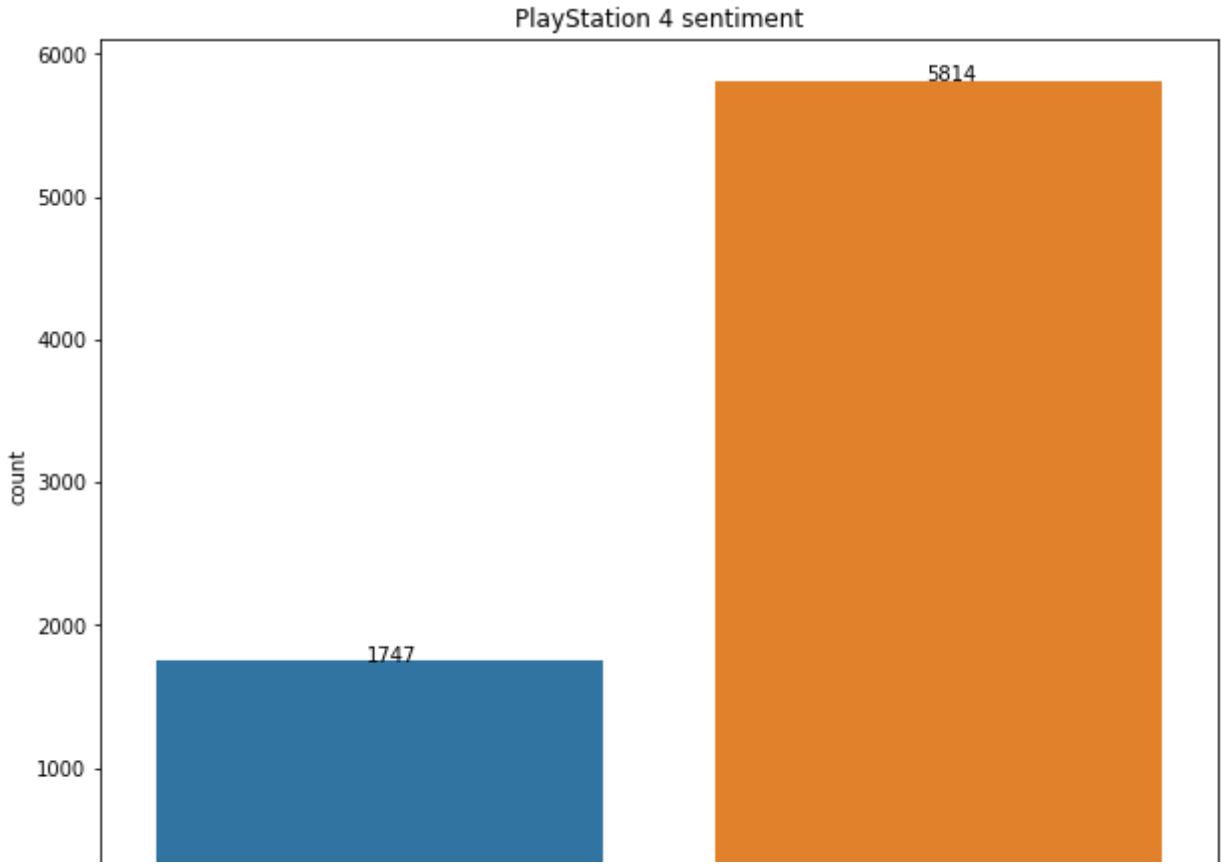
```
# YOUR CODE HERE
df_playstation['overall'].value_counts()
```

```
     5.0    5123
     1.0    1327
     4.0     691
     3.0     269
     2.0     151
     Name: overall, dtype: int64
```

```
# Check the distribution of ratings for Xbox One

# YOUR CODE HERE
df_xbox['overall'].value_counts()
```

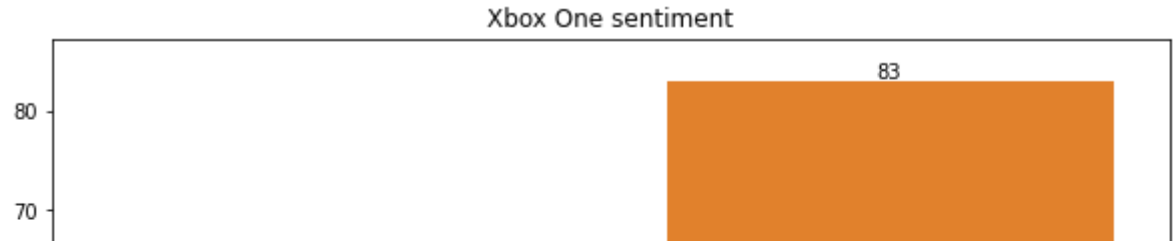
```
     5.0    71
     1.0    23
     4.0    12
     2.0     3
     3.0     3
     Name: overall, dtype: int64
```

```
# For each product create new Data Frame and leave there only review text and rating
# df_playstation_text = ...
# df_xbox_text = ...

# YOUR CODE HERE
df_playstation_text = df_playstation[['reviewText','overall']].reset_index(drop=True)
df_xbox_text = df_xbox[['reviewText','overall']].reset_index(drop=True)


# Create column "Sentiment" for sentiment values (0 - negative, 1 - positive). Sentiment i
# df_playstation_text["Sentiment"] =
# df_xbox_text["Sentiment"] =

# YOUR CODE HERE
df_playstation_text["Sentiment"] = df_playstation_text['overall'].apply(lambda x: 1 if (x
df_xbox_text["Sentiment"] = df_xbox_text['overall'].apply(lambda x: 1 if (x >=4.0) else 0)


# Plot sentiment for PlayStation 4
# DO NOT MODIFY THIS CELL

ax_1 = plt.figure(figsize = (10, 8))
ax_1 = sns.countplot(data = df_playstation_text, x = 'Sentiment')
for p, label in zip(ax_1.patches, df_playstation_text['Sentiment'].value_counts()[::-1]):
  ax_1.annotate(label, (p.get_x()+0.375, p.get_height()+0.15))
plt.title("PlayStation 4 sentiment")
plt.show()
```




```
# Plot sentiment for Xbox One
# DO NOT MODIFY THIS CELL

plt.figure(figsize = (10, 8))
ax_2 = plt.figure(figsize = (10, 8))
ax_2 = sns.countplot(data = df_xbox_text, x = 'Sentiment')
for p, label in zip(ax_2.patches, df_xbox_text['Sentiment'].value_counts()[::-1]):
    ax_2.annotate(label, (p.get_x()+0.375, p.get_height()+0.15))
plt.title("Xbox One sentiment")
plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```



Report - Part A

According to the bar chart for PlayStation 4, about 23% (1747 reviews) of the total reviews is negative for the PlayStation 4 and the figure is 77% for the positive reviews. The second bar chart shows that about 26% of the total reviews for Xbox (29 review) is negative.

```
# Create functions to clean the text
# DO NOT MODIFY THIS CELL

# Function to clean text
def clean_text(Review):
  Review = str(Review).lower() # convert to lowercase
  Review = re.sub('\[.*?\]', '', Review)  # Remove special symbols
  Review = re.sub('https?://\S+|www\.\S+', '', Review) # Remove URls
  Review = re.sub('<.*?>+', '', Review) # Remove special symbols
  Review = re.sub(r'[^a-z0-9\s]', '', Review) # Remove punctuation
  Review = re.sub('\n', '', Review) # Remove forced enter
  Review = re.sub('\w*\d\w*', '', Review) # Remove metacharachters
  return Review

# Function to convert list to string
def listToString(s):

  # initialize an empty string
  str1 = " "

  # return string
  return (str1.join(s))

# Function to remove stopwords
def remove_stopword(stop_words, sentence):
    return [word for word in nltk.word_tokenize(sentence) if word not in stop_words]


# Load standard list of stopwords
# stop_words = ...

# YOUR CODE HERE
# Load standard list of stopwords
stop_words = stopwords.words('english')

# Add own stop words and append to existing ones
stopwords = []
if stopwords is not []:
  stop_words.extend(stopwords)
```

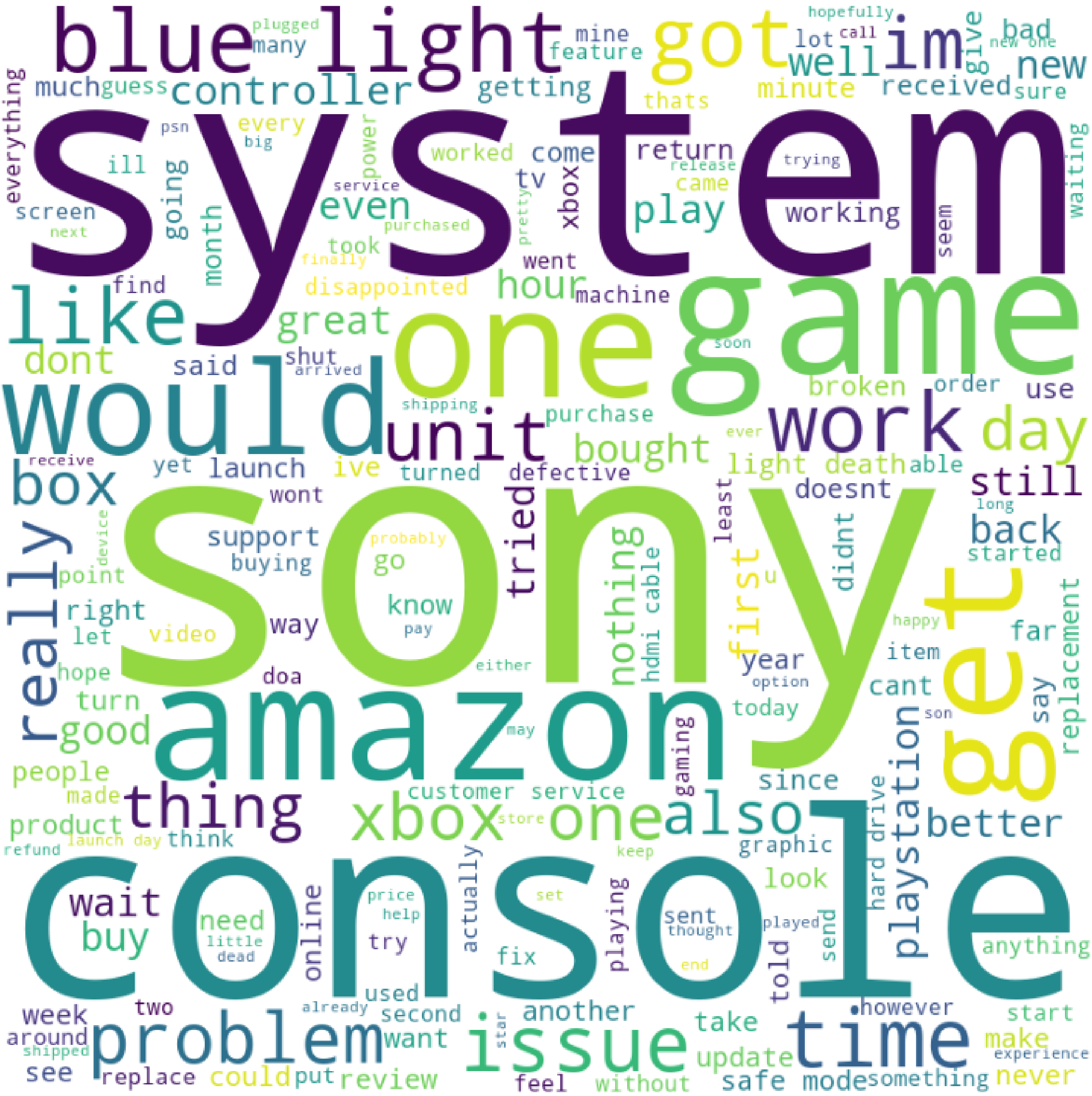
```
# Clean review texts for PlayStation 4
# df_playstation_text['review'] = df_playstation_text['reviewText'].apply(...)
# df_playstation_text['review'] = ...

# YOUR CODE HERE
df_playstation_text['review'] = df_playstation_text['reviewText'].apply(clean_text)
df_playstation_text['review'] = df_playstation_text['review'].apply(lambda row: remove_sto


# Clean review for Xbox One
# df_xbox_text['review'] = df_xbox_text['reviewText'].apply(...)
# df_xbox_text['review'] = ...

# YOUR CODE HERE
df_xbox_text['review'] = df_xbox_text['reviewText'].apply(clean_text)
df_xbox_text['review'] = df_xbox_text['review'].apply(lambda row: remove_stopword(stop_wor


# Check how clean reviews look for PlayStation 4 or Xbox

# YOUR CODE HERE
df_playstation_text.head(10)
```

	reviewText	overall	Sentiment	review
0	I'm so upset at how Sony is ripping people off...	1.0	0	im upset sony ripping people game system son r...
1	I gave it 4 out of 5 stars because of lack of ...	4.0	1	gave stars lack games otherwise would gotten c...
2	I have only Nintendo products in the past. Eve...	5.0	1	nintendo products past everything read playsta...
3	I love this PS4. Even though the game selectio...	5.0	1	love even though game selection still big qual...
4	I placed my order around June and received my ...	5.0	1	placed order around june received new day rele...
5	PlayStation 4 is kind of hard to review right ...	3.0	0	playstation kind hard review right missing man...
6	Simply amazing. A huge step up from	5.0	1	simply amazing huge step huge

```
# Create a Data Frame with negative reviews for PlayStation 4
# df_playstation_neg = ...

# YOUR CODE HERE
df_playstation_neg = df_playstation_text[df_playstation_text.Sentiment == 0][["review"]].r

# Create a Data Frame with positive reviews for PlayStation 4
# df_playstation_pos = ...

# YOUR CODE HERE
df_playstation_pos = df_playstation_text[df_playstation_text.Sentiment == 1][["review"]].r
```

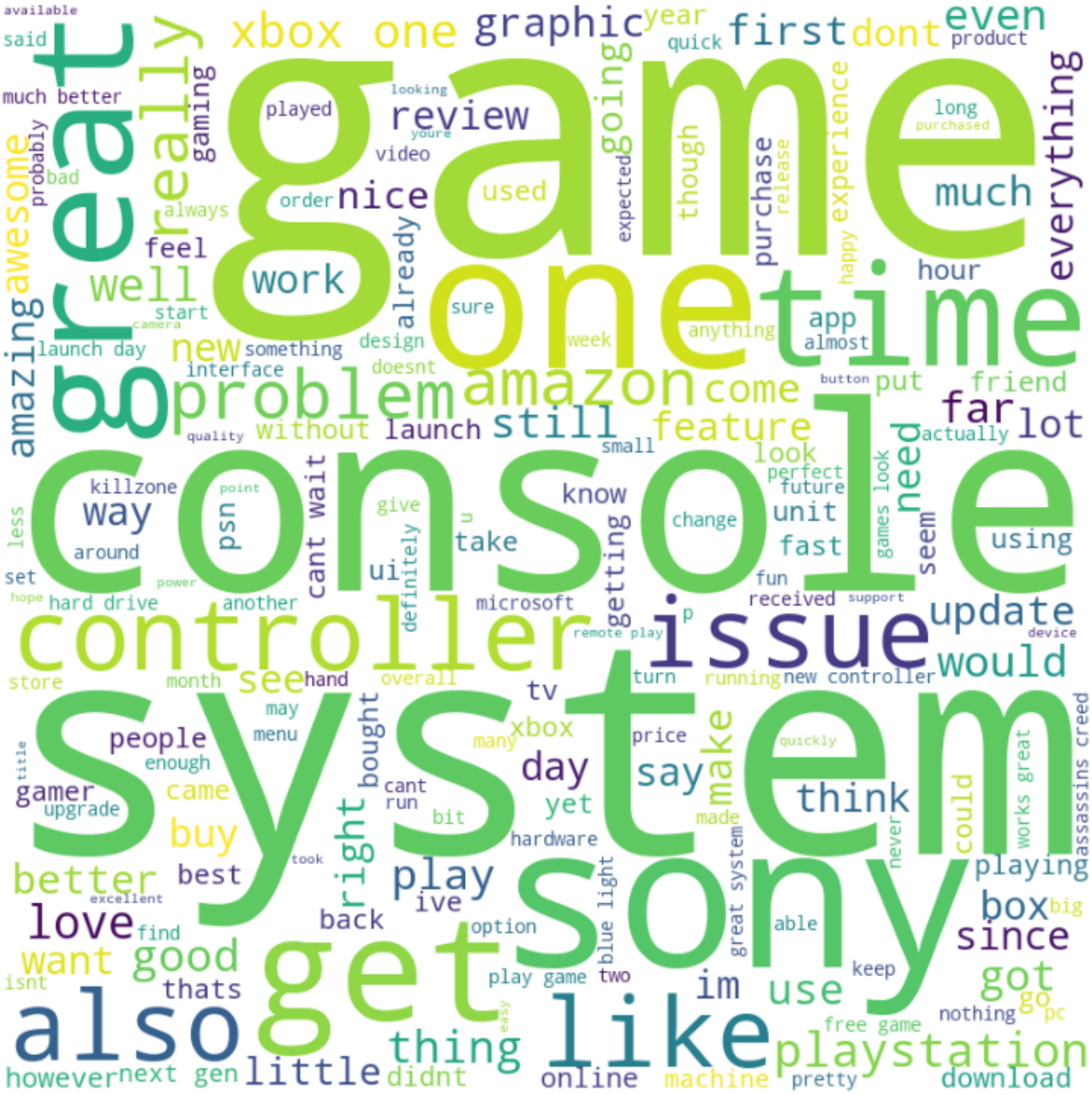
```
# Create a Data Frame with negative reviews for Xbox One
# df_xbox_neg = ...

# YOUR CODE HERE
df_xbox_neg = df_xbox_text[df_xbox_text.Sentiment == 0][["review"]].reset_index()
# Create a Data Frame with positive reviews for Xbox One
# df_xbox_pos = ...
df_xbox_pos = df_xbox_text[df_xbox_text.Sentiment == 1][["review"]].reset_index()
# YOUR CODE HERE


# Create function, that creates string for a word cloud
# DO NOT MODIFY THIS CELL
def create_string(col):
  string = ''
  for val in col:
    list = nltk.word_tokenize(val)
    for m in list:
      string += " " + m
  return string


# Create string for a word cloud for negative reviews for PlayStation 4
# string_playstation_neg = create_string()
string_playstation_neg = create_string(df_playstation_neg["review"])
# YOUR CODE HERE

# Create string for a word cloud for positive reviews for PlayStation 4
# string_playstation_pos = create_string()
string_playstation_pos = create_string(df_playstation_pos["review"])
# YOUR CODE HERE

# Create string for a word cloud for negative reviews for Xbox One
# string_xbox_neg = create_string()
string_xbox_neg = create_string(df_xbox_neg["review"])
# YOUR CODE HERE

# Create string for a word cloud for positive reviews for Xbox One
# string_xbox_pos = create_string()
string_xbox_pos = create_string(df_xbox_pos["review"])
# YOUR CODE HERE


# Create wordcloud for negative reviews for PlayStation 4
# wordcloud_playstation_neg = WordCloud(...).generate(...)

# YOUR CODE HERE
wordcloud_playstation_neg = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stop_words,
                min_font_size = 10).generate(string_playstation_neg)

# Plot the wordcloud
# DO NOT MODIFY THE CODE
plt.figure(figsize = (10, 10), facecolor = 'white', edgecolor='blue')
```

```
plt.figure(figsize = (10, 10), facecolor = 'white', edgecolor='blue')
plt.imshow(wordcloud_playstation_neg)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```




```
# Create wordcloud for positive reviews for PlayStation 4
# wordcloud_playstation_pos = WordCloud(...).generate(...)

# YOUR CODE HERE
wordcloud_playstation_pos = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stop_words,
                min_font_size = 10).generate(string_playstation_pos)
```

```
# Plot the wordcloud
# DO NOT MODIFY THE CODE
plt.figure(figsize = (10, 10), facecolor = 'white', edgecolor='blue')
plt.imshow(wordcloud_playstation_pos)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```




```
# Create wordcloud for negative reviews for Xbox One
# wordcloud_xbox_neg = WordCloud(...).generate(...)
```

```
# YOUR CODE HERE
wordcloud_xbox_neg = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stop_words,
                min_font_size = 10).generate(string_xbox_neg)

# Plot the wordcloud
# DO NOT MODIFY THE CODE
plt.figure(figsize = (10, 10), facecolor = 'white', edgecolor='blue')
plt.imshow(wordcloud_xbox_neg)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

```
# Create wordcloud for positive reviews for Xbox One
# wordcloud_xbox_pos = WordCloud(...).generate(...)

# YOUR CODE HERE
wordcloud_xbox_pos = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stop_words,
                min_font_size = 10).generate(string_xbox_pos)

# Plot the wordcloud
# DO NOT MODIFY THE CODE
plt.figure(figsize = (10, 10), facecolor = 'white', edgecolor='blue')
plt.imshow(wordcloud_xbox_pos)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```


```
# Update stop words if necessary and build new word clouds
stopwords_updated = ['game', 'games', 'console', 'system']
df_playstation_pos['review'] = df_playstation_pos['review'].apply(lambda row: remove_stopw
df_playstation_neg['review'] = df_playstation_neg['review'].apply(lambda row: remove_stopw
df_xbox_pos['review'] = df_xbox_pos['review'].apply(lambda row: remove_stopword(stopwords_
df_xbox_neg['review'] = df_xbox_neg['review'].apply(lambda row: remove_stopword(stopwords_
# Create an updated word cloud for positive
string_playstation_pos_2 = create_string(df_playstation_pos["review"])

wordcloud_pos_2 = WordCloud(width = 600, height = 600,
                background_color ='white',
                stopwords = stop_words,
                min_font_size = 10).generate(string_playstation_pos_2)

plt.figure(figsize = (10, 10), facecolor = 'white', edgecolor='blue')
plt.imshow(wordcloud_pos_2)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

Report - part B

From the keywords perspective, we could see clearly in the wordcloud, how the reviewer use to describe the product negatively and positively.

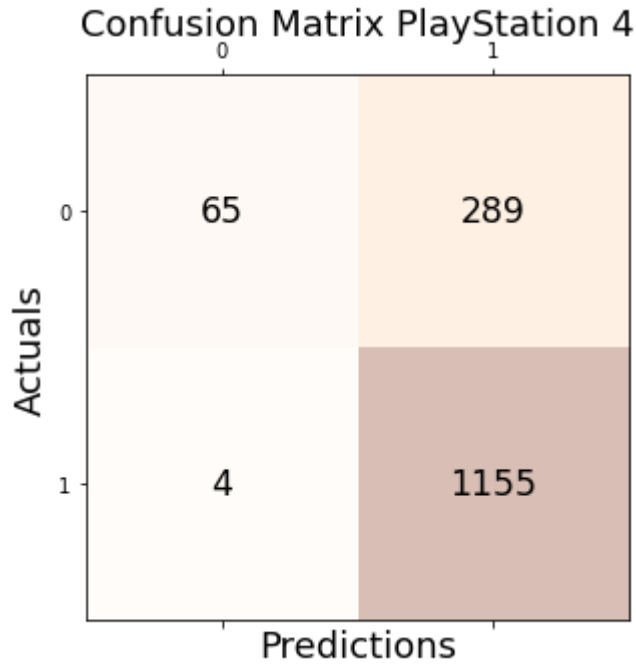
- For example, In the wordCloud for positive reviews of playstation 4, reviews frequently use word 'great','amazing', 'interface', 'awesome'.
- In contrast, in the negative reviews, the reviewer mention words 'problem', 'broken', 'defective', 'replace, 'hdmi cable'. Based on those words which are frequently metioned in the negative reviews, the company should put focus on, for example, what the problem users faces, whhich part is mostly defective or what part the users expect to be replaced. Some nouns like 'hdmi cable', 'hour' should also paid attention while looking at review.
- In the negative reviews for Xbox, some words that worth analyzing are: 'take hours', 'big issue', 'price', 'without', or 'snap'.
- In the positive reviews for Xbox, highlighted words are, 'kinect', 'good voice', 'feature', power' that worth digging more detail. For examplem which feature the user value of the product. If some features are already good, would the users compare to that of its competitor.


```
# Create vectors for features and labels for PlayStation 4
# X_playstation = ...
# y_playstation = ...

# YOUR CODE HERE
X_playstation = df_playstation_text['review']
y_playstation = df_playstation_text['Sentiment']


# Create vectors for features and labels for Xbox One
# X_xbox = ...
# y_xbox = ...

# YOUR CODE HERE
X_xbox = df_xbox_text['review']
y_xbox = df_xbox_text['Sentiment']


# Make a train-test split for PlayStation 4
# WHEN YOU MAKE A SPLIT, PLEASE, KEEP random_state = 42 - there will be points reduction i
# X_playstation_train, X_playstation_test, y_playstation_train, y_playstation_test =
```

```
# YOUR CODE HERE
# Make a split for test and train
X_playstation_train, X_playstation_test, y_playstation_train, y_playstation_test = train_t



# Make a train-test split for Xbox One
# WHEN YOU MAKE A SPLIT, PLEASE, KEEP random_state = 42 - there will be points reduction i
# X_xbox_train, X_xbox_test, y_xbox_train, y_xbox_test = ...

# YOUR CODE HERE
X_xbox_train, X_xbox_test, y_xbox_train, y_xbox_test = train_test_split(X_xbox, y_xbox, ra


# Create a pipeline to run a Naive Bayes Classifier for PlayStation 4
# clf_playstation = Pipeline (...)

# YOUR CODE HERE
clf_playstation = Pipeline([
    ('vect', CountVectorizer(stop_words= "english")),
    ('tfidf', TfidfTransformer()),
    ('classifier', MultinomialNB())])
# Fit train data to the model for PlayStation 4
# clf_playstation.

# YOUR CODE HERE
clf_playstation.fit(X_playstation_train, y_playstation_train)
# Predict results on test data for PlayStation 4
# y_playstation_pred =

# YOUR CODE HERE
y_playstation_pred = clf_playstation.predict(X_playstation_test)
# Print model metrics for PlayStation 4
# print('Test accuracy: %.3f' % ...)
# print('Precision: %.3f' % ...)
# print('Recall: %.3f' % ...)
# print('F1 Score: %.3f' % ...)

# YOUR CODE HERE
print('Test accuracy: %.3f' % accuracy_score(y_playstation_test, y_playstation_pred))
print('Precision: %.3f' % precision_score(y_playstation_test, y_playstation_pred))
print('Recall: %.3f' % recall_score(y_playstation_test, y_playstation_pred))
print('F1 Score: %.3f' % f1_score(y_playstation_test, y_playstation_pred))
# Create a confusion matrix for PlayStation 4 and print it
# conf_matrix_playstation = confusion_matrix(...)

# YOUR CODE HERE
conf_matrix_playstation = confusion_matrix(y_true=y_playstation_test, y_pred=y_playstation
# DO NOT MODIFY THE CODE FOR PRINTING
fig, ax = plt.subplots(figsize=(5, 5))
ax.matshow(conf_matrix_playstation, cmap=plt.cm.Oranges, alpha=0.3)
for i in range(conf_matrix_playstation.shape[0]):
    for j in range(conf_matrix_playstation.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix_playstation[i, j], va='center', ha='center', size='
plt.xlabel('Predictions', fontsize=18)
```

```
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix PlayStation 4', fontsize=18)
plt.show()
```

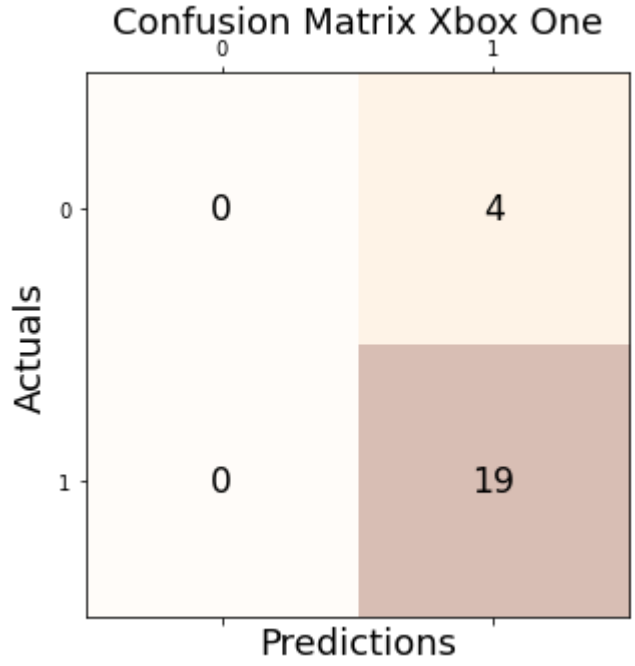
```
Test accuracy: 0.806
Precision: 0.800
Recall: 0.997
F1 Score: 0.887
```




```
# Create a pipeline to run a Naive Bayes Classifier for Xbox One
# clf_xbox = Pipeline (...)
clf_xbox = Pipeline([
    ('vect', CountVectorizer(stop_words= "english")),
    ('tfidf', TfidfTransformer()),
    ('classifier', MultinomialNB())])

# Fit train data to the model for Xbox One
# clf_xbox. ...

# YOUR CODE HERE
clf_xbox.fit(X_xbox_train, y_xbox_train)
# Predict results on test data for Xbox One
# y_xbox_pred =

# YOUR CODE HERE
y_xbox_pred = clf_xbox.predict(X_xbox_test)
# Print model metrics for Xbox One
# print('Test accuracy: %.3f' % ...)
# print('Precision: %.3f' % ...)
# print('Recall: %.3f' % ...)
# print('F1 Score: %.3f' % ...)

# YOUR CODE HERE
print('Test accuracy: %.3f' % accuracy_score(y_xbox_test, y_xbox_pred))
print('Precision: %.3f' % precision_score(y_xbox_test, y_xbox_pred))
print('Recall: %.3f' % recall_score(y_xbox_test, y_xbox_pred))
print('F1 Score: %.3f' % f1_score(y_xbox_test, y_xbox_pred))
```

```
# Create a confusion matrix for Xbox One and print it
# conf_matrix_xbox = confusion_matrix(...)

# YOUR CODE HERE
conf_matrix_xbox = confusion_matrix(y_true=y_xbox_test, y_pred=y_xbox_pred)
# DO NOT MODIFY THE CODE FOR PRINTING
fig, ax = plt.subplots(figsize=(5, 5))
ax.matshow(conf_matrix_xbox, cmap=plt.cm.Oranges, alpha=0.3)
for i in range(conf_matrix_xbox.shape[0]):
    for j in range(conf_matrix_xbox.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix_xbox[i, j], va='center', ha='center', size='xx-larg
plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix Xbox One', fontsize=18)
plt.show()
```

```
Test accuracy: 0.826
Precision: 0.826
Recall: 1.000
F1 Score: 0.905
```



Report part C

- Confusion matrix visualize the metrics that help evaluate the performance of an argorithm, in this case Naive Bayes Classifier model. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa, inclusing: True Positive, True Negative, False Positive, False Negative.
- Test Accuracy: The rate indicates the proportion of correct classifications. For example, the accuracy score of Playstation review is 80.6%, meaning that abour 81% od reviews are classified in correct class. The accuracy for Xbox is 82.6%.
- Precision or positive predictive value = TP/(TP+FP). The ratio of correct positive predictions to the total predicted positives. For example, in the Xbox, the precision is 0.826,

meaning 83% of the samples that are predicted positives is correctly classified positive. The ratio for playstaion 4 is 0.8

- Recall or true positive rate is the ratio of correct positive predictions to the total positives. The recall of playstation is 0.997 means that 99.7% of the actual positive examples is correctly classified. The rate for Xbox is 1.
- F_score is provides a way to combine both precision and recall into a single measure that captures both properties. It is formulated as (2 * precision * recall)/(precision+recall). The f-score measure can also be used to see if the data is balanced or not since to be able to have a good f-score, we need both good score of precision and recall. The f-score for playstation 4 is 88.7, while the Xbox is higher with the rate of 0.905.

However, if we look a bit more detail on the second model for Xbox, we can see that the model predict that all actual negatives are predicted incorrectly to false positive. It logically make sence since the true negative rate is 0 (TN/(TN + FP)). This mean that the model does not work well, when it comes to detecting the negative class.

In conclusion, confusion matrix and also the four metrics are very good measurement to evaluate the performance of the model but it does not mean they tell the whole story.

Colab paid products - Cancel contracts here

✓ 0s completed at 10:58 PM