In [1]:
```
import nltk

nltk.download('popular', quiet=True)
import demoji
from nltk import word_tokenize
from nltk.corpus import stopwords
import spacy
import string
nltk.download("wordnet", quiet=True)
nltk.download("stopwords", quiet=True)
from nltk.stem.wordnet import WordNetLemmatizer
from nltk import pos_tag
from nltk.corpus import wordnet
from collections import defaultdict
import pandas as pd
nlp = spacy.load("en_core_web_sm")
```

1. Perform necessary data preprocessing, e.g. removing punctuation and stop words, stemming, lemmatizing. You may use the outputs from previous weekly assignments. (10 points)

In [2]:
```
import os
import glob
def collect_data():
    text_file_pattern = "*.txt"  # You can adjust the pattern to match your file extensions
    text_files = glob.glob(os.path.join("../nhs/content", text_file_pattern))
    data = {}
    for file_path in text_files:
        with open(file_path, 'r', encoding='utf-8') as file:
            file_name = os.path.basename(file_path)
            file_content = file.read()
            data[file_name] = file_content
    return data
```

In [3]:
```
corpus = collect_data()
text = ""
for data in corpus:
    text += " " + data
```

In [4]:
```
def remove_punctuation(text):
    # Create a translation table to remove punctuation
    translator = str.maketrans('', '', string.punctuation)
    # Use translate method to remove punctuation
    cleaned_text = text.translate(translator)
    return cleaned_text

def remove_stop_words(text):
    nltk_stopwords = stopwords.words('english')
    spacy_stopwords = nlp.Defaults.stop_words

    stop_words = (*nltk_stopwords, *spacy_stopwords, "NHStxt")

    tokens = word_tokenize(text)
    tokens = [token for token in tokens if token not in stop_words]
    return " ".join(tokens)

def apply_lemmitization(text):
    tag_map = defaultdict(lambda : wordnet.NOUN)
    tag_map['V'] = wordnet.VERB
    tag_map['A'] = wordnet.ADJ
    tag_map['R'] = wordnet.ADJ

    lemmitizer = WordNetLemmatizer()
    lemmitized_result = ""
    tokens = word_tokenize(text)
    for token, tag in pos_tag(tokens):
        lemma = lemmitizer.lemmatize(token, tag_map[tag[0]])
        lemmitized_result = lemmitized_result + " " + lemma
    return lemmitized_result

def remove_emoji_and_smart_quotes(text):
    # replacing emojis with description
    text = demoji.replace_with_desc(text)
    #Removing smart quotes
    return text.replace("“", "\"").replace("”","\"")
```

In [5]:
```
def data_preprocessing(text):
    text = remove_emoji_and_smart_quotes(text)
    text = remove_punctuation(text)
    text = remove_stop_words(text)
    text = apply_lemmitization(text)
```

```
    return text

def apply_data_preprocessing_to_corpus(corpus):
    new_corpus = {}
    for idx, key in enumerate(corpus.keys()):
        new_corpus[key] = data_preprocessing(corpus[key])
        print(f"idx: {idx}")
    return new_corpus
```

In [6]:
```
processed_text = data_preprocessing(text)
with open('week8_1.txt', 'w') as file:
    file.write(f'{processed_text}')
```

2. Propose a binary classification problem from your project data and identify the columns that you will use to solve the problem. You may need to create new columns of data. (20 points)

In [7]:
```
```

In [93]:
```
df = pd.read_csv("../nhs/conditions_departments.csv", header=None)
df.columns = ["index", "condition", "department"]
department_dict = {
    row['condition']: row.drop('condition').to_dict()
    for index, row in df.iterrows()
}
```

In [135]:
```
"""
Get X as text data and Y as 0 if general medicine else 1
"""
X = []
y =[]
for file_name in list(corpus.keys()):
    idx = file_name.strip(" NHS.txt")
    text = data_preprocessing(corpus[file_name])
    X.append(text)
    y.append(0 if department_dict[idx]["department"] == "General Medicine" else 1)
```

In [136]:
```
"""
Ill be using this data for the binary classification
to determine department based on the text
"""
```

Out[136]:
```
'\nIll be using this data for the binary classification\nto determine department based on the text\n'
```

In [137]:
```
pd.DataFrame({"data":X,"general": y})
```

Out[137]:

	data	general
0	Research find possible link certain artificia...	0
1	If youre age 55 74 smoke offer NHS lung healt...	1
2	Carotid endarterectomy surgical procedure rem...	0
3	Middle East respiratory syndrome coronavirus ...	1
4	A kidney infection painful unpleasant illness...	1
...	...	...
973	Osteomyelitis painful bone infection It usual...	0
974	Dots line floater flash light vision common T...	0
975	TaySachs disease rare inherit condition mainl...	0
976	Slapped cheek syndrome call fifth disease com...	0
977	The mitral valve small flap heart stop blood ...	0

978 rows × 2 columns

3. Compute TF-IDF vectors on the text data. (10 points)

In [138]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
max_tokens = 1000

vectorizer = TfidfVectorizer(min_df = .1,
                             tokenizer = nltk.word_tokenize,
                             max_features = max_tokens)
X = vectorizer.fit_transform(X)
```

```
/Users/shireesh/opt/anaconda3/envs/COMP293/lib/python3.10/site-packages/sklearn/feature_extraction/text.py:525: UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
```

In [139]:
```
from sklearn.model_selection import train_test_split

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

4. Solve your binary classification problem with the Naïve Bayes classifier. (30 points)

In [140]:
```
from sklearn.naive_bayes import MultinomialNB
# Initialize the classifier and train it
classifier = MultinomialNB()
classifier.fit(X_train, y_train)
```

Out[140]:
```
MultinomialNB()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [141]:
```
from sklearn.metrics import accuracy_score
pred = classifier.predict(X_test)
accuracy_score(y_test, pred)
```

Out[141]:
```
0.8877551020408163
```

In [142]:
```
# View the results as a confusion matrix
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, pred, normalize=None)
print(conf_matrix)
```

```
[[171   0]
 [ 22   3]]
```

5. Solve your binary classification problem with the SVC classifier. (30 points)

In [144]:
```
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline


svc_tfidf = Pipeline([
    ("linear svc", SVC(kernel="linear"))
])
model = svc_tfidf
model.fit(X_train, y_train)
```

Out[144]:
```
Pipeline(steps=[('linear svc', SVC(kernel='linear'))])
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [147]:
```
svm_pred = model.predict(X_test)
accuracy_result = accuracy_score(y_test, svm_pred)
print("accuracy_result", accuracy_result)
# View the results as a confusion matrix
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test,
                               svm_pred,normalize=None)
print("conf_matrix\n", conf_matrix)
```

```
accuracy_result 0.9030612244897959
conf_matrix
 [[170   1]
 [ 18   7]]
```