Lab3 - Assignment Sentiment

This notebook describes the LAB-2 assignment of the Text Mining course. It is about sentiment analysis.

The aims of the assignment are:

- Learn how to run a rule-based sentiment analysis module (VADER)
- Learn how to run a machine learning sentiment analysis module (Scikit-Learn/ Naive Bayes)
- Learn how to run scikit-learn metrics for the quantitative evaluation
- Learn how to perform and interpret a quantitative evaluation of the outcomes of the tools (in terms of Precision, Recall, and F1)
- Learn how to evaluate the results qualitatively (by examining the data)
- Get insight into differences between the two applied methods
- Get insight into the effects of using linguistic preprocessing
- Be able to describe differences between the two methods in terms of their results
- Get insight into issues when applying these methods across different domains

In this assignment, you are going to create your own gold standard set from 50 tweets. You will the VADER and scikit-learn classifiers to these tweets and evaluate the results by using evaluation metrics and inspecting the data.

We recommend you go through the notebooks in the following order:

- **Read the assignment (see below)**
- **Lab3.2-Sentiment-analysis-with-VADER.ipynb**
- **Lab3.3-Sentiment-analysis.with-scikit-learn.ipynb**
- **Answer the questions of the assignment (see below) using the provided notebooks and submit**

In this assignment you are asked to perform both quantitative evaluations and error analyses:

- a quantitative evaluation concerns the scores (Precision, Recall, and F1) provided by scikit's classification_report. It includes the scores per category, as well as micro and macro averages. Discuss whether the scores are balanced or not between the different categories (positive, negative, neutral) and between precision and recall. Discuss the shortcomings (if any) of the classifier based on these scores
- an error analysis regarding the misclassifications of the classifier. It involves going through the texts and trying to understand what has gone wrong. It servers to get insight in what could be done to improve the performance of the classifier. Do you

observe patterns in misclassifications? Discuss why these errors are made and propose ways to solve them.

Credits

The notebooks in this block have been originally created by Marten Postma and Isa Maks. Adaptations were made by Filip Ilievski.

Part I: VADER assignments

Preparation (nothing to submit):

To be able to answer the VADER questions you need to know how the tool works.

- Read more about the VADER tool in this blog.

- VADER provides 4 scores (positive, negative, neutral, compound). Be sure to understand what they mean and how they are calculated.
- VADER uses rules to handle linguistic phenomena such as negation and intensification. Be sure to understand which rules are used, how they work, and why they are important.
- VADER makes use of a sentiment lexicon. Have a look at the lexicon. Be sure to understand which information can be found there (lemma?, wordform?, part-of-speech?, polarity value?, word meaning?) What do all scores mean? https://github.com/cjhutto/vaderSentiment/blob/master/vaderSentiment/vader_lexicon.txt)

[3.5 points] Question1:

Regard the following sentences and their output as given by VADER. Regard sentences 1 to 7, and explain the outcome **for each sentence**. Take into account both the rules applied by VADER and the lexicon that is used. You will find that some of the results are reasonable, but others are not. Explain what is going wrong or not when correct and incorrect results are produced.

```
INPUT SENTENCE 1 I love apples
VADER OUTPUT {'neg': 0.0, 'neu': 0.192, 'pos': 0.808, 'compound':
0.6369}

INPUT SENTENCE 2 I don't love apples
VADER OUTPUT {'neg': 0.627, 'neu': 0.373, 'pos': 0.0, 'compound': -
0.5216}

INPUT SENTENCE 3 I love apples :-)
VADER OUTPUT {'neg': 0.0, 'neu': 0.133, 'pos': 0.867, 'compound':
0.7579}

INPUT SENTENCE 4 These houses are ruins
VADER OUTPUT {'neg': 0.492, 'neu': 0.508, 'pos': 0.0, 'compound': -
```

```
0.4404}

INPUT SENTENCE 5 These houses are certainly not considered ruins
VADER OUTPUT {'neg': 0.0, 'neu': 0.51, 'pos': 0.49, 'compound':
0.5867}

INPUT SENTENCE 6 He lies in the chair in the garden
VADER OUTPUT {'neg': 0.286, 'neu': 0.714, 'pos': 0.0, 'compound': -
0.4215}

INPUT SENTENCE 7 This house is like any house
VADER OUTPUT {'neg': 0.0, 'neu': 0.667, 'pos': 0.333, 'compound':
0.3612}
```

The first two sentences are marked very intuitively. VADER considers 'love' a positive word and its negation then as negative. The third sentence is considered more positive than the first because VADER takes the smiley face into account. Fourth sentece is similar to the first, 'ruins' is a negative word and thusly the sentece is negative. The fifth sentence is again a negation of the fourth one and so is positive. The last two senteces contain the words 'lies' and 'like' which have a negative and positive sentiment respectively in certain contexts. Therefore, even though they are neutral in these sentences, VADER labels them with their respective sentiments.

[Points: 2.5] Exercise 2: Collecting 50 tweets for evaluation

Collect 50 tweets. Try to find tweets that are interesting for sentiment analysis, e.g., very positive, neutral, and negative tweets. These could be your own tweets (typed in) or collected from the Twitter stream.

We will store the tweets in the file **my_tweets.json** (use a text editor to edit). For each tweet, you should insert:

- sentiment analysis label: negative | neutral | positive (this you determine yourself, this is not done by a computer)
- the text of the tweet
- the Tweet-URL

from:

```
    "1": {
        "sentiment_label": "",
        "text_of_tweet": "",
        "tweet_url": "",
```

to:

```
"1": {
        "sentiment_label": "positive",
        "text_of_tweet": "All across America people chose to get
involved, get engaged and stand up. Each of us can make a difference,
```

```
and all of us ought to try. So go keep changing the world in 2018.",
        "tweet_url" :
"https://twitter.com/BarackObama/status/946775615893655552",
    },
```

You can load your tweets with human annotation in the following way.

```python
import json

my_tweets = json.load(open('my_tweets.json'))

for id_, tweet_info in my_tweets.items():
    print(id_, tweet_info)
    break
```

```
0 {'sentiment_label': 'negative', 'text_of_tweet': "@IamNotThatAlex
The infamous doxxing website named after a bird. If that's not enough,
count yourself lucky and stay away ", 'tweet_url':
'https://twitter.com/im_just_laur/status/1553870749290713089'}
```

[5 points] Question 3:

Run VADER on your own tweets (see function **run_vader** from notebook **Lab2-Sentiment-analysis-using-VADER.ipynb**). You can use the code snippet below this explanation as a starting point.

- [2.5 points] a. Perform a quantitative evaluation. Explain the different scores, and explain which scores are most relevant and why.
- [2.5 points] b. Perform an error analysis: select 10 positive, 10 negative and 10 neutral tweets that are not correctly classified and try to understand why. Refer to the VADER-rules and the VADER-lexicon. Of course, if there are less than 10 errors for a category, you only have to check those. For example, if there are only 5 errors for positive tweets, you just describe those.

Task 3a

```python
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
vader_model = SentimentIntensityAnalyzer()

def vader_output_to_label(vader_output):
    """
    map vader output e.g.,
    {'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.4215}
    to one of the following values:
    a) positive float -> 'positive'
    b) 0.0 -> 'neutral'
    c) negative float -> 'negative'

    :param dict vader_output: output dict from vader

    :rtype: str
```

```python
    :return: 'negative' | 'neutral' | 'positive'
    """
    compound = vader_output['compound']

    if compound < 0:
        return 'negative'
    elif compound == 0.0:
        return 'neutral'
    elif compound > 0.0:
        return 'positive'

assert vader_output_to_label( {'neg': 0.0, 'neu': 0.0, 'pos': 1.0,
'compound': 0.0}) == 'neutral'
assert vader_output_to_label( {'neg': 0.0, 'neu': 0.0, 'pos': 1.0,
'compound': 0.01}) == 'positive'
assert vader_output_to_label( {'neg': 0.0, 'neu': 0.0, 'pos': 1.0,
'compound': -0.01}) == 'negative'

tweets = []
all_vader_output = []
gold = []

# settings (to change for different experiments)
to_lemmatize = True
pos = set()

for id_, tweet_info in my_tweets.items():
    the_tweet = tweet_info['text_of_tweet']
    vader_output = vader_model.polarity_scores(the_tweet)
    vader_label = vader_output_to_label(vader_output)

    tweets.append(the_tweet)
    all_vader_output.append(vader_label)
    gold.append(tweet_info['sentiment_label'])

from sklearn.metrics import classification_report

print(classification_report(gold, all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.64      0.30      0.41        23
     neutral       0.42      0.56      0.48         9
    positive       0.48      0.72      0.58        18

    accuracy                           0.50        50
   macro avg       0.51      0.53      0.49        50
weighted avg       0.54      0.50      0.48        50
```

From the classification report we can observer that the tweets with the negative label have a high precision and lower recall, and the tweets with the neutral and positive label have a lower precision, and higher recall. This indicates that VADER outputs negatively labeled tweets less compared to the (23) gold negative tweet labels. Similarly, VADER labeled more tweets as neutral and positive than there actually were in the gold set (9 and 18 respectivley). The f1-scores indicate that VADER was most succesfull for labeling positive tweets.

Task 3b

```
#print 10 tweets that were misclassified
for label in set(gold):
    print('LABEL:', label.upper())
    c = 0
    for i in range(len(tweets)):
        if gold[i] == label and all_vader_output[i] != label:
            print(f"Tweet {i}: {tweets[i]}")
            print('gold:', gold[i], '|| vader:', all_vader_output[i])
            print()
            c += 1
        if c == 10:
            break
    print()
```

```
LABEL: POSITIVE
Tweet 2: @A2Lintra @PEMatson @NewfieldsToday it closed real early, so
we didn't even quite finish the museum much less explore any of the
grounds. So, there is a lot more to see on a future visit.
gold: positive || vader: neutral

Tweet 6: @KMNetter @cathleendecker @pkcapitol She was on MSNBC while
it was going on. She was hardly cowering in fear and afraid to let
anyone in. The lady sounded both outraged and ready to kick ass to me.
Rightfully so.
gold: positive || vader: negative

Tweet 17: Just as you feel when you look on the river and sky, so I
felt; Just as any of you is one of a living crowd, I was one of a
crowd  --Walt Whitman, whose 200th birthday is Friday #Whitman200
https://t.co/yFdTwklH9h
gold: positive || vader: neutral

Tweet 35: When Covid struck Michigan hard @GovWhitmer listened to
science &amp; saved thousands of lives with her swift action.  But
make no mistake, the pathogens of hate and division spread, incited by
the president and his complicit supporters.   We got your back
#BigGretch.
gold: positive || vader: negative

Tweet 37: White people: watch the Starbucks arrest video.  See the
white folks arguing with the police and asking why two innocent black
```

men were being arrested? THIS is how you use your privilege. Because if one of us had said something we'd get arrested too.
gold: positive || vader: negative

LABEL: NEGATIVE
Tweet 0: @IamNotThatAlex The infamous doxxing website named after a bird. If that's not enough, count yourself lucky and stay away
gold: negative || vader: positive

Tweet 3: @Scotty1872 It's very easy to say that when you're in the group that is well represented.
gold: negative || vader: positive

Tweet 5: Nationalize the cruise lines, turn the ships into hospitals. https://t.co/AleE3BFqDa
gold: negative || vader: neutral

Tweet 9: @realkatiejow @AOC congrats on conflating sociopathy with patriotism
gold: negative || vader: positive

Tweet 13: 4.Condemn and censure our colleague who wore a confederate flag on his face to session on Friday, 04.24.2020. In fact, you can also pass the legislation that prohibits white supremacist symbols from the Capitol completely. 6/x
gold: negative || vader: positive

Tweet 20: I went to Monticello last year and it struck with me is that even saying The Founding Fathers were geniuses who also owned slaves is wrong. They are woven together. The freedom they had for academic pursuits was one afforded to them by human capital https://t.co/rwSWCXns3o
gold: negative || vader: positive

Tweet 21: Time that women spend asking and reminding their male partners to do housework, and time women spend praising and rewarding their male partners when they do housework, is time those women themselves are spending on housework.
gold: negative || vader: positive

Tweet 22: If you were an alien showing up on Earth for the first time in the last 24 hours and all you had to go on was what you read on Twitter, you would probably thinking the thing Charlie Watts was most famous for was punching Mick Jagger in the face.
gold: negative || vader: neutral

Tweet 30: You allowed cops to utilize your studio lot yesterday so they could get their teargas and rubber bullets ready. They set up

```
base there. Make a statement on that.  All of us, especially those of
us in your industry, want to know why https://t.co/aguVw1G6dO
gold: negative || vader: positive

Tweet 31: I felt so optimistic at the start of this week about my plan
to spend less time on Twitter, but the intervening months have tested
my resolve.
gold: negative || vader: positive


LABEL: NEUTRAL
Tweet 4: My favorite Stable Genius. https://t.co/PsNNKjlXb3
gold: neutral || vader: positive

Tweet 10: More #SCOTUS/OSG news. Brian Fletcher was a clerk to now-
Attorney General Merrick Garland on the D.C. Circuit and to Justice
Ruth Bader Ginsburg. https://t.co/ZHFV0xCY3P
gold: neutral || vader: positive

Tweet 26: @mims This cuts both ways. My wife's idea for national
service is that every person be forced to live and work for two years
in a region of the country distinctly separate from their own in terms
of economy and culture.
gold: neutral || vader: negative

Tweet 36: I'm really hitting the "wish for cosmic justice" phase of
this pandemic. Particularly the kind of justice that comes from the
threefold law.
gold: neutral || vader: positive
```

NEUTRAL LABELS:

The neutral tweets that were labeled incorrectly by VADER, generally seem to do so as a result of having positive or negative words in the sentence, that do not necessarily contribute to the sentiment. Examples of the positive words can be seen in tweets 4, 10 and 26, with words such as 'favorate' and 'justice'. The only tweet labeled as negative (tweet 26) contains negative words such as 'forced' and 'cut'.

POSITIVE LABELS:

All misclasifications for the tweets that should have been marked as positive, are because VADER marked them as neutral instead. Most of these tweets state something positive, despite something negative (2, 6, 35, 37). These generally contain a mix of positive and negative words, which makes it understandable that it was classified as neutral. More complex reasoning is required to understand the true meaning/sentiment of such statements. Additionally, one of the tweets contained a metaphorical statement (17), which is also inherently difficult to classify correctly.

NEGATIVE LABELS:

Many of the negative tweets, VADER has labeled as positive instead. This is partially due to people using positive (or strengthening) words to describe negative things, to emphasise how bad it is (0, 3, 20, 21, 22, 30, 31). In some cases, it is even done sarcastically (9). Some tweets that VADER marked as neutral contain a more cryptic negative sentiment, and does not contain (many strong) negative or positive words (5, 22).

[4 points] Question 4:

Run VADER on the set of airline tweets with the following settings:

- Run VADER (as it is) on the set of airline tweets
- Run VADER on the set of airline tweets after having lemmatized the text
- Run VADER on the set of airline tweets with only adjectives
- Run VADER on the set of airline tweets with only adjectives and after having lemmatized the text
- Run VADER on the set of airline tweets with only nouns
- Run VADER on the set of airline tweets with only nouns and after having lemmatized the text
- Run VADER on the set of airline tweets with only verbs
- Run VADER on the set of airline tweets with only verbs and after having lemmatized the text
- [1 point] a. Generate for all separate experiments the classification report, i.e., Precision, Recall, and F1 scores per category as well as micro and macro averages. **Use a different code cell (or multiple code cells) for each experiment.**
- [3 points] b. Compare the scores and explain what they tell you.
-
  - Does lemmatisation help? Explain why or why not.
-
  - Are all parts of speech equally important for sentiment analysis? Explain why or why not.

Task 4a

```
def run_vader(textual_unit, lemmatize=False,
parts_of_speech_to_consider=None, verbose=0):
    doc = nlp(textual_unit)
    input_to_vader = []

    for sent in doc.sents:
```

```
        for token in sent:
            to_add = token.text
            if lemmatize:
                to_add = token.lemma_
                if to_add == '-PRON-': to_add = token.text
            if parts_of_speech_to_consider and token.pos_ in
parts_of_speech_to_consider:
                input_to_vader.append(to_add)
            else:
                input_to_vader.append(to_add)

    scores = vader_model.polarity_scores(' '.join(input_to_vader))

    if verbose >= 1:
        print()
        print('INPUT SENTENCE', sent)
        print('INPUT TO VADER', input_to_vader)
        print('VADER OUTPUT', scores)

    return scores

import spacy
from sklearn.datasets import load_files

nlp = spacy.load('en_core_web_sm')
airline_tweets_train = load_files('airlinetweets')

#run vader on tweets
all_vader_output = list()
for sent in airline_tweets_train.data:
    scores = vader_model.polarity_scores(str(sent))
    #print()
    #print('VADER OUTPUT', scores)
    vader_label = vader_output_to_label(scores)
    all_vader_output.append(vader_label)

print(classification_report([airline_tweets_train.target_names[i] for
i in airline_tweets_train.target], all_vader_output))

              precision    recall  f1-score   support

    negative       0.80      0.49      0.60      1750
     neutral       0.57      0.56      0.56      1515
    positive       0.56      0.83      0.67      1490

    accuracy                           0.62      4755
   macro avg       0.64      0.63      0.61      4755
weighted avg       0.65      0.62      0.61      4755
```

```
#lemmatized text
all_vader_output = list()
for i in airline_tweets_train.data:
    lemtext = run_vader(str(i), lemmatize=True)
    vader_label = vader_output_to_label(lemtext)
    all_vader_output.append(vader_label)

print(classification_report([airline_tweets_train.target_names[i] for
i in airline_tweets_train.target], all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.79      0.52      0.63      1750
     neutral       0.60      0.49      0.54      1515
    positive       0.56      0.88      0.68      1490

    accuracy                           0.62      4755
   macro avg       0.65      0.63      0.62      4755
weighted avg       0.65      0.62      0.62      4755
```

```
#only adjectives
all_vader_output = list()
for i in airline_tweets_train.data:
    lemtext = run_vader(str(i), lemmatize=False,
parts_of_speech_to_consider={'ADJ'}, verbose=0)
    vader_label = vader_output_to_label(lemtext)
    all_vader_output.append(vader_label)

print(classification_report([airline_tweets_train.target_names[i] for
i in airline_tweets_train.target], all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.80      0.51      0.63      1750
     neutral       0.60      0.51      0.55      1515
    positive       0.56      0.88      0.68      1490

    accuracy                           0.63      4755
   macro avg       0.65      0.63      0.62      4755
weighted avg       0.66      0.63      0.62      4755
```

```
#only adjectives and after having lemmatized the text
all_vader_output = list()
for i in airline_tweets_train.data:
    lemtext = run_vader(str(i), lemmatize=True,
parts_of_speech_to_consider={'ADJ'}, verbose=0)
    vader_label = vader_output_to_label(lemtext)
    all_vader_output.append(vader_label)
```

```
print(classification_report([airline_tweets_train.target_names[i] for
i in airline_tweets_train.target], all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.79      0.52      0.63      1750
     neutral       0.60      0.49      0.54      1515
    positive       0.56      0.88      0.68      1490

    accuracy                           0.62      4755
   macro avg       0.65      0.63      0.62      4755
weighted avg       0.65      0.62      0.62      4755
```

```
#only nouns
all_vader_output = list()
for i in airline_tweets_train.data:
    lemtext = run_vader(str(i), lemmatize=False,
parts_of_speech_to_consider={'NOUN'}, verbose=0)
    vader_label = vader_output_to_label(lemtext)
    all_vader_output.append(vader_label)

print(classification_report([airline_tweets_train.target_names[i] for
i in airline_tweets_train.target], all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.80      0.51      0.63      1750
     neutral       0.60      0.51      0.55      1515
    positive       0.56      0.88      0.68      1490

    accuracy                           0.63      4755
   macro avg       0.65      0.63      0.62      4755
weighted avg       0.66      0.63      0.62      4755
```

```
#only nouns and after having lemmatized the text
all_vader_output = list()
for i in airline_tweets_train.data:
    lemtext = run_vader(str(i), lemmatize=True,
parts_of_speech_to_consider={'NOUN'}, verbose=0)
    vader_label = vader_output_to_label(lemtext)
    all_vader_output.append(vader_label)

print(classification_report([airline_tweets_train.target_names[i] for
i in airline_tweets_train.target], all_vader_output))
```

```
              precision    recall  f1-score   support
```

```
    negative       0.79      0.52      0.63      1750
     neutral       0.60      0.49      0.54      1515
    positive       0.56      0.88      0.68      1490

    accuracy                           0.62      4755
   macro avg       0.65      0.63      0.62      4755
weighted avg       0.65      0.62      0.62      4755
```

```
#only verbs
all_vader_output = list()
for i in airline_tweets_train.data:
    lemtext = run_vader(str(i), lemmatize=False,
parts_of_speech_to_consider={'VERB'}, verbose=0)
    vader_label = vader_output_to_label(lemtext)
    all_vader_output.append(vader_label)

print(classification_report([airline_tweets_train.target_names[i] for
i in airline_tweets_train.target], all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.80      0.51      0.63      1750
     neutral       0.60      0.51      0.55      1515
    positive       0.56      0.88      0.68      1490

    accuracy                           0.63      4755
   macro avg       0.65      0.63      0.62      4755
weighted avg       0.66      0.63      0.62      4755
```

```
#only verbs and after having lemmatized the text
all_vader_output = list()
for i in airline_tweets_train.data:
    lemtext = run_vader(str(i), lemmatize=True,
parts_of_speech_to_consider={'VERB'}, verbose=0)
    vader_label = vader_output_to_label(lemtext)
    all_vader_output.append(vader_label)

print(classification_report([airline_tweets_train.target_names[i] for
i in airline_tweets_train.target], all_vader_output))
```

```
              precision    recall  f1-score   support

    negative       0.79      0.52      0.63      1750
     neutral       0.60      0.49      0.54      1515
    positive       0.56      0.88      0.68      1490

    accuracy                           0.62      4755
   macro avg       0.65      0.63      0.62      4755
```

```
weighted avg       0.65      0.62      0.62      4755
```

Task 4b

Comparing the classification reports of lemmatized and unlemmatized tweets we see that lemmatization does not really result in a significant difference. For every report the scores differ between 0.01 and 0.05. Lemmatization converts a word to its base form which does not really help VADER with semtiment classification in this case.

The best performance that we get using only one POS catagory is from the only adjectives report. The f1 score is: neg 0.34, neutr 0.56 and pos 0.53 with an accuracy of 0.50. Specifically the recall is really high (0.89) with a low precision (0.41) for the neutral tweets meaning it returns many results, but most of its predicted labels are incorrect when compared to the training labels. The opposite is true for the negative and positive tweets. There the recall is lower than the precision. VADER predicts the sentiment correctly of a low number of sentences.

The next best POS category is only using verbs and then nouns as tags. VADER however performs best when all the POS labels are used. The respective performance of f1-scores when all labels are used is: neg 0.60, neutr 0.56 and pos 0.67 with an accuracy of 0.62 which is the highest out of all.

All POS are helpfull since VADER can use all words with sentiment to figure out the overall sentiment of a sentance. If you only want to use one tag in this model then you are best off by using the adjective tag. However it will always be more accurate when all the tags are used.

Part II: scikit-learn assignments

[4 points] Question 5

Train the scikit-learn classifier (Naive Bayes) using the airline tweets.

- Train the model on the airline tweets with 80% training and 20% test set and default settings (TF-IDF representation, min_df=2)
- Train with different settings:
  - with respect to vectorizing: TF-IDF ('airline_tfidf') vs. Bag of words representation ('airline_count')
  - with respect to the frequency threshold (min_df). Carry out experiments with increasing values for document frequency (min_df = 2; min_df = 5; min_df =10)
- [1 point] a. Generate a classification_report for all experiments
- [3 points] b. Look at the results of the experiments with the different settings and try to explain why they differ:
  - which category performs best, is this the case for any setting?
  - does the frequency threshold affect the scores? Why or why not according to you?

Task 5a

```
import nltk
import warnings
from nltk.corpus import stopwords
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer

warnings.filterwarnings('ignore')

for min_df in [2, 5, 10]:
    # Create feature extractors
    airline_vec = CountVectorizer(min_df=min_df,
tokenizer=nltk.word_tokenize, stop_words=stopwords.words('english'))
    tfidf_transformer = TfidfTransformer()

    # Extract features
    airline_counts =
airline_vec.fit_transform(airline_tweets_train.data)
    airline_tfidf = tfidf_transformer.fit_transform(airline_counts)

    # Split train and test data
    tfidf_train, tfidf_test, tfidf_y_train, tfidf_y_test =
train_test_split(airline_counts, airline_tweets_train.target,
test_size=0.2)
    count_train, count_test, count_y_train, count_y_test =
train_test_split(airline_counts, airline_tweets_train.target,
test_size=0.2)

    # Train and predict
    print(f"-------------------------
MIN_DF={min_df}-------------------------")
    print("Trained with tfidf:")
    print(classification_report(tfidf_y_test,
MultinomialNB().fit(tfidf_train, tfidf_y_train).predict(tfidf_test)),
'\n')
    print("Trained with bow:")
    print(classification_report(count_y_test,
MultinomialNB().fit(count_train, count_y_train).predict(count_test)))
    print()
```

```
-------------------------MIN_DF=2-------------------------
Trained with tfidf:
              precision    recall  f1-score   support

           0       0.84      0.88      0.86       354
           1       0.81      0.69      0.75       298
           2       0.81      0.87      0.84       299
```

```
    accuracy                           0.82       951
   macro avg       0.82      0.81      0.81       951
weighted avg       0.82      0.82      0.82       951


Trained with bow:
              precision    recall  f1-score   support

           0       0.84      0.93      0.88       339
           1       0.84      0.72      0.78       313
           2       0.82      0.85      0.84       299

    accuracy                           0.84       951
   macro avg       0.84      0.83      0.83       951
weighted avg       0.84      0.84      0.83       951


-------------------------MIN_DF=5-------------------------
Trained with tfidf:
              precision    recall  f1-score   support

           0       0.83      0.93      0.88       369
           1       0.83      0.71      0.77       278
           2       0.85      0.84      0.85       304

    accuracy                           0.84       951
   macro avg       0.84      0.83      0.83       951
weighted avg       0.84      0.84      0.84       951


Trained with bow:
              precision    recall  f1-score   support

           0       0.84      0.92      0.88       362
           1       0.80      0.78      0.79       295
           2       0.86      0.79      0.82       294

    accuracy                           0.83       951
   macro avg       0.83      0.83      0.83       951
weighted avg       0.83      0.83      0.83       951


-------------------------MIN_DF=10-------------------------
Trained with tfidf:
              precision    recall  f1-score   support

           0       0.81      0.94      0.87       310
           1       0.85      0.76      0.80       317
           2       0.86      0.82      0.84       324
```

```
    accuracy                           0.84       951
   macro avg       0.84      0.84      0.84       951
weighted avg       0.84      0.84      0.84       951


Trained with bow:
              precision    recall  f1-score   support

           0       0.82      0.91      0.86       350
           1       0.84      0.74      0.79       317
           2       0.83      0.83      0.83       284

    accuracy                           0.83       951
   macro avg       0.83      0.83      0.83       951
weighted avg       0.83      0.83      0.83       951
```

Task 5b

Comparing the accuracy of both settings, they seem to be very similar. For every frequency threshold the results are almost identical.

For min_df = 2 the accuracy is 0.85 for TF-IDF compared to 0.84 for Bag of words.

For min_df = 5 the accuracy is 0.84 for TF-IDF compared to 0.85 for Bag of words.

For min_df = 10 the accuracy is 0.84 for both settings.

Looking at these results we can also see that the frequency threshold does not really affect the scores. By increasing the min_df, more terms should be ignored that appear too infrequent. But by removing these terms, the accuracy does not increase. But a higher frequency threshold may be beneficial still by reducing the dimensionality of the imput vector without negatively impacting performance.

The reason for why a higher frequency threshold does not reduce accuracy might be because the words are still considered relatively infrequent and so the model is not able to learn an association between them and the sentiment of the tweet.

[4 points] Question 6: Inspecting the best scoring features

- Train the scikit-learn classifier (Naive Bayes) model with the following settings (airline tweets 80% training and 20% test; Bag of words representation ('airline_count'), min_df=2)
- [1 point] a. Generate the list of best scoring features per class (see function **important_features_per_class** below) [1 point]
- [3 points] b. Look at the lists and consider the following issues:
  - [1 point] Which features did you expect for each separate class and why?
  - [1 point] Which features did you not expect and why ?

- [1 point] The list contains all kinds of words such as names of airlines, punctuation, numbers and content words (e.g., 'delay' and 'bad'). Which words would you remove or keep when trying to improve the model and why?

Task 6a

```
airline_vec = CountVectorizer(min_df=min_df,
tokenizer=nltk.word_tokenize, stop_words=stopwords.words('english'))
airline_counts = airline_vec.fit_transform(airline_tweets_train.data)
data, _, labels, _ = train_test_split(airline_counts,
airline_tweets_train.target, test_size=0.2)
clf = MultinomialNB().fit(data, labels)

def important_features_per_class(vectorizer, classifier, n=80):
    class_labels = classifier.classes_
    feature_names = vectorizer.get_feature_names()
    topn_class1 = sorted(zip(classifier.feature_count_[0],
feature_names),reverse=True)[:n]
    topn_class2 = sorted(zip(classifier.feature_count_[1],
feature_names),reverse=True)[:n]
    topn_class3 = sorted(zip(classifier.feature_count_[2],
feature_names),reverse=True)[:n]
    print("Important words in negative documents")
    for coef, feat in topn_class1:
        print(class_labels[0], coef, feat)
    print("-----------------------------------------")
    print("Important words in neutral documents")
    for coef, feat in topn_class2:
        print(class_labels[1], coef, feat)
    print("-----------------------------------------")
    print("Important words in positive documents")
    for coef, feat in topn_class3:
        print(class_labels[2], coef, feat)

important_features_per_class(airline_vec, clf)

Important words in negative documents
0 1529.0 @
0 1389.0 united
0 1249.0 .
0 416.0 ``
0 409.0 flight
0 401.0 ?
0 371.0 !
0 325.0 #
0 222.0 n't
0 160.0 ''
0 139.0 's
0 114.0 :
0 111.0 service
```

```
0 108.0 virginamerica
0 99.0 get
0 99.0 cancelled
0 92.0 delayed
0 91.0 bag
0 89.0 customer
0 86.0 time
0 84.0 plane
0 74.0 ...
0 74.0 'm
0 73.0 http
0 73.0 -
0 69.0 hours
0 68.0 gate
0 65.0 ;
0 64.0 hour
0 61.0 still
0 59.0 airline
0 58.0 late
0 58.0 help
0 57.0 would
0 57.0 &
0 56.0 ca
0 56.0 2
0 55.0 flights
0 52.0 amp
0 51.0 worst
0 51.0 like
0 50.0 one
0 50.0 $
0 48.0 flightled
0 47.0 delay
0 46.0 've
0 45.0 waiting
0 45.0 never
0 43.0 us
0 43.0 3
0 43.0 (
0 40.0 really
0 40.0 lost
0 40.0 ever
0 40.0 )
0 39.0 back
0 38.0 wait
0 37.0 u
0 37.0 last
0 37.0 check
0 36.0 seat
0 36.0 due
0 36.0 another
```

```
0 35.0 fly
0 35.0 day
0 33.0 seats
0 33.0 people
0 33.0 luggage
0 33.0 bags
0 32.0 ticket
0 32.0 thanks
0 32.0 could
0 32.0 airport
0 31.0 hold
0 31.0 guys
0 31.0 even
0 30.0 problems
0 30.0 need
0 29.0 trying
0 29.0 staff
-----------------------------------------
Important words in neutral documents
1 1394.0 @
1 523.0 ?
1 521.0 .
1 304.0 jetblue
1 266.0 :
1 261.0 united
1 259.0 southwestair
1 249.0 #
1 244.0 flight
1 240.0 ``
1 186.0 americanair
1 169.0 !
1 168.0 http
1 164.0 usairways
1 134.0 's
1 83.0 get
1 73.0 ''
1 72.0 -
1 71.0 virginamerica
1 65.0 flights
1 64.0 please
1 62.0 help
1 58.0 )
1 54.0 n't
1 52.0 need
1 52.0 ...
1 48.0 (
1 45.0 ;
1 43.0 us
1 40.0 dm
1 39.0 would
```

```
1 39.0 tomorrow
1 37.0 know
1 37.0 &
1 35.0 “
1 35.0 way
1 35.0 thanks
1 35.0 fleet
1 35.0 fleek
1 34.0 change
1 33.0 hi
1 31.0 ”
1 31.0 like
1 31.0 could
1 31.0 'm
1 29.0 flying
1 28.0 cancelled
1 28.0 amp
1 27.0 fly
1 26.0 travel
1 26.0 time
1 26.0 one
1 26.0 number
1 25.0 see
1 25.0 new
1 25.0 check
1 23.0 ticket
1 22.0 today
1 22.0 airport
1 21.0 destinationdragons
1 21.0 2
1 20.0 rt
1 20.0 make
1 20.0 going
1 20.0 chance
1 19.0 use
1 19.0 next
1 19.0 go
1 19.0 booked
1 19.0 back
1 18.0 sent
1 18.0 guys
1 18.0 gate
1 18.0 follow
1 18.0 first
1 17.0 trying
1 17.0 tickets
1 17.0 start
1 17.0 seat
1 17.0 reservation
----------------------------------------
```

```
Important words in positive documents
2 1321.0 @
2 1038.0 !
2 751.0 .
2 304.0 southwestair
2 296.0 thanks
2 290.0 #
2 283.0 jetblue
2 252.0 united
2 244.0 ``
2 241.0 thank
2 182.0 flight
2 165.0 :
2 162.0 americanair
2 138.0 usairways
2 136.0 great
2 96.0 service
2 92.0 )
2 87.0 virginamerica
2 67.0 much
2 67.0 http
2 65.0 love
2 65.0 customer
2 64.0 guys
2 60.0 best
2 57.0 awesome
2 57.0 's
2 56.0 ;
2 51.0 -
2 48.0 good
2 45.0 time
2 44.0 airline
2 43.0 amazing
2 42.0 got
2 41.0 us
2 41.0 crew
2 40.0 &
2 39.0 n't
2 36.0 fly
2 35.0 today
2 35.0 help
2 35.0 get
2 34.0 ''
2 33.0 made
2 33.0 amp
2 32.0 response
2 30.0 flying
2 29.0 home
2 29.0 appreciate
2 29.0 ...
```

```
2 26.0 see
2 26.0 day
2 26.0 back
2 26.0 ?
2 25.0 work
2 25.0 like
2 24.0 u
2 24.0 nice
2 24.0 gate
2 24.0 'm
2 23.0 team
2 23.0 new
2 23.0 know
2 23.0 first
2 23.0 (
2 22.0 well
2 22.0 tonight
2 22.0 ever
2 21.0 would
2 21.0 please
2 21.0 're
2 20.0 yes
2 20.0 southwest
2 20.0 quick
2 20.0 helpful
2 20.0 getting
2 19.0 plane
2 19.0 job
2 19.0 class
2 18.0 staff
2 18.0 happy
```

Task 6b

Expected features for the negative class are cancelled, delayed, worst, last etc since they are clearly negative. There are a lot of neutral words listed which are a bit unexpected such as united, flight, service.

Expected features for the neutral class are words like names such as jetblue, southwestair, americair and tomorrow, know. We did not expect 'help' and 'please' to be that highly ranked in the neutral list.

For the positive list we see positive words like thanks, great, love etc in the upper part of the list. It is unexpected however that names such as the airlines are ranked really high (southwestair and jetblue higher than thanks).

The tweets are about airlines so a high occurence of names of the airline companies are expected. Maybe deleting these would improve the ranking of the model. On the other hand, certain airlines may generally have worse or better client satisfaction and may be correlated with the sentiment.

To further improve the model we would probably delete the standard make-up of the tweets such as: @, # and http. We would also remove stop-words. We would keep negations since negation handling plays an important role in classification. An example would be n't which is vital in sentences.