PS3

April 11, 2022

1 Problem Set 3

For this problem set, you will expand on PS2 to perform and evaluate various sentiment classification methods.

Your name: Ashley Cortez

You abc123: gql524

1.1 Submission Instructions

After completing the exercises below, generate a pdf of the code **with** outputs. After that create a zip file containing both the completed exercise and the generated PDF/HTML. You are **required** to check the PDF/HTML to make sure all the code **and** outputs are clearly visible and easy to read. If your code goes off the page, you should reduce the line size. I generally recommend not going over 80 characters.

Finally, name the zip file using a combination of your the assigment and your name, e.g., ps3_rios.zip

1.2 Exercise 1 (1 point)

For this step, you will load the training and test sentiment datasets "twitdata_TEST.tsv" and "allTrainingData.tsv". The data should be loaded into 4 lists of strings: X_txt_train, X_txt_test, y_test, y_train.

Note, when using csvreader, you need to pass the "quoting" the value csv.QUOTE_NONE.

[1]:
```python
import csv
file_train = open("allTrainingData.tsv")
reader_train = csv.reader(file_train,delimiter="\t", quoting=csv.QUOTE_NONE)

X_txt_train = []
y_train = []

for row in reader_train:
    y= row[2]
    X_txt = "\t".join(row[3:])
    X_txt_train.append(X_txt)
    y_train.append(y)
```

```
file_test = open("twitdata_TEST.tsv")
reader_test = csv.reader(file_test,delimiter="\t", quoting=csv.QUOTE_NONE)

X_txt_test = []
y_test = []

for row in reader_test:
    y = row[2]
    X_txt = "\t".join(row[3:])
    X_txt_test.append(X_txt)
    y_test.append(y)
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

[2]:
```
assert(type(X_txt_train) == type(list()))
assert(type(X_txt_train[0]) == type(str()))
assert(type(X_txt_test) == type(list()))
assert(type(X_txt_test[0]) == type(str()))
assert(type(y_test) == type(list()))
assert(type(y_train) == type(list()))
assert(len(X_txt_test) == 3199)
assert(len(y_test) == 3199)
assert(len(X_txt_train) == 8018)
assert(len(y_train) == 8018)
print("Asserts Completed Successfully!")
```

```
Asserts Completed Successfully!
```

1.3 Exercise 2 (2 point)

This part is similar to HW2 (using the positive_words and negative_words variables). We will compare last homework's lexicon-based classification method with supervised models. Only make predictions on the test split and store all predictions in the list lex_test_preds. Next, calculate the **macro** precision, macro recall, and macro f1 scores using the lex_test_preds list.

You can learn more about lexicon-based classification in Chapter 19.6. If you are interested, the chapter is available online for free at the following link: Speech and Language Processing

Also, note that I have wrote the code for this exercise, you job is to read the code, understand it, and apply it to make the predictions.

INTUITION: For PS2 you implemented a "lexicon-based classifier". You looked at a few examples and manually accessed it's performance. Howeover, that was arbitary. Now we want to see how well it actually works. Hence, in this homework (for this exercise), you will use the class I provided that implements the lexicon-based classifier and use the provided "annotatoed dataset" (loaded in Exercise 1) to see how well it performance. If it worked 100% accurately, the F1 would be 1.00. However, it does not, so you should expect a smaller score.

[3]:
```python
# DO NOT MODIFY THE CODE IN THIS CELL
class LexiconClassifier():
    def __init__(self):
        """
            Initalize the Lexicon classifer by loading lexicons.
        """
        self.positive_words = set()
        with open('positive-words.txt', encoding = 'utf-8') as iFile:
            for row in iFile:
                self.positive_words.add(row.strip())

        self.negative_words = set()
        with open('negative-words.txt', encoding='iso-8859-1') as iFile:
            for row in iFile:
                self.negative_words.add(row.strip())

    def predict(self, sentence):
        """
            Returns a sentiment prediction give an input string.

            Keyword arguments:
            sentence -- string (e.g., "This is good good good")

            Returns:
            pred -- a string ("postive, "negative", or "neutral")
        """
        num_pos_words = 0
        num_neg_words = 0
        for word in sentence.lower().split():
            if word in self.positive_words:
                num_pos_words += 1
            elif word in self.negative_words:
                num_neg_words += 1

        pred = 'neutral'
        if num_pos_words > num_neg_words:
            pred = 'positive'
        elif num_pos_words < num_neg_words:
            pred = 'negative'

        return pred

    def count_pos_words(self, sentence):
        """
            Returns the number of positive words in string

            Keyword arguments:
```

```
            sentence -- string (e.g., "This is good good good")

            Returns:
            pred -- an integer (e.g., 3)
        """
        num_pos_words = 0
        for word in sentence.lower().split():
            if word in self.positive_words:
                num_pos_words += 1
        return num_pos_words

    def count_neg_words(self, sentence):
        """
            Returns the number of negative words in string

            Keyword arguments:
            sentence -- string (e.g., "This is good good good")

            Returns:
            pred -- an integer (e.g., 3)
        """
        num_neg_words = 0
        for word in sentence.lower().split():
            if word in self.negative_words:
                num_neg_words += 1
        return num_neg_words
```

[4]:
```
# WRITE CODE HERE
import numpy as np
from sklearn.metrics import precision_score, recall_score, f1_score

lex_test_preds = [] # Initialize this as an empty list

LC = LexiconClassifier()
# Loop over X_txt_test
#    for each string in X_txt_test (i.e., for each item in the list), pass it␣
 ↪to LexiconClassifiers .predict() method
#    append the prediction to lex_test_preds

for string in X_txt_test:
    lex_test_preds.append(LC.predict(string))


precision = precision_score(y_test, lex_test_preds, average = 'macro')
recall = recall_score(y_test, lex_test_preds, average = 'macro')
f1 = f1_score(y_test, lex_test_preds, average = 'macro')
```

```
print("Precision: {:.4f}".format(precision))
print("Recall: {:.4f}".format(recall))
print("F1: {:.4f}".format(f1))
```

```
Precision: 0.5500
Recall: 0.5443
F1: 0.5452
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

[5]:
```
assert(type(lex_test_preds) == type(list()))
assert(type(lex_test_preds[0]) == type(str()))
assert(set(lex_test_preds) == set(["positive", "negative", "neutral"]))
assert(len(lex_test_preds) == len(y_test))
assert(type(precision) == type(float()) or type(precision) == type(np.
 ↪float64()))
assert(type(recall) == type(float()) or type(recall) == type(np.float64()))
assert(type(f1) == type(float()) or type(f1) == type(np.float64()))
print("Asserts Completed Successfully!")
```

```
Asserts Completed Successfully!
```

1.4 Exercise 3 (1 point)

Again, using the LexiconClassifier, write code to generate a lists of lists where each sublist contains the number of positive words and negative words in a tweet. For example, assume we are given the following train and test datasets

```
X_txt_train = ["good good", "bad bad"]
X_txt_test = ["great", "bad bad great"]
```

you should write code that creates two lists of lists as follows:

```
X_train_lexicon_features = [[2, 0], [0,2]] # [2, 0] means the first tweeta has 2 positive word
X_test_lexicon_features = [[1, 0], [1, 2]]
```

Why are we doing this? We will use these as addition features in Exercise 5, combining it with the ngram features. Combining different sets of features is called "Feature Engineering" and is one of the most important steps of many machine learning tasks. In this case, we are using the lexicons to generate additional features. But, we could also count the number of capitalized words, number of punctuation marks, etc. We would come up with different feature sets via trial-and-error. We can guess what type of features would help our task. For instance, for sentiment prediction, we may guess that having many capitalized words is predictive of something negative (e.g., "WHY ARE YOU DOING THIS!!!!!").

[6]:
```
# WRITE CODE HERE

X_train_lexicon_features = [] # Initailze to an empty list. This will be a list␣
 ↪of lists
```

```
X_test_lexicon_features = [] #  Initailze to an empty list. This will be a list␣
↪of lists

# Loop over X_txt_test
#    for each string in X_txt_test (i.e., for each item in the list), pass it␣
↪to LexiconClassifiers .count_pos_words() and count_neg_words method
#    append a list with the counts to X_test_lexicon_features

for string in X_txt_test:
    X_test_lexicon_features.append([LC.count_pos_words(string),
    LC.count_neg_words(string)])

# Loop over X_txt_train
#    for each string in X_txt_train (i.e., for each item in the list), pass it␣
↪to LexiconClassifiers .count_pos_words() and count_neg_words method
#    append a list with the counts to X_train_lexicon_features

for string in X_txt_train:
    X_train_lexicon_features.append([LC.count_pos_words(string),
    LC.count_neg_words(string)])
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

[7]:
```
assert(type(X_train_lexicon_features) == type(list()))
assert(type(X_test_lexicon_features) == type(list()))
assert(type(X_test_lexicon_features[0]) == type(list()))
assert(len(X_train_lexicon_features) == len(X_txt_train))
assert(len(X_test_lexicon_features) == len(X_txt_test))
assert(len(X_train_lexicon_features[0]) == 2)
assert(len(X_test_lexicon_features[0]) == 2)
print("Asserts Completed Successfully!")
```

```
Asserts Completed Successfully!
```

1.5 Exercise 4 (2 points)

For this task you should creat a feature matrix using CountVectorizer and train a LinearSVC model from scikit-learn. On the train split, use GridSearchCV to find the best LinearSVC C values (0.0001, 0.001, 0.001, 0.01, 0.1, 1, 10, or 100) based on the **macro** f1 scoring metric (hint: "macro" average) and set the cv parameter to 5. Also, with the CountVectorizer, only use unigrams (i.e., set ngram_range = (1,1)). Note that GridSearchCV will retrain the final classifier using the best parameters, so you don't need to do it manually.

INTUITION: For this exercise, you are implementing a simple linear model using bag-of-words features. This is generally a very strong and simple baseline for text classification. Compare the scores from this exercise to the results in Exercise 2. You will find that the machine learning-based model implemented here acheives better performance.

[9]:
```python
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import precision_score, recall_score, f1_score

import numpy as np
np.random.seed(42)
import random
random.seed(42)

vec = CountVectorizer(ngram_range = (1,1))

X_train = vec.fit_transform(X_txt_train)
X_test = vec.transform(X_txt_test)

svc = LinearSVC()
params = {'C':[0.0001, 0.001, 0.01, 0.1, 1., 10, 100]}
clf = GridSearchCV(svc, params, cv = 5)
clf.fit(X_train, y_train)


validation_score = clf.best_score_ # Get the score from the GridSearchCV "best␣
↪score"
print("Validation F1: {:.4f}".format(validation_score))

svm_test_predictions = clf.predict(X_test) # "predict" on X_test

precision = precision_score(y_test, svm_test_predictions, average = 'macro') #␣
↪Get scores using svm_test_predictions and y_test with the precision_score␣
↪method
recall = recall_score(y_test, svm_test_predictions, average = 'macro')
f1 = f1_score(y_test, svm_test_predictions, average = 'macro')

print("Precision: {:.4f}".format(precision))
print("Recall: {:.4f}".format(recall))
print("F1: {:.4f}".format(f1))
```

```
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
```

```
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(

Validation F1: 0.6593
Precision: 0.6525
Recall: 0.5740
F1: 0.5878
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

[10]:
```
from scipy.sparse import csr_matrix
assert(type(X_train) == type(csr_matrix(0)) or type(X_train) == type(np.
 ↪array(0)))
assert(type(X_test) == type(csr_matrix(0)) or type(X_test) == type(np.array(0)))
assert(X_train.shape[0] == len(X_txt_train))
assert(X_test.shape[0] == len(X_txt_test))
assert(X_train.shape[1] == X_test.shape[1])
assert(type(precision) == type(float()) or type(precision) == type(np.
 ↪float64()))
assert(type(recall) == type(float()) or type(recall) == type(np.float64()))
assert(type(f1) == type(float()) or type(f1) == type(np.float64()))
print("Asserts Completed Successfully!")
```

```
Asserts Completed Successfully!
```

1.6 Exercise 5 (2 points)

Repeat the experiment from exercise 4, but include the lexicon features (from exercise 3) with the CountVectorizer features. Specifically, you need to concatenate the variables `X_train_lexicon_features` and `X_test_lexicon_features` with `X_train` and `X_test`, respectively. Intuitively, we are performing feature engineering by adding "lexicon features".

HINT: You will need to convert the lexicon features to numpy arrays then call hstack from the scipy.sparse library (https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.hstack.html)

```
from scipy.sparse import hstack
new_matrix = hstack([ngram_matrix, lexicon_matrix])
```

Again, this is a "Feature Engineering" exercise. This is common in all machine learning tasks.

INTUITION: How do we improve the model Exercise 4? You could try different machine learning models. However, it is generally better to focus on feature engineering. What information can you provide the model to make better predictions? Here we will try to combine counts using the Lexicon from Exercise 3 as additional features (i.e., combined with the bag-of-word features) from Exercise 4.

[15]:
```
import scipy.sparse as sp
import numpy as np
np.random.seed(42)
import random
random.seed(42)

# WRITE CODE HERE

# Summary:
```

```
# 1. Convert X_txt_train and X_txt_test to matricies of numbers (i.e., use␣
 ↪CountVectorizer)

vec = CountVectorizer(ngram_range = (1,1))

X_train_w_lex = vec.fit_transform(X_txt_train) # This will be the matrix from␣
 ↪CountVectorizer (X_txt_train)
X_test_w_lex = vec.transform(X_txt_test)

# Now we need to convert X_train_lexicon_features and X_test_lexicon_features␣
 ↪to numpy arrays

# "hstack" X_train_lexicon_features with X_train_w_lex

X_train_w_lex = sp.hstack([X_train_lexicon_features, X_train_w_lex]).toarray()

# "hstack" X_test_lexicon_features with X_test_w_lex

X_test_w_lex = sp.hstack([X_test_lexicon_features, X_test_w_lex]).toarray()

# Initialize the classifier LinearSVC

svc = LinearSVC()

# Create the params with the C values

params = {'C':[0.0001, 0.001, 0.01, 0.1, 1., 10, 100]}

# Initialize GridSearchCV

clf = GridSearchCV(svc, params, cv = 5)

# "fit" the model  on X_train_w_lex

clf.fit(X_train_w_lex, y_train)


validation_score = clf.best_score_
print("Validation F1: {:.4f}".format(validation_score))

svm_lex_test_predictions = clf.predict(X_test_w_lex) # Get predictions on␣
 ↪X_test_w_lex

precision = precision_score(y_test, svm_lex_test_predictions, average =␣
 ↪'macro') # Get scores using svm_test_predictions and y_test with the␣
 ↪precision_score method
recall = recall_score(y_test, svm_lex_test_predictions, average = 'macro')
```

```python
f1 = f1_score(y_test, svm_lex_test_predictions, average = 'macro')


print("Precision: {:.4f}".format(precision))
print("Recall: {:.4f}".format(recall))
print("F1: {:.4f}".format(f1))
```

```
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
```

```
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/svm/_base.py:1206: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
```

```
Validation F1: 0.6729
Precision: 0.6894
Recall: 0.5719
F1: 0.5875
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

[12]:
```python
from scipy.sparse import csr_matrix
assert(X_train_w_lex.shape[0] == len(X_txt_train))
assert(X_test.shape[0] == len(X_txt_test))
assert(X_train_w_lex.shape[1] == X_test.shape[1] + 2)
assert(X_train_w_lex.shape[1] == X_test_w_lex.shape[1])
assert(type(precision) == type(float()) or type(precision) == type(np.
 ↪float64()))
assert(type(recall) == type(float()) or type(recall) == type(np.float64()))
assert(type(f1) == type(float()) or type(f1) == type(np.float64()))
print("Asserts Completed Successfully!")
```

```
Asserts Completed Successfully!
```

1.7 Exercise 6 (1 point)

For this exercise, you will perform manual analysis of the predictions. Answer the questions below.

[13]:
```python
num_tweets = 0
for text, svm_pred, svm_lex_pred, lex_pred, y  in zip(X_txt_test,␣
 ↪svm_test_predictions, svm_lex_test_predictions, lex_test_preds, y_test):
    print("Tweet: {}".format(text))
    print("Ground-Truth Class: {}".format(y))
    print("SVM Prediction: {}".format(svm_pred))
    print("SVM+Lexicon Prediction: {}".format(svm_lex_pred))
    print("Lexicon Model Prediction: {}".format(lex_pred))
    print()

    num_tweets += 1
    if num_tweets == 20:
        break
```

```
Tweet: Musical awareness: Great Big Beautiful Tomorrow has an ending, Now is the
time does not
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: positive
Lexicon Model Prediction: positive

Tweet: On Radio786 100.4fm 7:10 Fri Oct 19 Labour analyst Shawn Hattingh:
Cosatu's role in the context of unrest in the mining http://t.co/46pjzzl6
Ground-Truth Class: neutral
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: negative

Tweet: Kapan sih lo ngebuktiin,jan ngomong doang Susah Susah.usaha Aja blm udh
nyerah,inget.if you never try you'll never know.cowok kok gentle bgt
Ground-Truth Class: negative
SVM Prediction: neutral
SVM+Lexicon Prediction: positive
Lexicon Model Prediction: positive

Tweet: Tomorrow come and hear @DavidWillettsMP&amp;@MASieghart debate
"Navigating the new Higher Education market" 5.30pm, Jurys Inn #CPC12
Ground-Truth Class: neutral
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

Tweet: Excuse the connectivity of this live stream, from Baba Amr, so many
activists using only one Sat Modem. LIVE http://t.co/U283IhZ5 #Homs
Ground-Truth Class: neutral
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: negative

Tweet: Show your LOVE for your local field &amp; it might win an award!
Gallagher Park #Bedlington current 4th in National Award http://t.co/WeiMDtQt
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: positive
Lexicon Model Prediction: positive

Tweet: @firecore Can you tell me when an update for the Apple TV 3rd gen becomes
available? The missing update holds me back from buying #appletv3
Ground-Truth Class: positive
```

```
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

Tweet: @Heavensbasement The Crown, Filthy McNastys, Katy Dalys or the Duke if
York in Belfast! Can't wait to catch you guys tomorrow night!
Ground-Truth Class: positive
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: negative

Tweet: Uncover the Eternal City! Return flights to Rome travel on the 21st
January, for 3 nights Augustea, 3 star Hotel… http://t.co/tw0Jeh9g
Ground-Truth Class: neutral
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

Tweet: My #cre blog Oklahoma Per Square Foot returns to the @JournalRecord blog
hub tomorrow. I will have some interesting local data to share.
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: positive
Lexicon Model Prediction: positive

Tweet: "@bbcburnsy: Loads from SB; talks with Chester continue; no deals 4 out
of contract players 'til Jan; Dev t Roth ,Coops to Chest'ld #hcafc"
Ground-Truth Class: negative
SVM Prediction: negative
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

Tweet: Trey Burke has been suspended for the Northern Michigan game (exhibition)
tomorrow. http://t.co/oefkAElW
Ground-Truth Class: negative
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

Tweet: W.O.W Wednesday!Marni lands this Lumberjack vest for the ladies looking
to bring a little Tom boy toughness  http://t.co/7NyCbdJR
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: negative

Tweet: Activists in Deir Ezzor captured this image of Musab Bin Umair Mosque
after regime forces set it on fire Wednesday. http://t.co/MRcoprCE
```

```
Ground-Truth Class: negative
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

Tweet: @karaotr You will appreciate this.. Sunday brunch coffee: Normal cup in
b/g and then the BOWL of java. Yowza. http://t.co/XhbtaCvm
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: positive

Tweet: Join me Wed for a live webcast on cost optimization for IT, for the SMB
crowd. http://t.co/tyJn4RES  &lt;&lt; send your questions in! #DellWebcast
Ground-Truth Class: positive
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

Tweet: Special THANKS to EVERYONE for coming out to Taboo Tuesday With DST
tonight! It was FUN&amp;educational!!! :) @XiEtaDST
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: positive
Lexicon Model Prediction: negative

Tweet: @fatimasule That was the revelation I mentioned on sunday evening. I am
still in Abj. How are u &amp; where have u been again?
Ground-Truth Class: positive
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: positive

Tweet: Kim Hyung Jun - Football Team the 2nd A Match at YeongDeungPo-gu
DaeRimDong [12.10.27] Credit : tlxhah #6 http://t.co/u7mPTl0X
Ground-Truth Class: neutral
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

Tweet: The audio booth is ready to blow the roof off the Comcast Center
tomorrow! Are you? #MDMadness http://t.co/B19fECgY
Ground-Truth Class: positive
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral
```

Complete the following tasks:

- Manually annotate all of the tweets printed above:
  1. Tweet 1 Sentiment: Positive
  2. Tweet 2 Sentiment: Neutral
  3. Tweet 3 Sentiment: Neutral
  4. Tweet 4 Sentiment: Neutral
  5. Tweet 5 Sentiment: Neutral
  6. Tweet 6 Sentiment: Positive
  7. Tweet 7 Sentiment: Neutral
  8. Tweet 8 Sentiment: Positive
  9. Tweet 9 Sentiment: Neutral
  10. Tweet 10 Sentiment: Positive
  11. Tweet 11 Sentiment: Negative
  12. Tweet 12 Sentiment: Negative
  13. Tweet 13 Sentiment: Positive
  14. Tweet 14 Sentiment: Neutral
  15. Tweet 15 Sentiment: Positive
  16. Tweet 16 Sentiment: Neutral
  17. Tweet 17 Sentiment: Positive
  18. Tweet 18 Sentiment: Neutral
  19. Tweet 19 Sentiment: Neutral
  20. Tweet 20 Sentiment: Neutral
- How many of your annotations match the ground truth labels? Do you think the datasets labels are correct? (Use your intuition)
  - about 18 of my annotations match the ground truth labels. I think the labels are subjective, but mostly represent the "correct" label given the majority.
- How many of your annotations match the lexicon-based model's predictions?
  - about 15
- How many of your annotations match the SVM's predictions?
  - about 18
- How many of your annotations match the SVM+Lexicon's predictions?
  - about 14
- Do you see any major limitations of the linear SVM model? Use your intuition, I will accept most answers, as long as it makes some sense. Please describe and provide examples below:
  - a limitation of the SVM model is that it is not suitable for large datasets, and SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.

1.8 Exercise 6 (1 point)

For this exercise, you should come up with 10 other potential features that could be useful for sentiment analysis. You do not need to implement them. You simply need to list this. Make sure it is easy for me to understand the feature you describe. An example could be "The count of the number of capitalized words in the text".

1. Idea 1 - speed with automation to score sentiments
2. Idea 2 - identify any emojis in text
3. Idea 3 - multilingual - being able to score in different languages

4. Idea 4 - social media - looking at captions and hashtags to determine sentiment
5. Idea 5 - multimedia - analyze videos such as reels and tiktoks
6. Idea 6 - automation to sort data into sentiment categories: positive, negative, neutral
7. Idea 7 - real time analysis so companies can quickly address negative/positive comments
8. Idea 8 - scoring for APIs to help accuracy - similar to one hot coding
9. Idea 9 - dashboard to very insight statistics - helps read/view sentiments quickly
10. Idea 10 - analyze brand monitoring - see where else your business is tagged outside of twitter, ie. Reddit, Facebook etc.

1.9 Extra Credit 1 (2 points)

For this extra credit the only goal is to improve your model on the test set (i.e., increase the **macro** f1 score). You may create new features, grid search over more parameters, try different feature weighting methods (e.g., TfidfVectorizer), or test different machine learning models. You can do whatever you want as long as the final test score improves, I will provide you with the extra credit points.

DO NOT TRAIN ON THE TEST SET. That is cheating!

[16]:
```
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
from sklearn.model_selection import GridSearchCV


params = {'n_estimators': [10, 100]}

rf = RandomForestClassifier()

clf_rf = GridSearchCV(rf, params, scoring = 'f1', cv = 2)

clf_rf.fit(X_train_w_lex, y_train)


validation_score = clf_rf.best_score_
print("Validation F1: {:.4f}".format(validation_score))

svm_lex_test_predictions_rf = clf_rf.predict(X_test_w_lex)

precision = precision_score(y_test, svm_lex_test_predictions_rf, average =␣
 ↪'macro')
recall = recall_score(y_test, svm_lex_test_predictions_rf, average = 'macro')
f1 = f1_score(y_test, svm_lex_test_predictions_rf, average = 'macro')


print("Precision: {:.4f}".format(precision))
print("Recall: {:.4f}".format(recall))
print("F1: {:.4f}".format(f1))
```

```
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/model_selection/_validation.py:770: UserWarning: Scoring
failed. The score on this train-test partition for these parameters will be set
to nan. Details:
Traceback (most recent call last):
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/model_selection/_validation.py", line 761, in _score
    scores = scorer(estimator, X_test, y_test)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_scorer.py", line 216, in __call__
    return self._score(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_scorer.py", line 264, in _score
    return self._sign * self._score_func(y_true, y_pred, **self._kwargs)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1123, in f1_score
    return fbeta_score(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1261, in fbeta_score
    _, _, f, _ = precision_recall_fscore_support(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1544, in
precision_recall_fscore_support
    labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1365, in
_check_set_wise_labels
    raise ValueError(
ValueError: Target is multiclass but average='binary'. Please choose another
average setting, one of [None, 'micro', 'macro', 'weighted'].

  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/model_selection/_validation.py:770: UserWarning: Scoring
failed. The score on this train-test partition for these parameters will be set
to nan. Details:
Traceback (most recent call last):
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/model_selection/_validation.py", line 761, in _score
    scores = scorer(estimator, X_test, y_test)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_scorer.py", line 216, in __call__
    return self._score(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_scorer.py", line 264, in _score
    return self._sign * self._score_func(y_true, y_pred, **self._kwargs)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1123, in f1_score
```

```
    return fbeta_score(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1261, in fbeta_score
    _, _, f, _ = precision_recall_fscore_support(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1544, in
precision_recall_fscore_support
    labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1365, in
_check_set_wise_labels
    raise ValueError(
ValueError: Target is multiclass but average='binary'. Please choose another
average setting, one of [None, 'micro', 'macro', 'weighted'].

  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/model_selection/_validation.py:770: UserWarning: Scoring
failed. The score on this train-test partition for these parameters will be set
to nan. Details:
Traceback (most recent call last):
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/model_selection/_validation.py", line 761, in _score
    scores = scorer(estimator, X_test, y_test)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_scorer.py", line 216, in __call__
    return self._score(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_scorer.py", line 264, in _score
    return self._sign * self._score_func(y_true, y_pred, **self._kwargs)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1123, in f1_score
    return fbeta_score(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1261, in fbeta_score
    _, _, f, _ = precision_recall_fscore_support(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1544, in
precision_recall_fscore_support
    labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1365, in
_check_set_wise_labels
    raise ValueError(
ValueError: Target is multiclass but average='binary'. Please choose another
average setting, one of [None, 'micro', 'macro', 'weighted'].

  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/model_selection/_validation.py:770: UserWarning: Scoring
failed. The score on this train-test partition for these parameters will be set
to nan. Details:
Traceback (most recent call last):
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/model_selection/_validation.py", line 761, in _score
    scores = scorer(estimator, X_test, y_test)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_scorer.py", line 216, in __call__
    return self._score(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_scorer.py", line 264, in _score
    return self._sign * self._score_func(y_true, y_pred, **self._kwargs)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1123, in f1_score
    return fbeta_score(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1261, in fbeta_score
    _, _, f, _ = precision_recall_fscore_support(
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1544, in
precision_recall_fscore_support
    labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)
  File "/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/metrics/_classification.py", line 1365, in
_check_set_wise_labels
    raise ValueError(
ValueError: Target is multiclass but average='binary'. Please choose another
average setting, one of [None, 'micro', 'macro', 'weighted'].

  warnings.warn(
/Users/ashley/opt/anaconda3/lib/python3.9/site-
packages/sklearn/model_selection/_search.py:969: UserWarning: One or more of the
test scores are non-finite: [nan nan]
  warnings.warn(
```

```
Validation F1: nan
Precision: 0.6226
Recall: 0.5261
F1: 0.5304
```