dataclean

December 6, 2023

[126]:

```
# Importing necessary libraries for dataclean
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.corpus import stopwords
import re
import string
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
import os
import pandas as pd
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /home/5a19efe1-b761-4c88-855d-
[nltk_data]     8dc5b6196096/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/5a19efe1-b761-4c88-855d-
[nltk_data]     8dc5b6196096/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /home/5a19efe1-b761-4c88-855d-
[nltk_data]     8dc5b6196096/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[128]:

```
# Importing tools for preprocessing and analysis
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import chi2
import numpy as np
```

[130]:

```
#Import Data
data = pd.read_csv('./collected_data_raw_initial2.csv', encoding = 'utf-8-sig')
data.head(10)
```

```
[130]:    Unnamed: 0     Video Id                                              Title  \
       0           0  vGj_eEDLRyY      tw e d | high restriction what i eat in a day
       1           1  2U0zFtD-FSQ  TW:ED | What I Eat in a Week  High Restrictio…
       2           2  zenSuGmybJI  tw 3d | what I 3at in a week - 5 days of mid r…
       3           3  wYjd6y2y_ZU  tw ed | vlog: what i eat in a week of high res…
       4           4  67vGc6s-IcI    TW: ED / What I eat in a day / high restriction
       5           5  ut5X3KeEZfY  tw ed // what i eat in a day // high restricti…
       6           6  -mw2J31CP0A  What I Eat In A Day: DisOrded Eating Edition. …
       7           7  0s6ohW06YXY       tw ed / what I eat in a day (restricting) #1
       8           8  eoxBkhLbI_g  tw ed // what I eat in a week // high res &amp…
       9           9  VOWCK0iCtKw  ( tw: ed) what I eat in a day | high res | die…

                                                Description  Comments  \
       0                                        love you  :       NaN
       1  TW:ED // I am not promoting this lifestyle .. …       NaN
       2                                              #edtwt       NaN
       3  #eating \n\n            \nhi !! \nth…        NaN
       4  #shorts\n\n TW \nI do not promote eating dis…       NaN
       5  first vid ^^;!!\nHi this is what a day on my h…       NaN
       6                                                 NaN       NaN
       7  this is my own document of my diet\n\noriginal…       NaN
       8  I’m not here to promote EDs. I’m here to share…       NaN
       9                                                 NaN       NaN

                                       Query  Pro_or_Con  Informative  Markup_terms
       0  what i eat in a day high restriction           1            0             1
       1  what i eat in a day high restriction           1            0             1
       2  what i eat in a day high restriction           1            0             1
       3  what i eat in a day high restriction           1            0             1
       4  what i eat in a day high restriction           1            0             1
       5  what i eat in a day high restriction           1            0             1
       6  what i eat in a day high restriction           1            0             1
       7  what i eat in a day high restriction           1            0             1
       8  what i eat in a day high restriction           1            0             1
       9  what i eat in a day high restriction           1            0             1
```

```
[132]: ####STEP 1: DATACLEAN####

       data['Title'] = data['Title'].map(lambda x: re.sub(r'\d+', '', str(x)))
       data['Description'] = data['Description'].map(lambda x: re.sub(r'\d+', '',␣
        ↪str(x)))



       data['Title'] = data['Title'].map(lambda x: x.lower())
       data['Description'] = data['Description'].map(lambda x: x.lower())
```

```
data['Title']  = data['Title'].map(lambda x: x.translate(x.maketrans('', '',␣
 ↪string.punctuation)))
data['Description']  = data['Description'].map(lambda x: x.translate(x.
 ↪maketrans('', '', string.punctuation)))

data['Title'] = data['Title'].map(lambda x: x.strip())
data['Description'] = data['Description'].map(lambda x: x.strip())

data['Title'] = data['Title'].map(lambda x: word_tokenize(x))
data['Description'] = data['Description'].map(lambda x: word_tokenize(x))

data['Title'] = data['Title'].map(lambda x: [word for word in x if word.
 ↪isalpha()])
data['Description'] = data['Description'].map(lambda x: [word for word in x if␣
 ↪word.isalpha()])

stop_words = set(stopwords.words('english'))
data['Title'] = data['Title'].map(lambda x: [w for w in x if not w in␣
 ↪stop_words])
data['Description'] = data['Description'].map(lambda x: [w for w in x if not w␣
 ↪in stop_words])
#df['TextColumn'] = df['TextColumn'].replace('[nan]', '', regex=True)

#word lemmatization
lem = WordNetLemmatizer()
data['Title'] = data['Title'].map(lambda x: [lem.lemmatize(word,"v") for word␣
 ↪in x])
data['Description'] = data['Description'].map(lambda x: [lem.
 ↪lemmatize(word,"v") for word in x])

#take off links people put in the description
data['Description'] = data['Description'].map(lambda lst: [word for word in lst␣
 ↪if not word.startswith("https")])

data.head(10)
```

```
[132]:    Unnamed: 0     Video Id                                               Title  \
       0           0  vGj_eEDLRyY                [tw, e, high, restriction, eat, day]
       1           1  2U0zFtD-FSQ    [twed, eat, week, high, restriction, diet, vlog]
       2           2  zenSuGmybJI                  [tw, week, days, mid, restriction]
       3           3  wYjd6y2y_ZU       [tw, ed, vlog, eat, week, high, restrictions]
       4           4  67vGc6s-IcI               [tw, ed, eat, day, high, restriction]
       5           5  ut5X3KeEZfY   [tw, ed, eat, day, high, restriction, edition,…
       6           6  -mw2J31CP0A   [eat, day, disorded, eat, edition, lowcalorie,…
       7           7  0s6ohW06YXY                        [tw, ed, eat, day, restrict]
       8           8  eoxBkhLbI_g   [tw, ed, eat, week, high, res, amp, binge, ed,…
```

```
9           9  VOWCKOiCtKw          [tw, ed, eat, day, high, res, diet, vlog]

                                           Description  Comments  \
0                                               [love]       NaN
1  [twed, promote, lifestyle, please, read, discl…       NaN
2                                              [edtwt]       NaN
3  [eat, hi, first, video, pls, leave, comment, a…       NaN
4  [short, promote, eat, disorder, want, glamoriz…       NaN
5  [first, vid, hi, day, high, restriction, diet,…       NaN
6                                                [nan]       NaN
7                 [document, diet, original, music]       NaN
8  [promote, eds, share, struggle, one, bother, s…       NaN
9                                                [nan]       NaN

                                 Query  Pro_or_Con  Informative  Markup_terms
0  what i eat in a day high restriction           1            0             1
1  what i eat in a day high restriction           1            0             1
2  what i eat in a day high restriction           1            0             1
3  what i eat in a day high restriction           1            0             1
4  what i eat in a day high restriction           1            0             1
5  what i eat in a day high restriction           1            0             1
6  what i eat in a day high restriction           1            0             1
7  what i eat in a day high restriction           1            0             1
8  what i eat in a day high restriction           1            0             1
9  what i eat in a day high restriction           1            0             1
```

[134]:

```
#turn list back to string
data['Title'] = data['Title'].map(lambda x: ' '.join(x))
data['Description'] = data['Description'].map(lambda x: ' '.join(x))
data.head(10)
```

[134]:

```
   Unnamed: 0     Video Id                                      Title  \
0           0  vGj_eEDLRyY                  tw e high restriction eat day
1           1  2U0zFtD-FSQ        twed eat week high restriction diet vlog
2           2  zenSuGmybJI                   tw week days mid restriction
3           3  wYjd6y2y_ZU           tw ed vlog eat week high restrictions
4           4  67vGc6s-IcI                  tw ed eat day high restriction
5           5  ut5X3KeEZfY     tw ed eat day high restriction edition cals
6           6  -mw2J31CP0A  eat day disorded eat edition lowcalorie wieiad…
7           7  0s6ohW06YXY                         tw ed eat day restrict
8           8  eoxBkhLbI_g         tw ed eat week high res amp binge ed vlog
9           9  VOWCKOiCtKw                tw ed eat day high res diet vlog

                                           Description  Comments  \
0                                                 love       NaN
1  twed promote lifestyle please read disclaimer …       NaN
2                                                edtwt       NaN
```

```
3  eat hi first video pls leave comment anything …      NaN
4  short promote eat disorder want glamorize roma…      NaN
5  first vid hi day high restriction diet look li…      NaN
6                                                nan       NaN
7                         document diet original music       NaN
8  promote eds share struggle one bother simply w…      NaN
9                                                nan       NaN

                                  Query  Pro_or_Con  Informative  Markup_terms
0  what i eat in a day high restriction           1            0             1
1  what i eat in a day high restriction           1            0             1
2  what i eat in a day high restriction           1            0             1
3  what i eat in a day high restriction           1            0             1
4  what i eat in a day high restriction           1            0             1
5  what i eat in a day high restriction           1            0             1
6  what i eat in a day high restriction           1            0             1
7  what i eat in a day high restriction           1            0             1
8  what i eat in a day high restriction           1            0             1
9  what i eat in a day high restriction           1            0             1
```

```
[136]: #for videos that do not have a description
       data['Description'] = data['Description'].replace(['nan'], '', regex=True)
       data.head(10)
```

```
[136]:    Unnamed: 0     Video Id                                               Title  \
       0           0  vGj_eEDLRyY                       tw e high restriction eat day
       1           1  2U0zFtD-FSQ            twed eat week high restriction diet vlog
       2           2  zenSuGmybJI                        tw week days mid restriction
       3           3  wYjd6y2y_ZU               tw ed vlog eat week high restrictions
       4           4  67vGc6s-IcI                      tw ed eat day high restriction
       5           5  ut5X3KeEZfY         tw ed eat day high restriction edition cals
       6           6  -mw2J31CP0A  eat day disorded eat edition lowcalorie wieiad…
       7           7  0s6ohW06YXY                              tw ed eat day restrict
       8           8  eoxBkhLbI_g           tw ed eat week high res amp binge ed vlog
       9           9  V0WCK0iCtKw                    tw ed eat day high res diet vlog

                                                Description  Comments  \
       0                                                love       NaN
       1  twed promote lifestyle please read disclaimer …      NaN
       2                                               edtwt       NaN
       3  eat hi first video pls leave comment anything …      NaN
       4  short promote eat disorder want glamorize roma…      NaN
       5  first vid hi day high restriction diet look li…      NaN
       6                                                           NaN
       7                         document diet original music       NaN
       8  promote eds share struggle one bother simply w…      NaN
       9                                                           NaN
```

```
                                Query  Pro_or_Con  Informative  Markup_terms
0  what i eat in a day high restriction           1            0             1
1  what i eat in a day high restriction           1            0             1
2  what i eat in a day high restriction           1            0             1
3  what i eat in a day high restriction           1            0             1
4  what i eat in a day high restriction           1            0             1
5  what i eat in a day high restriction           1            0             1
6  what i eat in a day high restriction           1            0             1
7  what i eat in a day high restriction           1            0             1
8  what i eat in a day high restriction           1            0             1
9  what i eat in a day high restriction           1            0             1
```

[138]:

```
####STEP 2.1: USE TF IDF FOR KEYWORDS####
#encode classes by key word (tried by pro and con but too vague)
le = LabelEncoder()
le.fit(data.Query)
data.Query = le.transform(data.Query)
```

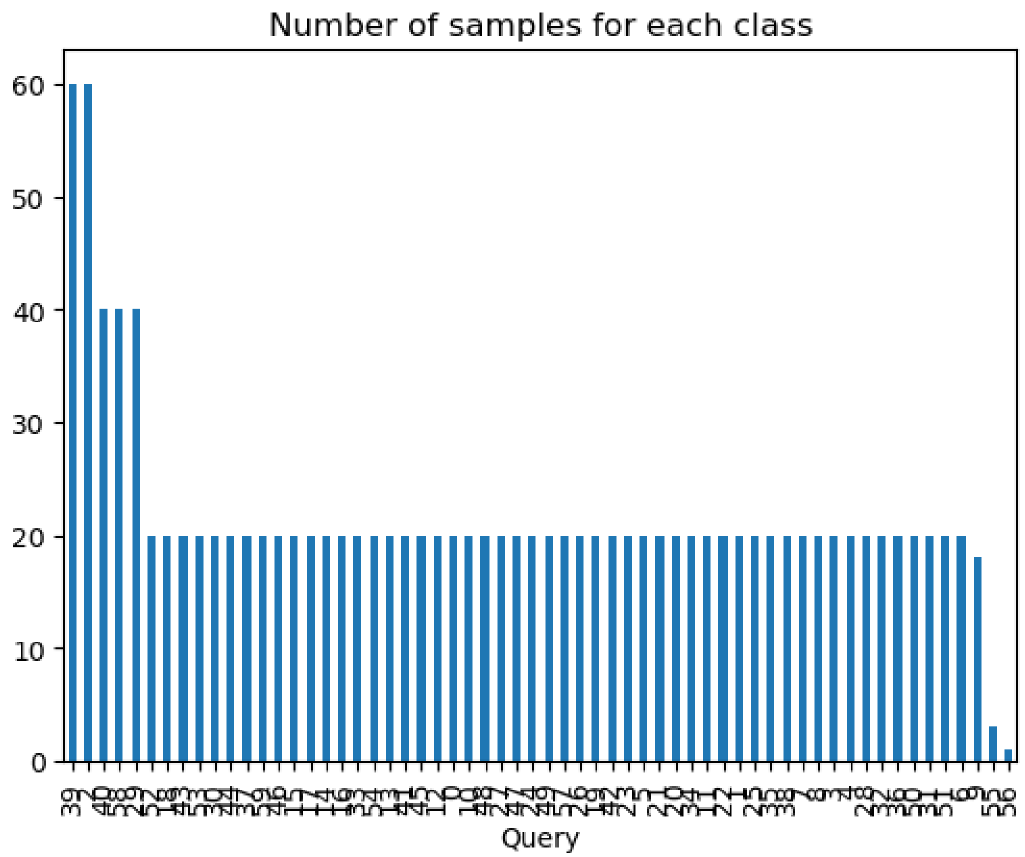
[140]:

```
# TF-IDF => high score = keywords / important descriptors
tfidf_title = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2',␣
 ↪encoding='latin-1', ngram_range=(1, 2), stop_words='english')
tfidf_desc = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2',␣
 ↪encoding='latin-1', ngram_range=(1, 2), stop_words='english')
labels = data.Query

features_title = tfidf_title.fit_transform(data.Title).toarray()
features_description = tfidf_desc.fit_transform(data.Description).toarray()
print('Title Features Shape: ' + str(features_title.shape))
print('Description Features Shape: ' + str(features_description.shape))
```

```
Title Features Shape: (1302, 464)
Description Features Shape: (1302, 6099)
```

[142]:

```
####STEP 3.1: UNIGRAM+BIGRAM ANALYSIS####
# Plotting class distribution
data['Query'].value_counts().sort_values(ascending=False).plot(kind='bar',␣
 ↪y='Number of Samples',title='Number of samples for each class')
plt.show()
```



[144]:
```
# Get 10 best keywords for each keyword, Title features (we only print the␣
↪first 2)
N = 10
count1 = 1
count2 = 1



MAX_PRINT = 2
for current_class in list(le.classes_):
    if count1 > MAX_PRINT:
        break
    current_class_id = le.transform([current_class])[0]
    features_chi2 = chi2(features_title, labels == current_class_id)
    indices = np.argsort(features_chi2[0])
    feature_names = np.array(tfidf_title.get_feature_names_out())[indices]
    unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
    bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
    print("# '{}':".format(current_class))
    print("Most correlated unigrams:")
    print('-' *30)
    print('. {}'.format('\n. '.join(unigrams[-N:])))
    print("Most correlated bigrams:")
    print('-' *30)
    print('. {}'.format('\n. '.join(bigrams[-N:])))
    print("\n")
    count1 += 1

print("**************************************************************************************
```

```
# '1000 calorie diet':
Most correlated unigrams:
------------------------------
. loss
. plan
. result
. days
. look
. update
. day
. calories
. doctor
. calorie
Most correlated bigrams:
------------------------------
. day fat
. lose weight
. weight loss
. fat loss
. eat calories
. diet plan
. loss diet
. look like
. calories day
. calorie day


# 'anarecovery anorexia recovery':
Most correlated unigrams:
------------------------------
. allin
. unglamorizing
. years
. veronica
. wright
. anorexiarecovery
. tiktoks
. recovery
. anorexia
. react
Most correlated bigrams:
------------------------------
. recreate video
. recovery anorexiarecovery
. allin anorexia
. unglamorizing eat
. tw eat
. story anorexia
. veronica wright
. recovery veronica
. day anorexia
. anorexia recovery


********************************************************************************
**************************************************
```

[146]:

```python
# Get 10 best keywords for each keyword, Desc features (we only print the first 2)
N = 10
for current_class in list(le.classes_):
    if count2 > MAX_PRINT:
        break
    current_class_id = le.transform([current_class])[0]
    features_chi2 = chi2(features_description, labels == current_class_id)
    indices = np.argsort(features_chi2[0])
    feature_names = np.array(tfidf_desc.get_feature_names_out())[indices]
    unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
    bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
    print("# '{}':".format(current_class))
    print("Most correlated unigrams:")
    print('-' *30)
    print('. {}'.format('\n. '.join(unigrams[-N:])))
    print("Most correlated bigrams:")
    print('-' *30)
    print('. {}'.format('\n. '.join(bigrams[-N:])))
    print("\n")
    count2 += 1
```

```
# '1000 calorie diet':
Most correlated unigrams:
------------------------------
. babin
. retainer
. department
. tier
. scott
. sauce
. acft
. army
. calorie
. bland
Most correlated bigrams:
------------------------------
. certify obesity
. obesity family
. board certify
. access accurate
. increase access
. bland board
. dr scott
. facebook page
. calories day
. scott bland


# 'anarecovery anorexia recovery':
Most correlated unigrams:
------------------------------
. clearly
. grateful
. depop
. shoplittlerose
. commute
. heather
. roisinmitc
. didnt
. rorecovering
. grey
Most correlated bigrams:
------------------------------
. shoplittlerose tiktok
. anorexia recovery
. tiktok rorecovering
. depop shoplittlerose
. public library
. music public
. library commute
. roisinmitc depop
. instagram rorecovering
. instagram roisinmitc
```

[ ]:

```
#Organizing by keyword gives insignificant results, let's try computing bigrams␣
 ↪per side (pro or con)
```

[148]:

```
####STEP 2.2: USE TF IDF FOR SIDES####
#encode classes by key word (tried by pro and con but too vague)
le = LabelEncoder()
le.fit(data.Pro_or_Con)
data.Pro_or_Con = le.transform(data.Pro_or_Con)
```

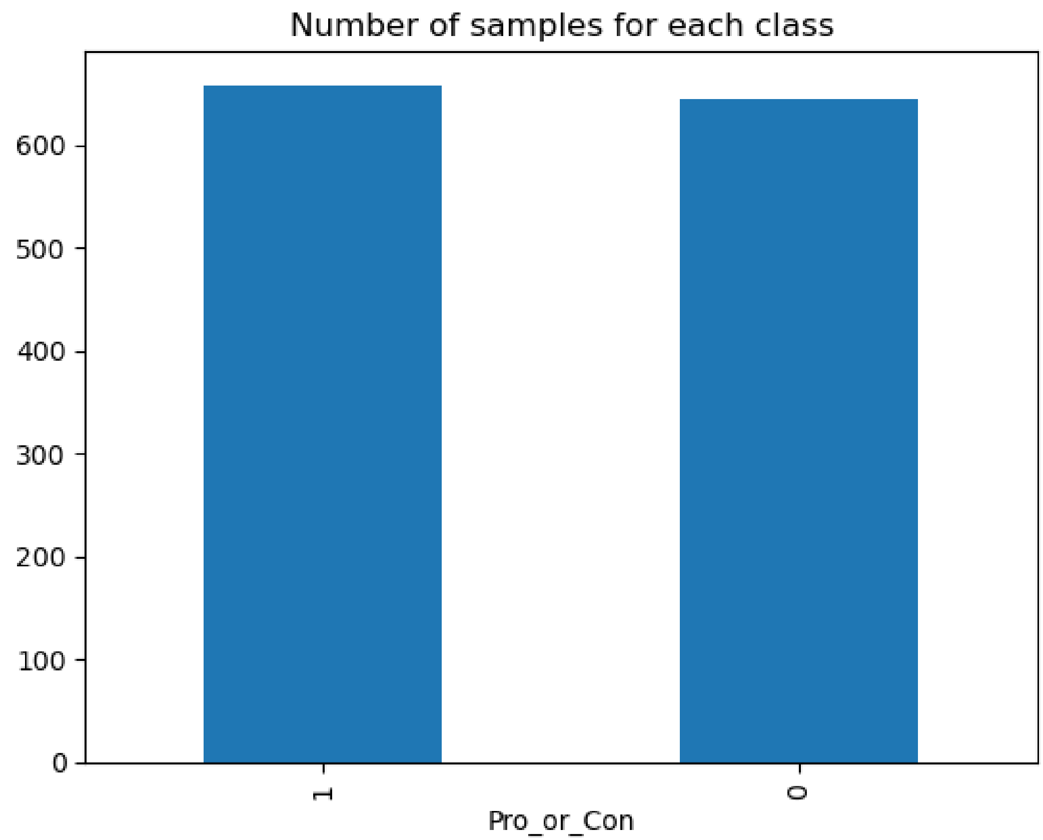
[150]:

```
tfidf_title = TfidfVectorizer(sublinear_tf=True, min_df=8, norm='l2',␣
 ↪encoding='latin-1', ngram_range=(1, 2), stop_words='english')
tfidf_desc = TfidfVectorizer(sublinear_tf=True, min_df=8, norm='l2',␣
 ↪encoding='latin-1', ngram_range=(1, 2), stop_words='english')
labels = data.Pro_or_Con

features_title = tfidf_title.fit_transform(data.Title).toarray()
features_description = tfidf_desc.fit_transform(data.Description).toarray()
print('Title Features Shape: ' + str(features_title.shape))
print('Description Features Shape: ' + str(features_description.shape))
```

```
Title Features Shape: (1302, 249)
Description Features Shape: (1302, 3036)
```

[152]:

```
####STEP 3.2: UNIGRAM+BIGRAM ANALYSIS####
# Plotting class distribution
data['Pro_or_Con'].value_counts().sort_values(ascending=False).plot(kind='bar',␣
 ↪y='Number of Samples',title='Number of samples for each class')
plt.show()
#slightly more samples for pro_ed content
```



[154]:

```
# Get 10 best keywords for each keyword, Title features (we only print the first 2)
N = 20
count1 = 1
count2 = 1


MAX_PRINT = 2
for current_class in list(le.classes_):
    if count1 > MAX_PRINT:
        break
    current_class_id = le.transform([current_class])[0]
    features_chi2 = chi2(features_title, labels == current_class_id)
    indices = np.argsort(features_chi2[0])
    feature_names = np.array(tfidf_title.get_feature_names_out())[indices]
    unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
    bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
    print("# '{}':".format(current_class))
    print("Most correlated unigrams:")
    print('-' *30)
    print('. {}'.format('\n. '.join(unigrams[-N:])))
    print("Most correlated bigrams:")
    print('-' *30)
    print('. {}'.format('\n. '.join(bigrams[-N:])))
    print("\n")
    count1 += 1

print("*************************************************************************************
```

```
# '0':
Most correlated unigrams:
------------------------------
. size
. recover
. gap
. days
. antidiet
. story
. calories
. workout
. culture
. fear
. thigh
. eat
. subliminal
. anorexia
. slim
. tw
. skinny
. fat
. recovery
. disorder
Most correlated bigrams:
------------------------------
. self love
. pro ana
. awareness week
. lose weight
. food freedom
. eat recovery
. body image
. disorder awareness
. national eat
. ed recovery
. fat loss
. body positivity
. diet culture
. thigh gap
. fear foods
. tw ed
. binge eat
. disorder recovery
. anorexia recovery
. eat disorder


# '1':
Most correlated unigrams:
------------------------------
. size
. recover
. gap
. days
. antidiet
. story
. calories
. workout
. culture
. fear
. thigh
. eat
. subliminal
. anorexia
. slim
. tw
. skinny
. fat
. recovery
. disorder
Most correlated bigrams:
------------------------------
. self love
. pro ana
. awareness week
. lose weight
. food freedom
. eat recovery
. body image
. disorder awareness
. national eat
. ed recovery
. fat loss
. body positivity
. diet culture
. thigh gap
. fear foods
. tw ed
. binge eat
. disorder recovery
. anorexia recovery
. eat disorder


*********************************************************************************
**************************************************
```

[ ]:

```
"""
Indicates there complete / significant overlap in the words used for both␣
 ↪sides, this is because
1) harmful content is hidden under 'tw' and other positive and recovery␣
 ↪sounding terms, to avoid being banned
2) bigrams might not be able to capture the differences in how terms are used␣
 ↪in one context or another
"""
```

[156]:

```
# Importing ML models for training
from sklearn.naive_bayes import MultinomialNB
from sklearn import linear_model
from sklearn.ensemble import AdaBoostClassifier
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt

from keras.layers import Dropout
from keras.regularizers import l1_l2
```

[158]:

```
####STEP 4: MODEL TRAINING####
X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, 2:4],␣
 ↪data['Pro_or_Con'], random_state = 0)
X_train_title_features = tfidf_title.transform(X_train['Title']).toarray()
X_train_desc_features = tfidf_desc.transform(X_train['Description']).toarray()
features = np.concatenate([X_train_title_features, X_train_desc_features],␣
 ↪axis=1)
```

[160]:

```
#3 models
#Naive Bayes
nb = MultinomialNB().fit(features, y_train)
#SVM
```

```
svm = linear_model.SGDClassifier(loss='modified_huber',max_iter=1000, tol=1e-3).
 ↪fit(features,y_train)
#AdaBoost
adaboost = AdaBoostClassifier(n_estimators=40,algorithm="SAMME").
 ↪fit(features,y_train)
```

[162]:

```
#Pre processing data for training
# Most frequently will not go over 20000
MAX_NB_WORDS = 20000
MAX_SEQUENCE_LENGTH = 50
# fixed
EMBEDDING_DIM = 100

# combine titles and descriptions into 1 sentence
titles = data['Title'].values
descriptions = data['Description'].values
data_for_lstms = []
for i in range(len(titles)):
    temp_list = [titles[i], descriptions[i]]
    data_for_lstms.append(' '.join(temp_list))

tokenizer = Tokenizer(num_words=MAX_NB_WORDS, filters='!"#$%&()*+,-./:;<=>?
 ↪@[\]^_`{|}~', lower=True)
tokenizer.fit_on_texts(data_for_lstms)
word_index = tokenizer.word_index

#number of rows in our data = num of videos total
print('Found %s unique tokens.' % len(word_index))
```

```
Found 13435 unique tokens.
```

[164]:

```
# convert the data to padded sequences
X = tokenizer.texts_to_sequences(data_for_lstms)
X = pad_sequences(X, maxlen=MAX_SEQUENCE_LENGTH)


# One-hot Encode labels
Y = pd.get_dummies(data['Pro_or_Con']).values


# Splitting into training and test set, choose 20-80%
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, random_state = 42)

print('Shape of data tensor:', X.shape) #input length is 50
print('Shape of label tensor:', Y.shape) #classification model hence normal to␣
 ↪get 2
```

```
Shape of data tensor: (1302, 50)
Shape of label tensor: (1302, 2)
```

[273]:

```
#LSTM Model first
# Convert labels to binary format if not already
Y_train = data['Pro_or_Con'].values
Y_test = data['Pro_or_Con'].values

# Modify the output layer for binary classification
model = Sequential()
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=X.shape[1]))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2)) #one layer, tried␣
 ↪adding up to 3, did worse so model becomes to complex for dataset

model.add(Dense(1, activation='sigmoid')) #since we are doing binary␣
 ↪classification

# Y_train is one-dimensional w binary labels
is_binary_labels = len(Y_train.shape) == 1

# Compile the model
model.compile(loss='binary_crossentropy' if is_binary_labels else␣
 ↪'categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_13"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_13 (Embedding)    (None, 50, 100)           2000000

 spatial_dropout1d_13 (Spat  (None, 50, 100)           0
 ialDropout1D)

 lstm_24 (LSTM)              (None, 100)               80400

 dense_11 (Dense)            (None, 1)                 101

=================================================================
Total params: 2080501 (7.94 MB)
Trainable params: 2080501 (7.94 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

[ ]:

```
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2)) # 20-80%
model.add(Dense(49, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
```

[281]:

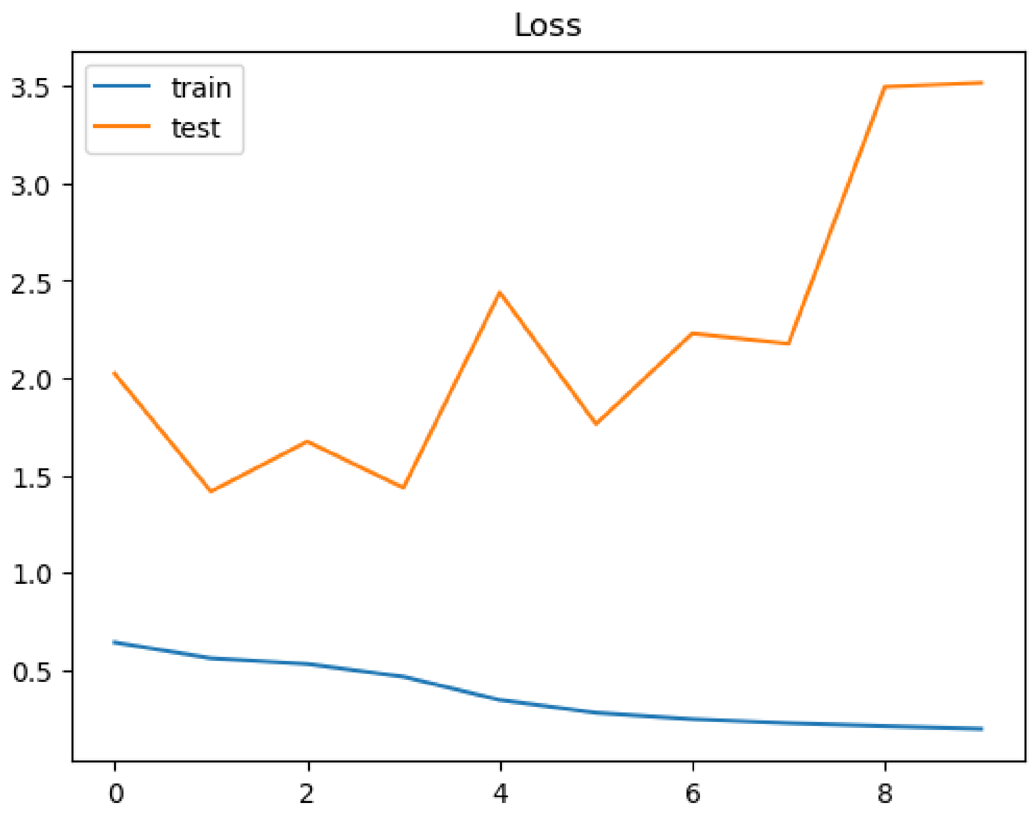
```
# Training the LSTM Model
epochs = 20
batch_size = 64
history = model.fit(X_train, Y_train, epochs=epochs, batch_size=batch_size,␣
 ↪validation_split=0.1)
```

```
Epoch 1/20
14/14 [==============================] - 2s 117ms/step - loss: 0.1833 -
accuracy: 0.9043 - val_loss: 2.6800 - val_accuracy: 0.1939
Epoch 2/20
14/14 [==============================] - 2s 115ms/step - loss: 0.1824 -
accuracy: 0.9089 - val_loss: 4.6362 - val_accuracy: 0.1224
Epoch 3/20
14/14 [==============================] - 2s 124ms/step - loss: 0.1739 -
accuracy: 0.9112 - val_loss: 2.8188 - val_accuracy: 0.1939
Epoch 4/20
14/14 [==============================] - 2s 125ms/step - loss: 0.1649 -
accuracy: 0.9066 - val_loss: 4.0615 - val_accuracy: 0.1837
Epoch 5/20
14/14 [==============================] - 2s 121ms/step - loss: 0.1599 -
accuracy: 0.9066 - val_loss: 3.5202 - val_accuracy: 0.1735
Epoch 6/20
14/14 [==============================] - 2s 120ms/step - loss: 0.1690 -
accuracy: 0.9032 - val_loss: 2.2947 - val_accuracy: 0.2143
Epoch 7/20
14/14 [==============================] - 2s 119ms/step - loss: 0.1858 -
accuracy: 0.9077 - val_loss: 4.8478 - val_accuracy: 0.1020
Epoch 8/20
14/14 [==============================] - 2s 112ms/step - loss: 0.1685 -
accuracy: 0.9100 - val_loss: 4.4184 - val_accuracy: 0.1837
Epoch 9/20
14/14 [==============================] - 2s 121ms/step - loss: 0.1636 -
accuracy: 0.9112 - val_loss: 3.7692 - val_accuracy: 0.2041
Epoch 10/20
14/14 [==============================] - 2s 115ms/step - loss: 0.1617 -
accuracy: 0.9077 - val_loss: 4.1073 - val_accuracy: 0.1735
Epoch 11/20
14/14 [==============================] - 2s 112ms/step - loss: 0.1538 -
accuracy: 0.9021 - val_loss: 3.7772 - val_accuracy: 0.2041
Epoch 12/20
14/14 [==============================] - 2s 114ms/step - loss: 0.1498 -
accuracy: 0.9021 - val_loss: 4.2782 - val_accuracy: 0.1939
```

```
Epoch 13/20
14/14 [==============================] - 2s 152ms/step - loss: 0.1531 -
accuracy: 0.9146 - val_loss: 3.9288 - val_accuracy: 0.2143
Epoch 14/20
14/14 [==============================] - 2s 111ms/step - loss: 0.1442 -
accuracy: 0.9123 - val_loss: 4.2566 - val_accuracy: 0.2143
Epoch 15/20
14/14 [==============================] - 2s 119ms/step - loss: 0.1415 -
accuracy: 0.9191 - val_loss: 4.7460 - val_accuracy: 0.1837
Epoch 16/20
14/14 [==============================] - 4s 321ms/step - loss: 0.1466 -
accuracy: 0.9021 - val_loss: 3.6217 - val_accuracy: 0.2347
Epoch 17/20
14/14 [==============================] - 3s 181ms/step - loss: 0.1547 -
accuracy: 0.9055 - val_loss: 4.9837 - val_accuracy: 0.1939
Epoch 18/20
14/14 [==============================] - 2s 125ms/step - loss: 0.1500 -
accuracy: 0.9100 - val_loss: 2.3122 - val_accuracy: 0.2347
Epoch 19/20
14/14 [==============================] - 2s 128ms/step - loss: 0.1607 -
accuracy: 0.9157 - val_loss: 4.3466 - val_accuracy: 0.1224
Epoch 20/20
14/14 [==============================] - 2s 126ms/step - loss: 0.1421 -
accuracy: 0.9203 - val_loss: 3.5665 - val_accuracy: 0.1633
```

[277]:

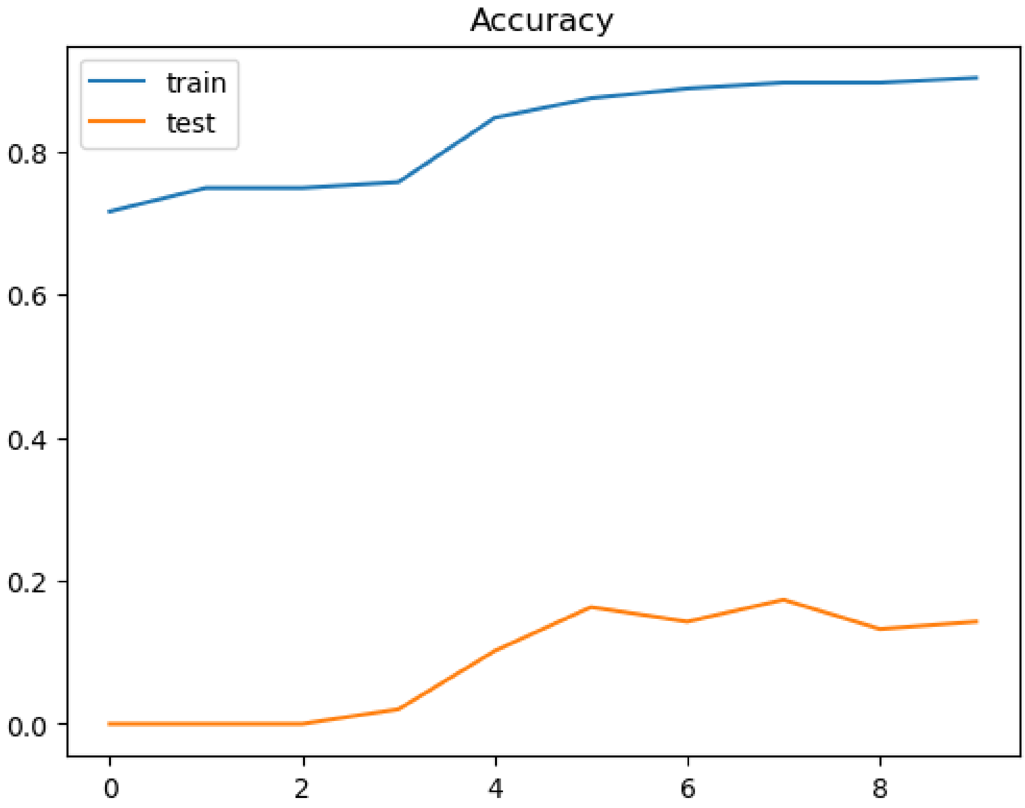
```python
## LOSS PLOT ##
plt.title('Loss')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show()
plt.savefig('loss_plot_lstm.png')
```



```
<Figure size 640x480 with 0 Axes>
```

[279]:

```
## ACCURACY PLOT ##

plt.title('Accuracy')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='test')
plt.legend()
plt.show()
```



[174]:
```
# Imports for Naive Bayes
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
import scikitplot as skplt
```

[176]:
```
X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, 2:4],␣
 ↪data['Pro_or_Con'], random_state = 0)
X_test_title_features = tfidf_title.transform(X_test['Title']).toarray()
X_test_desc_features = tfidf_desc.transform(X_test['Description']).toarray()
test_features = np.concatenate([X_test_title_features, X_test_desc_features],␣
 ↪axis=1)
```

[178]:
```
#Now we try Bayes
X_test_title_features = tfidf_title.transform(X_test['Title']).toarray()
X_test_desc_features = tfidf_desc.transform(X_test['Description']).toarray()
test_features = np.concatenate([X_test_title_features, X_test_desc_features],␣
 ↪axis=1)


# Naive Bayes
```

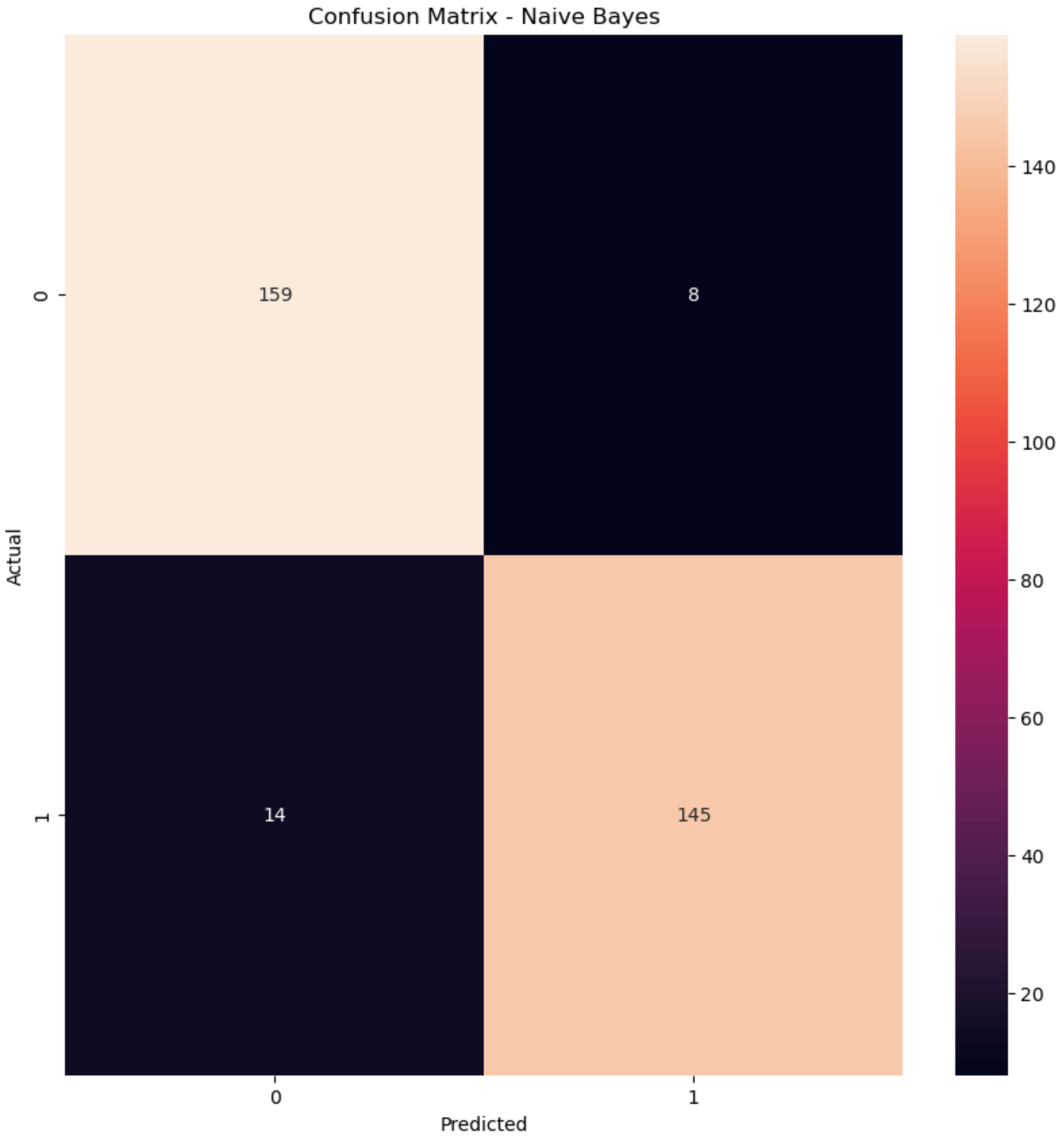
```
y_pred = nb.predict(test_features)
y_probas = nb.predict_proba(test_features)

class_labels = [str(cls) for cls in le.classes_]
print(metrics.classification_report(y_test, y_pred, target_names=class_labels))
```

```
              precision    recall  f1-score   support

           0       0.92      0.95      0.94       167
           1       0.95      0.91      0.93       159

    accuracy                           0.93       326
   macro avg       0.93      0.93      0.93       326
weighted avg       0.93      0.93      0.93       326
```

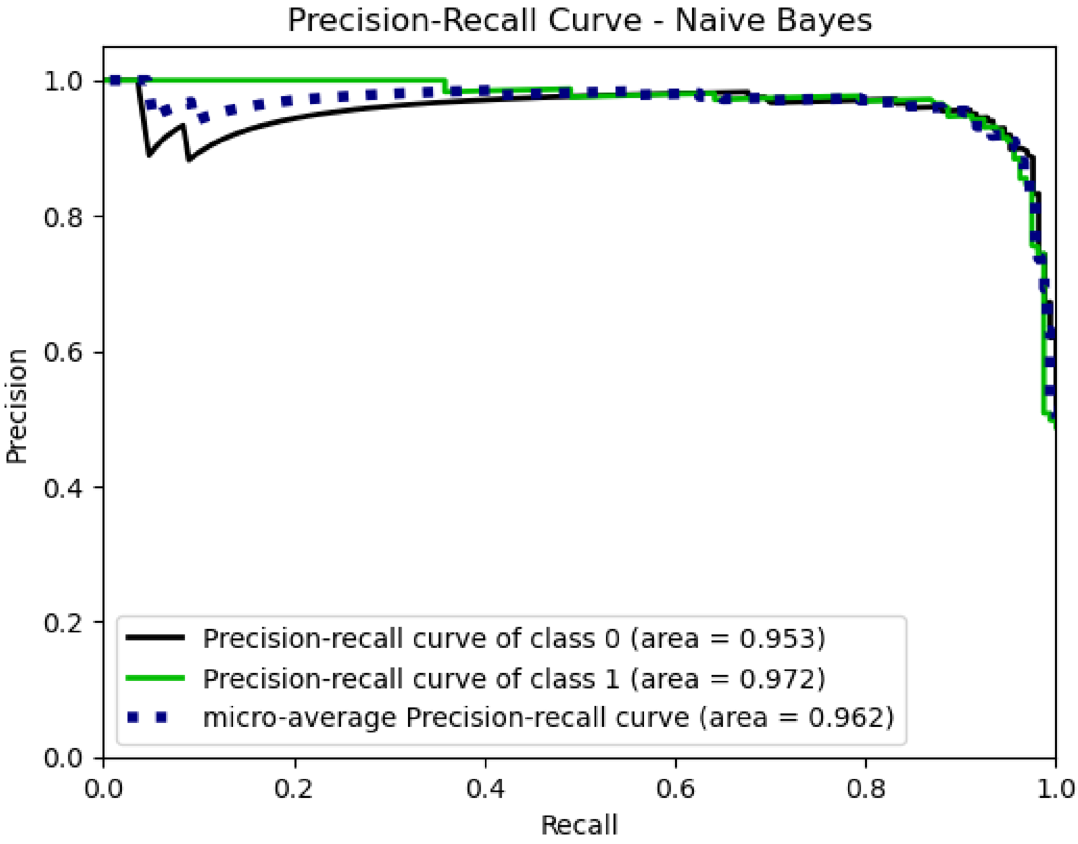
Strong performance with an overall accuracy of 0.93. Precision slightly higher for pro-ED data (0.95) compared to con-ED data (0.92), recall is slightly higher for class 0.

[189]:

```
conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(conf_mat, annot=True, fmt='d', xticklabels=list(le.classes_),␣
 ↪yticklabels=list(le.classes_))
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix - Naive Bayes')
plt.show()

skplt.metrics.plot_precision_recall_curve(y_test, y_probas)
plt.title('Precision-Recall Curve - Naive Bayes')
plt.show()
```



```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/utils/deprecation.py:86: FutureWarning: Function
plot_precision_recall_curve is deprecated; This will be removed in v0.5.0.
Please use scikitplot.metrics.plot_precision_recall instead.
  warnings.warn(msg, category=FutureWarning)
```



[195]:

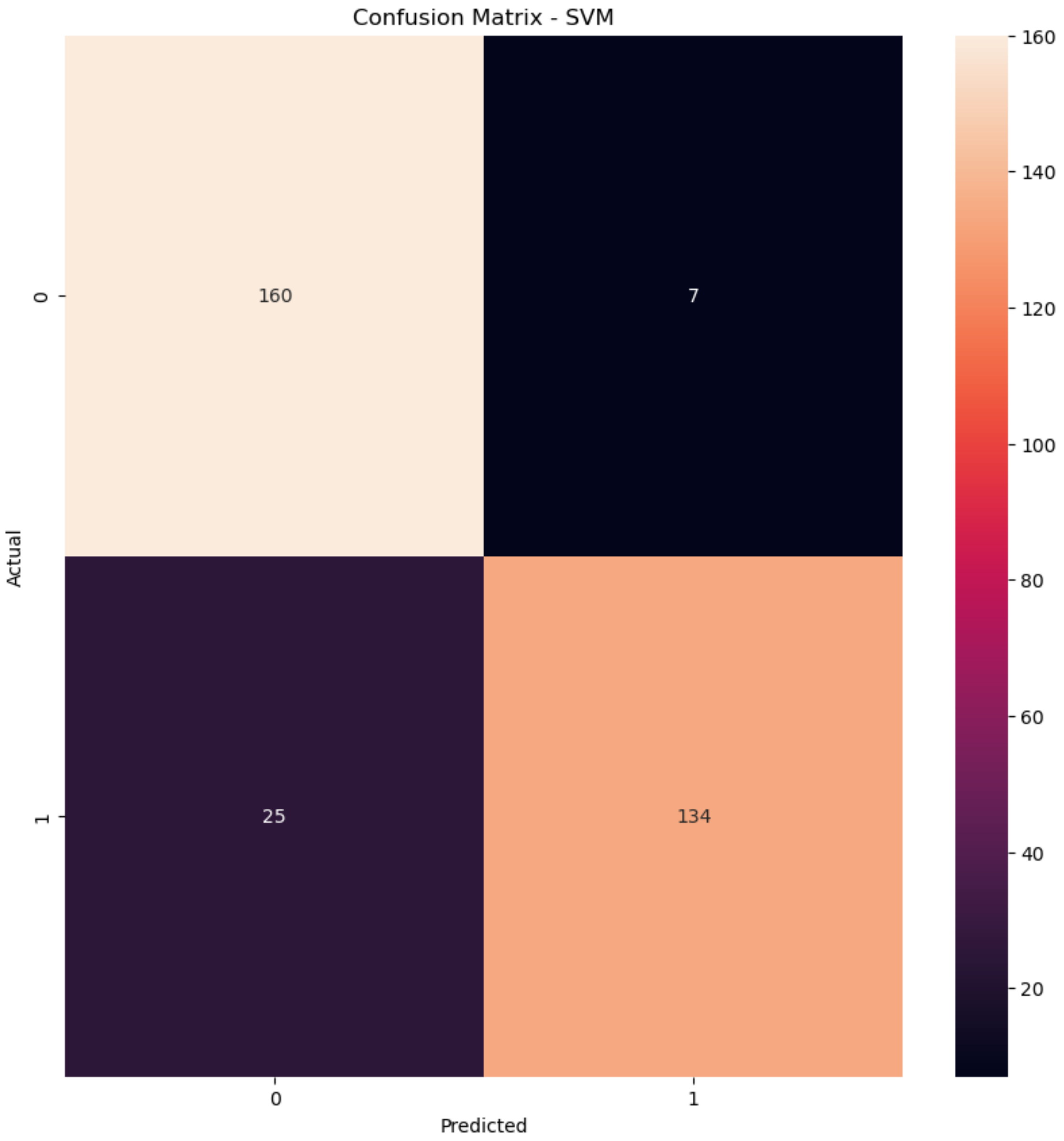
```
#Now SVM
y_pred = svm.predict(test_features)
y_probas = svm.predict_proba(test_features)

class_labels = [str(cls) for cls in le.classes_]
print(metrics.classification_report(y_test, y_pred, target_names=class_labels))
```

```
              precision    recall  f1-score   support

           0       0.86      0.96      0.91       167
           1       0.95      0.84      0.89       159

    accuracy                           0.90       326
   macro avg       0.91      0.90      0.90       326
weighted avg       0.91      0.90      0.90       326
```

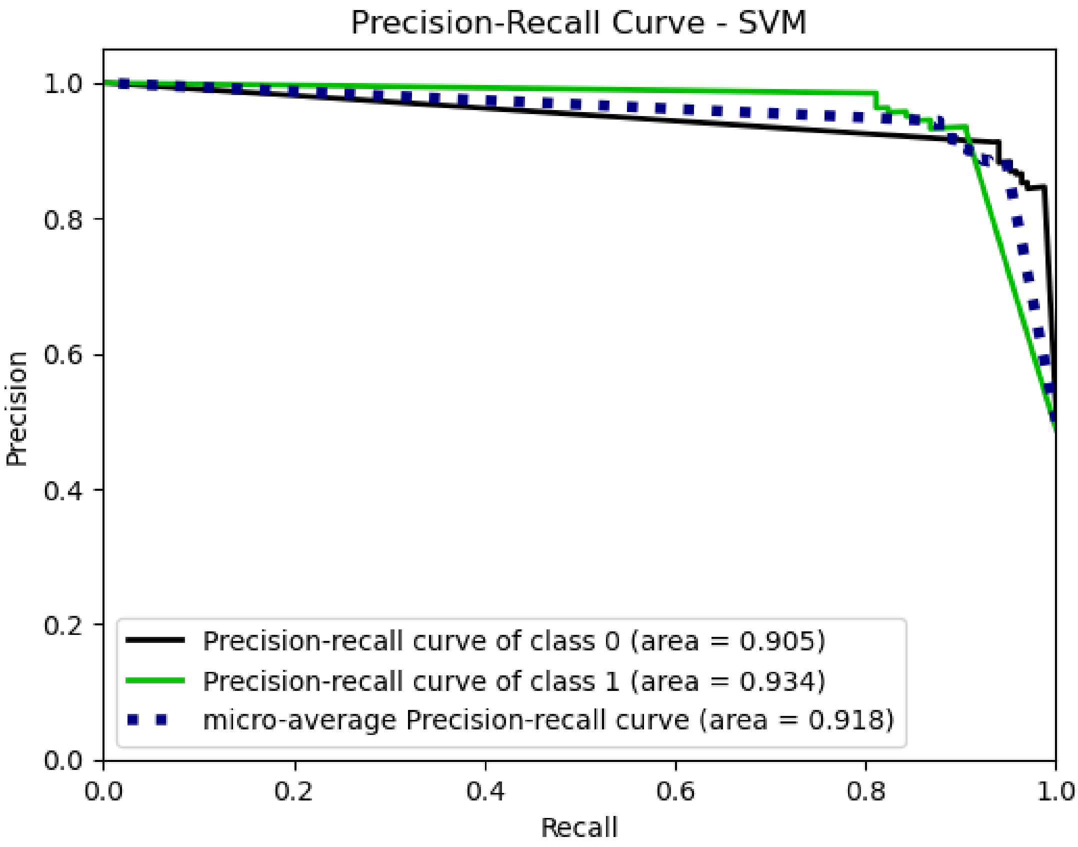
[197]:

```
conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(10,10))
```

```
sns.heatmap(conf_mat, annot=True, fmt='d', xticklabels=list(le.classes_),␣
 ↪yticklabels=list(le.classes_))
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix - SVM')
plt.show()

skplt.metrics.plot_precision_recall_curve(y_test, y_probas)
plt.title('Precision-Recall Curve - SVM')
plt.show()
```



```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/utils/deprecation.py:86: FutureWarning: Function
plot_precision_recall_curve is deprecated; This will be removed in v0.5.0.
Please use scikitplot.metrics.plot_precision_recall instead.
  warnings.warn(msg, category=FutureWarning)
```



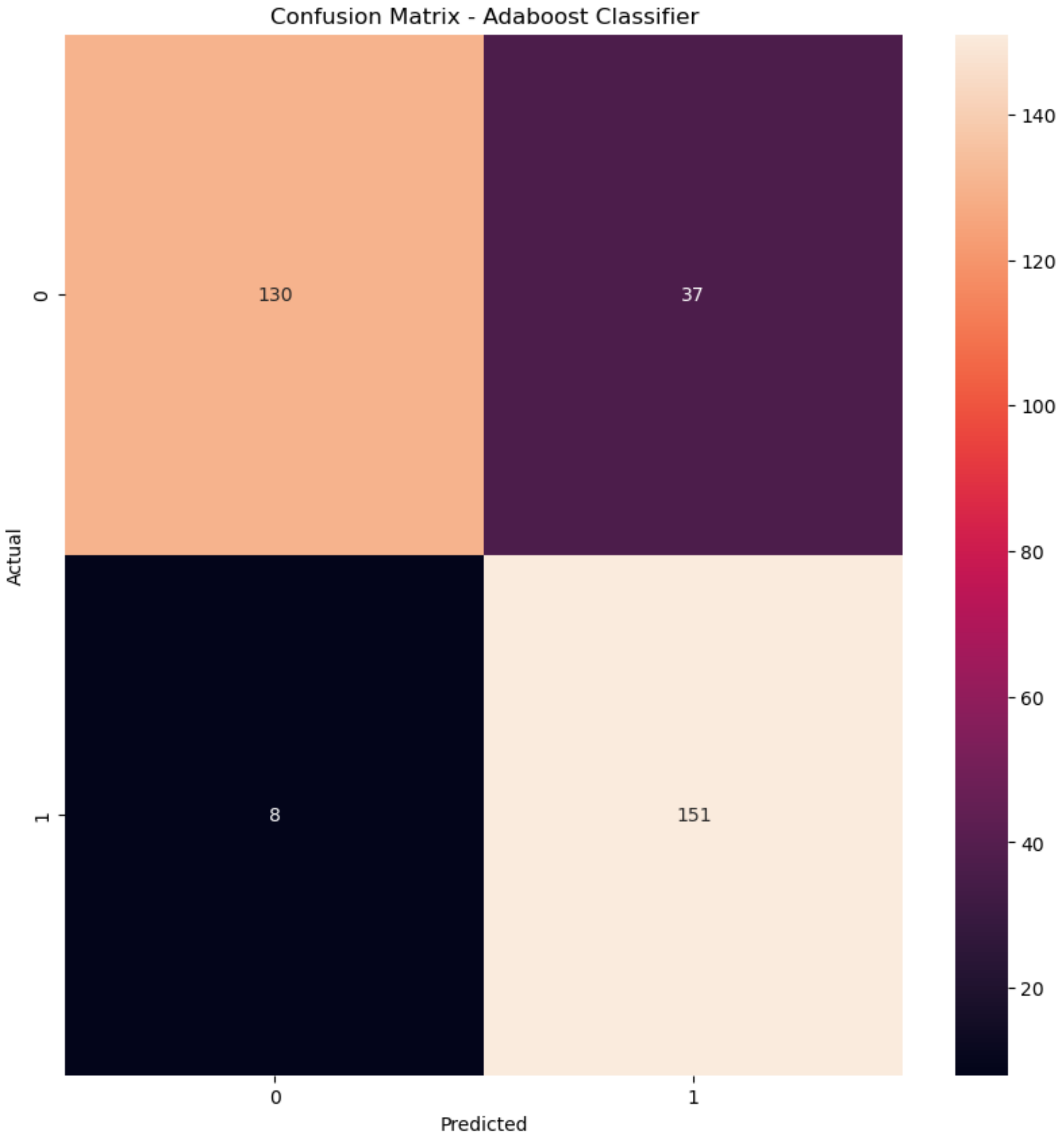
[199]:

```
# Finally Adaboost Classifier
y_pred = adaboost.predict(test_features)
y_probas = adaboost.predict_proba(test_features)

class_labels = [str(cls) for cls in le.classes_]
print(metrics.classification_report(y_test, y_pred, target_names=class_labels))
```

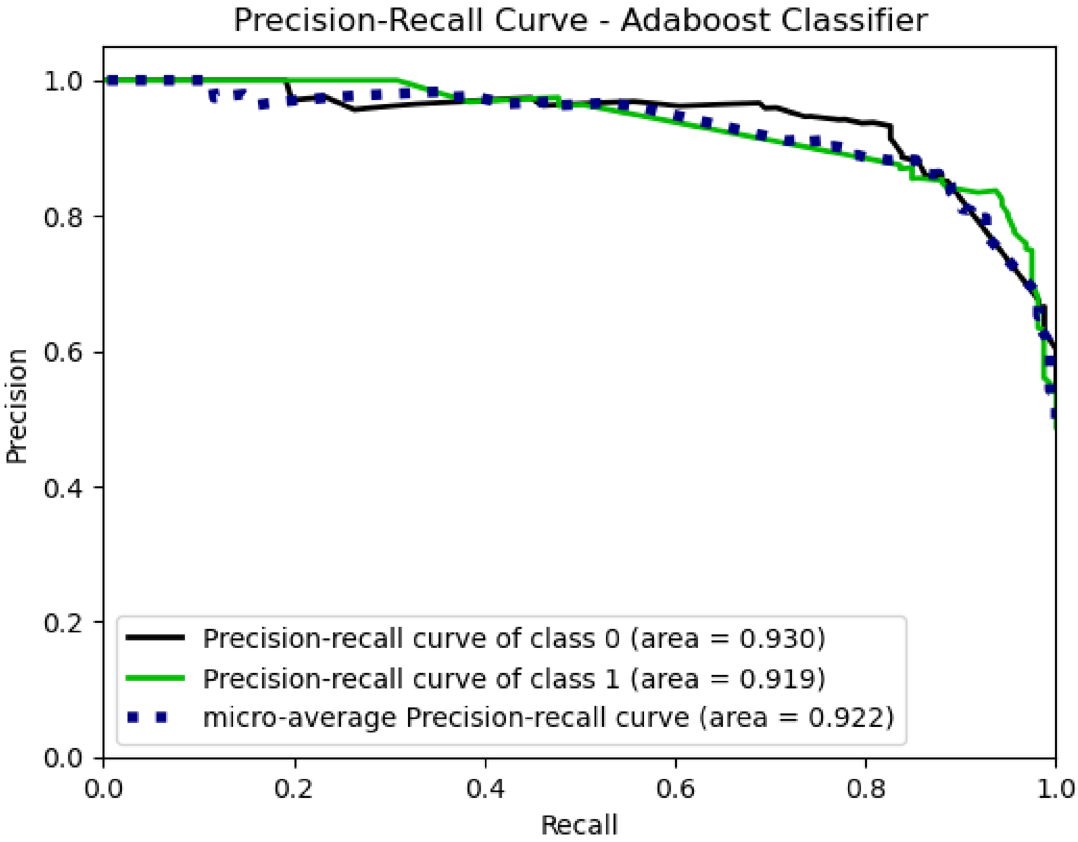
```
              precision    recall  f1-score   support

           0       0.94      0.78      0.85       167
           1       0.80      0.95      0.87       159

    accuracy                           0.86       326
   macro avg       0.87      0.86      0.86       326
weighted avg       0.87      0.86      0.86       326
```

[201]:

```
conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(conf_mat, annot=True, fmt='d', xticklabels=list(le.classes_),␣
 ↪yticklabels=list(le.classes_))
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix - Adaboost Classifier')
plt.show()

skplt.metrics.plot_precision_recall_curve(y_test, y_probas)
plt.title('Precision-Recall Curve - Adaboost Classifier')
plt.show()
```



```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/utils/deprecation.py:86: FutureWarning: Function
plot_precision_recall_curve is deprecated; This will be removed in v0.5.0.
Please use scikitplot.metrics.plot_precision_recall instead.
  warnings.warn(msg, category=FutureWarning)
```



[283]:

```python
# LSTM
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, random_state = 42)
y_probas = model.predict(X_test)
y_pred = np.argmax(y_probas, axis=1)
y_test = np.argmax(Y_test, axis=1)

class_labels = [str(cls) for cls in le.classes_]
print(metrics.classification_report(y_test, y_pred, target_names=class_labels))
```

```
11/11 [==============================] - 0s 12ms/step
              precision    recall  f1-score   support

           0       0.49      1.00      0.66       159
           1       0.00      0.00      0.00       167

    accuracy                           0.49       326
   macro avg       0.24      0.50      0.33       326
weighted avg       0.24      0.49      0.32       326


/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
```

```
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

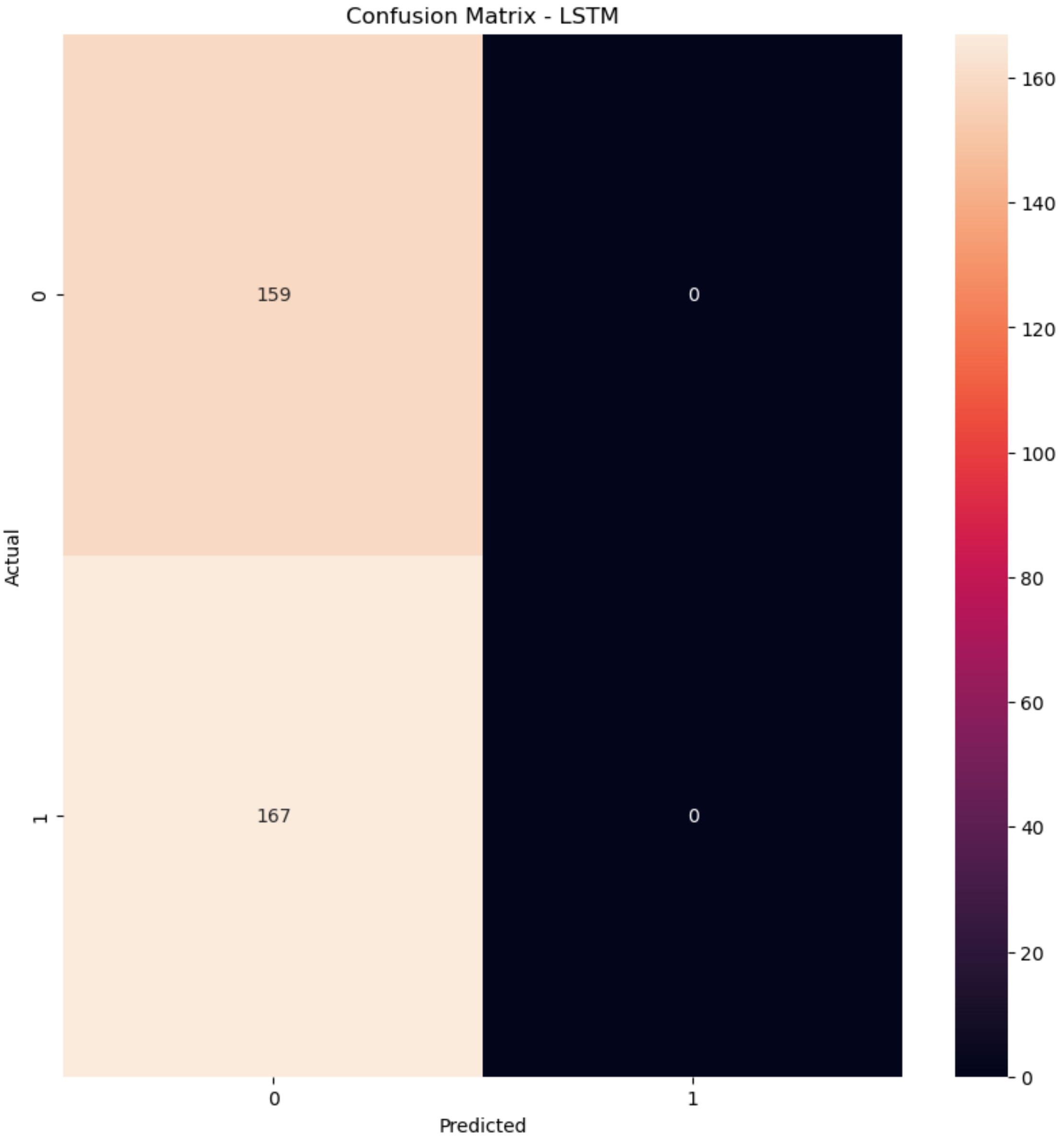
[285]:

```
model.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_13 (Embedding)    (None, 50, 100)           2000000

 spatial_dropout1d_13 (Spat  (None, 50, 100)           0
 ialDropout1D)

 lstm_24 (LSTM)              (None, 100)               80400

 dense_11 (Dense)            (None, 1)                 101

=================================================================
Total params: 2080501 (7.94 MB)
Trainable params: 2080501 (7.94 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[287]:

```
conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(conf_mat, annot=True, fmt='d', xticklabels=list(le.classes_),␣
 ↪yticklabels=list(le.classes_))
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix - LSTM')
plt.show()

skplt.metrics.plot_precision_recall_curve(y_test, y_probas)
plt.title('Precision-Recall Curve - LSTM')
plt.show()
```



```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/utils/deprecation.py:86: FutureWarning: Function
plot_precision_recall_curve is deprecated; This will be removed in v0.5.0.
Please use scikitplot.metrics.plot_precision_recall instead.
  warnings.warn(msg, category=FutureWarning)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[287], line 9
```

```
      6 plt.title('Confusion Matrix - LSTM')
      7 plt.show()
----> 9 skplt.metrics.plot_precision_recall_curve(y_test, y_probas)
     10 plt.title('Precision-Recall Curve - LSTM')
     11 plt.show()

File /opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/
 ↪sklearn/utils/deprecation.py:87, in deprecated._decorate_fun.<locals>.
 ↪wrapped(*args, **kwargs)
     84 @functools.wraps(fun)
     85 def wrapped(*args, **kwargs):
     86     warnings.warn(msg, category=FutureWarning)
---> 87     return fun(*args, **kwargs)

File ~/.local/lib/python3.11/site-packages/scikitplot/metrics.py:625, in␣
 ↪plot_precision_recall_curve(y_true, y_probas, title, curves, ax, figsize,␣
 ↪cmap, title_fontsize, text_fontsize)
    622 average_precision = dict()
    623 for i in range(len(classes)):
    624     precision[i], recall[i], _ = precision_recall_curve(
--> 625         y_true, probas[:, i], pos_label=classes[i])
    627 y_true = label_binarize(y_true, classes=classes)
    628 if len(classes) == 2:

IndexError: index 1 is out of bounds for axis 1 with size 1
```

```
[ ]:
```