Hate Speech Detection

מטרות :

מטרת הפרויקט היא לזהות בין ציוץ רע לציוץ טוב. (ציוץ = משפט\טקסט.)

על האלגוריתם יכולת להבחין במהירות וביעילות ולסווג בין ציוצים רעים לטובים.

-עלינו להשיג אחוז גבוה של דיוק. כלומר לאחר שנאמן מודל עלינו להגיע לאחוז דיוק גבוה ע"י test שנעשה.

-עלינו להשתמש בכמות דאטה עשירה ב train על מנת לעלות את אחוז הדיוק ולהתעלם מרעשים בדאטה.

-לבסוף המטרה הראשית היא לדעת להבדיל בין ציוץ רע לציוץ טוב. נרצה לדעת למגר ציוץ "רע" ולא לתת לו במה\שימוש ברחבי הרשת.

גישה:

גישה בסיסית לזיהוי דברי "רועה" היא שימוש בגישה מבוססת מילות מפתח. באמצעות אונטולוגיה או מילון, מזהים טקסט שמכיל מילות מפתח שעלולות לשנוא. (לדוגמה, קללה חמורה).

נרצה לתחזק כל הזמן מאגר של מילות מפתח אשר יעזרו לנו לזהות דברי שנאה. נשים לב שהטרמינולוגיה משתנה כל הזמן ונוספות לנו דברי "שנאה" מדי יום ולכן אחת הבעיות העיקריות בנושא זה היא שעלינו לתחזק כל הזמן את הדאטה לפי המתרחש ברחבי הגלובוס.

גישות מבוססות מילות מפתח הן מהירות ופשוטות להבנה. עם זאת, יש להם מגבלות חמורות. זיהוי השמצות גזעיות בלבד יביא למערכת מדויקת ביותר אך עם זכירה נמוכה כאשר הדיוק הוא אחוז הרלוונטיות מהקבוצה שזוהתה והזכירה היא אחוז הרלוונטיים מתוך האוכלוסייה העולמית. במילים אחרות, מערכת שמסתמכת בעיקר על מילות מפתח לא תזהה תוכן שנאה שאינו משתמש במונחים אלה. לעומת זאת, הכללת מונחים שיכולים אך אינם תמיד מעוררי שנאה (למשל, "זבל", "חזירים וכו') תיצור יותר מדי אזעקות שווא, יגביר את ההיזכרות על חשבון הדיוק.

מידע נוסף ממדיה חברתית יכול לעזור להבין יותר את המאפיינים של הפוסטים ואולי להוביל לגישת זיהוי טובה יותר. מידע כגון דמוגרפיה של המשתמש המפרסם, מיקום, חותמת זמן, או אפילו מעורבות חברתית בפלטפורמה יכולים כולם לתת הבנה נוספת של הפוסט בפירוט שונה.

עם זאת, מידע זה אינו זמין לעתים קרובות לחוקרים חיצוניים שכן פרסום נתונים עם מידע משתמש רגיש מעלה בעיות פרטיות. ייתכן שלחוקרים חיצוניים יש רק חלק או אפילו אף אחד ממידע המשתמש. לפיכך, הם אולי פותרים את החידה הלא נכונה או לומדים על סמך ידע שגוי מהנתונים. לדוגמה, מערכת מאומנת על נתונים אלה עשויה להטות באופן טבעי לסימון תוכן על ידי משתמשים או קבוצות מסוימות כדברי שטנה על סמך מאפיינים מקריים של מערך נתונים.

שימוש במידע משתמש עלול להעלות כמה בעיות אתיות. מודלים או מערכות עשויים להיות מוטים כלפי משתמשים מסוימים ולעיתים קרובות מסמנים את הפוסטים שלהם כמעוררי שנאה, גם אם חלק מהם לא. באופן דומה, הסתמכות רבה מדי על מידע דמוגרפי עלולה להחמיץ פוסטים ממשתמשים שאינם מפרסמים בדרך כלל תוכן שנאה. סימון פוסטים כשנאה על סמך נתונים סטטיסטיים של משתמשים עלול ליצור אפקט מצמרר על הפלטפורמה ובסופו של דבר להגביל את חופש הביטוי.

-תכנון עבודה :

התחלנו פרויקט זה בלמידת המודלים הרלוונטיים ולמידת החומר הנדרש. חשוב לציין כי לא היה לנו רקע בתחום זה לפני תחילת העבודה של הפרויקט .

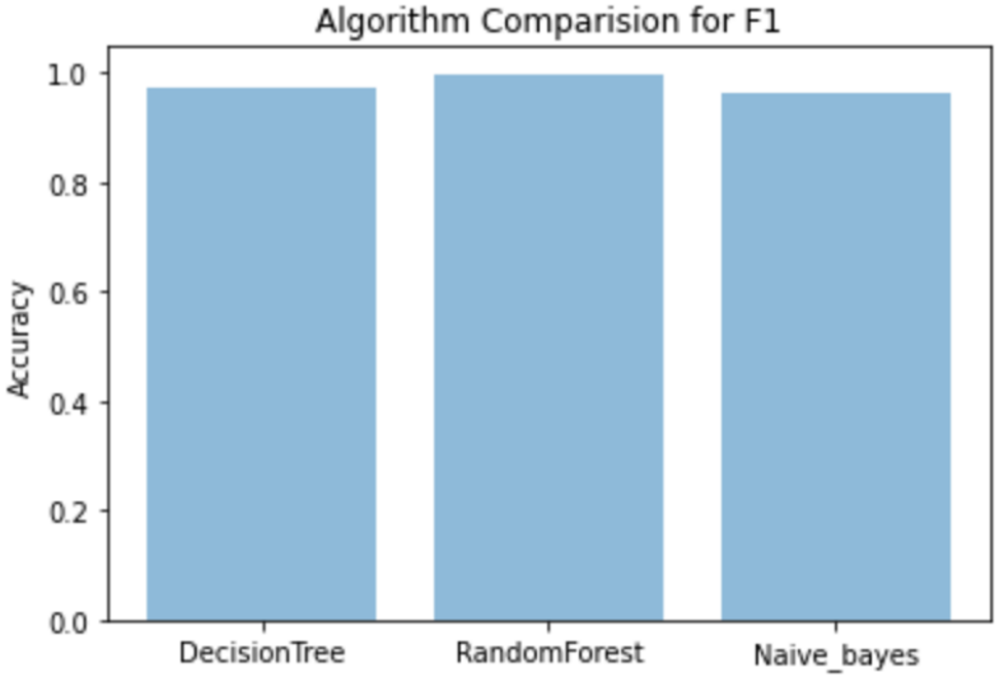
תכננו את מבנה שלד הפרויקט ובחירת מודלים מתאימים . כמו כן צירפנו את ה dataset שהשתמשו בו אשר נקלח מהמודל.

במבנה הפרויקט נעזרו בשלד של המעבדה במודל(שלבי העבודה לפי הצעתך).

מבנה : 1)קריאת הדאטה . 2)ניקוי הדאטה. 3) לעלות את מספר ה samples כך שנמנע מ overfitting . 4)יצירת המודל והתאמתו ל test ו train כולל cross validation. 5)הדפסת אחוז דיוק. 6)הדפסת גרפים רלוונטיים.

-שיפור המודל במהלך הפרויקט :

```
DecisionTreeClassifier: 0.9713069830421046
MultinomialNB: 0.9608490260790784
RandomForestClassifier: 0.9978141352586607
```



בתחילת הפרויקט חשבנו לעבוד לפי מודל naive Bayes כאשר עבודה זו הזכירה לנו את שיטת העבודה של spam detection .

לאחר הרבה מחשבה וניסיונות חוזרים הצלחנו להתאים את מודל naive Bayes לרמת דיוק של 96 אחוז.

רמת הדיוק עלתה ככל ששיפרנו יותר ויותר את ה data שלנו. כלומר ניקנו את הדאטה מרעשים.

לאחר קבלת תוצאות ביצענו cross-validation על מנת לשפר את בחירת המודל וקיבלנו את התוצאות מהגרף למעלה.

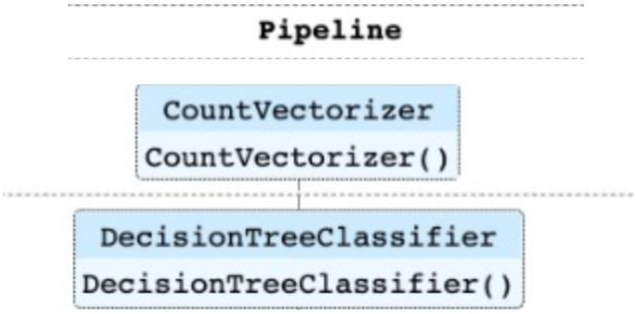
נבחין כי random forest מניב לנו תוצאות דיוק של 99 אחוז אך חשוב לזכור כי לפי הגישה שהצגנו למעלה הטרמינולוגיה תמיד משתנה ואסור לאחז ב דאטה אחידה (over-fitting) .

לכן בחירת המודל היא לא חד משמעית אבל עדיף לבחור מודל כמו naive Bayes אשר יש לו bias יותר גבוה אך ה variance שלו נמוך יותר.

ארכיטקטורה ובחירת מודל :

-pipeline :

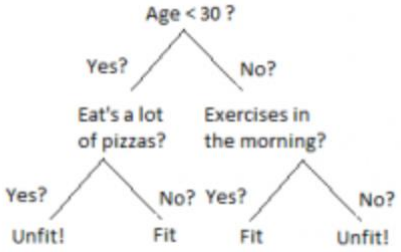
נעזרנו ב pipeline על מנת לשלב בין וקטור התדירויות למודל שלנו, ייצוג שכזה קריא יותר ונוח לשימוש



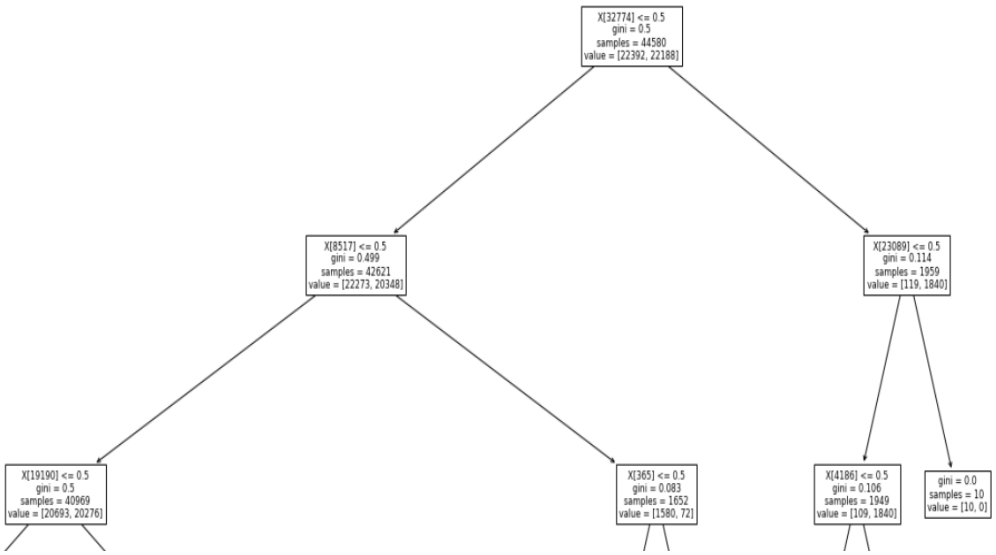
Decision Tree - :

עץ החלטה היודע גם כעץ סיווג הוא מודל חיזוי המספק מיפוי בין תצפיות לערכים המתאימים עבורן. עץ החלטה יכול לשמש כמודל חיזוי, הממפה תצפיות על פריט ויוצר מסקנות על ערך היעד של הפריט. במבנה של עצים אלה, עלים מייצגים סיווגים אפשריים וענפים מייצגים צירופים של תכונות אשר יובילו למחלקות הסיווג.

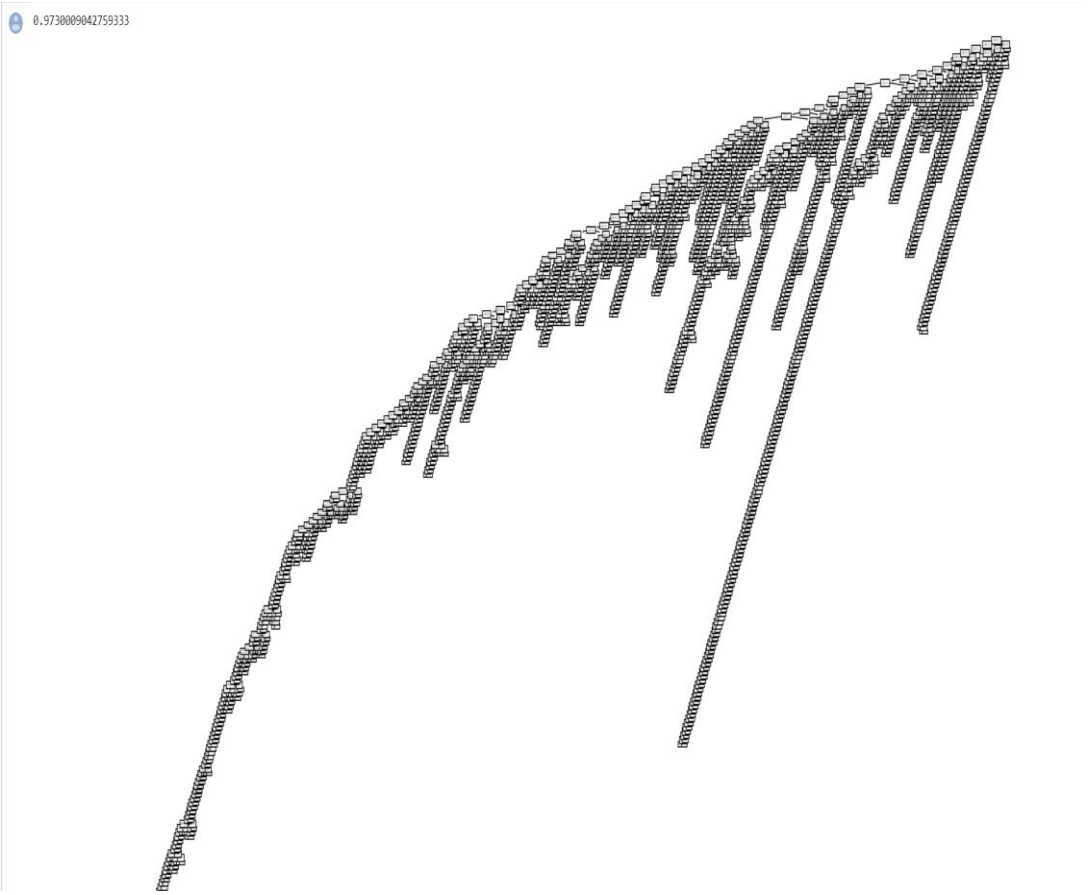
עץ סיווג בזום אין נראה כך : (דוגמא)



מבני העץ האמיתי בפרויקט שלנו לאחר התאמת הדאטה ל Hate Speech Detection: (במבט מקורב מאוד)



מבנה העץ לאחר התאמת המודל במבט על כולל אחוז דיוק :



הסבר על הקוד -הרחבה- Hate speech detection:

1)קריאת הדאטה train and test by using pandas libraries

2)ניקוי הדאטה מגורמי רעשים בתוך הדאטה (מנקים רעשים מ train & test)

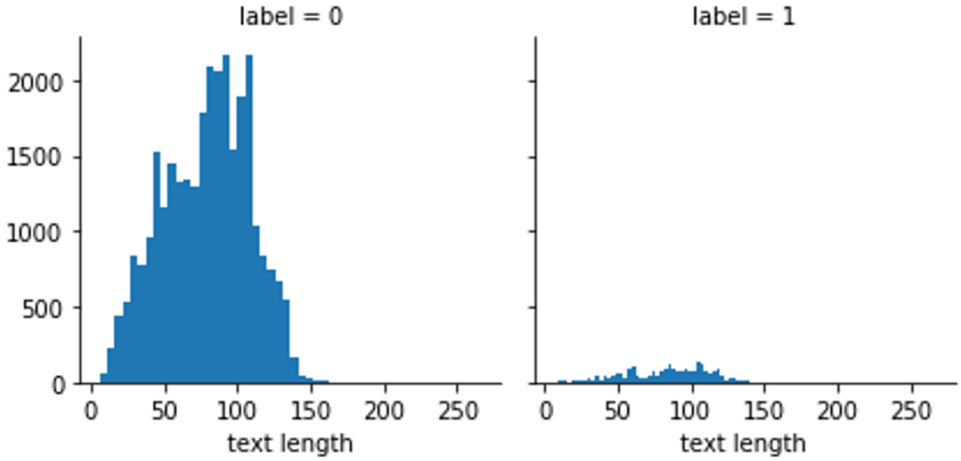
א) החלפת כל הדאטה לאותיות קטנות .

ב)ניקוי רווחים ,מספרים וסימנים.

ג)שימוש ב pandas dropna() אשר מנקה מהדאטה את כל ערכי ה nan\Null שלנו.

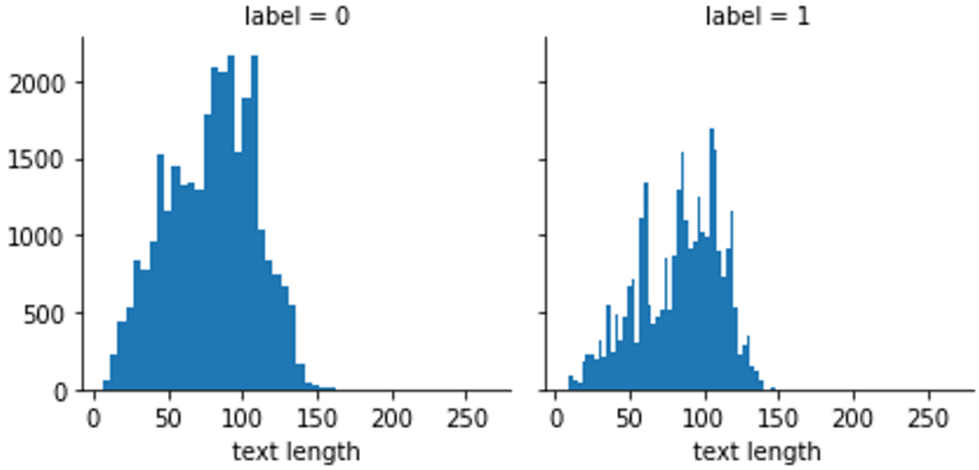
3) כעט נמשיך להכין ולסדר את הדאטה על מנת להתאים אותו כמה שיותר טוב למודל .
בפרויקט שלנו אנחנו צריכים לזהות בין דיבור "רע" לדיבור "טוב". נראה כי בדאטה שלנו יש המון משפטים שהם דיבור "טוב" – 29720 ציוצים שאכן מוגדרים כציוצים "טובים" ועוד 2242 ציוצים אשר מוגדרים כ "לא טובים".

התפלגות הדאטה הראשונית : ניתן לראות את התפלגות הדאטה שלנו לפי האורך שלה והכמות מכל label (label = 1 מייצג משפטים רעים)

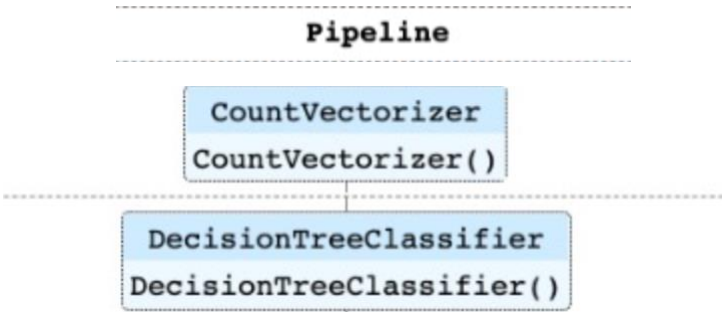


נשים לב שמצב זה יש לנו דגימת יתר של ציוצים "טובים" בעוד שיש לנו מיעוט רציני מהצד של המשפטים ה"רעים". נרצה לעלות את מספר הדגימות של הציוצים ה"רעים". נשתמש בפונקצית resample של sklearn על לעלות את כמות הדגימות ה"רעות" מ 2242 ל 29720 וכך להשוות בין הציוצים ה"טובים" ל"רעים" וכך נשפר משמעותית את למידת הדאטה שלנו ואת קבלת ההחלטות.

לבסוף אחרי ניקוי הדאטה ושיפור הדגימות משני ה קלאסים שלנו, נאחד את הדאטה שלנו ל data train מאוחד(59440 samples) נקי ומוכן להתאמת המודל:



4) כעט נשתמש ב pipeline של sklearn על מנת להשתמש בשני מודלים בצורה הבאה :



א)count vectorizer עוזר לנו להפוך את הציוצים ל "וקטור תדירויות" אשר סופר את התדירות של כל מילה בדאטה .

ב)שימוש בעץ החלטה כמודל שלנו.

5) כעט לפני שנאמן את המודל שלנו נרצה לפצל אותו ל train ו test ע"י שימוש ב sklearn .

6) נאמן את המודל אחרי היציאה מה pipeline :

א)fit the model

ב) ניתן למודל לעבוד על ה test שלנו. כלומר נעשה predict .

ג)נשתמש בפונקצית f1-score על מנת לשערך את רמת הדיוק של המודל שלנו. נשערך את ה test אל מול ה predict של ה train .

אחוז דיוק :

```
DecisionTreeClassifier: 0.9713069830421046
MultinomialNB: 0.9608490260790784
RandomForestClassifier: 0.9978141352586607
```

הסבר על המודל השני שלנו : ***Naïve Bayes*** :

שיטה המבוססת על חוק בייס ועל ההנחה ה"נאיבית" שאין תלות בין תכונות האובייקטים המסווגים כאשר כבר ידוע סיווגם. המשימה היא לסווג אובייקט לאחת מ k-קטגוריות כאשר האובייקט מאופיין על ידי וקטור התכונות .ידועות ההסתברויות האפריוריות(priors)של הקטגוריות לכל k ,וגם ידועות ההסתברויות המותנות(likelihood) של כל תכונה בנפרד בהינתן הקטגוריה. הסתברויות אלה ניתנות להערכה מתוך מדגם מייצג. בנוסף מניחים את ההנחה ה"נאיבית" שאם ידועה הקטגוריה אזי התכונות אינן תלויות זו בזו.

כדי למצוא את הקטגוריה הסבירה ביותר משווים את ההסתברויות המותנות ובוחרים את התשובה שמובילה להסתברות המותנית המקסימלית .

לפי חוק בייס :

$$p(C_k|x) = \frac{p(C_k)\, p(x|C_k)}{p(x)}$$

הרצת הקוד :

מקבלים קובץ Hate_Speech_Detection.py

הקוד רץ על כל IDE שתומך ב python . לפני הרצת הקוד עלינו לוודא שיש לנו את כל ספריות ה import הרלוונטיות לפרויקט : NumPy, matplotlib , re , pandas , sklearn .

עלינו לדאוג כי קבצי ה data שלנו נמצאים בתיקייה אחת יחד עם הקוד (כלומר באותו workspace).

מצורף גם קובץ py להרצה בסביבת פייטון וגם קובץ להרצת בסביבה google colab .

סקירת הקוד:

1)ייבוא ספריות שימושיות :

```
import pandas as pd
import re
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.metrics import f1_score
from wordcloud import WordCloud
from sklearn.utils import resample
import numpy as np
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import plot_tree
```

2)פונקציה לניקוי רעשים מדאטה :

```
# data cleaning by using the re library
def clean_text(data, tweet):
    temp = 0;
    data[tweet] = data[tweet].str.lower()
    data[tweet] = data[tweet].apply(lambda elem: re.sub(r"(@[A-Za-z0-9]+)|(\w+:\/\/\S+) | \n |  | \t", "", elem))
    return data
```

3)טעינת הדאטה + ניקוי רעשים :

```
[9]  if __name__ == "__main__":
         #reading the datasets
         train = pd.read_csv('train.csv')
         test = pd.read_csv('test.csv')
```

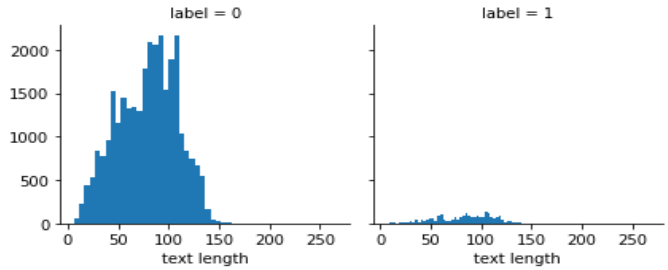
```
[10]     #clean noises from data by colum
         testClean = clean_text(test, "tweet")
         testData = test['tweet']
         trainClean = clean_text(train, "tweet")
         trainData = train['tweet']

         testClean.dropna()
         trainClean.dropna()
         testData.dropna()
```

```
0        #studiolife #aislife #requires #passion #dedic...
1          #white #supremacists want everyone to see th...
2        safe ways to heal your #acne!!#altwaystoheal #...
3        is the hp and the cursed child book up for res...
4        3rd #bihday to my amazing, hilarious #nephew e...
                                  ...
17192    thought factory: left-right polarisation! #tru...
17193    feeling like a mermaid ð□□□ #hairflip #neverre...
17194    #hillary #campaigned today in #ohio((omg)) &am...
17195    happy, at work conference: right mindset leads...
17196    my song "so glad" free download!#shoegaze #new...
Name: tweet, Length: 17197, dtype: object
```

4)הצגת מדגם מהדאטה לאחר ניקוי + גרפים המצגים את כמות המשפטים הטובות אל מול הרעים :

```
[11]
print(trainClean.head())
#graph of number of sentence and their size between the callses 0\1
trainClean['text length'] = trainClean['tweet'].apply(len)
graph = sns.FacetGrid(data=trainClean, col='label')
graph.map(plt.hist, 'text length', bins=50)

print(trainClean.head())
```

```
   id  label                                              tweet
0   1      0   when a father is dysfunctional and is so sel...
1   2      0   thanks for #lyft credit i can't use cause th...
2   3      0                                bihday your majesty
3   4      0  #model i love u take with u all the time in ur...
4   5      0                  factsguide: society now#motivation
   id  label                                              tweet  text length
0   1      0   when a father is dysfunctional and is so sel...           95
1   2      0   thanks for #lyft credit i can't use cause th...          108
2   3      0                                bihday your majesty           19
3   4      0  #model i love u take with u all the time in ur...           82
4   5      0                  factsguide: society now#motivation           35
```



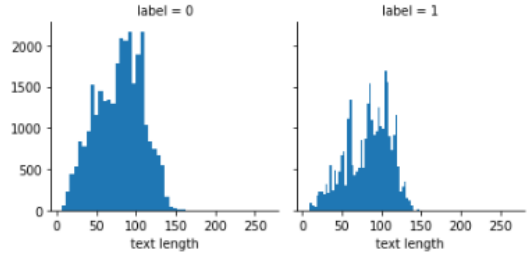
5)הצגת מילות המפתח של מילות "רעות" ו "טובות" בצורה ויזואלית נוחה לעין :

```
#plot normal words from the data
hatred_words = ' '.join([text for text in trainClean['tweet'][trainClean['label'] == 0]])
wordcloud = WordCloud(width=800, height=500,random_state=21, max_font_size=110).generate(hatred_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()

#plot hate words from the data
hatred_words = ' '.join([text for text in trainClean['tweet'][trainClean['label'] == 1]])
wordcloud = WordCloud(width=800, height=500,random_state=21, max_font_size=110).generate(hatred_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

6) סידור האי סדר בין כמות המשפטים הטובים אל מול הרעים + הצגת גרף לאחר תיקון אי הסדר (גרף המשך מהגרף הקודם)

```
#Fixing imbalance in data
trainNonHate = trainClean[trainClean.label==0]

print(trainNonHate.shape)
trainHate = trainClean[trainClean.label==1]
print(trainHate.shape)
trainHateResample = resample(trainHate, replace=True, n_samples=len(trainNonHate), random_state=111)
print(trainHateResample.shape)
trainReShape = pd.concat([trainHateResample, trainNonHate])
print(trainReShape.shape)
trainReShape['label'].value_counts()

trainReShape['text length'] = trainReShape['tweet'].apply(len)
graph = sns.FacetGrid(data=trainReShape, col='label')
graph.map(plt.hist, 'text length', bins=50)
```

```
(29720, 4)
(2242, 4)
(29720, 4)
(59440, 4)
<seaborn.axisgrid.FacetGrid at 0x7fea03f69610>
```



7) מודל decision tree + אחוז דיוק

```
[18]
# we will use the Scikit-Learn's pipeline with an DecisionTreeClassifier,
# before training our model
pipeline_nb = Pipeline([
    #used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.
    ('countvect', CountVectorizer()),
    ('dtc', DecisionTreeClassifier())
])

#split the data into a training set and a test set

X_train, X_test, y_train, y_test = train_test_split(trainReShape['tweet'],trainReShape['label'],random_state = 1)

# train the model and predict the results on the test set using the F1 score method
model = pipeline_nb.fit(X_train, y_train)
y_predict = model.predict(X_test)
print(f"DecisionTreeClassifier: {f1_score(y_test, y_predict)}")
acc = f1_score(y_test, y_predict)
```

```
DecisionTreeClassifier: 0.9720591082144932
```

מודל NB + אחוז דיוק

```
#MultinomialNB
pipeline_nb2 = Pipeline([
    ('countvect', CountVectorizer()),
    ('dtc', MultinomialNB())
])

X1, X2, y1, y2 = train_test_split(trainReShape['tweet'], trainReShape['label'], random_state=1)

model2 = pipeline_nb2.fit(X1, y1)
y_predict = model2.predict(X2)
print(f"MultinomialNB: {f1_score(y2, y_predict)}")
acc2 = f1_score(y2, y_predict)
```

```
MultinomialNB: 0.9608490260790784
```

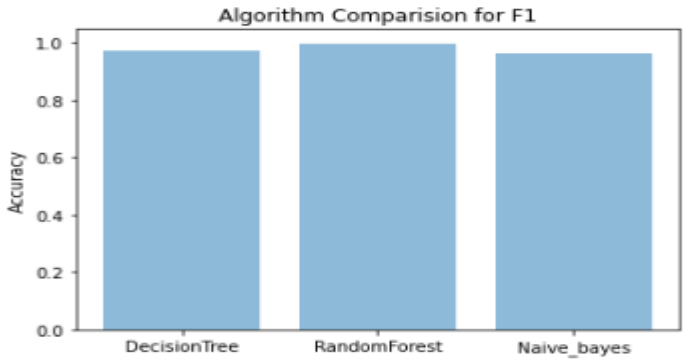
מודל random forest + אחוז דיוק :

```
[17]
#RandomForestClassifier
pipeline_nb3 = Pipeline([
    #used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.
    ('countvect', CountVectorizer()),
    ('dtc', RandomForestClassifier())
])

X3, X4, y3, y4 = train_test_split(trainReShape['tweet'], trainReShape['label'], random_state=1)

# train the model and predict the results on the test set using the F1 score method
model3 = pipeline_nb3.fit(X3, y3)
y_predict = model3.predict(X4)
print(f"RandomForestClassifier: {f1_score(y4, y_predict)}")
acc3 = f1_score(y4, y_predict)
```

```
RandomForestClassifier: 0.9976819656930923
```

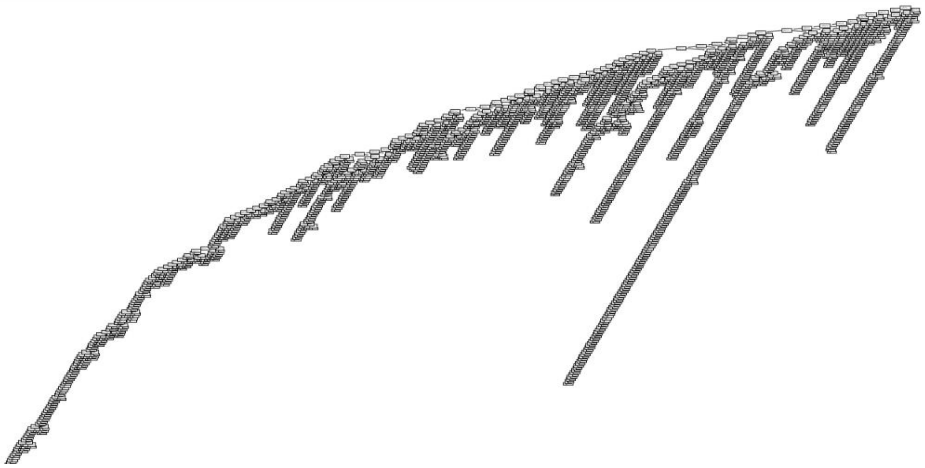
8)הצגת גרף אחוז דיוק בין המודלים השונים:

```
[19]
#plots accuracy of all models
objects = ('DecisionTree', 'RandomForest', 'Naive_bayes')
y_pos = np.arange(len(objects))
performance = [acc,acc3,acc2]
plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy')
plt.title('Algorithm Comparision for F1')
plt.show()
```



9) הדפסת העץ במלואו :

```
#plot the decision tree

fig = plt.figure(figsize=(25, 20))
plot_tree(pipeline_nb['dtc'])
plt.show()
```



סיכום ומסקנות :

ככל שציוציי "רועה" ממשיכים להוות בעיה חברתית, הצורך במערכות אוטומטיות לזיהוי דברי "רועה" הולך וגובר. הצגנו את הגישות הנוכחיות למשימה זו וכן גם מערכת חדשה המשיגה דיוק גבוה. כמו כן, הצענו גישה חדשה שיכולה לעלות על מערכות קיימות במשימה זו, עם היתרון הנוסף של יכולת פרשנות משופרת. לאור כל האתגרים שנותרו, יש צורך במחקר נוסף על בעיה זו, כולל עניינים טכניים ומעשיים כאחד.

לסיכום אנחנו ממליצים על מערכות שמסווג דברי שנאה .כך נוכל למגר הרבה מקרי זדון והשפלות הדדיות אך חשוב לציין כי עלינו לשים לב שאנו לא פוגעים בחופש הביטוי.