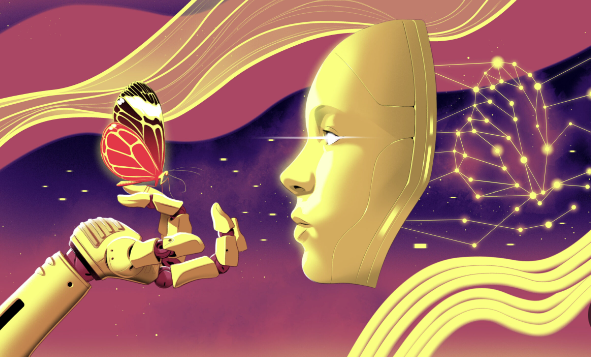
סיכום ודיווח ממצאים פרויקט אחזור מידע



תוכן

[2 הקדמה 2](#_Toc128321279)

[2.1 ספריות שהשתמשנו 2](#_Toc128321280)

[2.2 קצת על הקוד והדוח 3](#_Toc128321281)

[2.3 הפניות – References 3](#_Toc128321282)

[3 3](#_Toc128321283)

[3.1 ממצאים אחרי מחיקת ה- 5](#_Toc128321284)

[3.2 6](#_Toc128321285)

[3.3 - גזירה 6](#_Toc128321286)

[4 8](#_Toc128321287)

[4.1 10](#_Toc128321288)

[4.2 חלקות הנתונים לנתוני אימון ובדיקה לפי הדרישות 11](#_Toc128321289)

[4.3 הסיווג 12](#_Toc128321290)

[4.3.1 12](#_Toc128321291)

[4.4 16](#_Toc128321292)

[4.5 19](#_Toc128321293)

[4.6 KNN 22](#_Toc128321294)

[4.7 דיון לגבי התוצאות והשוואה בין מודלים וכמו כן הסבר על למה קיבלנו ציונים אלו במדדים של הערכה 25](#_Toc128321295)

[5 26](#_Toc128321296)

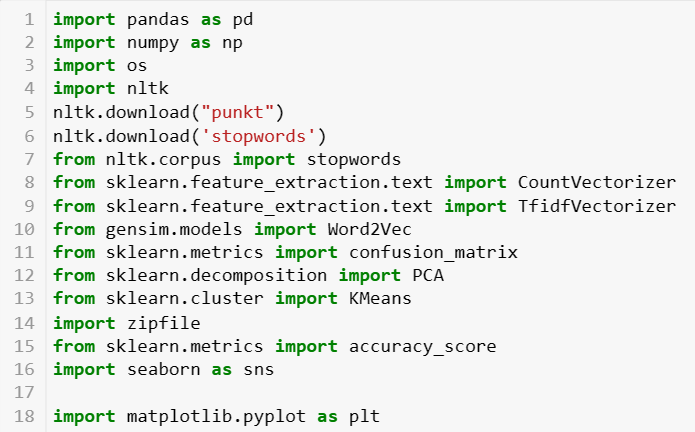
[5.1 התהליך 26](#_Toc128321297)

[5.1.1 ביצענו תהליך של הורדת ממדיות על ידי שימוש ב PCA לצורך קבלת תחושה על המסמכים: 28](#_Toc128321298)

[5.1.2 ביצוע KMEANS ו תוצאות שקיבלנו ו הצגת גרפים והסבר למה היה טעויות 29](#_Toc128321299)

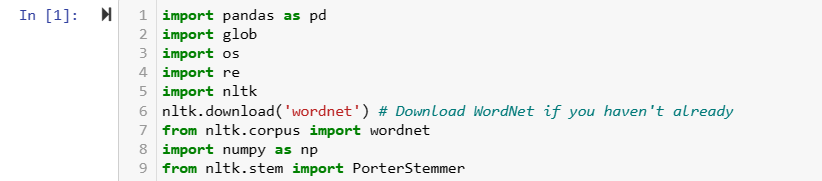
# הקדמה

## ספריות שהשתמשנו



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי



## קצת על הקוד והדוח

הדוח כולל הסבר מקיף על הקוד בכל השלבים שלו וכמו כן הוא מכיל דיון כך שיכסה את כלל הדרישות של המשימה. יישמנו את הדברים שלמדנו בקורס. אם התיעוד לא מובן בקוד הוא יהיה מובן פה.

## הפניות – References

* <https://scikit-learn.org/stable/index.html>
* <https://numpy.org/>
* <https://pandas.pydata.org/>
* <https://www.wikipedia.org/>
* <https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf>
* <https://mw12.haifa.ac.il/course/view.php?id=2867>
* <https://chat.openai.com/auth/login>

# 

התחלנו את המשימה על ידי קריאת כל המסמכים של הנושא שבחרנו, הקריאה נעשתה על ידי שימוש בספריה פשוטה של קריאת מסמכים בפייתון:

מה שקיבלנו זה טקסט אחד גדול שהוא מיזוג של כל הטקסטים שיש לנו:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

הטקסט היה מלא במילים שאין להם אף משמעות בשפה האנלית בכדי לטפל בזה ראשית אנחנו החלטנו לשמור רק מחרוזות שהם מורכבות אך ורק מאותיות באנגלית וזה נעשה על ידי שימוש ב regular expression:



ההחלטה הזו דיי מספקת את המטרה הסופית ולא פוגעת קשות במטרה הסופית. כי רוב המילים שיש להם משמעות בטקסטים הינם מילים שמורכבים אך ורק מאותיות, וגם במנועי חיפוש בכללי דיי נדיר שמישהו ירצה לחפש מספרים או כל משהו אחר למטרת צורך המידע שלו.

אחרי זה אנחנו שמנו לב שיש מלא צירופי אותיות שאין להם שום משמעות בשפה הטבעית שלנו, אז החלטנו להסיר צירופים אלו על ידי שימוש בספריית NLTK:



יש לשים לב להבדל בין האורך של רשימת ה- TOKENS שלנו לפני ואחרי פעולה זו:

לפני:

אחרי:

**בניית המילון על ידי הסתכלות על הטקסט שלנו כמודל שפה של UNIGRAM:**

עשינו זאת על ידי בניית פונקציה שמקבצת את כל המילים לרשימה של מפתחות ואת כל ה TF שלהם לרשימת הערכים על ידי הפונקציה הבאה:

סיכום המודל הבסיסי:

|  |  |
| --- | --- |
| Unique words | Word counts |
| 17558 | 316399 |

## ממצאים אחרי מחיקת ה-

פה נתמקד בהשפעה של הסרת ה- stop words שניתנה לנו על ידי צוות הקורס. אז הממצאים שמצאנו הינם:

סיכום המודל אחרי הסרת STOP WORDS:

|  |  |
| --- | --- |
| Unique words | Word count |
| 17025 | 226484 |

ניתן לראות שאחרי שהורדנו את המילות תפל אנחנו איבדנו קצת מידע אבל גודל ה VOCABULARY לא באמת השתנה לא הייתה ירידה דרסטית, הירידה הייתה פשוט מ- 17558  ל- 17025  אבל יש לשים לב שרשימת הפרסומים תרד משמעותית

## 

קיפול רישיות הוא תהליך המרת כל האותיות במחרוזת לאותיות נפוצות (בדרך כלל אותיות קטנות) על מנת להפוך את המחרוזת לקלה יותר להשוואה ולתפעול. בעיבוד טקסט ועיבוד שפה טבעית, קיפול רישיות הוא שלב נפוץ בנורמליזציה של טקסט, שהוא תהליך המרת נתוני טקסט לפורמט סטנדרטי שניתן להשתמש בו לניתוח נוסף.

אנחנו נעשה פעולה זו על ידי שורה אחת בפייתון:



|  |  |
| --- | --- |
| Unique words | Word counts |
| 12791 | 316399 |

ניתן לראות את כמות המונחים שנשארה לנו אחרי ביצוע תהליך זה וזה נובע מכך שהרבה מילים יאוחדו למילה אחת.

## - גזירה

במורפולוגיה הלשונית ובשליפת מידע, גזירה היא תהליך של הפחתת מילים מוטות (או לפעמים נגזרות) לצורת בסיס או שורש של המילה שלהן - בדרך כלל צורת מילה כתובה. הבסיס אינו חייב להיות זהה לשורש המורפולוגי של המילה; בדרך כלל מספיק שמילים קשורות ממפות לאותו בסיס, גם אם בסיס זה אינו שורש תקף בפני עצמו. אלגוריתמים ליישום STEMMING נחקרו במדעי המחשב מאז שנות ה-60. מנועי חיפוש רבים מתייחסים למילים עם אותו בסיס כמילים נרדפות כסוג של הרחבת שאילתה, תהליך הנקרא קונפלציה.

אנחנו השתמשנו ב אלגוריתם של פורטר:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

|  |  |
| --- | --- |
| Unique words | Word counts |
| 7619 | 316399 |

פה רואים ירידה משמעותית ב- unique words וזה קורה בגלל שהרבה מילים שהם בצורות שונות אבל יש להם את אותה תחילת למשל: .

ניתן לראות שכל השיטות האלו גרמו להשפעה על כמות המילים ועל גודל המילון שיהיה לנו וכמו כן אם נרצה לעשות אינדקס למשל אנחנו נחסוך מידע שאולי הוא נחוץ ואולי לא, בעולם האחזור מידע אנחנו תמיד מנסים יוריסטיקות שונות כדי להחליט איך לעשות את התהליך הזה. כל בחירה הינה תלויה במשימה שלה. פה ניתן לראות למעלה איך כל שיטה השפיעה וכל אחת יש לה את הסיכונים שלה, בעולם שלנו סיכון זה משהו שיכול להיות קריטי ואולי לא. אצלנו לא היו ממצאים חריגים ביחס למשימה אפשר לעשות את כלל השיטות ועדיין לקבל מודלים טובים.

**המילה הכי שכיחה שלא הושפעה מאף שיטה:**

**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי**



# 

המטרה שלנו בחלק הזה הוא לבנות 4 מסווגים שונים כדי לבחון ביצועים על נתוני אימון שאנחנו בונים בעצמנו בהתייחס לדרישות.

נתחיל בהקדמה קצרה של מה זה סיווג, סיווג טקסט הוא המשימה של סיווג או תיוג אוטומטי של מסמכי טקסט על סמך תוכנם. זה כרוך באימון אלגוריתם למידת מכונה על קבוצה של מסמכי טקסט מסומנים, כאשר לכל מסמך מוקצית קטגוריה או תוויות אחת או יותר. לאחר מכן האלגוריתם משתמש בנתוני האימון הללו כדי ללמוד דפוסים ויחסים בטקסט, ויכול לסווג מסמכי טקסט חדשים שלא נראים לקטגוריות המתאימות על סמך הדפוסים הנלמדים הללו.

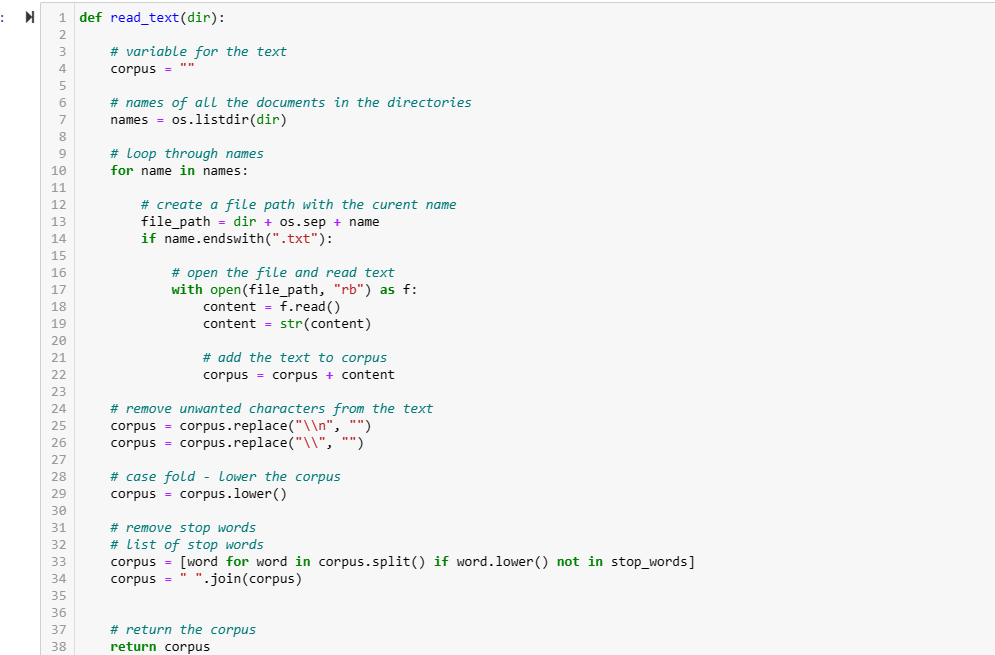
לסיווג טקסט יש מגוון רחב של יישומים, כולל סינון דואר זבל, ניתוח סנטימנטים, מודלים של נושאים והמלצת תוכן, בין היתר. זה יכול לשמש בתחומים שונים, כגון שיווק, בריאות, חינוך וכספים, אם להזכיר כמה. מטרת סיווג הטקסט היא להפוך את תהליך הסיווג או התיוג של מסמכי טקסט לאוטומטיים, דבר שעלול לצרוך זמן רב ונוטה לשגיאות כאשר נעשה באופן ידני, ולאפשר ניתוח מהיר ומדויק של כמויות גדולות של נתוני טקסט.

המסווגים שהשתמשנו בהם:

1. Gaussian Naive Bayes
2. Bernoulli Naive Bayes
3. Rocchio
4. KNN

קודם כל התהליך שלנו התחיל על בחירת המסמכים משאר התקיות שהיו בדרייב:

ואז הגרנו פונקציה שתקרא את המסמכים שיש בכל אחד מנתיבים הללו:



התהליכים דיי דומים כמו שהסברנו קודם בסעיף הקודם.

אחרי שקיבלנו את כל המסמכים תייגנו את המסמכים באופן בינארי באופן הבא:

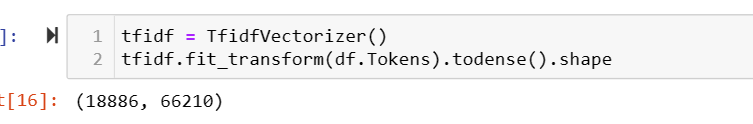
בהתאמה.

מה שקיבלנו:

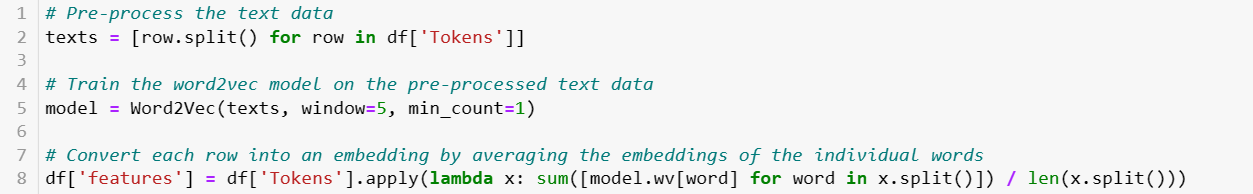
תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

אחרי זה כדי להשתמש במסווגים של למידת מכונה, אנחנו צריכים לייצג כל מסמך\תת מסמך להיות ווקטור שערכים שלו הינם משקלי ה- tf-idf של כל מונח. זה נעשה תוך שימוש בספריה שמכילה את האובייקט המתאים לזה: TfidfVectorizer:



אחרי זה בנינו embedding כדי שכל מילה תהיה תכונה שממנה המסווג ילמד לתת את הסיווגים שלו:



מה שקיבלנו:

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

אחרי זה אנחנו בנינו DATAFRAME חדש שמכיל את התכונות של הווקטור ייצג את המסמך:

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

החלטנו גם ליישם פונקציה של בחירת תכונות כדי לבדוק את התוצאות.

## 

תמונה שמכילה טקסט

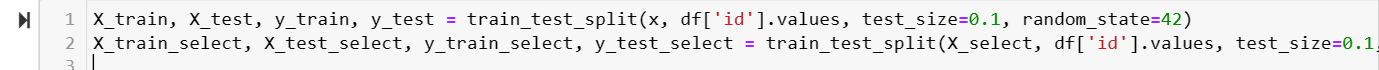
התיאור נוצר באופן אוטומטי

קוד זה מבצע בחירת תכונה באמצעות שיטת בחירת תכונות רקורסיבית (RFE) עם מודל רגרסיה לוגיסטית. להלן הסבר על כל שורה:

* X\_select = pd.DataFrame(x.copy()): יוצר DataFrame X\_select של פנדה חדש עם אותם נתונים כמו x, שהוא ככל הנראה מטריצה של תכונות לסיווג טקסט.
* y\_select = df['id'].values: יוצר מערך numpy חדש y\_select עם התגיות (LABELS) עבור משימת סיווג הטקסט.
* model = LogisticRegression(): יוצר מודל חדש של רגרסיה לוגיסטית.
* rfe = RFE(model, n\_features\_to\_select=15): יוצר אובייקט RFE חדש עם מודל הרגרסיה הלוגיסטית בתור האומד, ומציין שעליו לבחור את 15 התכונות המובילות.
* rfe.fit(X\_select, y\_select): מתאים את אובייקט ה-RFE לנתונים ב-X\_select ו-y\_select, ובוחר את 15 התכונות הטובות ביותר.
* best\_features = X\_select.columns[rfe.support\_].tolist(): מאחזר את שמות 15 התכונות המובילות מאובייקט RFE, וממיר אותם לרשימה.
* X\_select = X\_select[best\_features]: יוצר DataFrame X\_select חדש הכולל רק את 15 התכונות המובילות.

המטרה של קוד זה היא לבצע בחירת תכונות במטריצת הקלט x עבור משימת סיווג טקסט. שיטת RFE משמשת לזיהוי 15 התכונות המובילות החשובות ביותר עבור משימת הסיווג, ונוצר DataFrame X\_select חדש הכולל רק את התכונות הללו. זה יכול לעזור לשפר את הדיוק והיעילות של מודל סיווג הטקסט, על ידי הפחתת הממדיות של תכונות הקלט והתמקדות באלה האינפורמטיבית ביותר.

## חלקות הנתונים לנתוני אימון ובדיקה לפי הדרישות



קוד זה משמש לפיצול הנתונים לקבוצות הדרכה ובדיקות לסיווג טקסט. להלן הסבר על כל שורה:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, df['id'].values, test\_size=0.1, random\_state=42): מפצל את מטריצת הקלט x ומתווית df['id'].values לקבוצות אימון ובדיקה . הפרמטר test\_size מציין את הפרופורציה של הנתונים שישמשו לבדיקה (במקרה זה, 10%), והפרמטר random\_state מגדיר את ה-SEEDS האקראי לשחזור. המשתנים המתקבלים הם X\_train (תכונות אימון), X\_test (תכונות בדיקה), y\_train (תוויות אימון) ו-y\_test (תוויות בדיקה).

מטרת קוד זה היא לפצל את הנתונים למערכות הדרכה ובדיקות להדרכה והערכה של מודלים, הן עבור התכונות המקוריות והן עבור התכונות הנבחרות. לאחר מכן, המשתנים המתקבלים יכולים לשמש כקלט לקוד האימון של המודל, כגון רגרסיה לוגיסטית או k-NN, כדי לאמן ולהעריך את ביצועי המודל. על ידי שימוש הן בתכונות המקוריות והן בתכונות שנבחרו, הקוד יכול להשוות את הביצועים של המודלים עם ערכות תכונות שונות, ולבחור את זו שמשיגה את הציון הטוב ביותר.

## הסיווג

### 

Gaussian Naive Bayes הוא אלגוריתם סיווג שמניח שהתכונות בנתונים מתפלגות נורמאלית , והוא מחשב את הסבירות של כל תכונה להיות שייכת למחלקה מסוימת. לאחר מכן הוא משלב את ההסתברויות כדי לחזות את המחלקה עם ההסתברות הגבוהה ביותר. האלגוריתם פשוט, מהיר ועובד היטב עם מערכי נתונים קטנים עד בינוניים, מה שהופך אותו לבחירה פופולרית למשימות סיווג טקסט. עם זאת, ייתכן שהוא לא מתפקד טוב עם מערכי נתונים גדולים ומורכבים מאוד או כאשר הנחת האי תלות של התכונות אינה מתקיימת.

יישום:

קוד זה מבצע hyperparameter tuning עבור מסווג גאוס נאיבי באמצעות GridSearchCV:

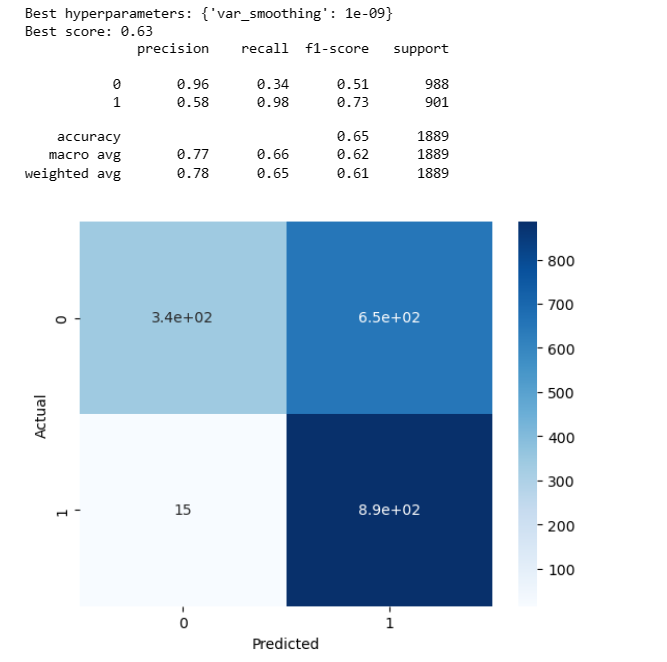
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

* estimator\_NB = GaussianNB(): יוצר אובייקט מסווג גאוס נאיבי חדש בתור האומד עבור ה- grid search.
* param\_grid\_NB = {'var\_smoothing': [1e-9, 1e-8, 1e-7]}: מגדיר מילון של היפרפרמטרים לחיפוש. במקרה זה, אנו עושים tunning רק ל- פרמטר var\_smoothing, שהוא פרמטר החלקה שקובע את עוצמת ה- regularization על הערכות השונות של התכונות .
* kfold = KFold(n\_splits=10, shuffle=True, random\_state=42): יוצר אובייקט חדש של של K-fold עם 10 פיצולים, ערבוב הנתונים לפני הפיצול והגדרת ה-seed האקראי לשחזור.
* grid\_search = GridSearchCV(estimator\_NB, param\_grid\_NB, cv=kfold, n\_jobs=-1): יוצר אובייקט GridSearchCV חדש עם אומדן ה-Gaussian Naive Bayes, מילון ההיפרפרמטרים, אובייקט K-fold ומספר העבודות להפעלה במקביל (n\_jobs=-1 פירושו להשתמש בכל ליבות המעבד הזמינות).
* grid\_search.fit(X\_train, y\_train): מתאים את אובייקט GridSearchCV לנתוני האימון X\_train ו-y\_train, על ידי חיפוש במרחב ההיפרפרמטר שהוגדר ב-param\_grid\_NB והערכת ביצועי המודל עם אימות צולב.
* y\_pred = grid\_search.predict(X\_test): משתמש במודל הטוב ביותר שנמצא כדי לחזות את התוויות עבור נתוני הבדיקה X\_test.

מטרת קוד זה היא לבצע hyperparameter tuning עבור המסווג גאוס נאיבי, על ידי חיפוש על הפרמטר var\_smoothing ובחירה באחד שמשיג את ציון ה-KFOLD הטוב ביותר. ניתן להשתמש בהיפרפרמטרים ובניקוד הטובים ביותר המתקבלים כדי לאמן מודל חדש על כל נתוני האימון, ולהעריך את הביצועים שלו על נתוני הבדיקה. ניתן להשתמש ב- y\_pred גם לחישוב מדדי הערכה שונים.

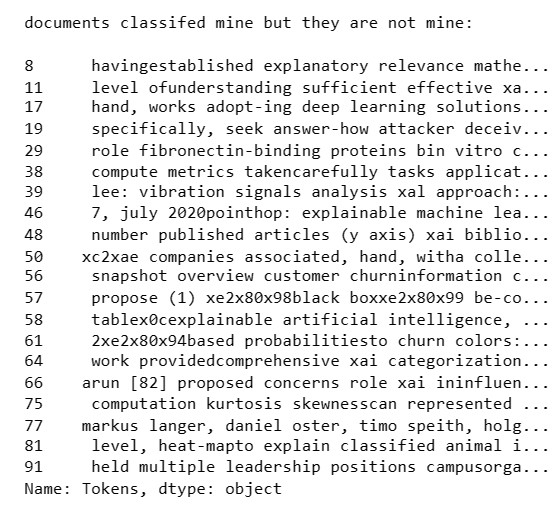
התוצאות שקיבלנו:



ה- ACCURACY שווה ל 0.65 שכלומר המודל שלנו יצליח לסווג נכון בהסתברות של 0.65

עבור מחלקה 0, ה-PRECISION הוא 0.96, מה שאומר שכאשר המודל חוזה שמסמך נמצא במחלקה 0, הוא צודק ב-96% מהמקרים. עם זאת, ה-RECALL הוא רק 0.34, מה שאומר שהמודל מסוגל לזהות נכון רק 34% מהמסמכים האמיתיים של מחלקה 0. זה מצביע על כך שהמודל טוב מאוד בזיהוי ה- negative class כשהיא מופיעה, אבל הוא לא טוב מאוד בזיהוי כל ה- negative instances.

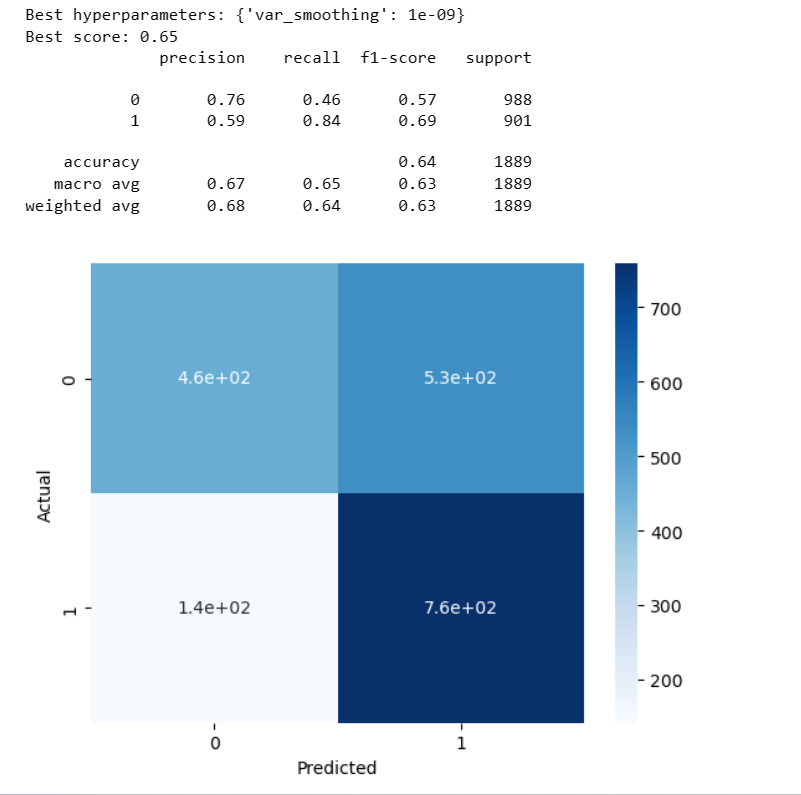
עבור מחלקה 1, ה- PRECISION הוא רק 0.58, מה שאומר שכאשר המודל חוזה שמסמך נמצא במחלקה 1, הוא צודק רק 58% מהמקרים. עם זאת, ה-RECALL הוא 0.98, מה שאומר שהמודל מסוגל לזהות נכון 98% מהמסמכים במחלקה 1 בפועל. זה מצביע על כך שהמודל אינו מדויק מאוד בזיהוי כל ה- positive instances, אך הוא טוב מאוד בזיהוי ה- positive instances כאשר הוא כן מזהה אותם.



תמונה שמכילה טקסט, עיתון, צילום מסך, מסמך

התיאור נוצר באופן אוטומטי

אחרי שימוש רק בתכונות שנבחרו על ידי ה-RFE:



תמונה שמכילה טקסט, עיתון, מסמך

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, עיתון, מסמך

התיאור נוצר באופן אוטומטי

אין הבדל משמעותי בין הממצאים של שימוש במודל אחרי ולפני בחירת תכונות, הביצועים כמעט זהים לכן אותו הסבר כמו קודם.

## 

Bernoulli Naive Bayes הוא גרסה של Naive Bayes המשמשת לבעיות סיווג בינארי עם תכונות בעלות ערך בינארי (0 או 1), כגון סיווג טקסט. זה מחשב את ההסתברות שכל תכונה שייכת למחלקה מסוימת ומשלבת אותם כדי לקבוע את ההסתברות שנקודת נתונים שייכת למחלקה. המודל הזה מניח שתכונות אינן תלויות זו בזו, מה שהופך אותו לפשוט ומהיר, אך מודל זה עשוי שלא לעבוד טוב עם מערכי נתונים גדולים או מורכבים או כאשר הנחת האי תלות מופרת.

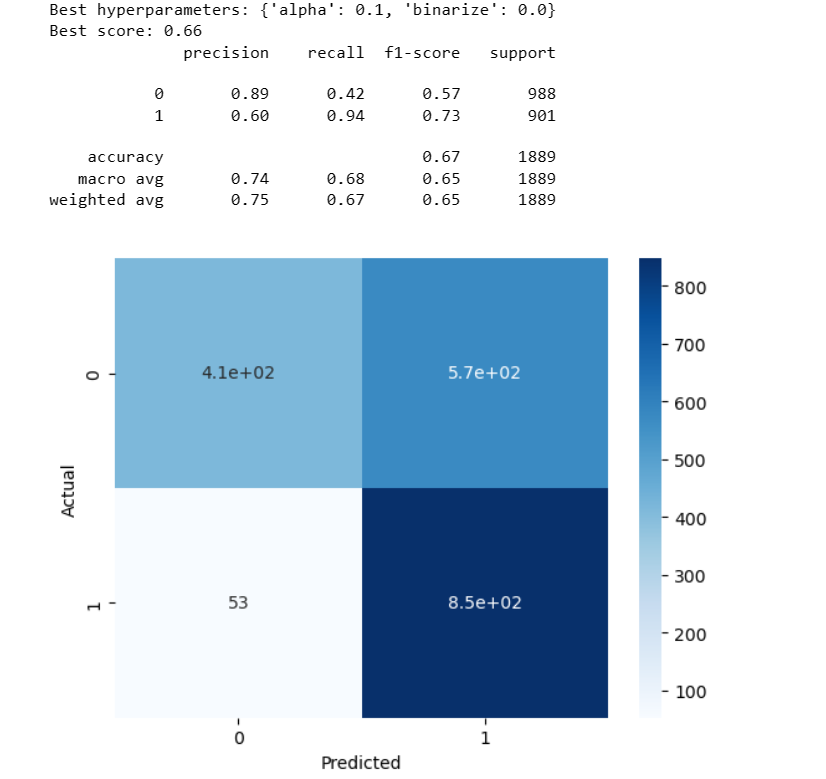


למסווג BernoulliNB יש שני היפרפרמטרים שניתן לעשות להם TUNING:

* alpha: זהו פרמטר החלקה, השולט ביכולת ההכללה של המודל. ערך קטן יותר של אלפא פירושו פחות החלקה והמודל רגיש יותר לנתוני האימון, בעוד שערך גדול יותר של אלפא מביא ליותר החלקה והמודל ניתן להכללה יותר.
* binarize: פרמטר זה משמש לבינאריזציה של התכונות, כלומר המרת ערכי התכונה ל-0 או 1 על סמך ערך סף. אם בינאריות מוגדרת ל-0, התכונות אינן בינאריות, בעוד שערך שאינו אפס ישמש כערך הסף. סף נמוך יותר יביא ליותר תכונות בינאריות.

מילון param\_grid מכיל את הערכים של ההיפרפרמטרים הללו שינוסו על ידי GridSearchCV. cv=kfold מגדיר את שיטת ה- CROSS VALIDATION לשימוש, ו-n\_jobs=-1 אומר לפונקציה להשתמש בכל המעבדים הזמינים כדי להאיץ את החישוב. המשתנה y\_pred מכיל את הערכים החזויים עבור נתוני הבדיקה.

התוצאות שקיבלנו:

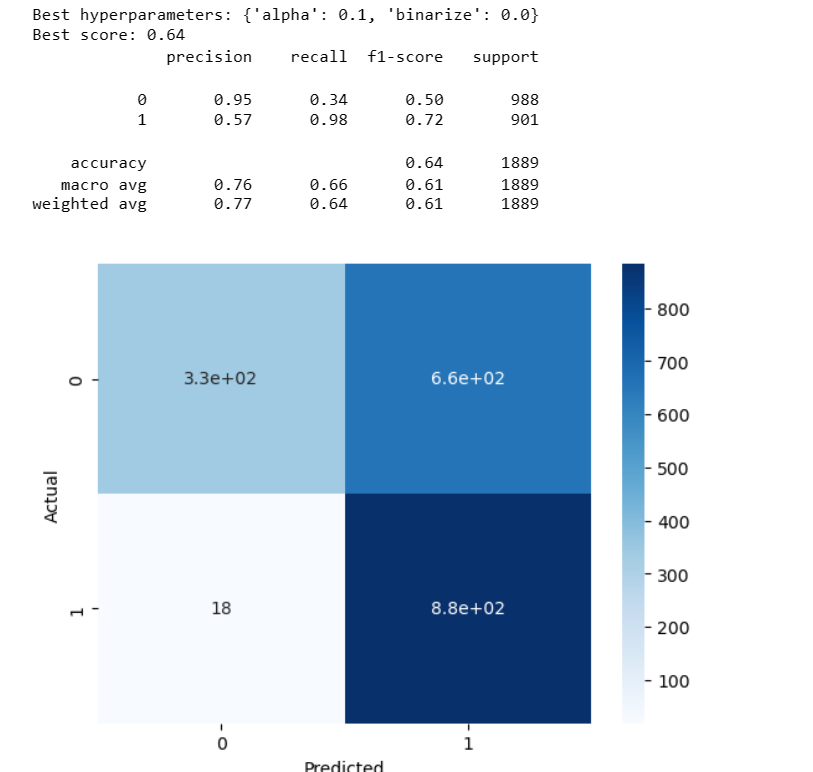


תוצאות דיי דומות למודל הקודם לכן ההסבר יהיה זהה.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

ממצאים אחרי שימוש במודל על התכונות שנבחרו על ידי RFE:



היה שיפור קטן בערכים של ה- RECALL וה- PRECISION עבור מחלקה 0 ומחלקה 1. וההסבר הכולל זהה למודל הקודם מבחינת הערכים של כל המדדים.

## 

סיווג Rocchio הוא אלגוריתם השייך לאלגוריתמי ה- SUPERVISED המשמש למשימות סיווג טקסט, הפועל על ידי הקצאת מסמך למחלקה בהתבסס על הדמיון שלו למרכז המחלקה (CENTROID). המרכז מחושב כממוצע של וקטורי התכונה של כל מסמכי האימון השייכים לאותה מחלקה.

באלגוריתם זה, כל מחלקה מיוצגת על ידי וקטור מרכז במרחב התכונה. האלגוריתם מחשב תחילה את הסנטרואידים של כל המחלקות בהתבסס על נתוני האימון, ולאחר מכן משתמש במרכזים אלו כדי לסווג מסמכים חדשים.

כאשר מוצג מסמך חדש, האלגוריתם מחשב את וקטור התכונה שלו ומשווה אותו למרכזים של כל המחלקות. לאחר מכן, המסמך מסווג למחלקה עם המרכז הקרוב ביותר. מדד הדמיון המשמש להשוואת וקטור התכונה והסנטרואידים הוא בדרך כלל הדמיון הקוסינוס.

אלגוריתם Rocchio הוא פשוט, מהיר ויכול להיות יעיל עבור משימות סיווג טקסט, במיוחד עבור משימות שבהן מספר התכונות קטן יחסית. עם זאת, הוא יכול לסבול מבעיה של התפלגויות תכונות שהן חופפות, כאשר למחלקות שונות יש התפלגות תכונות דומות, מה שמקשה על הפרדתן על ידי הגבולות שהוא יוצר.

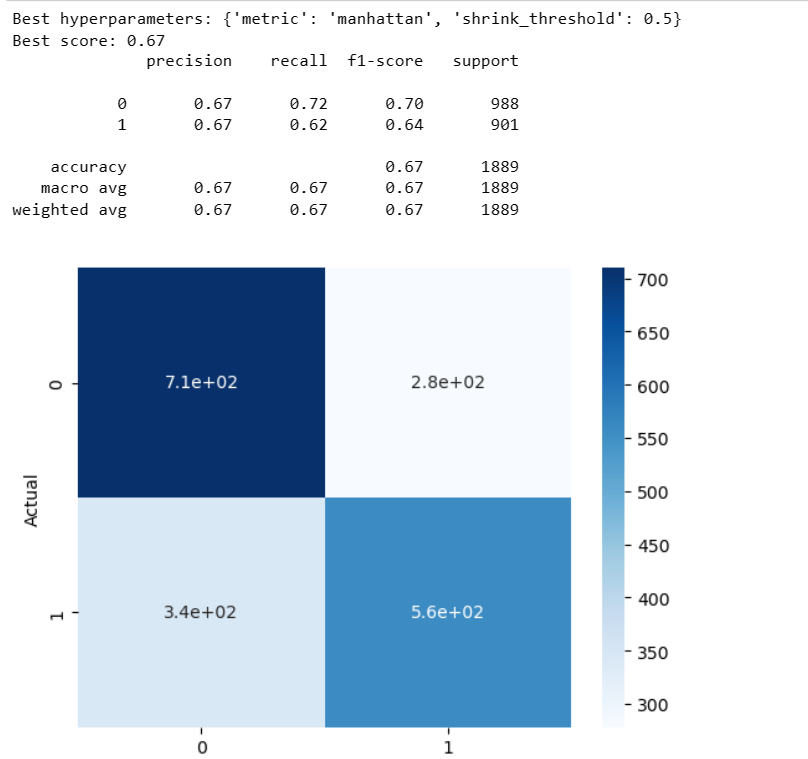
יישום:



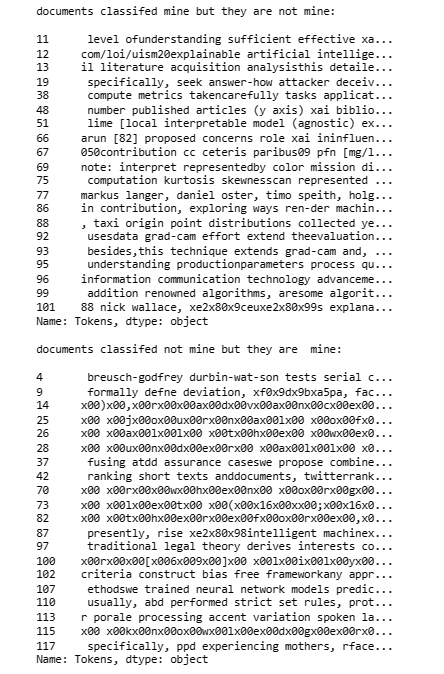
הסבר:

* metric: המדד המשמש לחישוב המרחק בין המרכזים לנקודות הנתונים. במקרה שלנו, האפשרויות הן אוקלידית, מנהטן ומינקובסקי.
* shrink\_threshold: פרמטר רגוליזציה המסייע במניעת התאמת יתר(OVERFITTING). הוא מכווץ את המרכזים לכיוון הממוצע הכולל של הנתונים כאשר המרחקים שלהם קטנים מהסף. במקרה שלנו, האפשרויות הן ללא, 0.1, 0.5 ו-1.0.

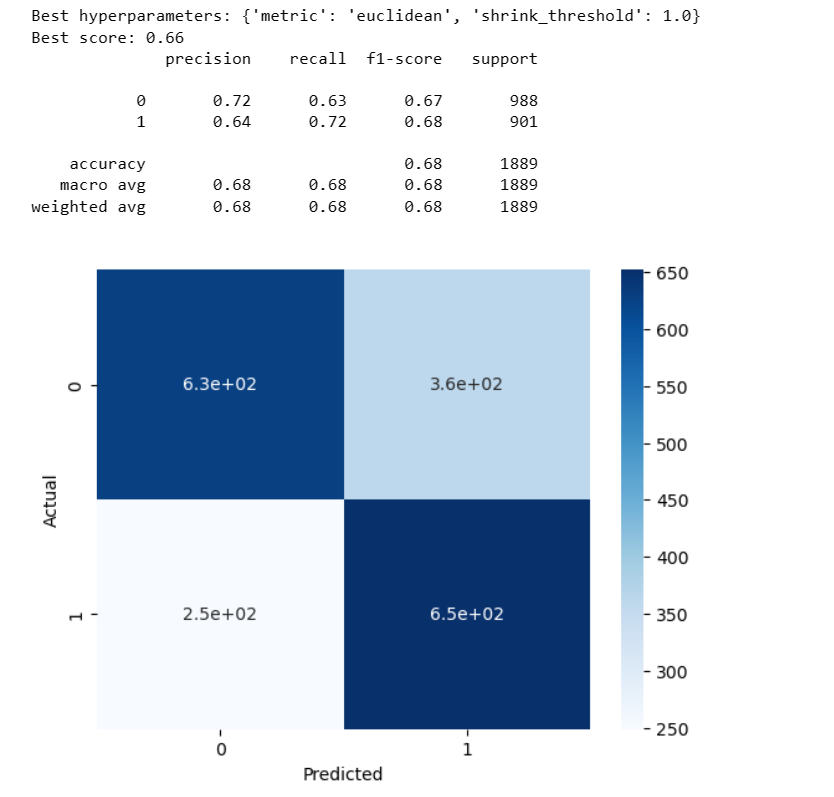
תוצאות שקיבלנו:



ניתן לראות פה שיש יותר איזון בתוצאות החיזויים של המודל הזה(נסביר בהמשך). ה-PRECISION של המסווג הוא 0.67 עבור מחלקה 0 ו-0.67 עבור מחלקה 1, מה שאומר שמתוך כל המסמכים שסווגו כמחלקה 0, 67% היו למעשה במחלקה 0, ומתוך כל המסמכים שסווגו כמחלקה 1, 67% היו בפועל במחלקה 1. ה-RECALL של המסווג הוא 0.72 עבור מחלקה 0 ו-0.62 עבור מחלקה 1, כלומר המסווג זיהה נכון 72% מכל מסמכי מחלקה 0 ו-62% מכל מסמכי מחלקה 1.



אחרי שימוש בתכונות שנבחרו על ידי ה-RFE:



ה-PRESICION של המסווג הוא 0.72 עבור מחלקה 0 ו-0.64 עבור מחלקה 1, מה שאומר שמתוך כל המסמכים שסווגו כמחלקה 0, 72% היו למעשה במחלקה 0, ומתוך כל המסמכים שסווגו כמחלקה 1, 64% היו בפועל במחלקה 1. ה-RECALL של המסווג הוא 0.63 עבור מחלקה 0 ו-0.72 עבור מחלקה 1, כלומר המסווג זיהה נכון 63% מכל מסמכי מחלקה 0 ו-72% מכל מסמכי מחלקה 1.

ציון F1 הוא הממוצע ההרמוני של ה-RECALL ו PRESICION, המספק מדד לאיזון בין שני המדדים. ציון F1 הוא 0.67 עבור שתי המחלקות, מה שמצביע על ביצועים דומים יחסית של המסווגן עבור שתי המחלקות. ה-ACCURACY של המסווג הוא 0.68, מה שאומר שהוא סיווג נכון 68% מכלל המסמכים.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

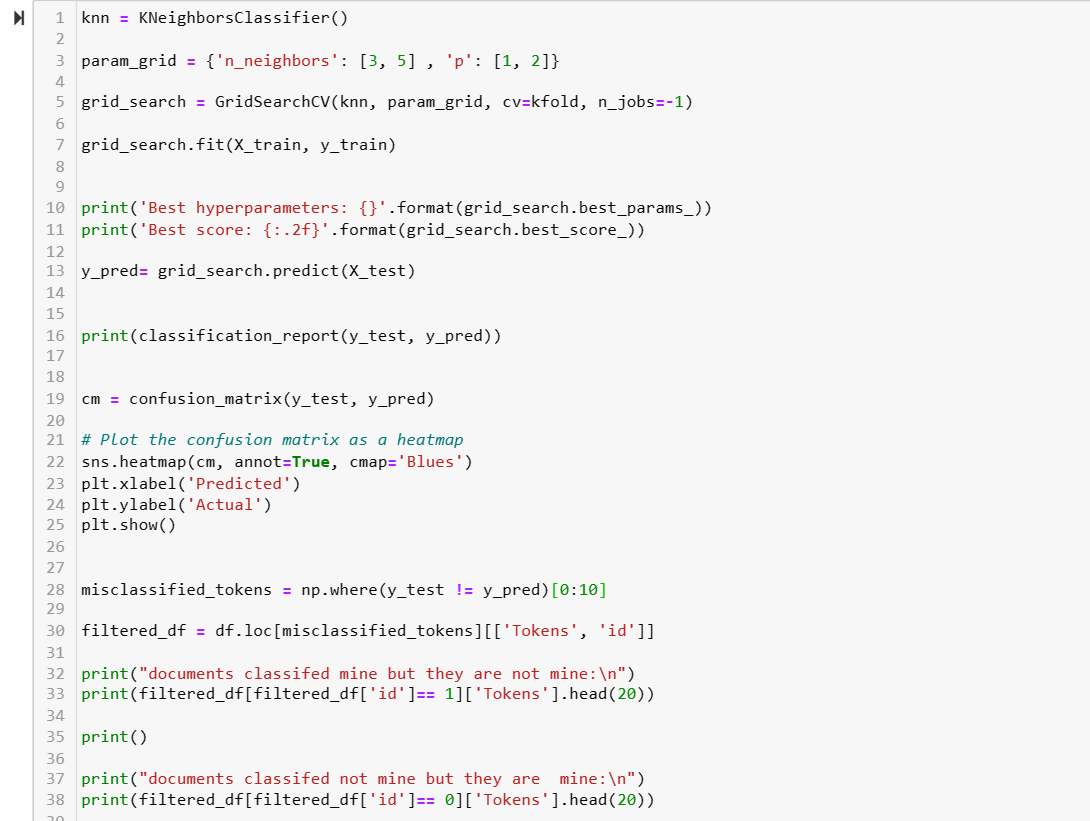
## KNN

KNN, הוא אלגוריתם למידת מכונה פופולרי המשמש במשימות סיווג טקסט. אלגוריתם ה-KNN פועל על ידי אימון תחילה על קבוצה של דוגמאות מתויגות, ולאחר מכן שימוש במרחקים בין הדוגמאות החדשות, ללא תווית, או נתוני הבדיקה, לבין הדוגמאות המתויגות כדי לסווג את הדוגמאות החדשות.

בסיווג טקסט, ניתן להשתמש ב-KNN כדי למצוא את k השכנים הקרובים ביותר לדוגמאות הבדיקה בנתוני האימון, כאשר "הקרוב ביותר" מוגדר על ידי מדד מרחק, כגון מרחק אוקלידי או דמיון קוסינוס, בין וקטורי הטקסט. המחלקה של רוב ה-k השכנים הקרובים ביותר משמשת לאחר מכן לסיווג נתונים לבדיקה.

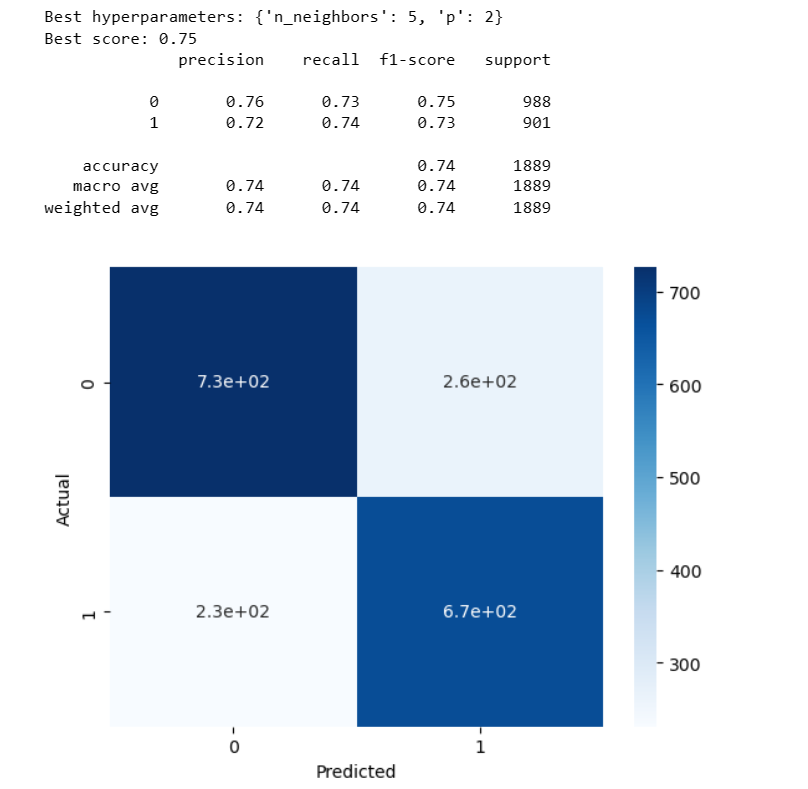
יתרון אחד בשימוש ב-KNN לסיווג טקסט הוא שזהו אלגוריתם פשוט וניתן לפירוש וכמו כן לא צריך אימון המודל. בנוסף, KNN אינו מניח הנחות לגבי ההתפלגות הבסיסית של הנתונים ויכולה להיות יעילה עם מערכי נתונים גדולים. עם זאת, KNN יכול להיות יקר מבחינה חישובית, במיוחד כאשר מספר התכונות, או גודל וקטור הטקסט, גדל. זה גם דורש הגדרת הערך של k, מה שיכול להיות מאתגר במקרים מסוימים.

יישום:



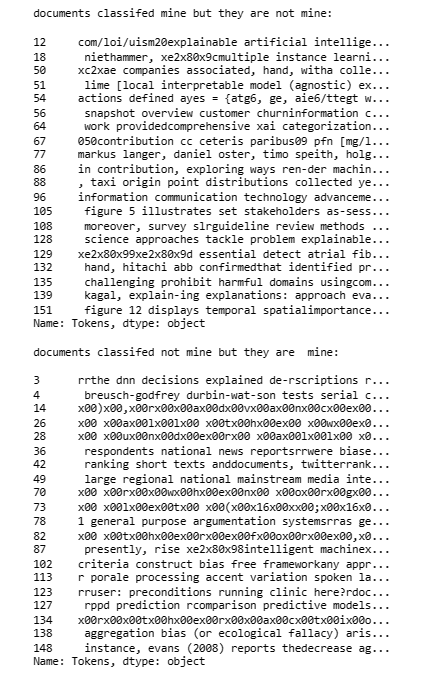
* n\_neighbors: היפרפרמטר זה שולט במספר השכנים הנחשבים בעת ביצוע חיזוי. ב-KNN, כל נקודת נתונים מסווגת על סמך מחלקת הרוב של K השכנים הקרובים ביותר. בקוד זה, אנו מחפשים את הערכים של 3 ו-5, כדי לראות איזה ערך נותן את הביצועים הטובים ביותר.
* p: היפרפרמטר זה שולט במדד המרחק המשמש לחישוב המרחק בין נקודות במרחב התכונה. ב-KNN, מדדי המרחק הנפוצים ביותר הם מרחק אוקלידי (p=2) ומרחק מנהטן (p=1). כאן אנו מחפשים את הערכים של 1 (מרחק מנהטן) ו-2 (מרחק אוקלידי).

תוצאות שקיבלנו:



ה-PRESICION של המסווג הוא 0.76 עבור מחלקה 0 ו-0.72 עבור מחלקה 1, מה שאומר שמתוך כל המסמכים שסווגו כמחלקה 0, 76% היו למעשה במחלקה 0, ומתוך כל המסמכים שסווגו כמחלקה 1, 72% היו בפועל במחלקה 1. ה-RECALL של המסווג הוא 0.73 עבור מחלקה 0 ו-0.74 עבור מחלקה 1, כלומר המסווג זיהה נכון 73% מכל מסמכי מחלקה 0 ו-74% מכל מסמכי מחלקה 1.

ציון F1 הוא הממוצע ההרמוני של ה-RECALL ו PRESICION, המספק מדד לאיזון בין שני המדדים. ציון F1 הוא 0.73 עבור שתי המחלקות, מה שמצביע על ביצועים דומים יחסית של המסווגן עבור שתי המחלקות. ה-ACCURACY של המסווג הוא 0.74, מה שאומר שהוא סיווג נכון 74% מכלל המסמכים.



החלטנו לא לעשות עוד מסווג KNN לתכונות שנבחרו על ידי ה- RFE כי ה- KNN לקח הרבה זמן כדי לעשות סיווג ולמידת ההיפרפרמטרים גם לקחה זמן רב

## דיון לגבי התוצאות והשוואה בין מודלים וכמו כן הסבר על למה קיבלנו ציונים אלו במדדים של הערכה

|  |  |
| --- | --- |
| **KNN** |  |
| **Rocchio** |  |
| **Bernoulli Naïve Bayes** |  |
| **Gaussian Naïve Bayes** |  |

ניתן לראות שאלגוריתם KNN עבד הכי טוב במקרה שלנו מבחינת כלל המדדים, לכן אם הייתה לנו משימה לבחור מסווג היינו בוחרים את KNN אבל ברמת בטיחות לא כל כך גבוהה. אחריו בא האלגוריתם של ROCHHIO שהוא גם נתן ביצועים טובים וכמעט מאוזנים בין כלל המדדים כמו ב- KNN. המודלים של NAÏVE BAYES נתנו תוצאות לא טובות וגם ציוני המדדים אינם מאוזנים בין שתי המחלקות.

הסיבה לקבלת תוצאות מאוזנות (כלומר, ערכי PRESCION ו RECALL דומים עבור שתי המחלקות) היא ככל הנראה משום שמסווג Rocchio מתאים את גבול ההחלטה על סמך המרחק של כל מסמך למרכז של כל מחלקה. בדרך זו, הוא יכול להתמודד עם מערכי נתונים לא מאוזנים טוב יותר מאשר מסווגים אחרים, שעלולים להיות מוטים למחלקת הרוב. ועבור KNN הוא יכול לבצע ביצועים טובים עבור משימות סיווג טקסט מכיוון שהוא יכול ללכוד את המבנה המקומי בנתונים ויכול להשתמש במבנה זה כדי לבצע תחזיות. בסיווג טקסט, משמעות הדבר היא ש-KNN יכול לזהות דוגמאות טקסט דומות ולהשתמש בהן כדי לבצע תחזיות. אם דוגמאות הטקסט בכל מחלקה דומות מבחינת השימוש במילה והקשר שלהן, אז KNN יכול לזהות את קווי הדמיון הללו ולהציג ביצועים טובים בניבוי תווית המחלקה הנכונה. בנוסף, מכיוון ש-KNN אינו מניח הנחות לגבי התפלגות הנתונים, הוא יכול להתמודד עם גבולות החלטה לא ליניאריים ואינטראקציות מורכבות של התכונות וכמו כן יכול להתמודד עם מערכי נתונים לא מאוזנים, מה שיכול להועיל במשימות סיווג טקסט.

בניגוד ל- ROCCHIO ו KNN המודלים NAÏVE BAYES נתנו תוצאות לא מאוזנות ואין לנו אפשרות למתן בטיחות בתוצאות שלהם. התוצאה יכולה להיגרם מהסיבה שמודלים אילו מניחים הנחה מאוד רגישה לגבי המונחים במסמכים וההנחה היא שהתכונות שלנו בלתי תלויים אחד בשני.

למה בכללי קיבלנו תוצאות לא כל כך טובות וסיווגים לא נכונים?

יש כל כך סיבות בהקשר של המשימה שלנו, הסיבה הראשונה שעלולה להיות הגורם היא עיבוד מקדים לא טוב על הנתונים שלנו כלומר לא יצרנו תכונות באופן מקצועי. וכמו כן ייתכן שיש אי איזון בנתוני האימון כלומר מחלקה אחת מחסלת את השנייה. בנוסף לכך חיפוש ההיפרפרמטרים נעשתה באופן GREEDY כלומר עצרנו כאשר הגענו למינימום מקומי וייתכן שפעולה זו הייתה נעצרת מהר מדי. וכמו כן ייתכן שהמסמכים שסופקו לנו על ידי בסטודנטים בקורס משתמשים במונחים דומים ואז יהיה לכולם תכונות דומים וזה מקשה על סיווג ומציאת גבולות בין המחלקות. וייתכן שקרה לנו OVERFITTING אן UNDERFITINNG בצורה כלשהי. אם נסתכל למעלה על המסמכים שסווגו לא נכון נראה שזה בגלל שמסמכים רבים מכילים הרבה מונחים של המסמכים שלי והם לא מדברים על אותו נושא מה שמקשה על אלגוריתמים כמו למשל NAÏVE BAYES לזהות דברים כאלה במיוחד כי הם מניחים אי תלות.

# 

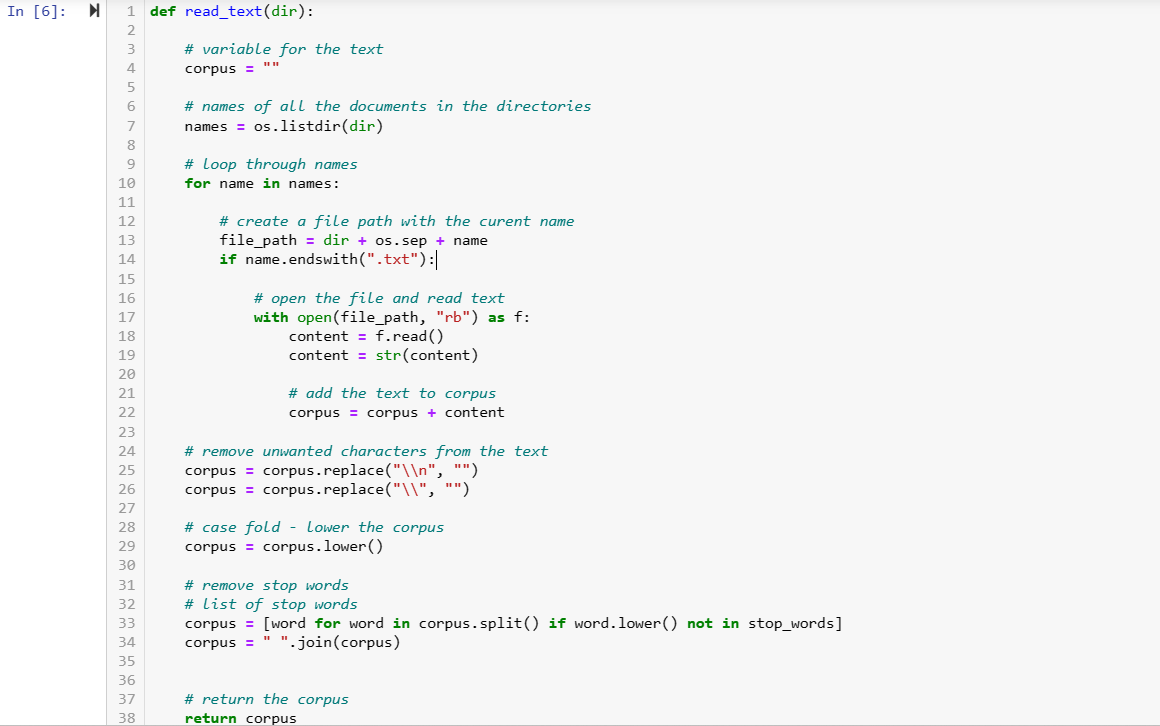
אשכול טקסט, הידוע גם בשם אשכול מסמכים, הוא המשימה לקבץ קבוצה של מסמכי טקסט לאשכולות על סמך קווי הדמיון ביניהם. אשכול כרוך בחלוקת המסמכים במערך נתונים לקבוצות, כך שהמסמכים בתוך כל קבוצה חולקים מאפיין משותף כלשהו, תוך שהם שונים מאלה שבקבוצות אחרות. המטרה של אשכול טקסט היא לזהות תבניות ומבנה בסיסי באוסף גדול של נתוני טקסט לא מובנים, ולהקל על ניווט ושליפה של המסמכים. תהליך אשכול טקסט כולל חילוץ תכונות מנתוני הטקסט, כגון TF או ציוני TF-IDF, ולאחר מכן שימוש באלגוריתמים של clustering כגון K-MEANS.

## התהליך

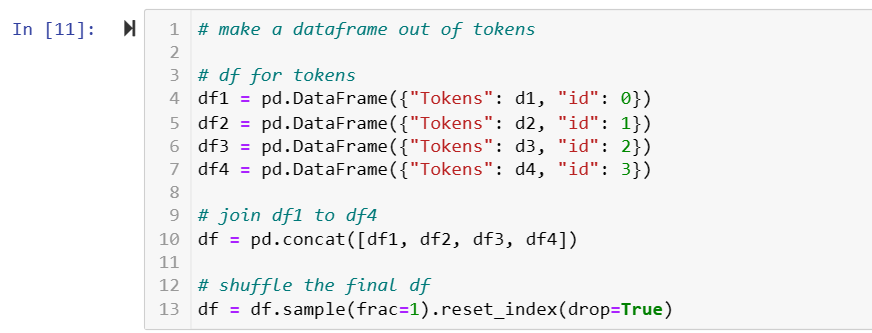
ראשית התחלנו את התהליך על ידי קריאת רשימת המילות עצורה שצוות הקורס נתן לנו:



אחרי זה השתמשנו בפונקציה שתקרא לנו את המסמכים ותוך כדי תבצע ניקוי למסמכים הללו:

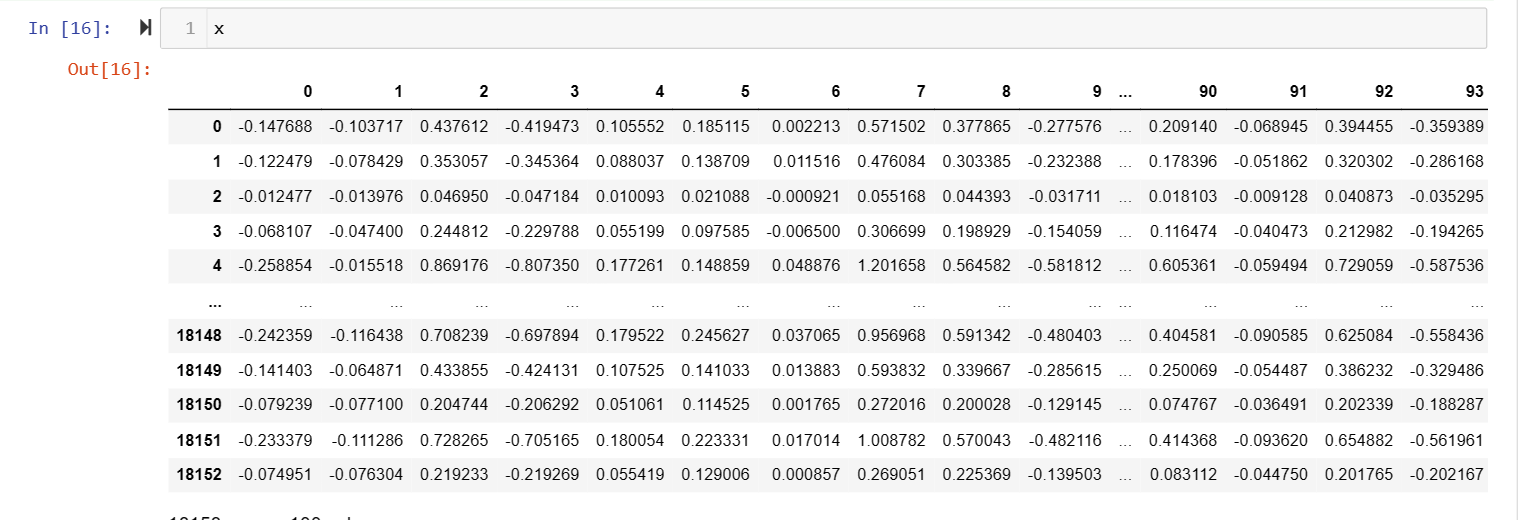


הסברנו על הפונקציה קודם לכן. אחרי זה אני ושותפתי תייגנו את המסמכים כדי לבדוק ביצעיו של אלגוריתם KMEANS, התיוג היה בהתאם למסמכים שבחרנו מהתקיות בדרייב:



|  |  |
| --- | --- |
| 0 | Our documents |
| 1 | artificial intelligence explainability |
| 2 | Twitter bias |
| 3 | Social bias |

ואז ייצגנו את המסמכים על ידי ווקטורים בייצוג TF-IDF וקיבלנו את הווקטורים הבאים:

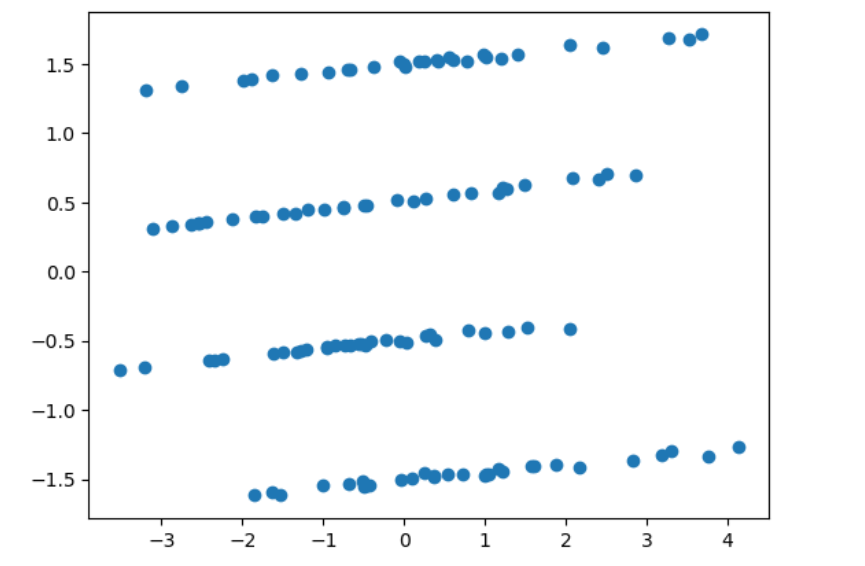


לצורך ייעול התהליך של הקיבוץ בחרנו מדגם מייצג של 120 תצפיות באופן רנדומלי:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

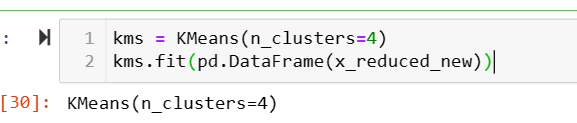
### ביצענו תהליך של הורדת ממדיות על ידי שימוש ב PCA לצורך קבלת תחושה על המסמכים:

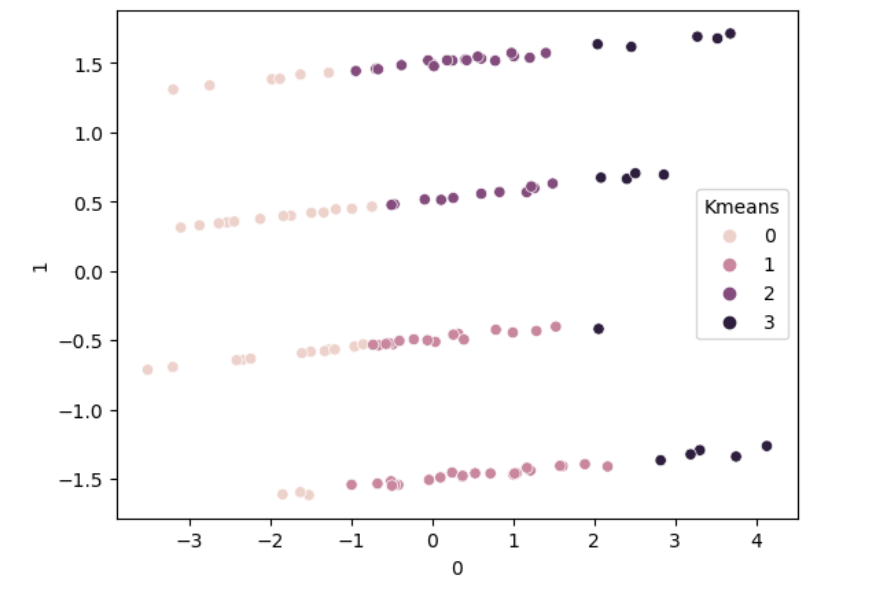


השאלה עכשיו היא האם KMEANS יצליח באמת להביא אותנו ל 4 אשכולות? התשובה שלנו לפני ביצוע האלגוריתם היא לא כי KMEANS לא מצליח הרבה באשכול מסמכים אם צורת ההתפלגות שלהם היא לא כדורית.

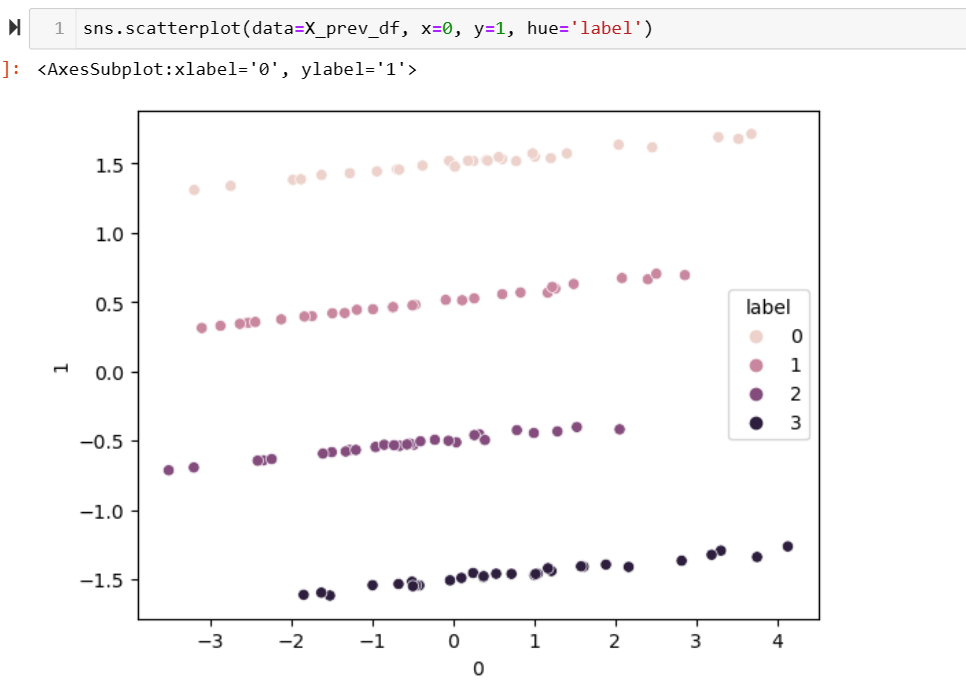
השתמשנו באלגוריתם וביצענו את האשכול.

### ביצוע KMEANS ו תוצאות שקיבלנו ו הצגת גרפים והסבר למה היה טעויות



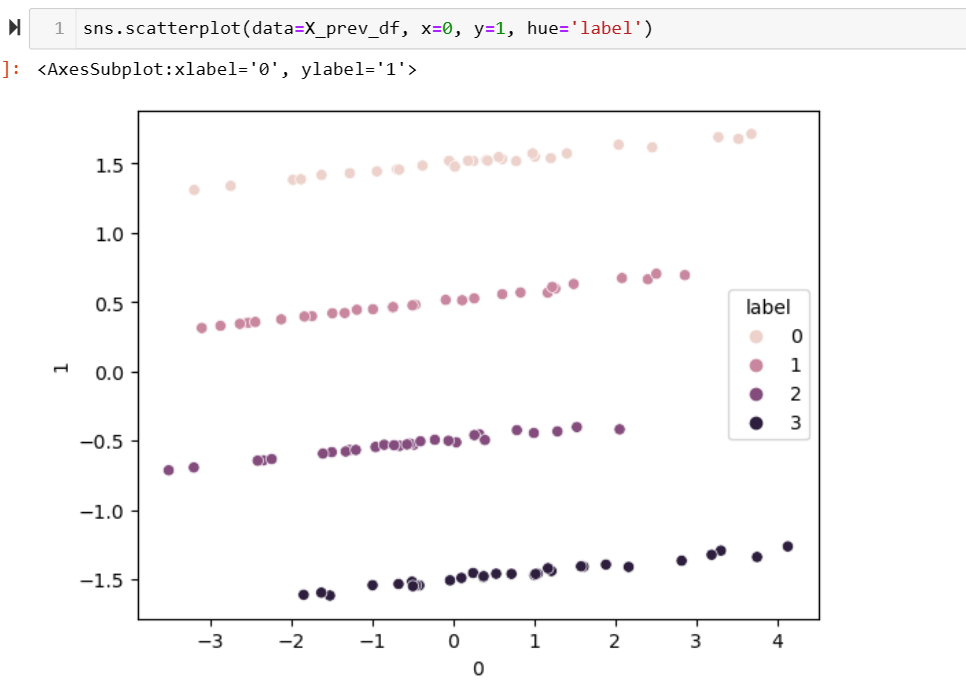


**מה שציפינו אליו בהתייחס לתגיות שנתנו למסמכים:**



הגרף הראשון מראה תוצאות של KMEANS שנתן ביצועים לא טובים על הנתונים שלנו. יש כמה סיבות למה KMEANS לא עבד טוב, ראשית הממדיות של הנתונים שלנו הינה גבוהה מה שיקשה על KMEANS בחישוב המרחקים שהוא מחשב, וכמו כן אולי השימוש במרחק האוקלידי הוא מדד לא טוב לביצוע המשימה הזו. לבסוף KMEANS מאוד רגיש להפלגות הנתונים ומאוד לא יעיל כאשר הנתונים אינם כדוריים, ייתכן ש-k-means clustering לא יעבוד טוב מכיוון שההנחה הבסיסית של אשכולות איזוטרופיים מופרת. נתונים שאינם כדוריים יכולים לכלול אשכולות מוארכים, אשכולות בעלי צורה לא סדירה או אשכולות עם צפיפות או שונות משתנים. במקרים אלה, האשכולות עלולים לחפוף או להיות בעלי גבולות מורכבים שקשה ל-k-means להפרידם.

לבסוף אנחנו לא היינו נותנים אימון בתוצאות ה-KMEANS ולא יכלנו להניב מסקנות אחרי הרצת ה לKMEANS, התיוג הידני היה יותר ברור:



תודה שקראת את כל העבודה!