ממ"ן 22

מגיש: נטף רנד

ת.ז. 040672339

מנחה: רועי רחמני

1.א+ב.

- השתמשתי בקובץ שהתקבל מממ"ן 21.

- השתמשתי באלגוריתם אפריורי והרצתי אותו מספר פעמים, תוך שינוי אחוזי התמיכה והביטחון המינימליים ומספר החוקים.

- כאשר הרצתי את האלגוריתם על כל 17 התכונות בהן השתמשתי במטלה 21, גיליתי שרובם המוחלט .is\_weekend של החוקים נסמך על התכונה

בצילום המסך למטה, ניתן לראות למשל, כי מתוך 41 החוקים שבין חוק 17 לחוק 57, 40 חוקים (!) מתייחסים לתכונה זו כאשר היא שלילית. הצילום נלקח מהרצה עבור תמיכה מינימלית 0.15 ובטחון מינימלי 0.5, אבל ההתנהגות הייתה זהה גם עבור ערכים אחרים. הסיבה להתנהגות זו נעוצה במספר השדות הרב שאינו מקיים את התכונה – 33349/38367.



לכן הורדתי את התכונה והרצתי את האלגוריתם שוב.

- עבור רמת תמיכה מינימלית 0.1, רמת ביטחון מינימלית 0.5 ו-100 חוקים, התקבלו חוקים רבים, אך רבים מהם בעלי רמת בטחון 1 ולכן לא מעניינים.

להלן טבלה של החוקים המעניינים, בעלי רמת ביטחון קטנה מ-1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lift | בטחון | חוקים מענייינים | מספר חוקים (חוקים שהתקבלו) | בטחון מינימום | תמיכת מינימום | אלגוריתם |
| 6.12 | 0.96 | number\_of\_imgs=1 self\_reference\_min\_shares='(-inf-369.5]' 4444 ==> self\_reference\_avg\_sharess='(-inf-368.5]' 4282 | 100 (100)  מתוכם 82 מעניינים (רמת ביטחון נמוכה מ-1) | 0.5 | 0.1 | Apriori |
| 6.12 | 0.96 | number\_of\_imgs=1 self\_reference\_min\_shares='(-inf-369.5]' 4444 ==> self\_reference\_max\_shares='(-inf-354.5]' 4280 |
| 6.12 | 0.96 | number\_of\_imgs=1 self\_reference\_min\_shares='(-inf-369.5]' 4444 ==> self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_avg\_sharess='(-inf-368.5]' 4280 |
| 4.45 | 0.95 | LDA\_02='(0.628306-inf)' 5517 ==> data\_channel=6 5223 |
| 5.96 | 0.94 | self\_reference\_min\_shares='(-inf-369.5]' 6441 ==> self\_reference\_avg\_sharess='(-inf-368.5]' 6038 |
| 5.96 | 0.94 | self\_reference\_min\_shares='(-inf-369.5]' 6441 ==> self\_reference\_max\_shares='(-inf-354.5]' 6035 |
| 5.96 | 0.94 | self\_reference\_min\_shares='(-inf-369.5]' 6441 ==> self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_avg\_sharess='(-inf-368.5]' 6035 |
| 2.18 | 0.89 | self\_reference\_avg\_sharess='(1896.166667-3195.75]' 6913 ==> self\_reference\_max\_shares='(2050-10950]' 6174 |
| 3.34 | 0.85 | data\_channel=3 6175 ==> LDA\_00='(0.201706-0.866666]' 5234 |
| 2.05 | 0.84 | self\_reference\_avg\_sharess='(3195.75-6998]' shares=Popular 4713 ==> self\_reference\_max\_shares='(2050-10950]' 3944 |
| 2.03 | 0.83 | self\_reference\_avg\_sharess='(3195.75-6998]' 7806 ==> self\_reference\_max\_shares='(2050-10950]' 6472 |
| 1.63 | 0.82 | kw\_avg\_avg='(2887.85692-3512.807243]' shares=Popular 5083 ==> kw\_max\_avg='(3752.607692-7547.980991]' 4144 |
| 1.6 | 0.8 | kw\_avg\_avg='(2887.85692-3512.807243]' 8582 ==> kw\_max\_avg='(3752.607692-7547.980991]' 6849 |
| 4.47 | 0.79 | self\_reference\_max\_shares='(1250-2050]' 5060 ==> self\_reference\_avg\_sharess='(1098.1-1896.166667]' 4016 |
| 1.8 | 0.78 | kw\_avg\_avg='(-inf-2531.06838]' shares=Popular 5221 ==> kw\_min\_avg='(-inf-44]' 4079 |
| 1.67 | 0.76 | kw\_max\_avg='(3493.984775-3752.607692]' 5370 ==> timedelta='(133.5-468.5]' 4097 |
| 2.16 | 0.71 | kw\_max\_avg='(3237.393984-3493.984775]' 6647 ==> kw\_avg\_avg='(-inf-2531.06838]' 4746 |
| 1.51 | 0.71 | self\_reference\_max\_shares='(-inf-354.5]' 6035 ==> number\_of\_imgs=1 4280 |
| 1.51 | 0.71 | self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_avg\_sharess='(-inf-368.5]' 6035 ==> number\_of\_imgs=1 4280 |
| 6.35 | 0.71 | self\_reference\_max\_shares='(-inf-354.5]' 6035 ==> number\_of\_imgs=1 self\_reference\_avg\_sharess='(-inf-368.5]' 4280 |
| 1.51 | 0.71 | self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_min\_shares='(-inf-369.5]' 6035 ==> number\_of\_imgs=1 4280 |
| 6.12 | 0.71 | self\_reference\_max\_shares='(-inf-354.5]' 6035 ==> number\_of\_imgs=1 self\_reference\_min\_shares='(-inf-369.5]' 4280 |
| 1.51 | 0.71 | self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_avg\_sharess='(-inf-368.5]' self\_reference\_min\_shares='(-inf-369.5]' 6035 ==> number\_of\_imgs=1 4280 |
| 6.35 | 0.71 | self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_min\_shares='(-inf-369.5]' 6035 ==> number\_of\_imgs=1 self\_reference\_avg\_sharess='(-inf-368.5]' 4280 |
| 6.12 | 0.71 | self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_avg\_sharess='(-inf-368.5]' 6035 ==> number\_of\_imgs=1 self\_reference\_min\_shares='(-inf-369.5]' 4280 |
| 6.35 | 0.71 | self\_reference\_max\_shares='(-inf-354.5]' 6035 ==> number\_of\_imgs=1 self\_reference\_avg\_sharess='(-inf-368.5]' self\_reference\_min\_shares='(-inf-369.5]' 4280 |
| 1.51 | 0.71 | self\_reference\_avg\_sharess='(-inf-368.5]' 6038 ==> number\_of\_imgs=1 4282 |
| 1.51 | 0.71 | self\_reference\_avg\_sharess='(-inf-368.5]' self\_reference\_min\_shares='(-inf-369.5]' 6038 ==> number\_of\_imgs=1 4282 |
| 6.12 | 0.71 | self\_reference\_avg\_sharess='(-inf-368.5]' 6038 ==> number\_of\_imgs=1 self\_reference\_min\_shares='(-inf-369.5]' 4282 |
| 6.35 | 0.71 | self\_reference\_avg\_sharess='(-inf-368.5]' 6038 ==> number\_of\_imgs=1 self\_reference\_max\_shares='(-inf-354.5]' 4280 |
| 6.35 | 0.71 | self\_reference\_avg\_sharess='(-inf-368.5]' self\_reference\_min\_shares='(-inf-369.5]' 6038 ==> number\_of\_imgs=1 self\_reference\_max\_shares='(-inf-354.5]' 4280 |
| 6.35 | 0.71 | self\_reference\_avg\_sharess='(-inf-368.5]' 6038 ==> number\_of\_imgs=1 self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_min\_shares='(-inf-369.5]' 4280 |
| 1.63 | 0.71 | kw\_avg\_avg='(-inf-2531.06838]' 12700 ==> kw\_min\_avg='(-inf-44]' 8989 |
| 1.49 | 0.7 | data\_channel=3 6175 ==> number\_of\_imgs=1 4322 |
| 1.47 | 0.69 | self\_reference\_min\_shares='(-inf-369.5]' 6441 ==> number\_of\_imgs=1 4444 |
| 1.58 | 0.69 | kw\_avg\_avg='(-inf-2531.06838]' number\_of\_imgs=1 7571 ==> kw\_min\_avg='(-inf-44]' 5206 |
| 1.28 | 0.68 | kw\_min\_avg='(2232.165858-inf)' 7200 ==> shares=Popular 4903 |
| 5.96 | 0.66 | self\_reference\_min\_shares='(-inf-369.5]' 6441 ==> number\_of\_imgs=1 self\_reference\_avg\_sharess='(-inf-368.5]' 4282 |
| 5.96 | 0.66 | self\_reference\_min\_shares='(-inf-369.5]' 6441 ==> number\_of\_imgs=1 self\_reference\_max\_shares='(-inf-354.5]' 4280 |
| 5.96 | 0.66 | self\_reference\_min\_shares='(-inf-369.5]' 6441 ==> number\_of\_imgs=1 self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_avg\_sharess='(-inf-368.5]' 4280 |
| 1.25 | 0.66 | self\_reference\_min\_shares='(2750-inf)' 9454 ==> shares=Popular 6267 <conf:(0.66)> lift:(1.25) lev:(0.03) [1238] conv:(1.39) |
| 1.51 | 0.66 | kw\_avg\_avg='(-inf-2531.06838]' shares=Not Popular 7479 ==> kw\_min\_avg='(-inf-44]' 4910 <conf:(0.66)> lift:(1.51) lev:(0.04) [1656] conv:(1.64) |
| 1.22 | 0.65 | kw\_avg\_avg='(3512.807243-4961.099444]' 7599 ==> shares=Popular 4929 |
| 1.29 | 0.64 | kw\_avg\_avg='(3512.807243-4961.099444]' 7599 ==> kw\_max\_avg='(3752.607692-7547.980991]' 4889 |
| 1.2 | 0.64 | data\_channel=5 7323 ==> shares=Popular 4690 |
| 4.45 | 0.64 | data\_channel=6 8159 ==> LDA\_02='(0.628306-inf)' 5223 |
| 1.2 | 0.64 | self\_reference\_max\_shares='(10950-57150]' 6420 ==> shares=Popular 4092 |
| 1.91 | 0.63 | kw\_min\_avg='(-inf-44]' number\_of\_imgs=1 8220 ==> kw\_avg\_avg='(-inf-2531.06838]' 5206 |
| 1.35 | 0.63 | kw\_max\_avg='(3237.393984-3493.984775]' 6647 ==> shares=Not Popular 4205 |
| 1.35 | 0.63 | kw\_max\_avg='(3237.393984-3493.984775]' 6647 ==> number\_of\_imgs=1 4199 |
| 2.48 | 0.63 | LDA\_02='(0.033334-0.050049]' 7429 ==> LDA\_04='(0.033334-0.106577]' 4654 |
| 1.89 | 0.63 | kw\_min\_avg='(-inf-44]' shares=Not Popular 7845 ==> kw\_avg\_avg='(-inf-2531.06838]' 4910 |
| 1.33 | 0.62 | kw\_avg\_avg='(-inf-2531.06838]' shares=Not Popular 7479 ==> number\_of\_imgs=1 4667 |
| 1.32 | 0.62 | data\_channel=6 8159 ==> shares=Not Popular 5053 |
| 1.32 | 0.62 | kw\_avg\_avg='(-inf-2531.06838]' number\_of\_imgs=1 7571 ==> shares=Not Popular 4667 |
| 1.31 | 0.62 | LDA\_00='(0.201706-0.866666]' 9745 ==> number\_of\_imgs=1 6003 |
| 1.15 | 0.61 | self\_reference\_max\_shares='(2050-10950]' self\_reference\_avg\_sharess='(3195.75-6998]' 6472 ==> shares=Popular 3944 |
| 1.14 | 0.61 | kw\_max\_avg='(3752.607692-7547.980991]' self\_reference\_max\_shares='(2050-10950]' 7899 ==> shares=Popular 4811 |
| 1.14 | 0.61 | LDA\_02='(-inf-0.028572]' 9691 ==> shares=Popular 5894 |
| 1.48 | 0.61 | self\_reference\_min\_shares='(2750-inf)' 9454 ==> self\_reference\_max\_shares='(2050-10950]' 5721 |
| 1.14 | 0.61 | kw\_avg\_avg='(2887.85692-3512.807243]' kw\_max\_avg='(3752.607692-7547.980991]' 6849 ==> shares=Popular 4144 |
| 1.14 | 0.6 | self\_reference\_avg\_sharess='(3195.75-6998]' 7806 ==> shares=Popular 4713 |
| 1.27 | 0.6 | kw\_avg\_avg='(-inf-2531.06838]' 12700 ==> number\_of\_imgs=1 7571 |
| 1.79 | 0.59 | data\_channel=6 8159 ==> kw\_avg\_avg='(-inf-2531.06838]' 4841 |
| 1.18 | 0.59 | kw\_min\_avg='(2232.165858-inf)' 7200 ==> kw\_max\_avg='(3752.607692-7547.980991]' 4265 |
| 1.11 | 0.59 | kw\_avg\_avg='(2887.85692-3512.807243]' 8582 ==> shares=Popular 5083 |
| 1.26 | 0.59 | data\_channel=2 6849 ==> shares=Not Popular 4043 |
| 4.47 | 0.59 | self\_reference\_avg\_sharess='(1098.1-1896.166667]' 6808 ==> self\_reference\_max\_shares='(1250-2050]' 4016 |
| 1.26 | 0.59 | kw\_avg\_avg='(-inf-2531.06838]' 12700 ==> shares=Not Popular 7479 |
| 1.11 | 0.59 | kw\_max\_avg='(3752.607692-7547.980991]' timedelta='(133.5-468.5]' 8982 ==> shares=Popular 5284 |
| 1.24 | 0.58 | data\_channel=6 8159 ==> number\_of\_imgs=1 4752 |
| 1.16 | 0.58 | LDA\_00='(0.201706-0.866666]' 9745 ==> kw\_max\_avg='(3752.607692-7547.980991]' 5656 |
| 1.09 | 0.58 | kw\_max\_avg='(3752.607692-7547.980991]' 19193 ==> shares=Popular 11128 |
| 1.23 | 0.58 | kw\_avg\_avg='(-inf-2531.06838]' kw\_min\_avg='(-inf-44]' 8989 ==> number\_of\_imgs=1 5206 |
| 1.09 | 0.58 | LDA\_00='(0.201706-0.866666]' 9745 ==> shares=Popular 5642 |
| 1.16 | 0.58 | data\_channel=2 6849 ==> kw\_max\_avg='(3752.607692-7547.980991]' 3963 |
| 1.09 | 0.58 | kw\_max\_avg='(3752.607692-7547.980991]' kw\_min\_avg='(-inf-44]' 8240 ==> shares=Popular 4763 |
| 1.09 | 0.58 | self\_reference\_max\_shares='(2050-10950]' timedelta='(133.5-468.5]' 7446 ==> shares=Popular 4304 |
| 2.97 | 0.57 | self\_reference\_avg\_sharess='(1098.1-1896.166667]' 6808 ==> self\_reference\_min\_shares='(1050-1650]' 3863 |
| 1.07 | 0.57 | self\_reference\_max\_shares='(2050-10950]' 15683 ==> shares=Popular 8893 |
| 1.13 | 0.56 | LDA\_02='(-inf-0.028572]' 9691 ==> kw\_max\_avg='(3752.607692-7547.980991]' 5475 |
| 1.06 | 0.56 | kw\_max\_avg='(3752.607692-7547.980991]' number\_of\_imgs=1 8431 ==> shares=Popular 4747 |

- כדי לצמצם את מספר החוקים המתקבלים, העליתי את רמת התמיכה ל-0.2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lift | בטחון | חוקים מענייינים | מספר חוקים (חוקים שהתקבלו) | בטחון מינימום | תמיכת מינימום | אלגוריתם |
| 1.63 | 0.71 | kw\_avg\_avg='(-inf-2531.06838]' 12700 ==> kw\_min\_avg='(-inf-44]' 8989 | 100 (11) | 0.5 | 0.2 | Apriori |
| 1.09 | 0.58 | kw\_max\_avg='(3752.607692-7547.980991]' 19193 ==> shares=Popular 11128 |  |  |  |  |
| 1.07 | 0.57 | self\_reference\_max\_shares='(2050-10950]' 15683 ==> shares=Popular 8893 |  |  |  |  |
| 1.04 | 0.55 | timedelta='(133.5-468.5]' 17523 ==> shares=Popular 9651 |  |  |  |  |
| 1.09 | 0.55 | shares=Popular 20409 ==> kw\_max\_avg='(3752.607692-7547.980991]' 11128 |  |  |  |  |
| 1.63 | 0.54 | kw\_min\_avg='(-inf-44]' 16691 ==> kw\_avg\_avg='(-inf-2531.06838]' 8989 |  |  |  |  |
| 1 | 0.53 | kw\_min\_avg='(-inf-44]' 16691 ==> shares=Popular 8846 |  |  |  |  |
| 1.1 | 0.52 | shares=Not Popular 17958 ==> number\_of\_imgs=1 9286 |  |  |  |  |
| 1.1 | 0.52 | number\_of\_imgs=1 18004 ==> shares=Not Popular 9286 |  |  |  |  |
| 1.02 | 0.51 | timedelta='(133.5-468.5]' 17523 ==> kw\_max\_avg='(3752.607692-7547.980991]' 8982 |  |  |  |  |
| 1.01 | 0.5 | self\_reference\_max\_shares='(2050-10950]' 15683 ==> kw\_max\_avg='(3752.607692-7547.980991]' 7899 |  |  |  |  |

.Lift- העמודה השמאלית מציגה את מדד ה-

משמעות מדד זה היא מידת הקורלציה של המאורעות. אם הוא קטן מ-1 – הקורלציה שלילית; אם הוא גדול מ-1 – הקורלציה חיובית; אם הוא שווה ל-1 – אין קורלציה בין האירועים – כלומר הם בלתי תלויים אחד בשני. ככל שהניקוד גבוה יותר תועלת החוק גבוהה יותר.

.FPGrowth ג. בחרתי שני אלגוריתמים: אפריורי ו-

ג.1. **אלגוריתם אפריורי**

האלגוריתם מסתמך על כך שבמידה וקיים סט נתונים שכיח בבסיס הנתונים אז כל קבוצה חלקית של אותה קבוצת פריטים שכיחה חייבת גם היא להיות שכיחה.

זהו אלגוריתם איטרטיבי, כאשר במחזור k מוצאים את כל קבוצות הפריטים השכיחות בגודל k. בכל מחזור באלגוריתם מתבצעים שלושה מעברים עיקריים:

1. יצירת מועמדים. מציאת מועמדים בגודל k על בסיס קבוצות הפריטים השכיחות שנמצאו בשלב k – 1. בשלב זה מתבצע צירוף של שתי קבוצות שנמצאו בשלב קודם, לצורך מציאת קבוצה חדשה.
2. גיזום מועמדים שלא מקיימים את התכונה האפריורית, כלומר מתעלמים מהצירופים שאינם שכיחים.
3. בדיקת התמיכה של קבוצות המועמדים שלא נגזמו בשלב 2 ע"י מניית מספר התנועות שבהן הם מופיעים בתוך בסיס הנתונים עצמו, כלומר בדיקה האם הקבוצות המועמדות באמת שכיחות.

ג.2.1. **אלגוריתם FPGrowth**

האלגוריתם משתמש בשיטת הפרד ומשול.

א. בתחילה הוא דוחס את המידע שבבסיס הנתונים על-ידי יצירת עץ תבניות שכיחות.

ב. אחר כך הוא מפריד את המידע בבסיס הנתונים לבסיסי נתונים מותנים, כל אחד מקושר לתבנית שנמצאה בשלב הראשון.

ג. בשלב בשלישי הוא מבצע כרייה נפרדת, לעומק, עבור כל בסיס נתונים שנוצר.

שלבי האלגוריתם בפסודו-קוד[[1]](#footnote-1):

(01) if Tree contains a single prefix path then { // Mining single prefix-path FP-tree

(02) let P be the single prefix-path part of Tree;

(03) let Q be the multipath part with the top branching node replaced by a null root;

(04) for each combination (denoted as ß) of the nodes in the path P do

(05) generate pattern ß ∪ a with support = minimum support of nodes in ß;

(06) let freq pattern set(P) be the set of patterns so generated;

}

(07) else let Q be Tree;

(08) for each item ai in Q do { // Mining multipath FP-tree

(09) generate pattern ß = ai ∪ a with support = ai .support;

(10) construct ß’s conditional pattern-base and then ß’s conditional FP-tree Tree ß;

(11) if Tree ß ≠ Ø then

(12) call FP-growth(Tree ß , ß);

(13) let freq pattern set(Q) be the set of patterns so generated;

}

(14) return(freq pattern set(P) ∪ freq pattern set(Q) ∪ (freq pattern set(P) × freq pattern set(Q)))

}

**Apriori**ד.1.

תוצאות הרצת האלגוריתם האפריורי מופיעות בסעיף א+ב.

**FPGrowth**ד.2.

לצורך הרצת האלגוריתם יש להמיר את התכונות בבסיס הנתונים לתכונות בינריות – התקבלו 106 תכונות (מתוך 16 התכונות המקוריות).

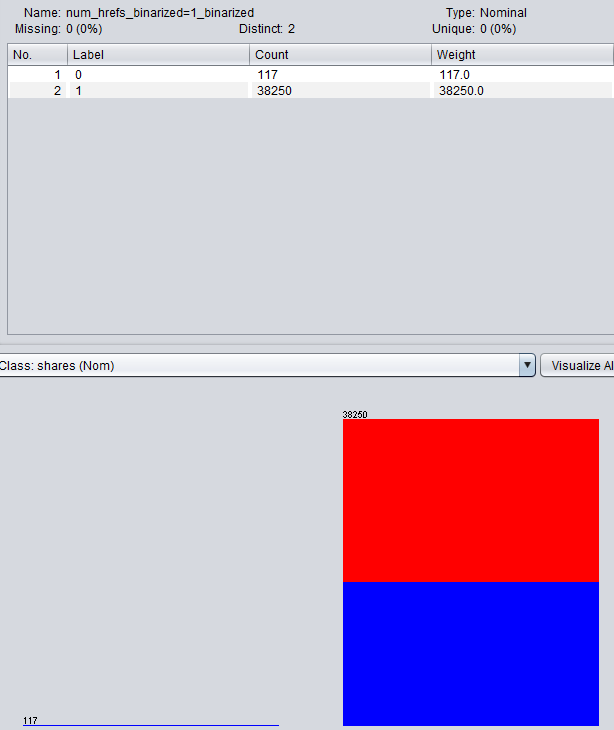
עבור רמת תמיכה 0.1 ו-100 חוקים, התקבלו אך ורק חוקים בעלי רמת ביטחון 1.

העליתי את רמת התמיכה ל-0.2. התקבלו 79 חוקים, 56 מתוכם בעלי רמת ביטחון 1.

הורדתי בהדרגה מספר תכונות, מכיוון שהן היו חד צדדיות ולכן דומיננטיות מדי ביצירת החוקים.

num\_hrefs\_binarized=1\_binarized=1לדוגמא, כפי שניתן לראות בצילומי המסך למטה, התכונה

הייתה חד צדדית מדי, ורובם המוחלט של החוקים נסמך עליה.





אחרי הסרת התכונה, קיבלתי חוקים עם תכונות דומיננטית נוספות:

self\_reference\_min\_shares='(-inf-369.5]'\_binarized=1

number\_of\_imgs\_binarized=1\_binarized=1

אחרי הורדת התכונות הנוספות האלה נשארו 103 תכונות.

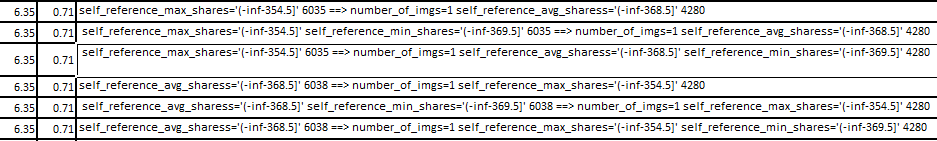
תוצאות ההרצה:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lift | בטחון | חוקים מענייינים | מספר חוקים (חוקים שהתקבלו) | בטחון מינימום | תמיכת מינימום | אלגוריתם |
| 5.96 | 0.94 | [self\_reference\_min\_shares='(-inf-369.5]'\_binarized=1]: 6441 ==> [self\_reference\_avg\_sharess='(-inf-368.5]'\_binarized=1]: 6038 | 100 (73) מתוכם 64 חוקים מעניינים (רמת ביטחון נמוכה מ-1) | 0.5 | 0.1 | FPGrowth |
| 5.96 | 0.94 | [self\_reference\_min\_shares='(-inf-369.5]'\_binarized=1]: 6441 ==> [self\_reference\_max\_shares='(-inf-354.5]'\_binarized=1]: 6035 |
| 5.96 | 0.94 | [self\_reference\_min\_shares='(-inf-369.5]'\_binarized=1]: 6441 ==> [self\_reference\_avg\_sharess='(-inf-368.5]'\_binarized=1, self\_reference\_max\_shares='(-inf-354.5]'\_binarized=1]: 6035 |
| 2.18 | 0.89 | [self\_reference\_avg\_sharess='(1896.166667-3195.75]'\_binarized=1]: 6913 ==> [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 6174 |
| 2.05 | 0.84 | [shares=Popular, self\_reference\_avg\_sharess='(3195.75-6998]'\_binarized=1]: 4713 ==> [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 3944 |
| 2.03 | 0.83 | [self\_reference\_avg\_sharess='(3195.75-6998]'\_binarized=1]: 7806 ==> [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 6472 |
| 1.63 | 0.82 | [shares=Popular, kw\_avg\_avg='(2887.85692-3512.807243]'\_binarized=1]: 5083 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 4144 |
| 1.6 | 0.8 | [kw\_avg\_avg='(2887.85692-3512.807243]'\_binarized=1]: 8582 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 6849 |
| 4.47 | 0.79 | [self\_reference\_max\_shares='(1250-2050]'\_binarized=1]: 5060 ==> [self\_reference\_avg\_sharess='(1098.1-1896.166667]'\_binarized=1]: 4016 |
| 1.8 | 0.78 | [shares=Popular, kw\_avg\_avg='(-inf-2531.06838]'\_binarized=1]: 5221 ==> [kw\_min\_avg='(-inf-44]'\_binarized=1]: 4079 |
| 1.67 | 0.76 | [kw\_max\_avg='(3493.984775-3752.607692]'\_binarized=1]: 5370 ==> [timedelta='(133.5-468.5]'\_binarized=1]: 4097 |
| 2.16 | 0.71 | [kw\_max\_avg='(3237.393984-3493.984775]'\_binarized=1]: 6647 ==> [kw\_avg\_avg='(-inf-2531.06838]'\_binarized=1]: 4746 |
| 1.63 | 0.71 | [kw\_avg\_avg='(-inf-2531.06838]'\_binarized=1]: 12700 ==> [kw\_min\_avg='(-inf-44]'\_binarized=1]: 8989 |
| 1.28 | 0.68 | [kw\_min\_avg='(2232.165858-inf)'\_binarized=1]: 7200 ==> [shares=Popular]: 4903 |
| 1.25 | 0.66 | [self\_reference\_min\_shares='(2750-inf)'\_binarized=1]: 9454 ==> [shares=Popular]: 6267 |
| 1.22 | 0.65 | [kw\_avg\_avg='(3512.807243-4961.099444]'\_binarized=1]: 7599 ==> [shares=Popular]: 4929 |
| 1.29 | 0.64 | [kw\_avg\_avg='(3512.807243-4961.099444]'\_binarized=1]: 7599 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 4889 |
| 1.2 | 0.64 | [self\_reference\_max\_shares='(10950-57150]'\_binarized=1]: 6420 ==> [shares=Popular]: 4092 |
| 2.48 | 0.63 | [LDA\_02='(0.033334-0.050049]'\_binarized=1]: 7429 ==> [LDA\_04='(0.033334-0.106577]'\_binarized=1]: 4654 |
| 1.15 | 0.61 | [self\_reference\_max\_shares='(2050-10950]'\_binarized=1, self\_reference\_avg\_sharess='(3195.75-6998]'\_binarized=1]: 6472 ==> [shares=Popular]: 3944 |
| 1.14 | 0.61 | [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1, self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 7899 ==> [shares=Popular]: 4811 |
| 1.14 | 0.61 | [LDA\_02='(-inf-0.028572]'\_binarized=1]: 9691 ==> [shares=Popular]: 5894 |
| 1.48 | 0.61 | [self\_reference\_min\_shares='(2750-inf)'\_binarized=1]: 9454 ==> [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 5721 |
| 1.14 | 0.61 | [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1, kw\_avg\_avg='(2887.85692-3512.807243]'\_binarized=1]: 6849 ==> [shares=Popular]: 4144 |
| 1.14 | 0.6 | [self\_reference\_avg\_sharess='(3195.75-6998]'\_binarized=1]: 7806 ==> [shares=Popular]: 4713 |
| 1.18 | 0.59 | [kw\_min\_avg='(2232.165858-inf)'\_binarized=1]: 7200 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 4265 |
| 1.11 | 0.59 | [kw\_avg\_avg='(2887.85692-3512.807243]'\_binarized=1]: 8582 ==> [shares=Popular]: 5083 |
| 4.47 | 0.59 | [self\_reference\_avg\_sharess='(1098.1-1896.166667]'\_binarized=1]: 6808 ==> [self\_reference\_max\_shares='(1250-2050]'\_binarized=1]: 4016 |
| 1.11 | 0.59 | [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1, timedelta='(133.5-468.5]'\_binarized=1]: 8982 ==> [shares=Popular]: 5284 |
| 1.16 | 0.58 | [LDA\_00='(0.201706-0.866666]'\_binarized=1]: 9745 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 5656 |
| 1.09 | 0.58 | [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 19193 ==> [shares=Popular]: 11128 |
| 1.09 | 0.58 | [LDA\_00='(0.201706-0.866666]'\_binarized=1]: 9745 ==> [shares=Popular]: 5642 |
| 1.09 | 0.58 | [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1, kw\_min\_avg='(-inf-44]'\_binarized=1]: 8240 ==> [shares=Popular]: 4763 |
| 1.09 | 0.58 | [timedelta='(133.5-468.5]'\_binarized=1, self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 7446 ==> [shares=Popular]: 4304 |
| 2.97 | 0.57 | [self\_reference\_avg\_sharess='(1098.1-1896.166667]'\_binarized=1]: 6808 ==> [self\_reference\_min\_shares='(1050-1650]'\_binarized=1]: 3863 |
| 1.07 | 0.57 | [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 15683 ==> [shares=Popular]: 8893 |
| 1.13 | 0.56 | [LDA\_02='(-inf-0.028572]'\_binarized=1]: 9691 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 5475 |
| 1.12 | 0.56 | [LDA\_04='(-inf-0.028572]'\_binarized=1]: 8529 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 4799 |
| 1.04 | 0.55 | [timedelta='(133.5-468.5]'\_binarized=1]: 17523 ==> [shares=Popular]: 9651 |
| 1.09 | 0.55 | [shares=Popular, timedelta='(133.5-468.5]'\_binarized=1]: 9651 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 5284 |
| 1.09 | 0.55 | [shares=Popular]: 20409 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 11128 |
| 1.09 | 0.54 | [kw\_avg\_avg='(2531.06838-2887.85692]'\_binarized=1]: 7170 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 3906 |
| 1.19 | 0.54 | [kw\_avg\_avg='(3512.807243-4961.099444]'\_binarized=1]: 7599 ==> [timedelta='(133.5-468.5]'\_binarized=1]: 4119 |
| 1.08 | 0.54 | [shares=Popular, self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 8893 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 4811 |
| 1.08 | 0.54 | [self\_reference\_avg\_sharess='(3195.75-6998]'\_binarized=1]: 7806 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 4216 |
| 1.63 | 0.54 | [kw\_min\_avg='(-inf-44]'\_binarized=1]: 16691 ==> [kw\_avg\_avg='(-inf-2531.06838]'\_binarized=1]: 8989 |
| 1.08 | 0.54 | [shares=Popular, kw\_min\_avg='(-inf-44]'\_binarized=1]: 8846 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 4763 |
| 1.08 | 0.54 | [self\_reference\_min\_shares='(2750-inf)'\_binarized=1]: 9454 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 5087 |
| 1.01 | 0.53 | [LDA\_02='(0.033334-0.050049]'\_binarized=1]: 7429 ==> [shares=Popular]: 3974 |
| 1.17 | 0.53 | [kw\_min\_avg='(2232.165858-inf)'\_binarized=1]: 7200 ==> [timedelta='(133.5-468.5]'\_binarized=1]: 3837 |
| 1 | 0.53 | [kw\_min\_avg='(-inf-44]'\_binarized=1]: 16691 ==> [shares=Popular]: 8846 |
| 0.99 | 0.53 | [LDA\_04='(-inf-0.028572]'\_binarized=1]: 8529 ==> [shares=Popular]: 4514 |
| 0.99 | 0.53 | [LDA\_01='(0.050001-0.483465]'\_binarized=1]: 8766 ==> [shares=Popular]: 4628 |
| 2.09 | 0.53 | [LDA\_04='(-inf-0.028572]'\_binarized=1]: 8529 ==> [LDA\_02='(-inf-0.028572]'\_binarized=1]: 4495 |
| 2.97 | 0.53 | [self\_reference\_min\_shares='(1050-1650]'\_binarized=1]: 7330 ==> [self\_reference\_avg\_sharess='(1098.1-1896.166667]'\_binarized=1]: 3863 |
| 1.05 | 0.53 | [LDA\_02='(0.033334-0.050049]'\_binarized=1]: 7429 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 3912 |
| 0.97 | 0.51 | [kw\_min\_avg='(1399.695833-2232.165858]'\_binarized=1]: 8360 ==> [shares=Popular]: 4304 |
| 1.13 | 0.51 | [LDA\_04='(-inf-0.028572]'\_binarized=1]: 8529 ==> [timedelta='(133.5-468.5]'\_binarized=1]: 4384 |
| 1.18 | 0.51 | [LDA\_04='(-inf-0.028572]'\_binarized=1]: 8529 ==> [kw\_min\_avg='(-inf-44]'\_binarized=1]: 4382 |
| 1.02 | 0.51 | [timedelta='(133.5-468.5]'\_binarized=1]: 17523 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 8982 |
| 1.11 | 0.51 | [self\_reference\_min\_shares='(2750-inf)'\_binarized=1]: 9454 ==> [timedelta='(133.5-468.5]'\_binarized=1]: 4802 |
| 1.16 | 0.51 | [LDA\_02='(-inf-0.028572]'\_binarized=1]: 9691 ==> [kw\_min\_avg='(-inf-44]'\_binarized=1]: 4905 |
| 2.18 | 0.51 | [self\_reference\_avg\_sharess='(3195.75-6998]'\_binarized=1]: 7806 ==> [shares=Popular, self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 3944 |
| 1.01 | 0.5 | [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 15683 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 7899 |
| 1.63 | 0.71 | [kw\_avg\_avg='(-inf-2531.06838]'\_binarized=1]: 12700 ==> [kw\_min\_avg='(-inf-44]'\_binarized=1]: 8989 | 100 (9) | 0.5 | 0.2 | FPGrowth |
| 1.09 | 0.58 | [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 19193 ==> [shares=Popular]: 11128 |
| 1.07 | 0.57 | [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 15683 ==> [shares=Popular]: 8893 |
| 1.04 | 0.55 | [timedelta='(133.5-468.5]'\_binarized=1]: 17523 ==> [shares=Popular]: 9651 |
| 1.09 | 0.55 | [shares=Popular]: 20409 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 11128 |
| 1.63 | 0.54 | [kw\_min\_avg='(-inf-44]'\_binarized=1]: 16691 ==> [kw\_avg\_avg='(-inf-2531.06838]'\_binarized=1]: 8989 |
| 1 | 0.53 | [kw\_min\_avg='(-inf-44]'\_binarized=1]: 16691 ==> [shares=Popular]: 8846 |
| 1.02 | 0.51 | [timedelta='(133.5-468.5]'\_binarized=1]: 17523 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 8982 |
| 1.01 | 0.5 | [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 15683 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 7899 |

ה. מדובר בשני אלגוריתמים דומים. ההבדלים העיקריים הם **דרך החיפוש**, כאשר אפריורי מונחה על-ידי יצירת מועמדים ואילו FPGrowth בחיפוש אחר תבניות שכיחות, **ואופן השימוש בתכונות** הדורש המרה לבינרי ב-FPGrowth. מעבר לכך, נצפה למצוא תוצאות דומות, כאשר המעניינות שבהן יהיו בעלי דירוג Lift גבוה.

ו. אלגוריתמים בעלי Lift גבוה מלמדים על העניין שבחוק ביחס לאוכלוסיה הכוללת. לכן מיינתי כל הרצה לפי מדד ה-Lift שלה. להלן 10 התוצאות הגבוהות ביותר עבור כל הרצה (אם ישנן), אחרי שסיננתי ידנית חוקים מאוד דומים, שהתקבלו באותה הרצה.

\*דוגמא לחוקים מאוד דומים:



ניתן לראות כי אלו חוקים בעלי בטחון וקורלציה זהים, וכי התכונות המופיעות בהם הן בעלות אופי משותף. במקרה זה, כל התכונות סובבות סביב מידה כזו או אחרת של שיתופים.

התוצאות:

אלגוריתם אפריורי, תמיכת מינימום של 0.1 ובטחון מינימום של 0.5:

|  |  |  |
| --- | --- | --- |
| Lift | Con | חוקים מענייינים |
| 6.35 | 0.71 | self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_min\_shares='(-inf-369.5]' 6035 ==> number\_of\_imgs=1 self\_reference\_avg\_sharess='(-inf-368.5]' 4280 |
| 6.12 | 0.96 | number\_of\_imgs=1 self\_reference\_min\_shares='(-inf-369.5]' 4444 ==> self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_avg\_sharess='(-inf-368.5]' 4280 |
| 5.96 | 0.94 | self\_reference\_min\_shares='(-inf-369.5]' 6441 ==> self\_reference\_max\_shares='(-inf-354.5]' self\_reference\_avg\_sharess='(-inf-368.5]' 6035 |
| 4.47 | 0.79 | self\_reference\_max\_shares='(1250-2050]' 5060 ==> self\_reference\_avg\_sharess='(1098.1-1896.166667]' 4016 |
| 4.45 | 0.95 | LDA\_02='(0.628306-inf)' 5517 ==> data\_channel=6 5223 |
| 3.34 | 0.85 | data\_channel=3 6175 ==> LDA\_00='(0.201706-0.866666]' 5234 |
| 2.97 | 0.57 | self\_reference\_avg\_sharess='(1098.1-1896.166667]' 6808 ==> self\_reference\_min\_shares='(1050-1650]' 3863 |
| 2.48 | 0.63 | LDA\_02='(0.033334-0.050049]' 7429 ==> LDA\_04='(0.033334-0.106577]' 4654 |
| 2.18 | 0.89 | self\_reference\_avg\_sharess='(1896.166667-3195.75]' 6913 ==> self\_reference\_max\_shares='(2050-10950]' 6174 |
| 2.16 | 0.71 | kw\_max\_avg='(3237.393984-3493.984775]' 6647 ==> kw\_avg\_avg='(-inf-2531.06838]' 4746 |

אלגוריתם אפריורי, תמיכת מינימום של 0.2 ובטחון מינימום של 0.5:

|  |  |  |
| --- | --- | --- |
| Lift | Con | חוקים מענייינים |
| 1.63 | 0.71 | kw\_avg\_avg='(-inf-2531.06838]' 12700 ==> kw\_min\_avg='(-inf-44]' 8989 |
| 1.1 | 0.52 | number\_of\_imgs=1 18004 ==> shares=Not Popular 9286 |
| 1.09 | 0.58 | kw\_max\_avg='(3752.607692-7547.980991]' 19193 ==> shares=Popular 11128 |
| 1.07 | 0.57 | self\_reference\_max\_shares='(2050-10950]' 15683 ==> shares=Popular 8893 |
| 1.04 | 0.55 | timedelta='(133.5-468.5]' 17523 ==> shares=Popular 9651 |
| 1.02 | 0.51 | timedelta='(133.5-468.5]' 17523 ==> kw\_max\_avg='(3752.607692-7547.980991]' 8982 |
| 1.01 | 0.5 | self\_reference\_max\_shares='(2050-10950]' 15683 ==> kw\_max\_avg='(3752.607692-7547.980991]' 7899 |
| 1 | 0.53 | kw\_min\_avg='(-inf-44]' 16691 ==> shares=Popular 8846 |

אלגוריתם FPGrowth, תמיכת מינימום של 0.1 ובטחון מינימום של 0.5:

|  |  |  |
| --- | --- | --- |
| Lift | Con | חוקים מענייינים |
| 5.96 | 0.94 | [self\_reference\_min\_shares='(-inf-369.5]'\_binarized=1]: 6441 ==> [self\_reference\_avg\_sharess='(-inf-368.5]'\_binarized=1, self\_reference\_max\_shares='(-inf-354.5]'\_binarized=1]: 6035 |
| 4.47 | 0.79 | [self\_reference\_max\_shares='(1250-2050]'\_binarized=1]: 5060 ==> [self\_reference\_avg\_sharess='(1098.1-1896.166667]'\_binarized=1]: 4016 |
| 2.97 | 0.57 | [self\_reference\_avg\_sharess='(1098.1-1896.166667]'\_binarized=1]: 6808 ==> [self\_reference\_min\_shares='(1050-1650]'\_binarized=1]: 3863 |
| 2.48 | 0.63 | [LDA\_02='(0.033334-0.050049]'\_binarized=1]: 7429 ==> [LDA\_04='(0.033334-0.106577]'\_binarized=1]: 4654 |
| 2.18 | 0.51 | [self\_reference\_avg\_sharess='(3195.75-6998]'\_binarized=1]: 7806 ==> [shares=Popular, self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 3944 |
| 2.16 | 0.71 | [kw\_max\_avg='(3237.393984-3493.984775]'\_binarized=1]: 6647 ==> [kw\_avg\_avg='(-inf-2531.06838]'\_binarized=1]: 4746 |
| 2.09 | 0.53 | [LDA\_04='(-inf-0.028572]'\_binarized=1]: 8529 ==> [LDA\_02='(-inf-0.028572]'\_binarized=1]: 4495 |
| 2.05 | 0.84 | [shares=Popular, self\_reference\_avg\_sharess='(3195.75-6998]'\_binarized=1]: 4713 ==> [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 3944 |
| 2.03 | 0.83 | [self\_reference\_avg\_sharess='(3195.75-6998]'\_binarized=1]: 7806 ==> [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 6472 |
| 1.8 | 0.78 | [shares=Popular, kw\_avg\_avg='(-inf-2531.06838]'\_binarized=1]: 5221 ==> [kw\_min\_avg='(-inf-44]'\_binarized=1]: 4079 |

אלגוריתם FPGrowth, תמיכת מינימום של 0.2 ובטחון מינימום של 0.5:

|  |  |  |
| --- | --- | --- |
| Lift | Con | חוקים מענייינים |
| 1.63 | 0.71 | [kw\_avg\_avg='(-inf-2531.06838]'\_binarized=1]: 12700 ==> [kw\_min\_avg='(-inf-44]'\_binarized=1]: 8989 |
| 1.09 | 0.58 | [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 19193 ==> [shares=Popular]: 11128 |
| 1.07 | 0.57 | [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 15683 ==> [shares=Popular]: 8893 |
| 1.04 | 0.55 | [timedelta='(133.5-468.5]'\_binarized=1]: 17523 ==> [shares=Popular]: 9651 |
| 1.02 | 0.51 | [timedelta='(133.5-468.5]'\_binarized=1]: 17523 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 8982 |
| 1.01 | 0.5 | [self\_reference\_max\_shares='(2050-10950]'\_binarized=1]: 15683 ==> [kw\_max\_avg='(3752.607692-7547.980991]'\_binarized=1]: 7899 |
| 1 | 0.53 | [kw\_min\_avg='(-inf-44]'\_binarized=1]: 16691 ==> [shares=Popular]: 8846 |

מהשוואה בין האלגוריתמים ניתן להגיע למספר מסקנות:

* האלגוריתמים מספקים תוצאות דומות, עבור תמיכות מינימום זהות. למשל, עבור תמיכה של 0.1, התכונה הדומיננטית בשני האלגוריתמים היא Self Reference (על הוריאציות השונות שלה); עבור תמיכה של 0.2, בשני האלגוריתמים ישנה דומיננטיות של תכונות מילות המפתח ושל מידת הפופולריות של השיתופים.
* עבור תמיכה גבוהה נקבל פחות תוצאות בעלות Lift גבוה. הדבר נובע מכך שהתכונה הדומיננטית Self Reference נתמכת פחות, עבור תמיכה מינימלית של 0.2.
* כתוצאה מהסעיף הקודם, בבחינה סובייקטיבית, נראה כי העלאת התמיכה מביאה לחוקי הקשר מעניינים יותר, אם כי בעלי קורלציה נמוכה יותר. כך, למשל, עבור תמיכה 0.2, ניתן למצוא חוקים רבים המתייחסים ישירות לתכונה המסווגת – Shares.
* הטענה המקובלת היא, כי עבור בסיסי נתונים קטנים יחסית, FPGrowth הינו אלגוריתם יעיל מ-Apriori. אך לא שמתי לב להבדלים משמעותיים בין האלגוריתמים, הן מבחינת זמן ההרצה והן מבחינת התוצאות.

2.א. ניתוח אשכולות נעשה כדי לאפשר הסקת מסקנות נוספת לנתונים שבידינו. מטרת הניתוח היא לחלק את האובייקטים השונים שבסיס הנתונים, לקבוצות שונות זו מזו, כאשר האובייקטים שבתוך כל קבוצה דומים זה לזה. בסיום הניתוח אמורות להתקבל קבוצות ברורות של נתונים, אשר מראות על מגמה מסוימת, שפעמים רבות לא הייתה נודעת בצורת ניתוח אחרת.

קיימים שני סוגי אישכול עיקריים: אישכול היררכי אשר מותאם לפרטים המועמדים לאישכול, ויוצר היררכיה בין פריט אחד לאוסף הפריטים הסופי; אישכול חלוקתי המגדיר מראש את מספר החלוקות הנדרש ומשתמש בדרך כלל בחישובי מרחק לשם החלוקה.

ב. לשם מדידת איכות האשכולות יש לבדוק את מידת הדמיון בין האובייקטים שבתוך האשכול ואת מידת חוסר-הדמיון שבין האשכולות. אישכול טוב ימדד גם בדפוסים החבויים שימצא, אשר לא היה ניתן לגלותם בדרך אחרת. לשם מדידת איכות האשכול ניתן להשתמש במטריקת דמיון ובפונקציית איכות, אשר מסתמכים לרוב על מדידת מרחקים. סכום השגיאות המרובעות המתקבל בסיום הרצה של כל אלגוריתם יראה על אשכול טוב ככל שהסכום נמוך יותר.

ג. קיימות שתי גישות עיקריות לאישכול: חלוקה והיררכיה.

באישכול חלוקה נעשה שימוש במדידות מרחק. תחילה בוחרים מספר נקודות, אקראיות או קבועות מראש, ובהמשך מצרפים אליהן את האובייקטים הקרובים ביותר. עבור כל איטרציה מחושב מרכז כל אשכול מחדש, בהתחשב באובייקטים החדשים שצורפו אליו, ולפיו נמדד המרחק עבור אובייקטים חדשים המועמדים להצטרף לאשכול.

האלגוריתם העיקרי שעושה שימוש בשיטת החלוקה הוא K-Means. היתרון באלגוריתם זה הוא חוסר ההתערבות הסובייקטיבי ביצירת אשכולות. האלגוריתם בוחר בעצמו את נקודות המרכז הראשוניות, בצורה אקראית. בצורה זו גם מובטח כי יתקבלו אשכולות, אפילו אם הנתונים ההתחלתיים אינם מפולגים בצורה ברורה. חסרונות האלגוריתם בכך שהוא ניתן לעבודה עם משתנים מספריים בלבד, ובכך שהתוצאה משתנה בין הרצה להרצה בהכרח, עקב נקודות ההתחלה השונות המתקבלות בתחילת כל הרצה. ניתן לראות גם את הצורך בבחירת מספר האשכולות על-ידי המשתמש כחסרון.

האלגוריתם:

* בחר מספר נקודות מתוך סט הנתונים, כמספר האשכולות שהוגדרו.
* חזור:
  + עבור כל אובייקט – הקצה אותו לאשכול בעל נקודת המרכז הכי קרובה אליוץ
  + חשב מחדש את נקודת המרכז של כל אשכול.
* הפסק כאשר אין שינוי.

באישכול היררכי נוצרים אשכולות מקוננים בצורה של עץ היררכי – הרמה העליונה היא סט הנתונים כולו, המהווה אשכול אחד גדול, והרמה התחתונה היא החלוקה הגדולה ביותר של האובייקטים בסט הנתונים, כאשר בכל אשכול מספר אובייקטים קטן יחסית.

בחרתי באלגוריתם ( EM (expectation maximization. הבחירה באלגוריתם זה נעשתה, בחלקה, עקב אילוצי ריצה וזמן. אלגוריתמים אחרים דוגמאת HierarchicalClusterer, Cobweb, DBSCAN, עפו בזמן הריצה, או נתקעו, או רצו למשך שעות ארוכות ללא תוצאה. גם EM רץ זמן רב, לכן החלטתי להריץ אותו ללא תכונות רבות, ולהשאיר רק את אלה, שכבר זוהו כבעלות משקל על-ידי K-Means. בנוסף, יש לציין כי התצוגה של האלגוריתם התקבלה בצורה גרפית בלבד ולא עצית.

EM הוא אלגוריתם היררכי-הסתברותי (Probabilsic Hierarchical), אגלומרטיבי-מצטבר (bottom-up). האלגוריתם מקצה חלוקה הסתברותית עבור כל משתנה, לגבי סיכוייו להכלל בכל אחד מן האשכולות. כך הוא מחליט אילו אובייקטים דומים מספיק כדי להכלל באותו אשכול, ומהי ההיררכיה שבין האשכולות. מספר האשכולות יכול להקבע על-ידי האלגוריתם, או על ידי המשתמש, מראש.

האלגוריתם משתמש במשתנה LogLikelihood, שמחשב את ההסתברות הממוצעת עבור כל סט הנתונים, בכל שלב בו נבדקים אובייקטים נוספים מהסט. עלייה בערך המשתנה גורמת ליצירת אשכול חדש. תקציר האלגוריתם כפי שהוא מובא בWEKA:

1. the number of clusters is set to 1.

2. the training set is split randomly into 10 folds.

3. EM is performed 10 times using the 10 folds the usual CV way.

4. the loglikelihood is averaged over all 10 results.

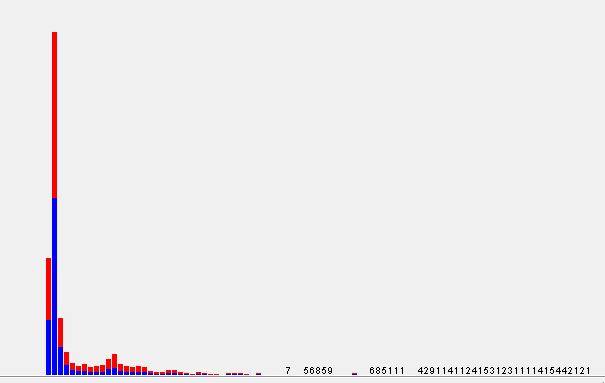
5. if loglikelihood has increased the number of clusters is increased by 1 and the program continues at step 2.

היתרון של אלגוריתם היררכי הסתברותי על-פני אלגוריתמים היררכיים אחרים, הוא בעיקר בכך שאלגוריתמים היררכיים מחפשים אפשרויות אשכול בצורה מקומית-נקודתית, עבור כל שלב בו הם נמצאים, ואילו היררכי-הסתברותי בוחן עצמו תמיד ביחס גלובלי (ההסתברות הכללית).

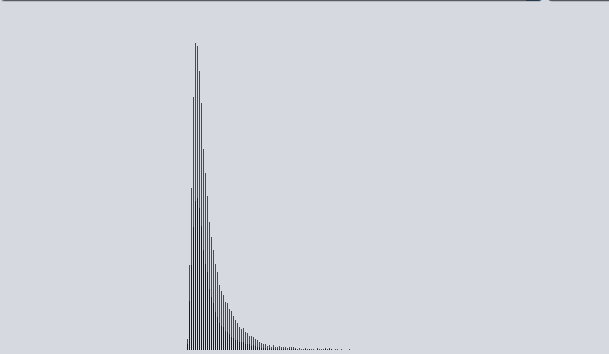
ד. K-Means

השתמשתי תחילה בכל 16 התכונות, כולל התכונה המסווגת. עבור כל מספר של אשכולות ודירוג (seed) אותם בחרתי, התקבל תמיד אשכול עם מספר אובייקטים גבוה בהרבה משאר האובייקטים. כך למשל, עבור 3 אשכולות התקבל אשכול אחד בו 48% מהאובייקטים.

כדי לנסות לאזן את האובייקטים, הורדתי בהדרגה 3 תכונות דומיננטיות: תחילה את shares בהיותה דומיננטית מעצם היותה התכונה המסווגת; שנייה את number\_of\_images שבדומה ליצירת חוקי ההקשר, ההתפלגות שלה מוטה מאוד ועלולה להשפיע על האלגוריתם החלוקתי. ניתן לראות זאת בקלות בגרף ההתפלגות שלה:



ולבסוף את num\_hrefs, מאותה סיבה:



לאחר הורדת התכונות האלו, אכן התקבלו אשכולות מאוזנים יותר, כאשר 3 האשכולות הגדולים ביותר נעים בין 25%-33% עבור פרמטר של 4 אשכולות ובין 37%-40% עבור פרמטר של 3 אשכולות.

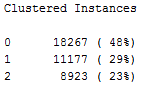
נראה שלפרמטר הדירוג ישנה השפעה רבה, הן על החלוקה בין האשכולות והן על סך השגיאות המרובעות (squared errors).

להלן הריצות (בריצה הראשונה הוספתי צילום של התכונות עצמן):

ריצה 1:

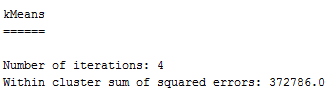
16 תכונות 3 אשכולות seed 10

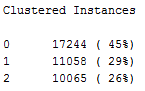




ריצה 2:

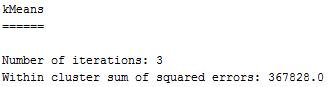
16 תכונות 3 אשכולות seed 13





ריצה 3:

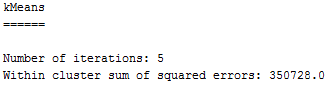
16 תכונות 4 אשכולות seed 10

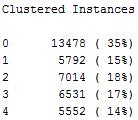




ריצה 4:

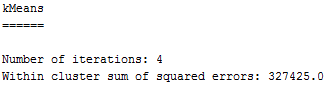
16 תכונות 5 אשכולות seed 83

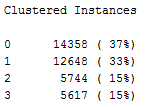




ריצה 5:

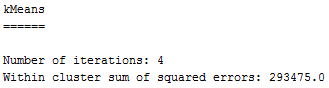
14 תכונות 4 אשכולות seed 12

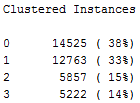




ריצה 6:

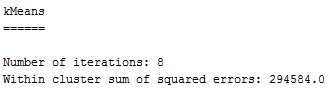
13 תכונות 4 אשכולות seed 12

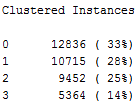




ריצה 7:

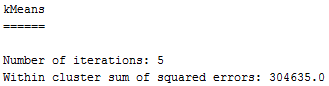
13 תכונות 4 אשכולות seed 17





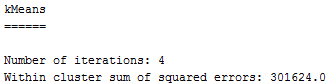
ריצה 8:

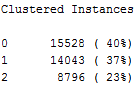
13 תכונות 3 אשכולות seed 14



ריצה 9:

13 תכונות 3 אשכולות seed 12





ניתן לראות כי ריצה 7 וריצה 9 הן המאוזנות ביותר עבור 4 ו-3 אשכולות, וכי ישנה השפעה של מספר התכונות ושל הדירוג על סכום השגיאות המרובעות.

EM – expectation maximisation

הרצתי את האלגוריתם 3 פעמים עבור פרמטרים שונים, כפי המופיע למטה. כאמור, נאלצתי לוותר על מספר תכונות, לשם הרצת האלגוריתם בזמן סביר. מכיוון שהרצתי את האלגוריתם ההיררכי אחרי האלגוריתם החלוקתי, בחרתי להשתמש בתכונות העיקריות שמצאתי באלגוריתם החלוקתי (סעיף ה'), ולמעשה להשתמש בו לשם וידוא התוצאות של האלגוריתם החלוקתי.

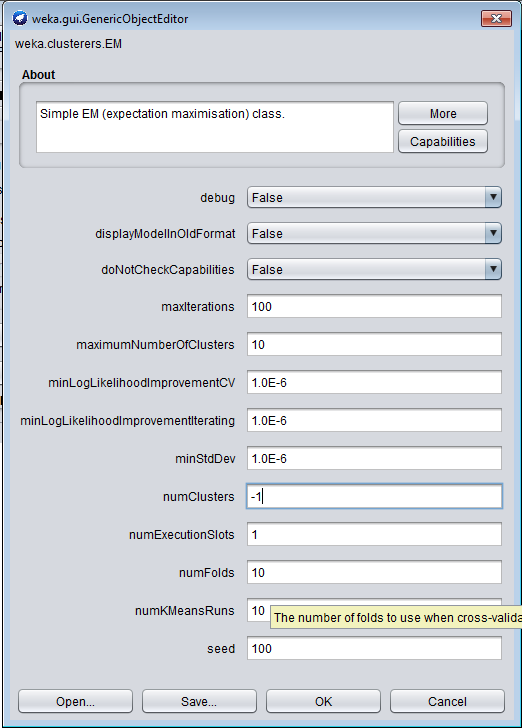
התכונות שהשארתי:

data\_channel, kw\_avg\_avg, LDA\_02, self\_reference\_avg\_sharess, weekday

התכונות מהן התעלמתי:

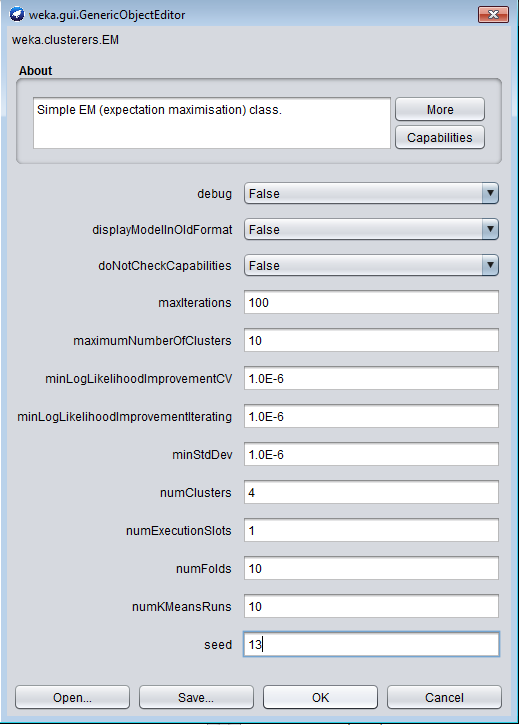
kw\_max\_avg, kw\_min\_avg, LDA\_00, LDA\_01, LDA\_04, num\_hrefs, number\_of\_imgs, self\_reference\_max\_shares, self\_reference\_min\_shares, timedelta, shares

ריצה 1:



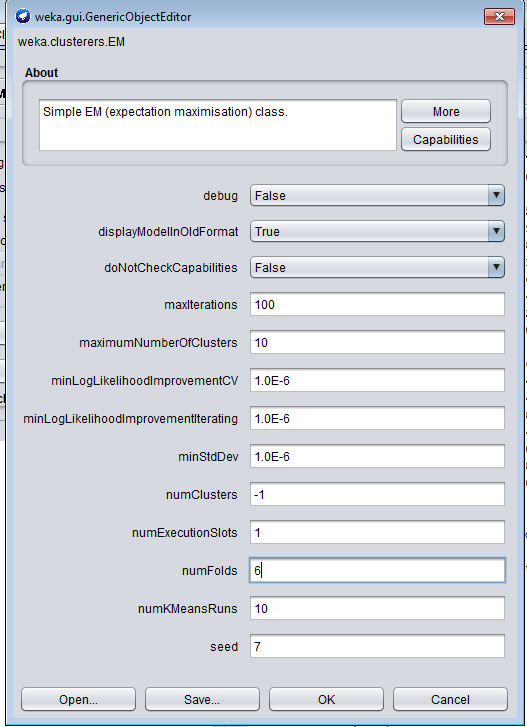
התקבלו 10 אשכולות.

ריצה 2:



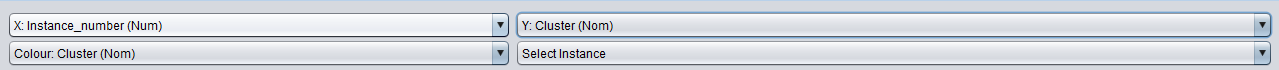
התקבלו 4 אשכולות.

ריצה 3:

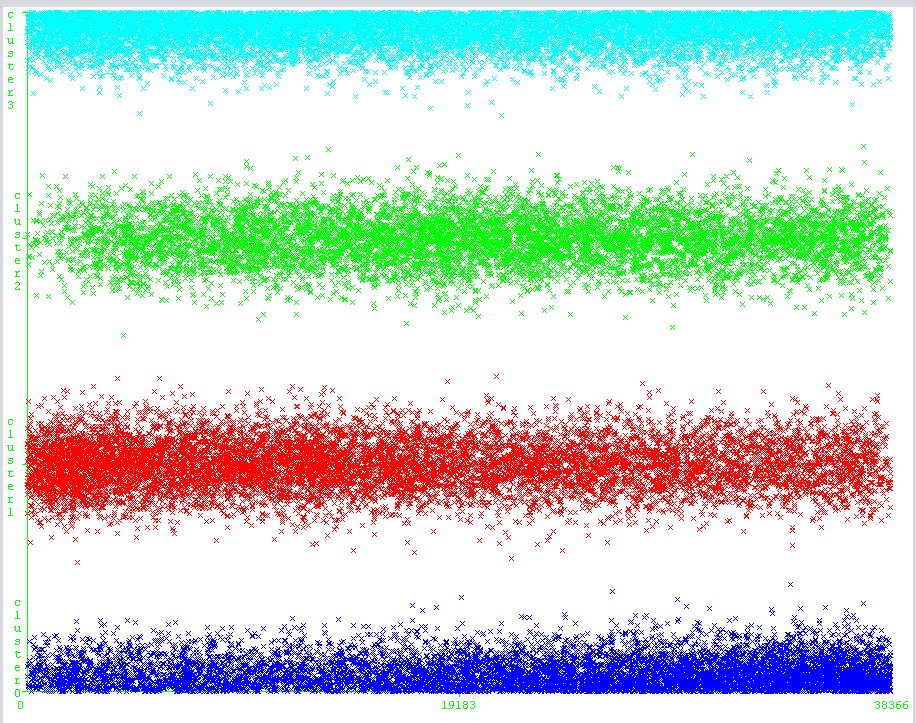


התקבלו 10 אשכולות.

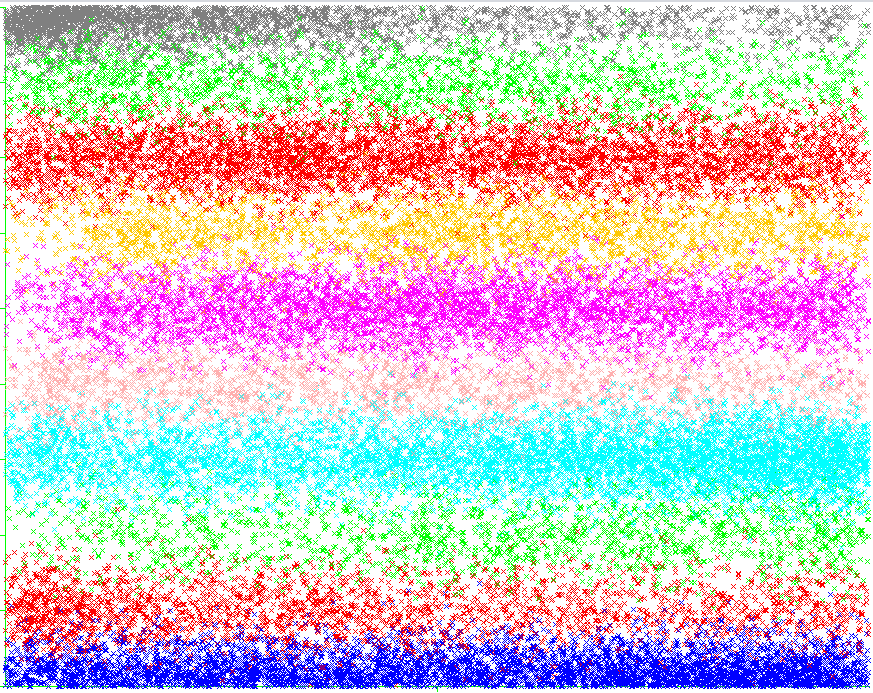
מבין 3 הריצות העדפתי לוותר על הריצה השנייה. בדקתי את החלוקה בין האשכולות על-ידי התבוננות בתצוגת משתנים-אשכולות:

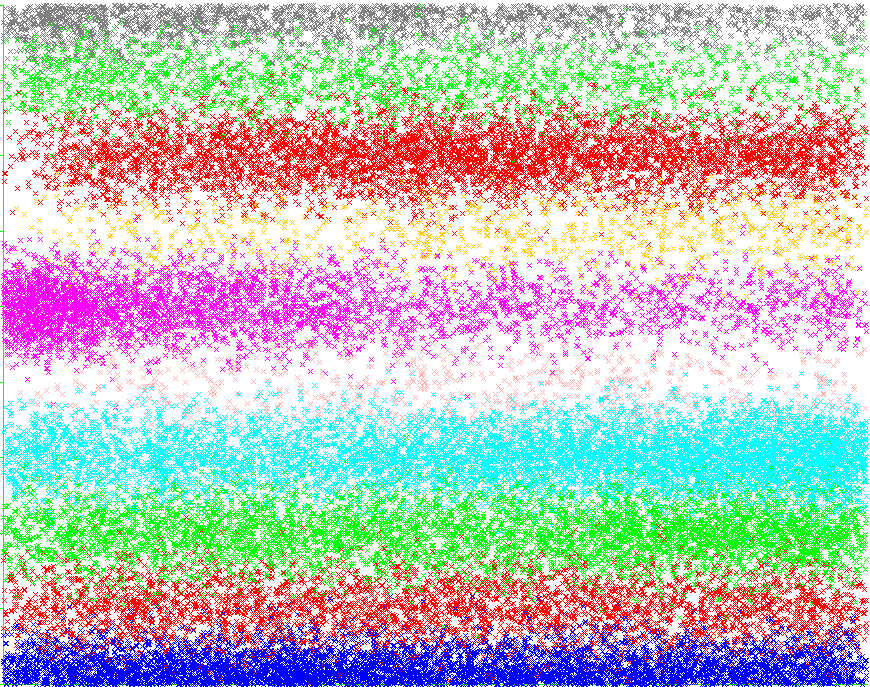


ניתן לראות כי האישכול ל-4 נותן אשכולות ברורים מדי, דבר המצביע על אישכול "מלאכותי" מדי וטשטוש הדמיון בין האובייקטים שבתוך כל אשכול.



לעומת זאת, בריצה הראשונה והשלישית, למרות שגם בהן האשכולות ברורים, ניתן למצוא מעט רעש:

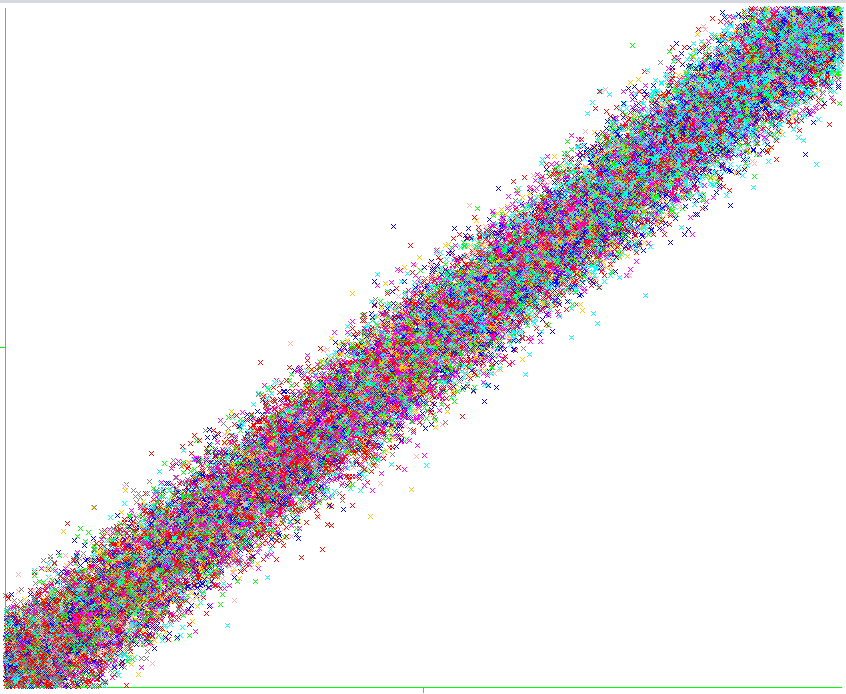




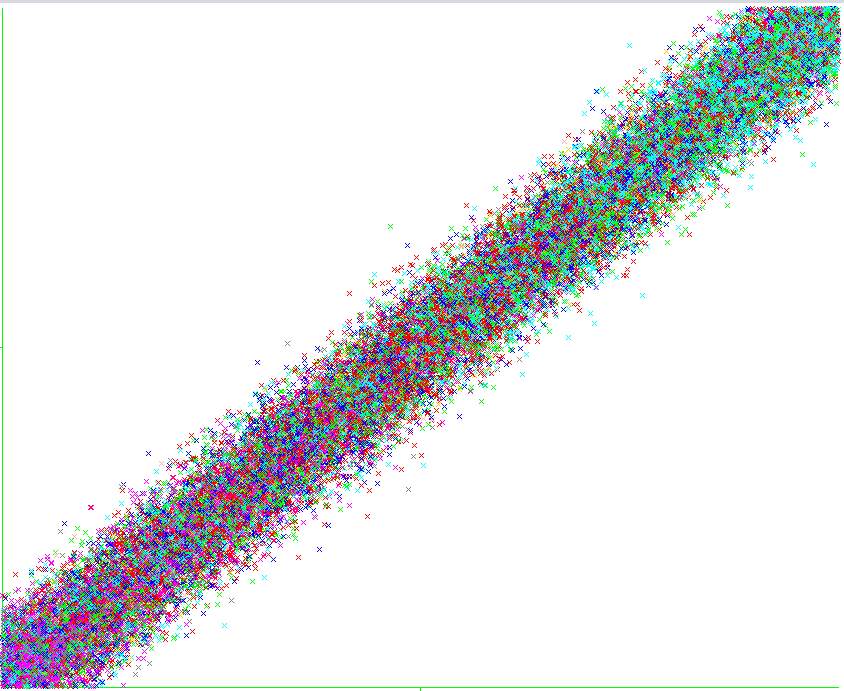
בחנתי את שתי הריצות לפי פיזור המשתנים בלבד:



ריצה 1:



ריצה 3:



נראה כי בריצה 3 יש השפעה גדולה יותר של סדר הנתונים מאשר בריצה הראשונה. למשל, למשתנים האחרונים בסט הנתונים יש נטייה להצטרף לאשכול הירוק. בריצה הראשונה יש מגמה דומה (המשתנים האחרונים הם בעלי גוון כחול), אך היא פחות ברורה, דבר המלמד על פיזור אחיד יותר. לכן בחרתי להשתמש בריצה 1.

ה.1. K-Means

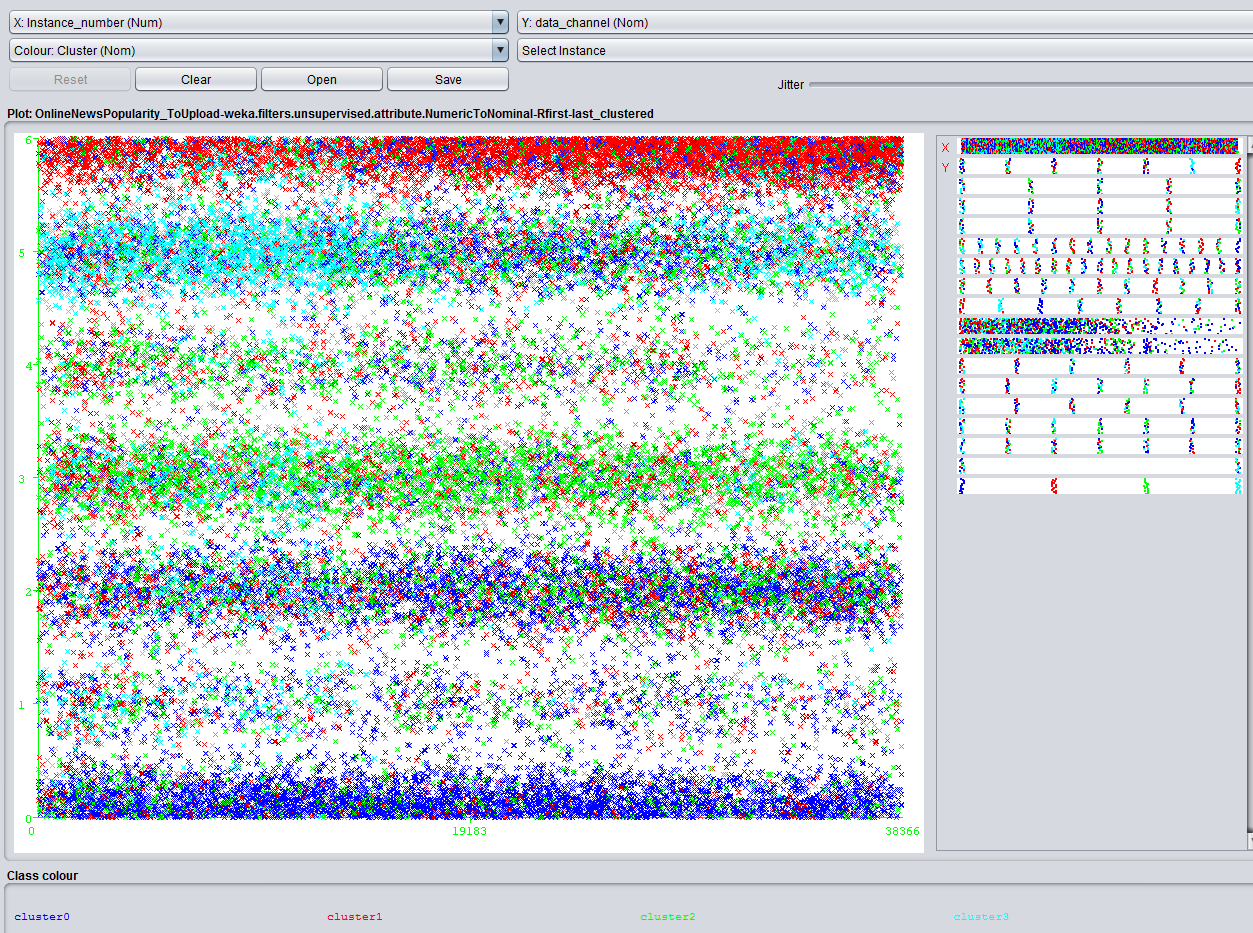
בחרתי להשתמש באלגוריתם בעל 4 אשכולות (ריצה 7), כיוון שהוא נראה הכי מאוזן ובעל סך השגיאות המרובעות הנמוך ביותר.

בחינה ויזואלית של התוצאות מראה כי ישנן תכונות בעלות דומיננטיות בין האשכולות השונים וישנן תכונות המתפזרות בצורה שווה. להלן מספר דוגמאות עבור תכונות בעלות מגמות ברורות בבסיס הנתונים, ועבור תכונות שאינן כאלה.

תכונות דומיננטיות:

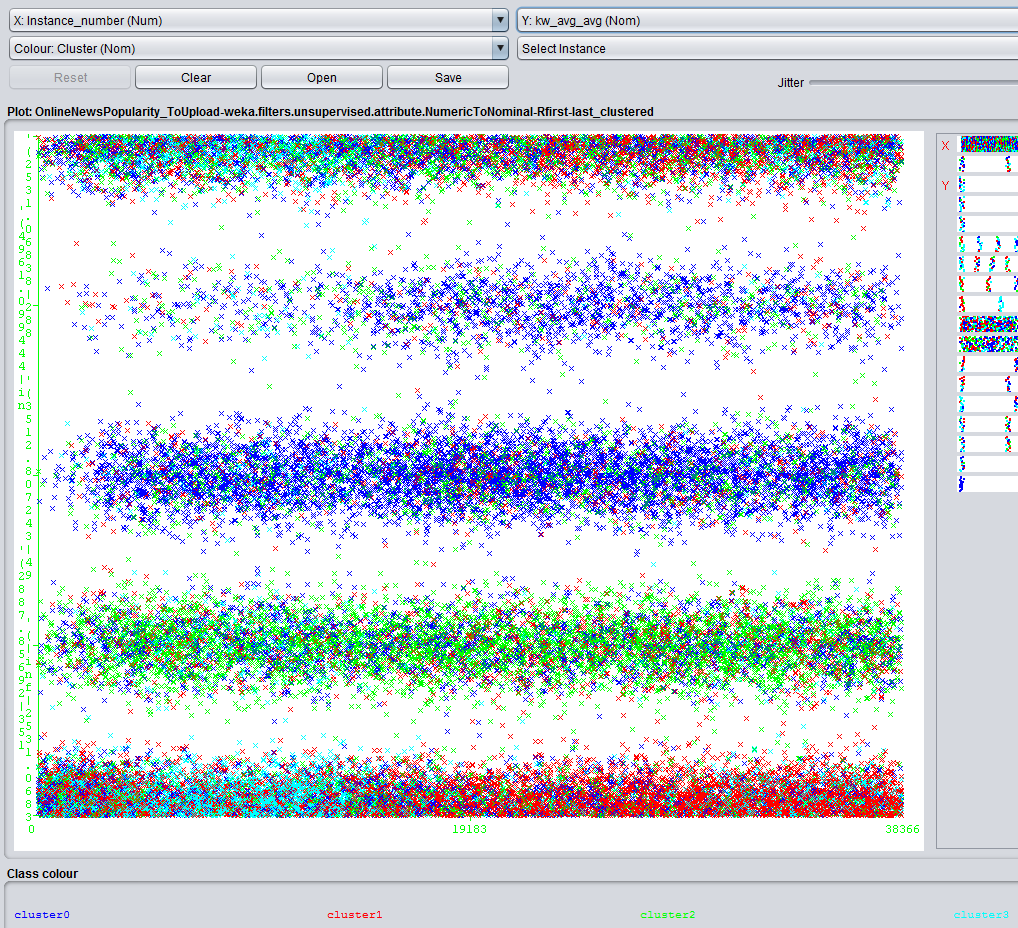
data\_channel

ניתן לראות בבירור כי ערוץ 6 (world) משויך לאשכול 1, ערוץ 5 (tech) משויך לאשכול 3, ערוצים 0 (other) ו-2 (entertainment) מזוהים בעיקר עם אשכול 0, ערוץ 3 (business) עם אשכול 2 ושאר הערוצים מפוזרים בין האשכולות.



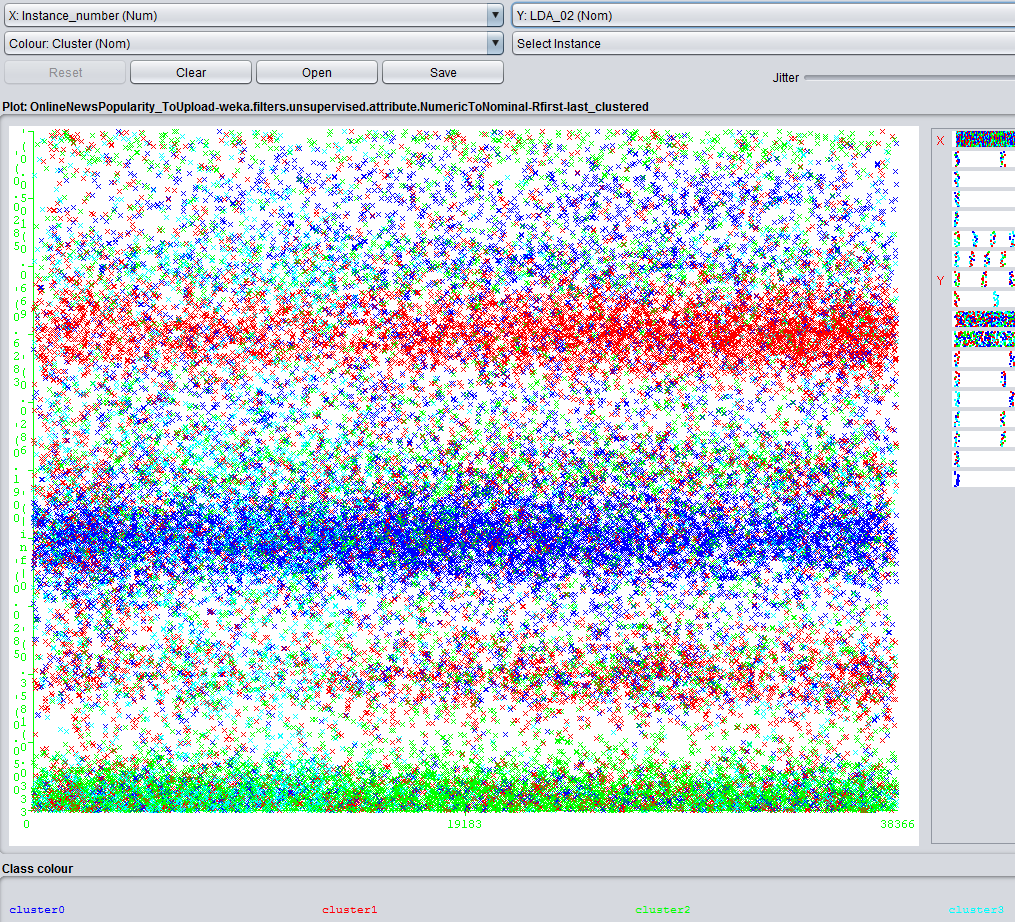
kw\_avg\_avg

התכונה דומיננטית בעיקר באשכול 0 (3512.807243-4961.099444) ובאשכול 2 (2887.85692-3512.807243), כאשר עבור אשכולות 1 ו-3 היא מקיימת את אותו טווח ערכים (-inf-2531.06838). עבור טווח ערכים גבוה, אין הבחנה בין האשכולות.



LDA\_02

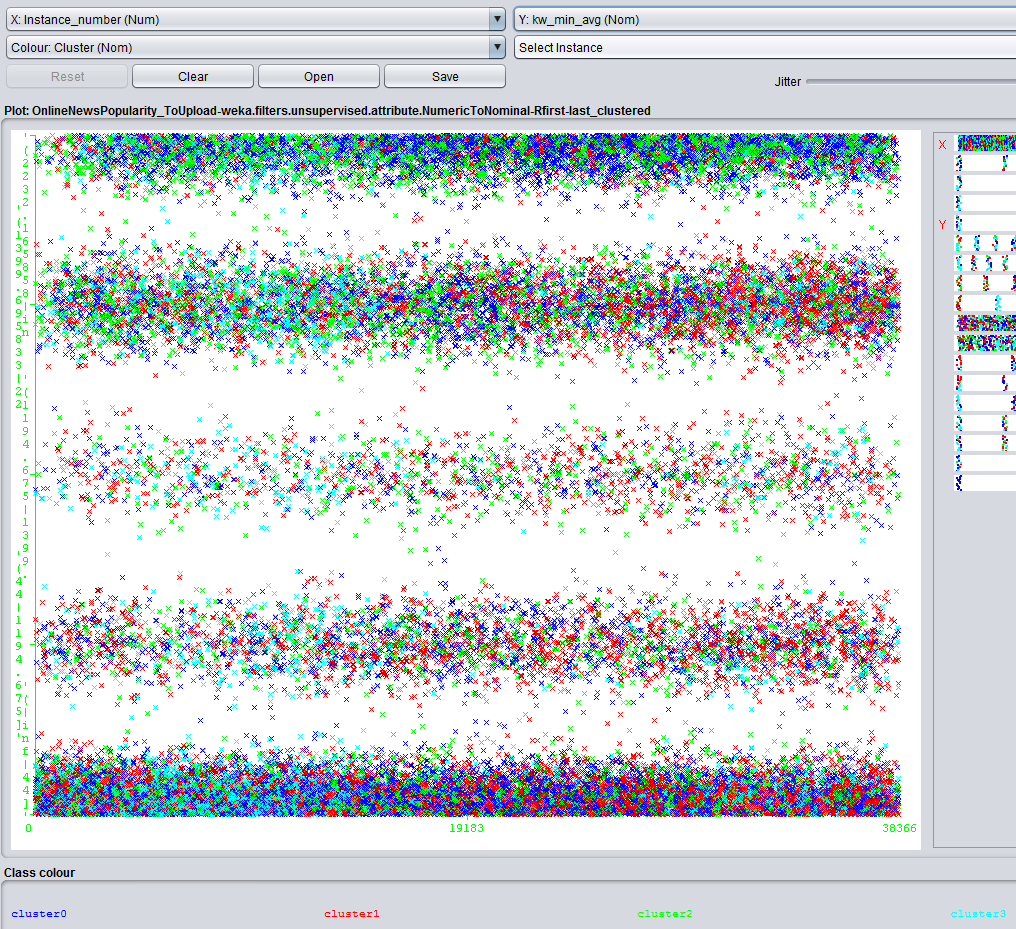
גם כאן ניתן להבחין במגמות ברורות. הערכים הנמוכים משוייכים לאשכול 2, ערכי האמצע לאשכול 0, והערכים הגבוהים לאשכול 1.



תכונות לא דומיננטיות:

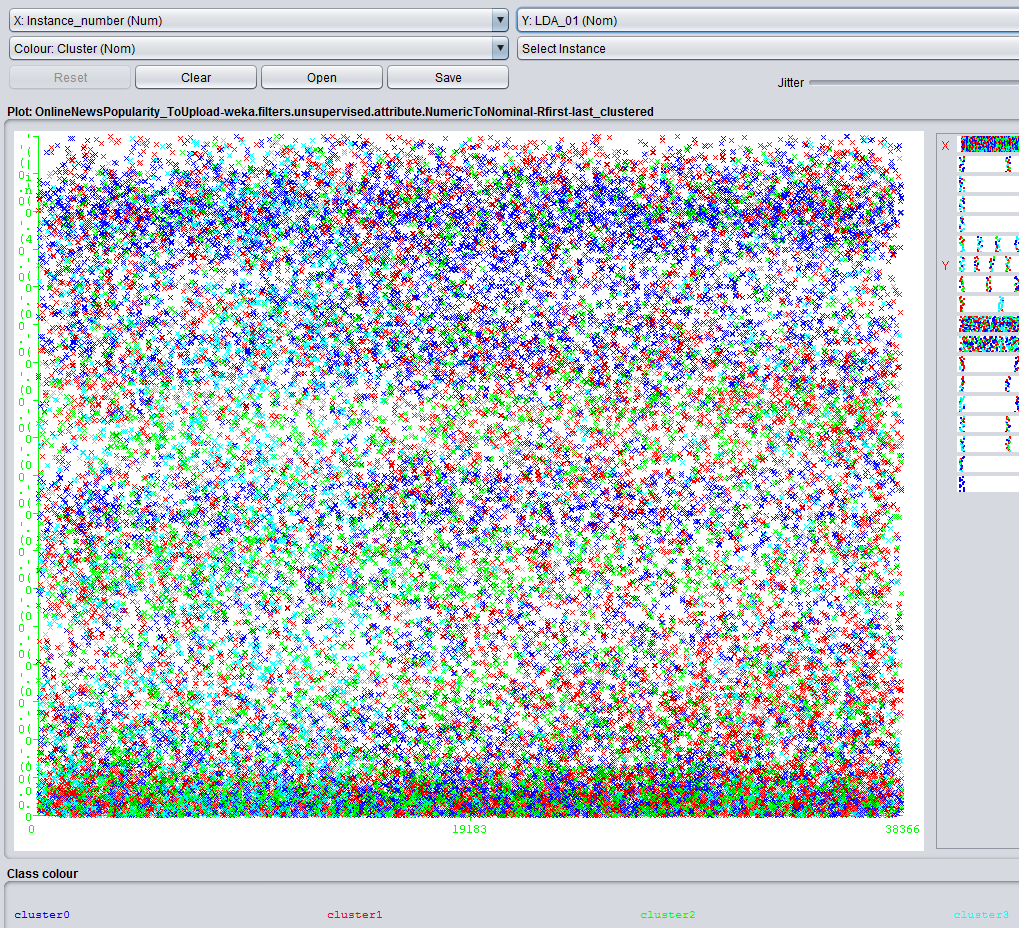
kw\_min\_avg

באופן מפתיע, למרות שמדובר בתכונה של מילות מפתח, כמו התכונה הדומיננטית שראינו קודם לכן, עבור תכונה זו קשה להבחין במגמות ברורות.



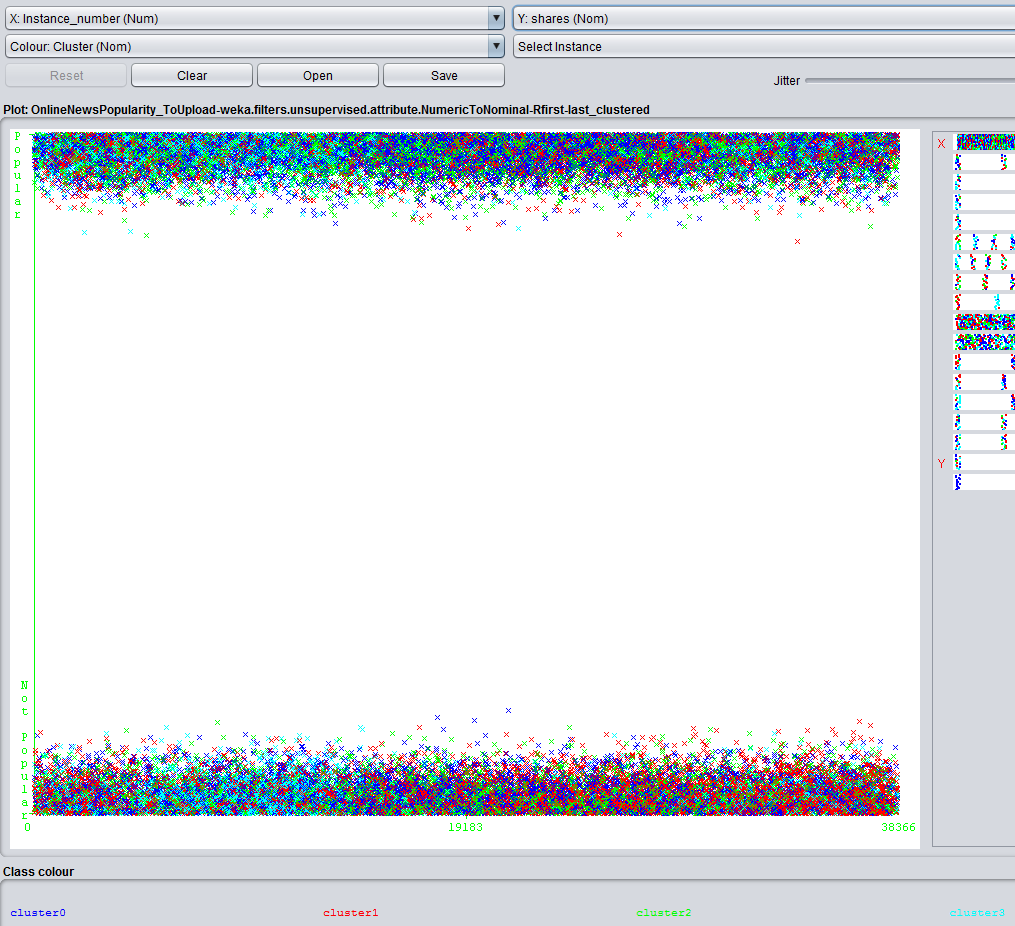
LDA\_01

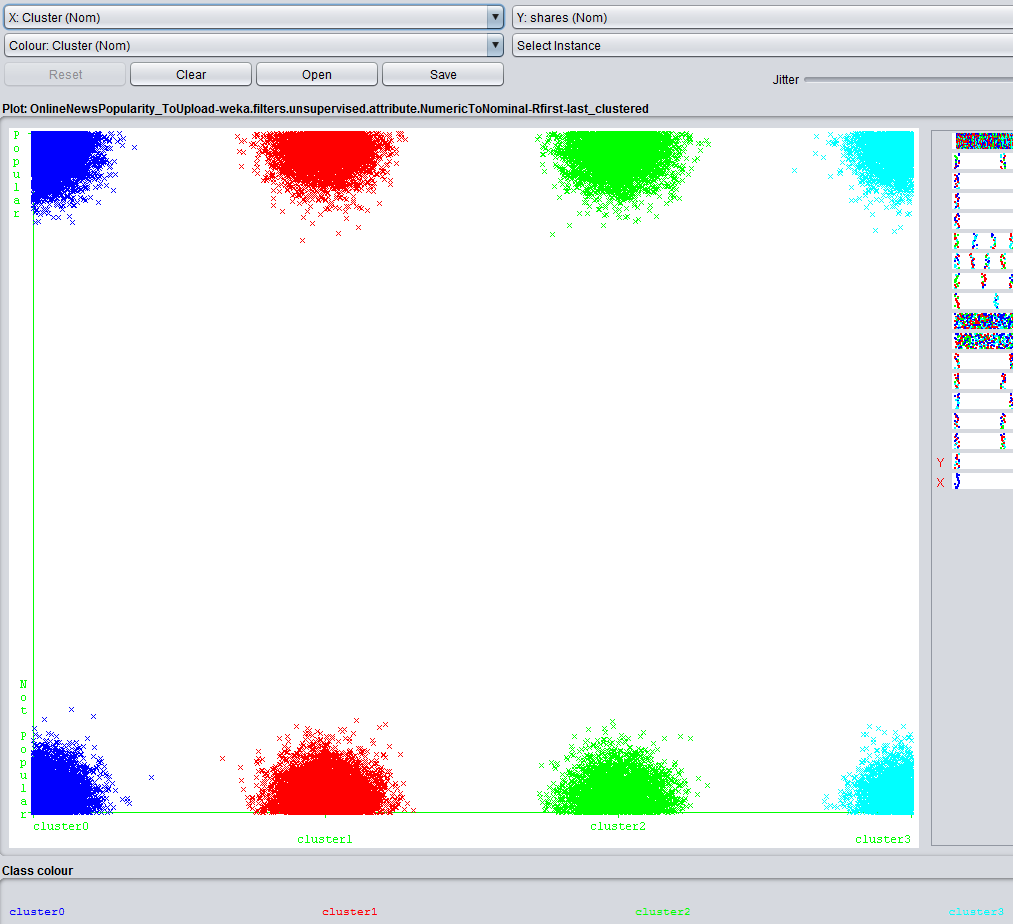
לעומת LDA\_02, תכונה זאת אינה בעלת מגמות ברורות, אם כי קיימת הזדהות מסוימת של הערכים הגבוהים עם אשכול 0:



shares

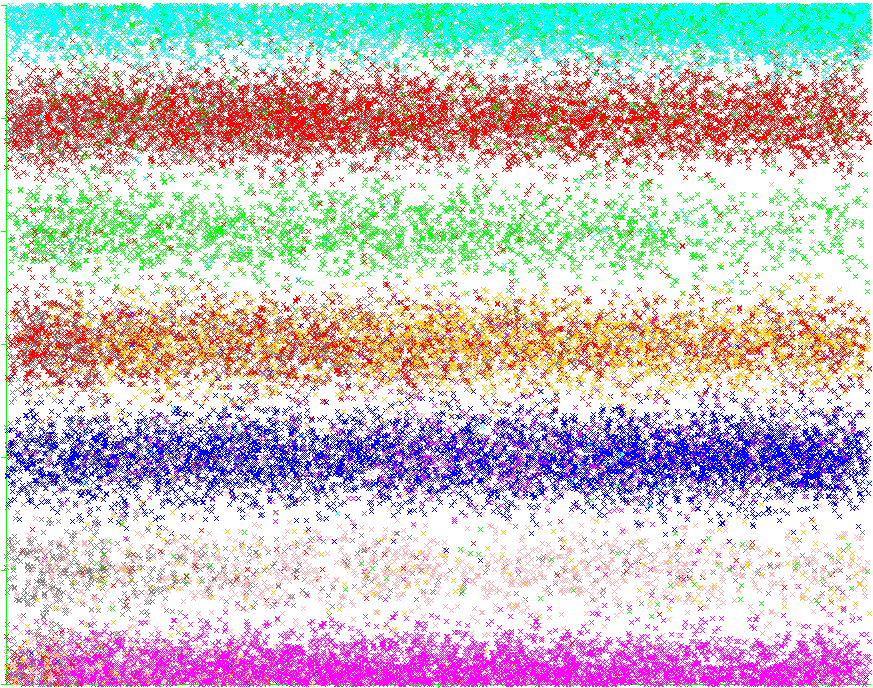
באופן לא מפתיע, תכונה זו אינה בעלת מגמה חזקה בתוך האשכולות, מכיוון שהיא הוצאה מסט הנתונים לצורך הרצת האלגוריתם. בהיותה בוליאנית ודי מאוזנת, אנו מקבלים, כפי שהיינו מצפים חלוקה לא מובהקת בתוך האשכולות, של פופולרי/לא פופולרי.

התפלגות התכונה לפי אשכולות מראה על שיוויוניות:

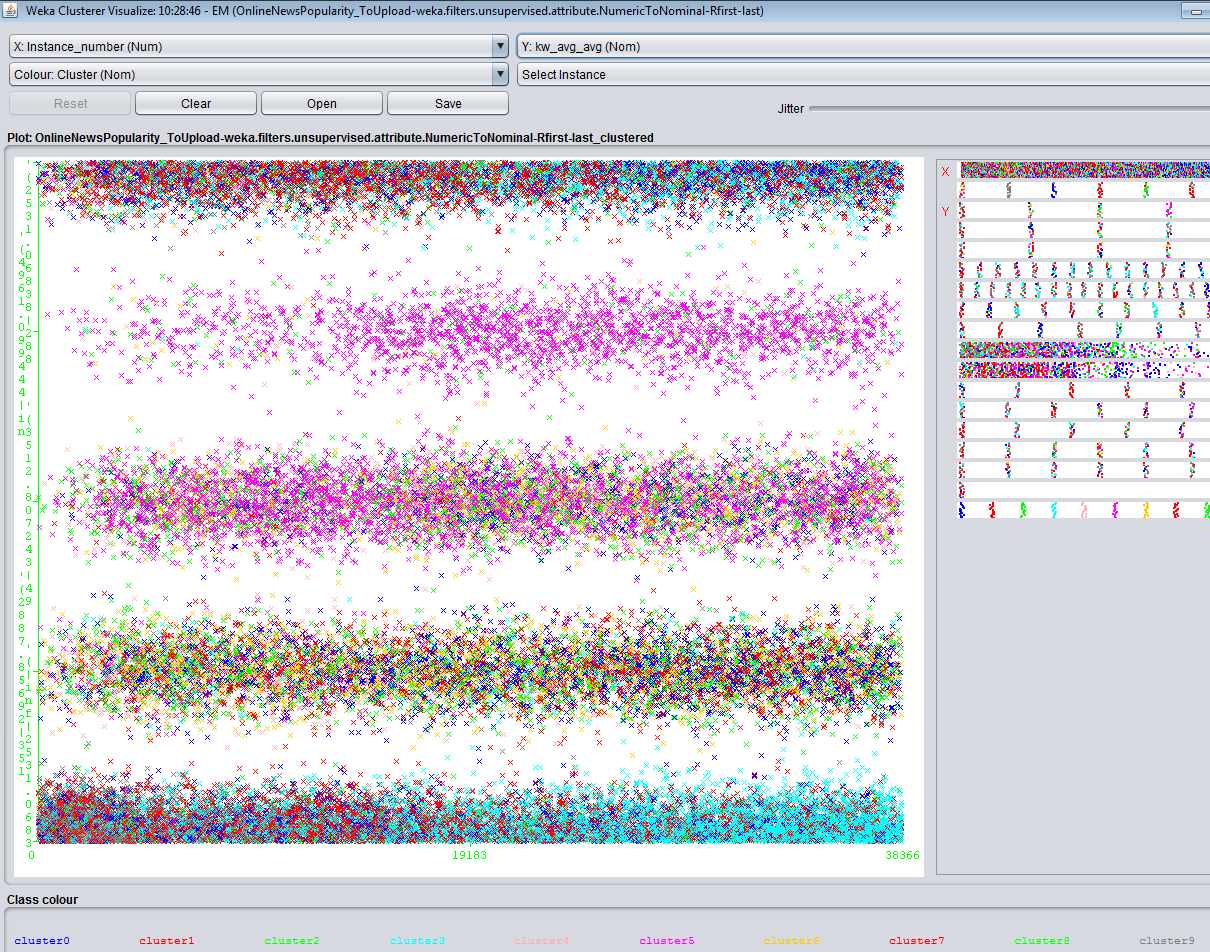


ה.2. EM

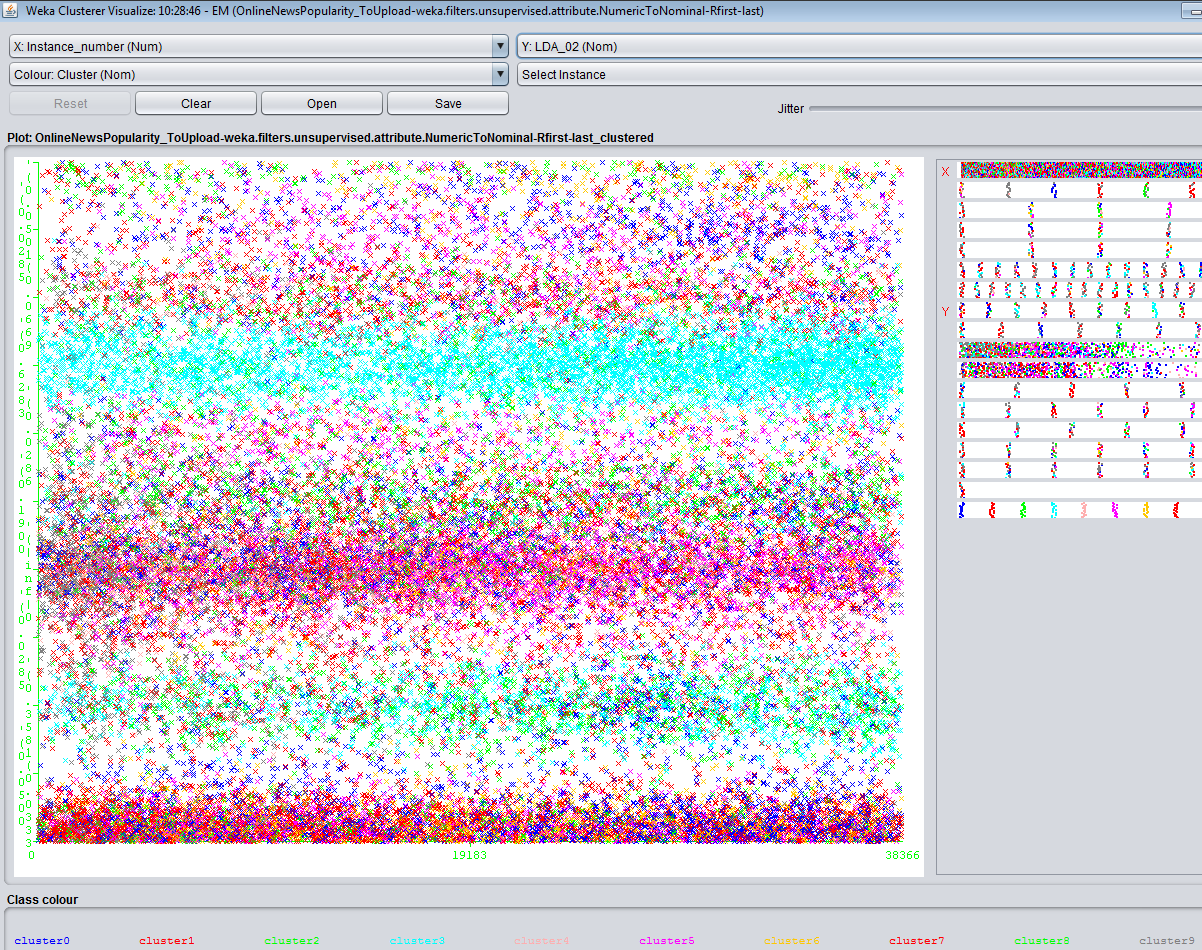
תוצאות ההרצה מראות על אישכול ברור למספר תכונות, גם עבור אלגוריתם ההיררכי, במיוחד עבור 3 התכונות הבאות:

data\_channel: 

kw\_avg\_avg:



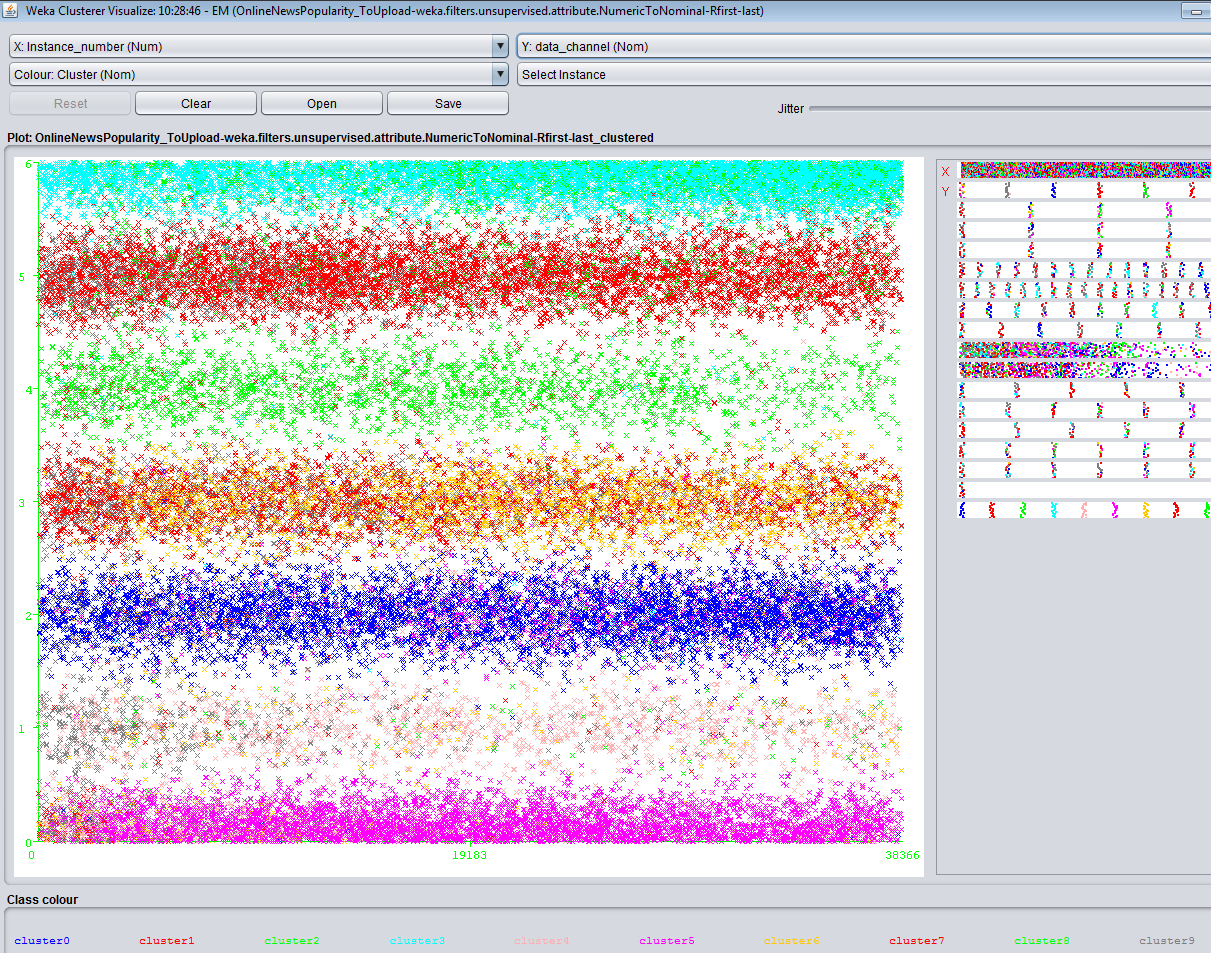
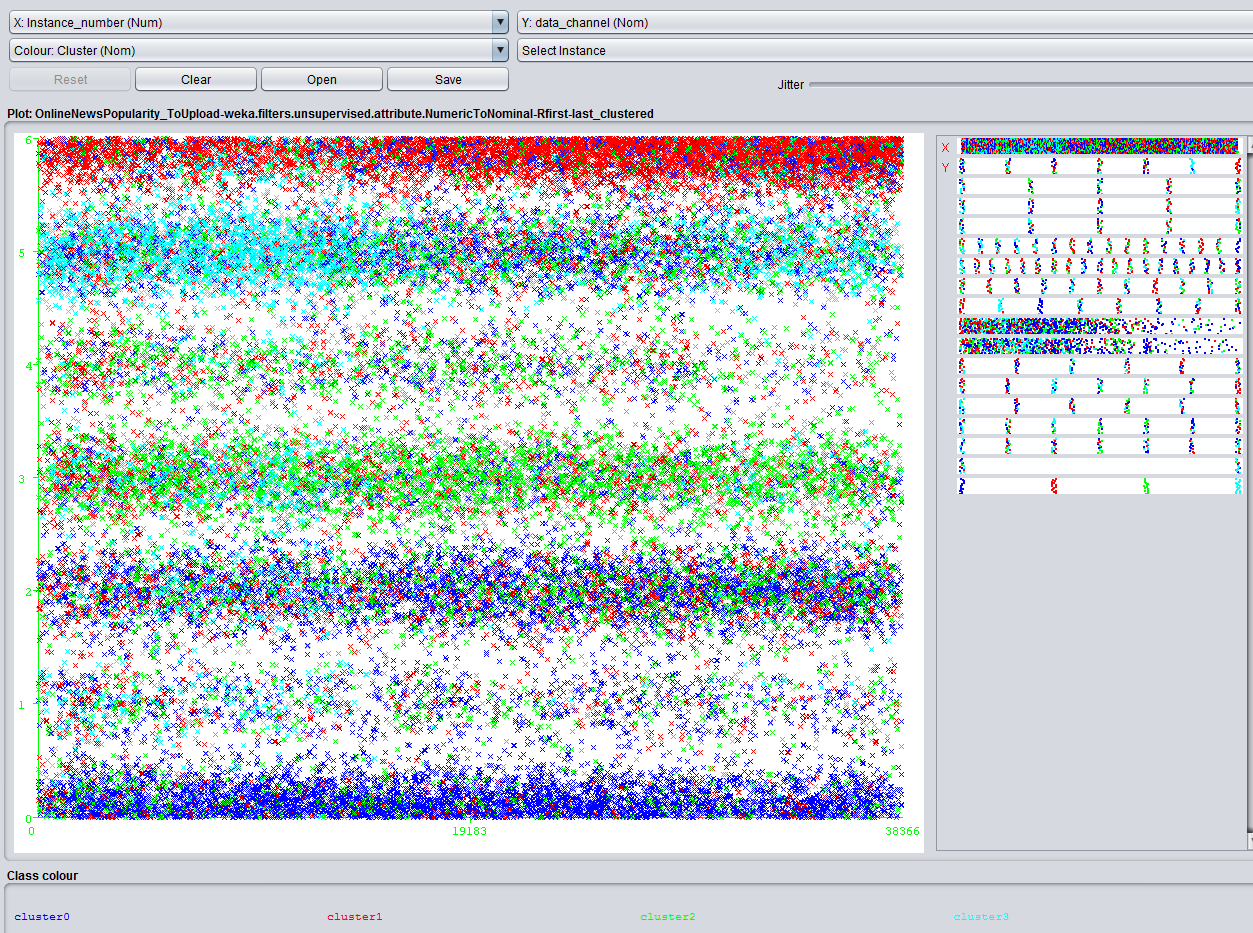
LDA\_02:



ו. נבחן את 3 התכונות הדומיננטיות שהתקבלו עבור EM, מול אותן תכונות, שהתקבלו עבור K-Means.

התכונה data\_channel מראה על דמיון בין EM ל-K-Means:

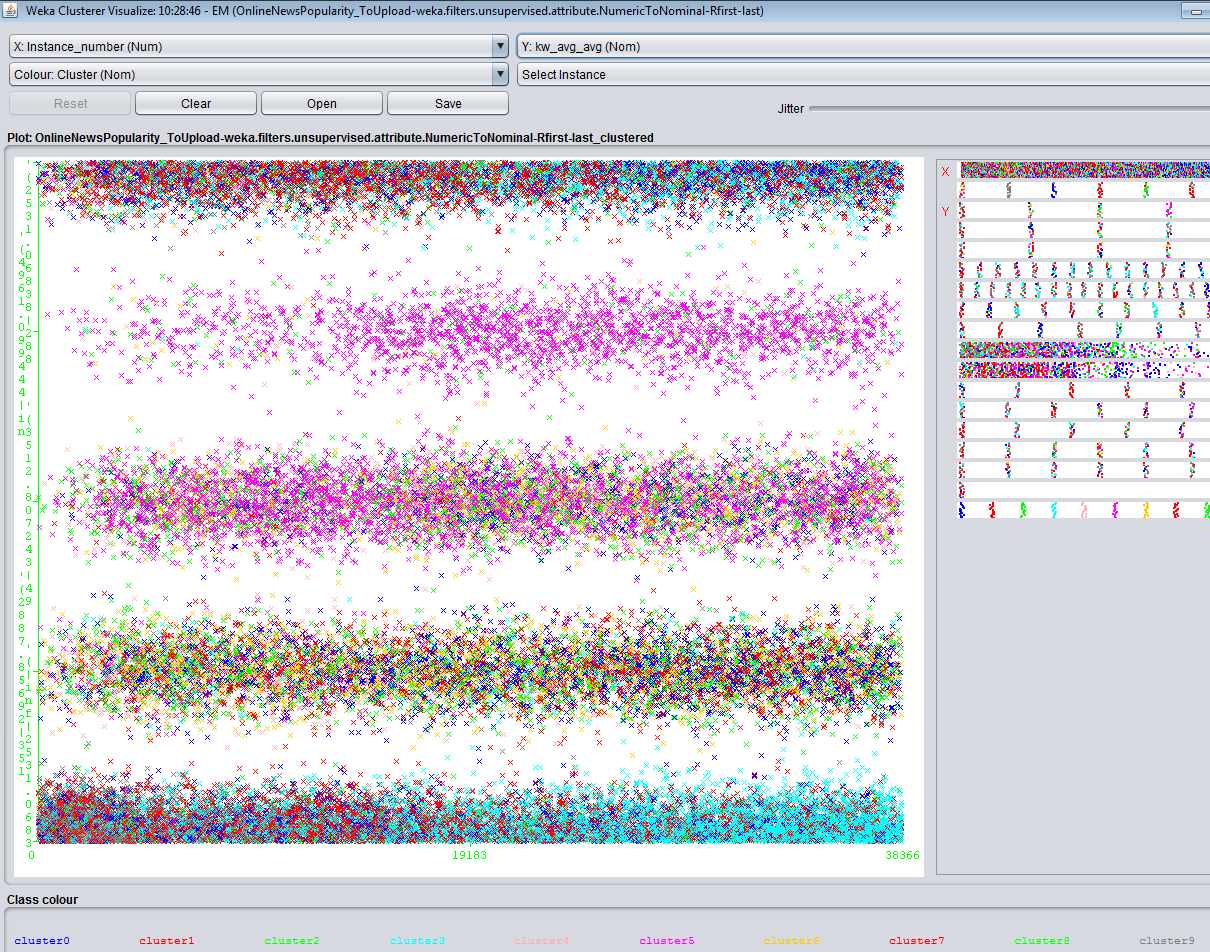
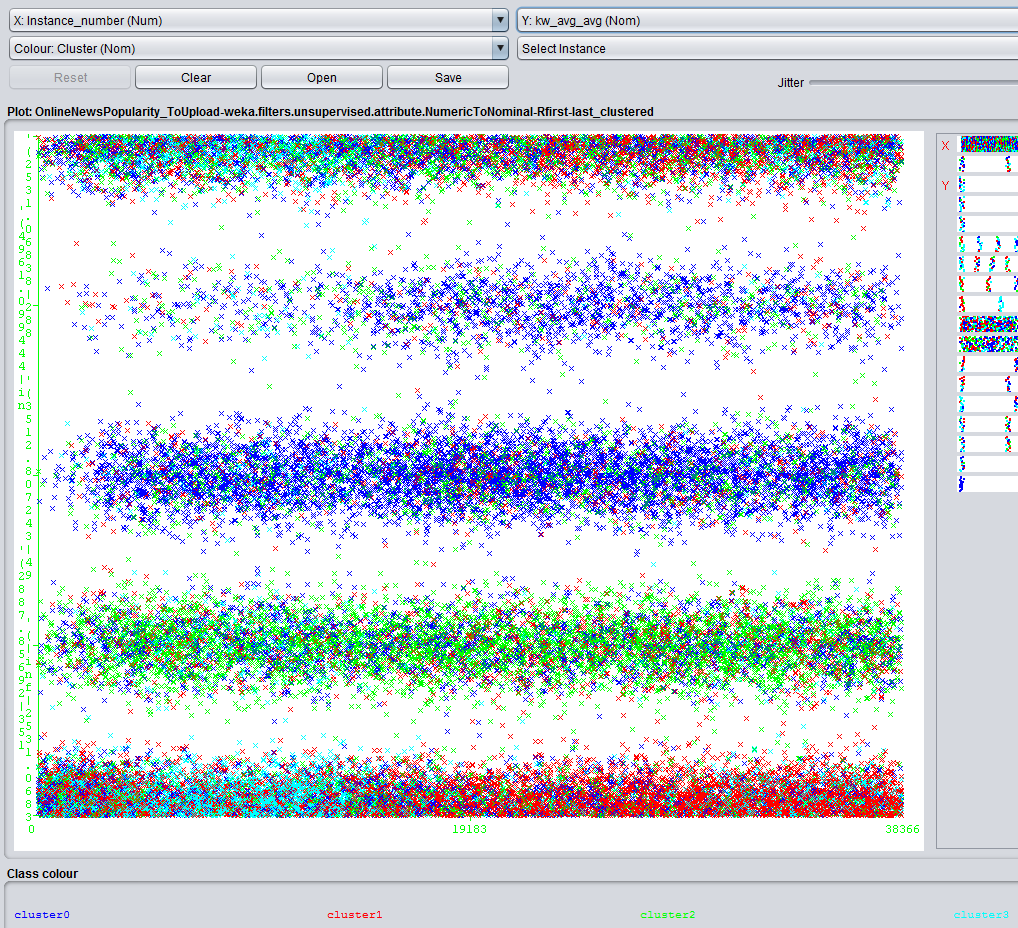
EM K-Means

יחד עם זאת האשכולות ב-EM ברורים הרבה יותר.

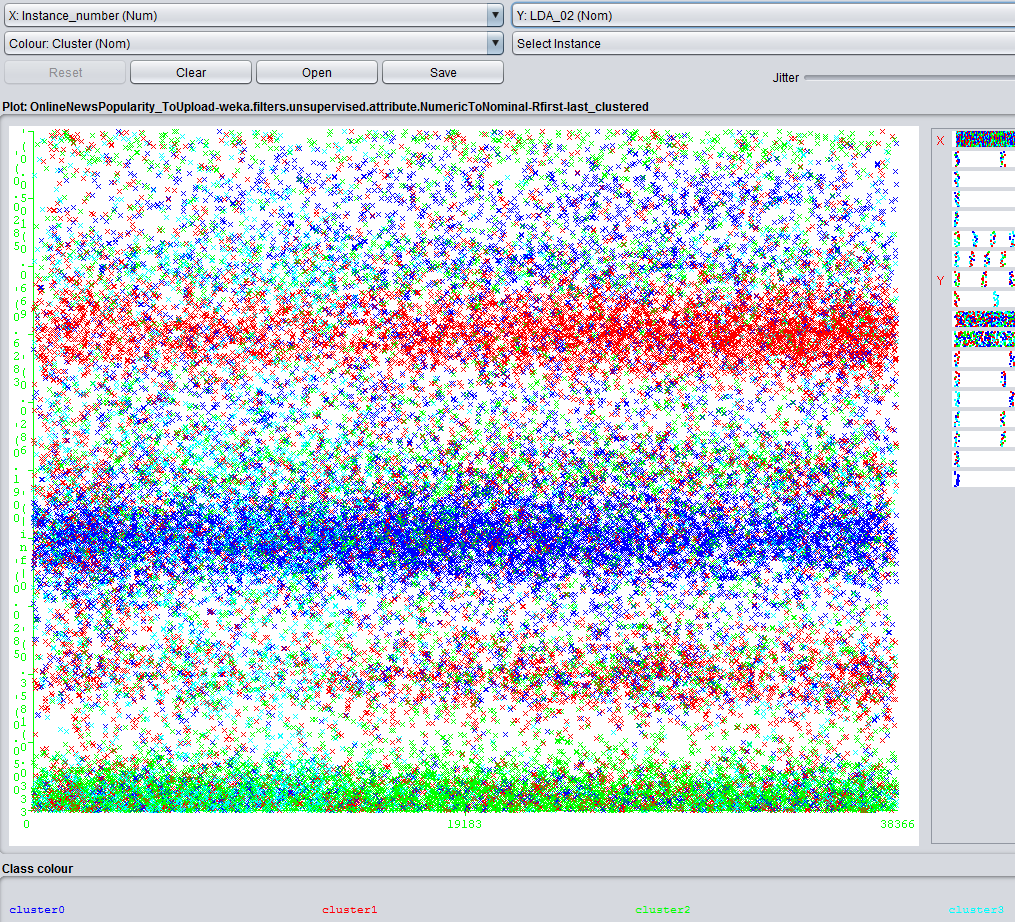
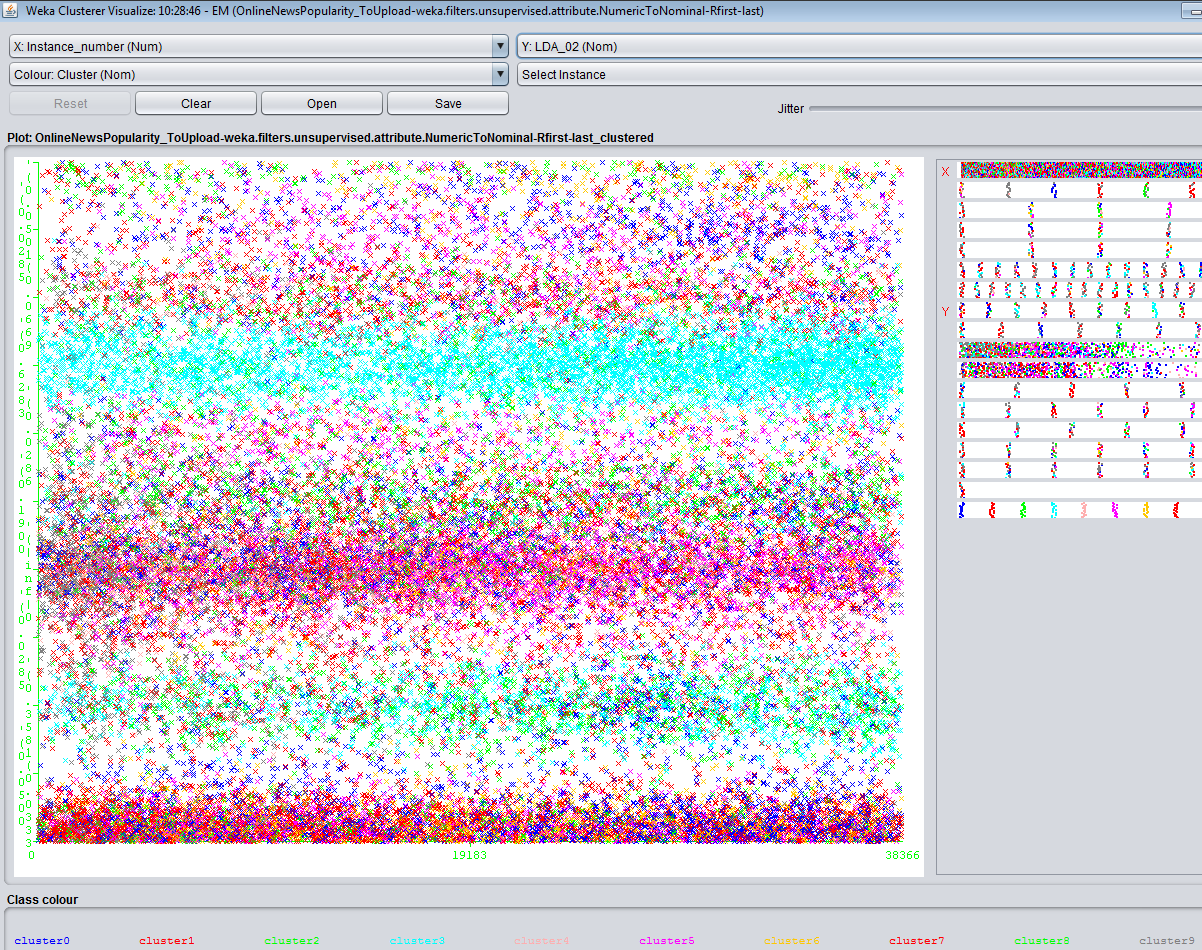
kw\_avg\_avg:

EM K-Means

LDA\_02:

EM K-Means

בשתי התכונות האחרונות הפיזור לאשכולות נראה דומה.

ניתן, אם כן, לסכם ולומר כי שני האלגוריתמים הניבו תוצאות דומות. אמנם EM חילק את הנתונים ל-10 אשכולות ו-K-Means ל-4, אך קשה למצוא הבדלים מובהקים המשתמעים מכך. 3 התכונות הדומיננטיות ביותר בשני האלגוריתמים הן data\_channel, kw\_avg\_avg, ו-LDA\_02. יתרון קל לטובת EM ניתן למצוא, כאמור, בכך ש- data\_channelמתפזר אצלו בצורה יפה יותר. שאר התכונות, בשני האלגוריתמים, התפזרו בצורה פחות ברורה לעין. היתרון של K-Means מתבטא בכך שזמן הריצה של האלגוריתמים ההיררכיים ארוך בהרבה, דבר שגרם להסרת תכונות לשם קבלת תוצאות מהירות, כך שלא ניתן לדעת מה היו המסקנות במצב אחר.

3. סיכום ומסקנות

מטרת העל של שני הפרויקטים – ממ"ן 21 וממ"ן 22, הייתה כריית מידע והסקת מסקנות לגבי פופולריות של מאמרים, שהתפרסמו במשך שנתיים באתר החדשות הפופולרי Mashable. אך המטרות בממ"ן 21 היו שונות לגמרי מהמטרות בממ"ן 22. ממ"ן 21 התמקד בטיוב הנתונים והכנתם לכריית המידע ובסיווג הנתונים בעזרת אלגוריתמים שונים של עצי החלטה. ממ"ן 22 עסק בניתוח המידע והסקת מסקנות, על-ידי יצירה של חוקי הקשר ובניית אשכולות.

בסיום ממ"ן 21 נותרנו עם 17 תכונות מתוך 61 תכונות בסט הנתונים המקורי. הדבר איפשר גישה קלה יותר לנתונים בממ"ן 22, ומתוך כך – סביבה נוחה להתמודדות עם המידע והוצאת התמצית מתוכו בדמות חוקי ההקשר ויצירת האשכולות.

המסקנות מממ"ן 21, מעבר לטיוב הנתונים, היו העדפת האלגוריתם J48 על-פני CART, לפחות לגבי סט הנתונים הנוכחי. J48 סיפק מספרים מעט טובים יותר, בכל הנוגע לדיוק הסיווג, ובנוסף זמן הריצה שלו נמוך לאין שיעור מזה של CART.

המסקנות בממ"ן 22, מתחלקות לשניים:

1. מבחינת הנתונים: מציאתן של שלוש תכונות עיקריות המראות על קשר מסוים בינן לבין מידת הפופולריות של מאמר: data\_channel, kw\_avg\_avg, ו-LDA\_02
2. מבחינת כלי הכרייה:
   1. מבחינת תוצאות: העדפה קל של האלגוריתם ההיררכי-הסתברותי EM על פני האלגוריתם החלוקתי K-Means.
   2. מבחינת זמן ריצה וידידותיות למשתמש: העדפה ברורה של K-Means.

1. נלקח מ:

   <https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Frequent_Pattern_Mining/The_FP-Growth_Algorithm#cite_note-HanPei2000-1> [↑](#footnote-ref-1)