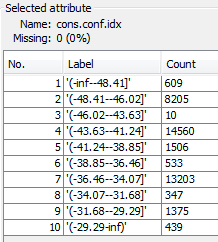
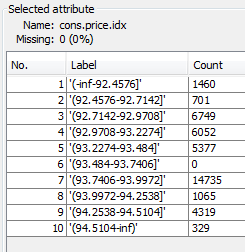
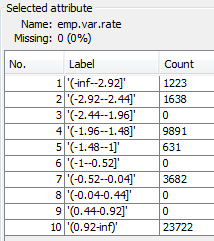
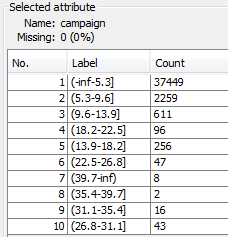
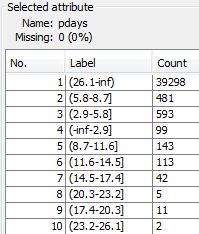
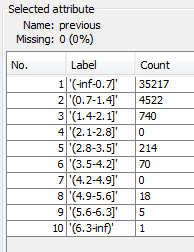
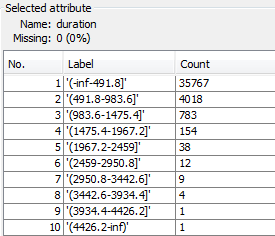
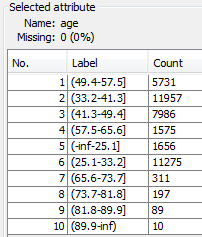
ממ"ן 22 - כריית מידע - אלי אברמוביץ

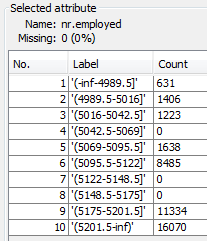
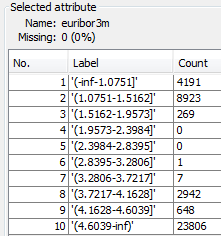
## מטרה:

איפיון פרופיל לקוח שיחתום על תנאי הפקדון שהוצעו לו במהלך שיחות

## חוקי הקשר

נשתמש בקובץ התוצר של הממ"ן הקודם עם 40787 רשומות. נבצע אחר דיסקרטיזציה על הנתונים על מנת שנוכל להריץ עליו אלגוריתמי חוקי הקשר:

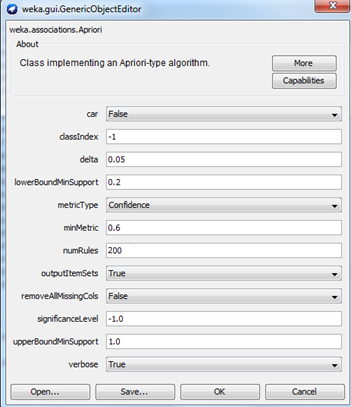




## שאלה 1.

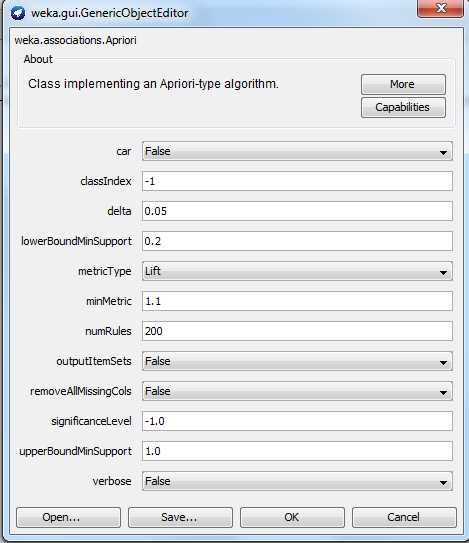
**סעיפים א+ב)**

להגדרת החוקים נריץ ב-WEKA תחת Associate את האלגוריתם האפריורי ב-minSupport=0.2 ו-minConfidence=0.6 ו-numRules עם כל מיני ערכים - מ-20 ועד 200 (תוצאות). הסיבה לכך היא שאנחנו רוצים לקבל גם ערכים שקטנים מ-1 על מנת שלא נקבל רק חוקים טריוויאלים ולא מעניינים.



ככל שעלינו בכמות החוקים ככה קיבלנו גם יותר חוקים עם Confidenceיותר נמוך, לדוגמא ב-10 חוקים הנמוך ביותר היה 1 בעוד שב-200 חוקים הConfidence הנמוך ביותר היה 0.91.

לאחר מכן נבדוק את נושא הגבלת מספר החוקים האפשריים לפי מדד ה-Lift. ככל שהערך של המדד גבוהה יותר ככה החוק אמור להיות חשוב יותר.



נתחיל ב-Lift=1.1 ונראה שקיבלנו את כל החוקים, כשנציב 1.8 נראה שקיבלנו חוקים יותר מורכבים ומסובכים - הסיבה לכך היא שזו הדרך היחידה שנקבל חוקים עם מדדים גבוהים מספיק כדי לעמוד במינימום LIFT.

נמשיך עם הערך 3 ונגלה שהפעם במקום לקבל 200 חוקים קיבלנו רק 162 חוקים שעומדים במינימום.

נמשיך עם הערך 4.8 ונגלה שהפעם קיבלנו רק 72 חוקים שעומדים במינימום.

אם נכוון את הערך ל-5 אז למעשה נקבל רשימה ריקה שכן אין חוק שיעמוד במינימום LIFT.

על מנת שחוק יעמוד במינימום התמיכה 20% הוא צריך להכיל לפחות 8157.4 טרנזקציות ע"פ חישוב:  
40787\*0.2=8157.4

* החוקים מצאנו עבור minSupport=0.2 ו-minConfidence=0.6: בקובץ rule1\_apriori.txt.
* החוקים עבור LIFT=1: בקובץ rule2\_lift.txt.
* כשמכוונים את LIFT ל-4.8: בקובץ rule3\_lift.txt.

**סעיף ג)**

**1.** נבחר את האלגוריתם ***Apriori***:

סיבה: בחרתי בו מאחר ובדיקות ביצועים הראו שהאלגוריתם יעיל וסקיילבילי(scalable).

הכנת המידע: על מנת להריץ את אלגוריתם זה היינו צריכים לעשות דיסקרטיזציה על הנתונים.

האלגוריתם: אלגוריתם אפריורי מחפש תדירות של ITEMSETS לצורך ביצוע אסוציאציות בצורה בלואנית. הוא מבוסס (על פי שמו Prior Knowledge ) על ידע מוקדם של תדירות המאפייני הסטים.

האלגוריתם פועל בצורה איטראטיבית בצורת חיפוש ע"פ רמות שבו משתמשים ב K-סטים כדי לחקור K+1 סטים. בהתחלה מסננים את הסטים שלא עובדים בדרישת המינימום תמיכה -נסמן את התוצאה בתור L1. משם משתמשים בL1 לחיפוש בL2 - הסטים הכי שכיחים. הם משומשים למצוא עבור L3 וכן הלאה עד שאין יותר K סטים שכיחים למצוא. כדי למצוא עבור כל אחד מLK  נדרש מעבר שלם על בסיס הנתונים.

כל מחזור של האלגוריתם מחולק ל-2 שלבים עיקרים:

1. Join step - 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) The Prune - בשלב הגיזום מתעלמים מצירופים שאינם שכיחים שאינם לא עומדים בתכונה האפריורית. בשלב זה גם מבוצעת בדיקה מהירה (subset testing) של הסטים שלא נגזמו - האם הסטים אכן שכיחים. השכיחות מתבצעת ע"י ספירת המתמודדים בטבלת גיבוב (hash).

ישנם גם דרכים נוספות לשפר את היעילות שלו כמו לדוגמא לקחת תת-סט של המידע ולבצע את הניתוח עליו, משפרים מאוד את המהירות אך מאבדים מהדיוק.

שיטה נוספת היא להשתמש בספירה דינמאית של הסטים - מועמדים חדשים יכולים להתווסף בכל שלב בניגוד לאלגוריתם האפריורי המקורי שסופר כבר בהתחלה את המועמדים ודורש סריקה שלמה של בסיס הנתונים.

**2.** נבחר את האלגוריתם ***FPGrowth***:

סיבה: בחרתי בו מאחר הוא מכיל את שלמות המידע ואת התדירות, כמו כן הוא מאוד קומפקטי (פריטים שאינם שכיחים נעלמים). הוא נותן תוצאה מסדורת ע"פ שכיחות כך שהאלמנטים שכנראה מעניינים יותר נמצאים בהתחלה.

הכנת המידע: על מנת להריץ את אלגוריתם זה על הנתונים הייתי צריך להמיר את כל הערכים בעזרת הפילטר-NominalToBinary ולאחר מכן בעזרת הפילטר NumerictoBinary כשאופציית ה-CLASS מכוונת ל-TRUE על מנת שהאלגוריתם יוכל לעבוד. למעשה מה שקרה למידע זה שהוא נהפך לאוסף של קבוצות בינאריות כשכל אחת מכילה את כמות השורות שתכונה מופיעה כ-1 וכמות המקומות שתכונה לא מופיעה כ-0.

האלגוריתם: פועל בשיטת הפרד-ומשול הוא דוחס את בסיס הנתונים אל תוך עץ שמציג את השכיחויות (FP-TREE) השומר את המידע לitemset האסוצייטיבי. הוא מפרק את בסיס הנתונים המכווץ לסט של בסיסי נתונים מותנים (מלשון:תנאי) ואז מבצע את הליך כריית המידע על כל אחד מהתת בסיסי נתונים בנפרד מה שמקטין משמעותית את כמות המידע שבה יש לחפש.

שלבים:

1) שלב מציאת המועמדים בדומה לאלגוריתם האפריורי ומייצרים רשימה שמכילה את הסטים שעומדים במינימום ואת מספר השיכחויות שלהם .

2) בניית עץ הFP-TREE- בהתחלה, בונים את שורש העץ שמסומן ב-null. סריקה נוספת של בסיס הנתונים שבה מסדרים את הסטים בסדר יורד ע"פ כמות השכיחויות ומייצרים ענף עבור כל טרנזקציה. כל ענף מכיל PREFIX שידאג שאם מכניסים את "אותו האיבר" במקום לייצר ענף חדש לאותו האיבר, פשוט יתווסף 1 לcounter של השכיחויות.

**סעיף ד)**

האלגוריתם הAPRIORI - התוצאות בקובץ apriori\_algo.tx

כפי שצויין בסעיף א', בהינתן support/confidence הנתונים ניתן לקבל מרחב עצום של החוקים. הגבלנו את מספר החוקים על מנת לבדוק את ההשפעה על טיב החוקים שמתקבלים. נמצא, שככל שמגדילים את מספר החוקים מקבלים יותר חוקים, אך מתקבלים המון חוקים עם confidence = 1, משמע - חוקים טריוויאליים ולא מעניינים. ניסינו מספר חלופות של מספר החוקים, כגון 500, 200, 100, 50, 20 עד שנמצאו 14 חוקים וה-confidence ירד מ- 1 דהיינו החוקים יותר משמעותיים.

האלגוריתם FP GROWTH - התוצאות נמצאות בקובץ fpgrowth\_algo.txt

גם אלגוריתם זה בהינתן support/confidence הנתונים ניתן לקבל מרחב עצום של החוקים. כמובן חלק מההגדרהnumRulesToFind.

**סעיף ה)** בשביל להעריך את תועלת החוקים נשתמש במדד LIFT עם ערך 4.8. ככל שהערך של המדד יותר גבוה, כך הוא יותר חשוב. כדי להמנע מחוקים לא מעניינים הורדתי את אלה שקיבלתי conf1 ונשארתי עם 22 חוקים. ככל שעלה מדד הLIFT ככה התקבלו יותר תוצאות (מעניינות ופחות מעניינות כאחד) וכמו כן החוקים נהיו יותר מסובכים:

32. nr.employed='(5095.5-5122]' 8485 ==> emp.var.rate='(-1.96--1.48]' euribor3m='(1.0751-1.5162]' 8216 conf:(0.97) < lift:(4.81)> lev:(0.16) [6506] conv:(25.1)

35. nr.employed='(5095.5-5122]' 8485 ==> default=no emp.var.rate='(-1.96--1.48]' euribor3m='(1.0751-1.5162]' 8216 conf:(0.97) < lift:(4.81)> lev:(0.16) [6506] conv:(25.1)

36. default=no nr.employed='(5095.5-5122]' 8485 ==> emp.var.rate='(-1.96--1.48]' euribor3m='(1.0751-1.5162]' 8216 conf:(0.97) < lift:(4.81)> lev:(0.16) [6506] conv:(25.1)

38. nr.employed='(5095.5-5122]' 8485 ==> cons.conf.idx='(-48.41--46.02]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

41. nr.employed='(5095.5-5122]' 8485 ==> default=no cons.conf.idx='(-48.41--46.02]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

42. default=no nr.employed='(5095.5-5122]' 8485 ==> cons.conf.idx='(-48.41--46.02]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

45. nr.employed='(5095.5-5122]' 8485 ==> emp.var.rate='(-1.96--1.48]' cons.conf.idx='(-48.41--46.02]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

46. emp.var.rate='(-1.96--1.48]' nr.employed='(5095.5-5122]' 8485 ==> cons.conf.idx='(-48.41--46.02]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

48. nr.employed='(5095.5-5122]' 8485 ==> cons.conf.idx='(-48.41--46.02]' euribor3m='(1.0751-1.5162]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

53. nr.employed='(5095.5-5122]' 8485 ==> default=no emp.var.rate='(-1.96--1.48]' cons.conf.idx='(-48.41--46.02]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

54. default=no nr.employed='(5095.5-5122]' 8485 ==> emp.var.rate='(-1.96--1.48]' cons.conf.idx='(-48.41--46.02]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

55. emp.var.rate='(-1.96--1.48]' nr.employed='(5095.5-5122]' 8485 ==> default=no cons.conf.idx='(-48.41--46.02]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

56. default=no emp.var.rate='(-1.96--1.48]' nr.employed='(5095.5-5122]' 8485 ==> cons.conf.idx='(-48.41--46.02]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

59. nr.employed='(5095.5-5122]' 8485 ==> default=no cons.conf.idx='(-48.41--46.02]' euribor3m='(1.0751-1.5162]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

60. default=no nr.employed='(5095.5-5122]' 8485 ==> cons.conf.idx='(-48.41--46.02]' euribor3m='(1.0751-1.5162]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

63. nr.employed='(5095.5-5122]' 8485 ==> emp.var.rate='(-1.96--1.48]' cons.conf.idx='(-48.41--46.02]' euribor3m='(1.0751-1.5162]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

64. emp.var.rate='(-1.96--1.48]' nr.employed='(5095.5-5122]' 8485 ==> cons.conf.idx='(-48.41--46.02]' euribor3m='(1.0751-1.5162]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

69. nr.employed='(5095.5-5122]' 8485 ==> default=no emp.var.rate='(-1.96--1.48]' cons.conf.idx='(-48.41--46.02]' euribor3m='(1.0751-1.5162]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

70. default=no nr.employed='(5095.5-5122]' 8485 ==> emp.var.rate='(-1.96--1.48]' cons.conf.idx='(-48.41--46.02]' euribor3m='(1.0751-1.5162]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

71. emp.var.rate='(-1.96--1.48]' nr.employed='(5095.5-5122]' 8485 ==> default=no cons.conf.idx='(-48.41--46.02]' euribor3m='(1.0751-1.5162]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

72. default=no emp.var.rate='(-1.96--1.48]' nr.employed='(5095.5-5122]' 8485 ==> cons.conf.idx='(-48.41--46.02]' euribor3m='(1.0751-1.5162]' 8205 conf:(0.97) < lift:(4.81)> lev:(0.16) [6498] conv:(24.12)

**סעיף ו)** בעוד שהאלגוריתם האפריורי (ללא LIFT) נתן לנו תוצאות רצויות לקח לו הרבה יותר זמן לרוץ מאשר לאלגוריתם הFPGROWTH. הסיבה העיקרית היא שהאלגוריתם האפריורי מבצע הרבה סריקות כל פעם מחדש על בסיס הנתונים בעוד ש-FPGROWTH עובד בתור עץ בינארי שמפצל את בסיס הנתונים למספר תת בסיסי נתונים ומגובה בטבלת גיבוב.

חשוב לציין שבFPGROWTH הCONF הנמוך ביותר היה 0.89 לעומת האפריורי שבו היה הCONF הנמוך ביותר 0.91.

באפריורי קיבלנו 51 תוצאות אינן CONF 1

בFPGROWTH קיבלנו 53 תוצאות אינן CONF 1, כלומר יותר חוקים שמעניינים אותנו.

בFPGROWTH קיבלנו למעשה יותר מידע בתוצאה מהאפריורי שנוכל לעבד בנוסף ולכן נעדיף אותו גם בקטגוריה זאת:

בצד ימין של כל חוק באפריורי קיבלנו 2 נתונים(ללא LIFT):

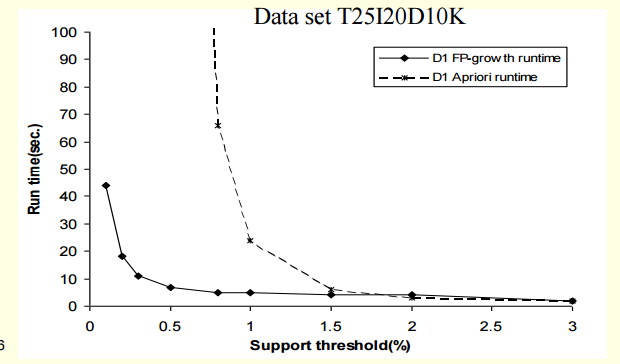
שכיחות,conf

בצד ימין של כל חוק ב-FPGROWTH סיפק לנו 5 נתונים:

שכיחות,conf,lift,lev,conv

בדוחות של האפריורי ניתן היה גם לקבל רשימה של סטים גדולים בכל אחת מהדרגות, מה המינימום תמיכה שהתקבל ומספר המחזורים, מה שיכול לסייע לנו להבין את התהליך יותר טוב ואולי אף לשפר אותו באמצעות שינוי הפרמטרים. בFPGROWTH קיבלנו רק את התוצאה הסופית.

מסקנה:בסיכומו של עניין שני האלגוריתמים עובדים בצורות שונות ובעוד שהאפריורי מספק יותר דוחות הביצועים של FPGROWTH מהירים פי כמה וכמה. נסייג לומר שהביצועים מאוד תלויים בסוג וכמות המידע כפי שניתן לראות מהטבלה למטה. לצורך העניין הייתי בוחר בFPGROWTH אם הייתי צריך לעבוד באופן תכוף יותר עם המידע לעומת האפריורי שבו הייתי נותן לו לרוץ במשך זמן רב יותר, כנראה עם LIFT מוציא דוח שמצמצם את כמות הנתונים ואז עובד איתם.



## ניתוח אשכולות

## שאלה 2.

**א.** אשכול היא קבוצת רשומות (subset מהקבוצה הכללית). המטרה בהליך אשכול היא ליצור חלוקה שבה איברי האשכולות דומים זה לזה בעוד שהאשכולות עצמם שונים זה מזה. ככה לדוגמא כשעושים ניתוח של תמונה אפשר באמצעות אשכול לחלק את התמונה לאזורים ע"פ צבע. הרבה פעמים מאפייני האשכולות אינם ידועים מראש ולכן צריך לגלות אותם מראש. בחזרה לדוגמא של ניתוח תמונה - בתמונה של חוף ים תהליך האשכול על בסיס צבע יחלק את התמונה לאשכול חוף ים,אשכול מים,אשכול שמיים ואשכול עננים בעוד שבתמונה של יער תהליך האשכול יחלק את התמונה לאשכול דשא ירוק, אשכול עלי העצים, אשכול של גזעי העץ ואשכול שמיים.

ככל שיש יותר מידע, כל תהליך האשכול נהיה כבד יותר מבחינת סיבוכיות ולוקח יותר זמן. כמו כן ככל שתהליך האשכול יהיה רגיש יותר ככה יווצרו יותר אשכולות מה שעלול לגרום לעודף מידע (לדוגמא כל עלה של עץ יהפוך להיות אשכול בפני עצמו) ומנגד תהליך האשכול יכול להיות לא רגיש מספיק ואז לאחד קבוצות שאיננו רוצים (לדוגמא: בתמונות שלנו יתאחדו הים והשמיים או הדשא ועלי העצים).

שימושים נפוצים בתהליך האשכול: חיפוש באינטרנט, ביולוגיה, אבטחה,תנועה\תחבורה בכבישים, עיבוד תמונה ובינה מלאכותית.

ניתן גם ליצור אשכולות ע"פ OUTLINER כלומר לחפש איברים שמאוד שונים אחד מהשני , דמיינו חוברת צביעה לילדים: תהליך האשכול יזהה את האיזורים הלבנים ע"י מציאת ההבדלים בין הלבן והשחור.

כמו כן ישנן 2 סוגי חלוקות:

חלוקה קשה- בה כל איבר יכול להשתייך לאשכול אחד בלבד.

חלוקה רכה - בה כל איבר יכול להשתייך למספר אשכולות.

מידות הדמיון בין הרשומות נעשות באמצעות פונקציות ויחסי דמיון ופונקציות מרחק (כגון: מרחק אוקלידי, מרחק המינג, מרחק מנהטן ומרחק צ'בישב).

ישנם סוגים שונים של אלגוריתמים של אשכול ובינהם אלגוריתמים היררכים (יוצרים אשכולות שמסודרים בעץ), אלגוריתמים צוברים (שעובדים בשיטת bottom-up שבו כל איבר מתחיל לבד ולאט לאט האיברים מתאחדים) ואלגוריתמים מפלגים (שעובדים בשיטת top-down שבו כל האיברים מתחיל באשכול אחד גדול ולאחר מכן מתפלגים למספר של אשכולות).

**ב.** ישנם מספר מדדים\גישות אורתוגונליות לקביעת איכות האשכול:

קריטריון החלוקה: מועיל במיוחד לתהליכי אשכול שטוחים (שאין להם היררכיה) ועוזר לנתח קבוצות שונות לדוגמא כשמחלקים את האיברים לקבוצות ומקנים לכל קבוצה מנהל.

הפרדה לאשכולות: כשרוצים לבצע הפרדות לקבוצות שונות רוצים לוודא שכל איבר שייך לקבוצה אחת ובמידה ויש חלוקה רכה שכל איבר שייך לפחות לקבוצה אחת. אם קיבלנו אשכולות עם איברים שלא עומדים בתנאים אז למעשה האיכות של האשכול היא פחות טובה.

מדד הדימיון: חלק מהחישובים לדימיון בין איברים נעשים ע"י חישוב מרחק בין איברים חישוב של מדד זה. ישנה גם פונקציית האיכות שבודקת עד כמה אשכול מסויים הוא טוב או עד כמה חלוקה מסויימת טובה - מאוד תלוי בכל מקרה.

שטח\חלל האשכול: אשכולים מפוזרים על גבי שטחים דו, תלת מימדים ואף יותר. אם קיבלנו אשכולות שמכסים את כל השטח כנראה שהאשכול הוא חסר משמעות.

מציאת תבניות: אם האלגוריתם של האשכול מצליח לזהות תבניות במידע נקבל תוצאות טובות יותר (דבר נפוץ בכלי גרפיקה של השלמת חלקים חסרים בתמונה).

**ג.נבחר בגישת החלוקה ( partitioning method):** גישה שבה יוצרים k קבוצות () כשכל קבוצה מכילה לפחות איבר אחד. איברים יכולים להיות דומים לחלוטין אחד לשני או שלעמעם טיפה באמצעות Fuzzy Logic.

רוב השיטות הללו מבוססות מרחק ופועלות באופן איטרטיבי כשבכל איטרציה איברים יכולים לעבור מקבוצה אחת לשנייה. אובייקטים שקרובים אחד לשני יהיו שייכים לאותו האשכול ואילו איברים רחוקים יהיו שייכים לאשכולות אחרים.

**בגישה זו נבחר בשיטת ה-K Means**:

באלגוריתם K-means האיטרטיבי אנו מחשבים את הממוצע\מרכז של האשכול ואת האיברים הקרובים אליו.

שלבים:

1. מחלקים את האובייקטים המצויים בבסיס הנתונים ל- K אשכולות בצורה אקראית
2. מחשבים ערך ממוצע "מוביל" לכל קבוצה K' על פי החלוקה לקבוצות שביצענו
3. מבצעים בדיקת שיוך של האובייקטים בכל אזור אל מול ערכי הממוצע של כל אשכול. האובייקטים עם המרחקים/ההפרשים הקטנים מול אותו ממוצע ישויכו לאשכול K' המתאים
4. במידה ושיוך האובייקטים בהתאם לערכי הממוצע השתנה חוזרים לסעיף 2
5. תהליך מחזורי זה יסתיים רק לאחר שלא יתקיימו יותר תזוזות של האובייקטים בין האשכולות בהתאם להגדרת ערכי הממוצע.

בחרתי בגישה זו מאחר והיא מאוד אפקטיבית עבור בסיסי נתונים בגדלים קטנים עד בינונים, לפרוייקט שלנו עם 40787 רשומות זה יתאים. וכמו כן מאחר אלגוריתם זה מאפשר לנו לבצע אישכול גם אם אין לנו שום פרמטר לפילוח מקדים.

**נבחר בגישת הצפיפות ( Density method):** בעוד שהגישה הקודמת התבססה על מרחק בין אובייקטים ויכולות למצוא צורות דומויות למעגל, בשיטת הצפיפות ממשיכים להגדיל את האשכול כל עוד יש כמות מספיקה של איברים (צפיפות) בשטח נתון בסביבה של האשכול ללא מגבלה לצורה מסויימת. בעוד שניתן להגדיר מרחק מקסימלי ממקור האשכול ניתן לדמיין את התנהגות האלגוריתם עצמו כאש שמתפשטת כל עוד יש מספיק צפיפות של חומר בעירה.

**בגישה זו נבחר באלגוריתם DBSCAN:**

באלגוריתם זה, מחשבים את צפיפות האיברים לנקודה מסויימת. לאיזורים עם "שכונות" צפופות בוחרים נקודה באופן רנדומלי. ע"י נתון שמספק המשתמש e-nieghborhood (e>0) מגדירים את הרדיוס של השכונה. כמו כן צריך לספק עוד פרמטר שיתחשב בדחיסות MinPts - כמות נקודות מינימלית שתחשב כתנאי הסף לאזור צפוף.

שלבים:

1. בוחרים, באופן אקראי, נקודה o בתוך הנתונים.
2. מחפשים בבסיס הנתונים את כל הנקודות שניתן להשיג אותן בהתבסס על צפיפות מהנקודה p, בהינתן ערך מסוים של Eps וגם ערך מסוים של MinPts.
3. אחרי הזיהוי של כל הנקודות שניתנות להשגה בהתבסס על צפיפות מהנקודה p, בודקים אם p מהווה נקודת ליבה (core).

* אם p היא אכן נקודת הליבה אז יוצרים אשכול.
  + אם התברר ש- p היא נקודת גבול, כלומר היא ניתנת להשגה בהתבסס על צפיפות מנקודות ליבה אחרות אבל אין לה מספיק נקודות בסביבה הקרובה שלה, ברדיוס Eps, אז מנסים לבדוק נקודה אחרת בתוך בסיס הנתונים וליצור אשכול מסביב לאותה נקודה.

4) התהליך נמשך עד שכל נקודה בתוך בסיס הנתונים משויכת לאיזשהו אשכול או מוכרזת כ- outlier.

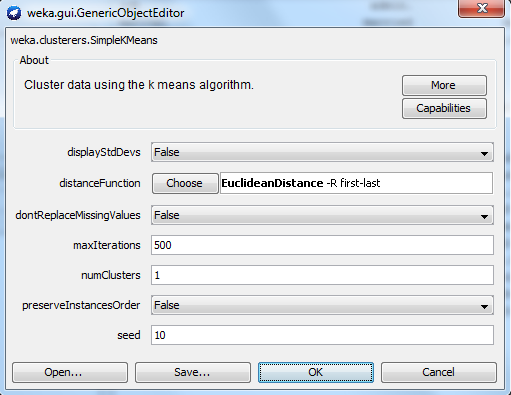
בחרתי להשתמש בגישה זו מאחר והיא לא מוגבלת במרחקים כמו הגישה הקודמת ותתן לנו פרספקטיבה נוספת לגבי בסיס הנתונים שלנו.

**ד+ה)**

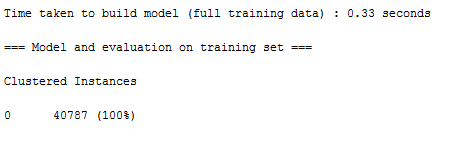
**K Means:**

קובץ הנתונים מכיל רשומות בעלות 21 מאפיינים (כולל y שאותו נרצה לנבא).

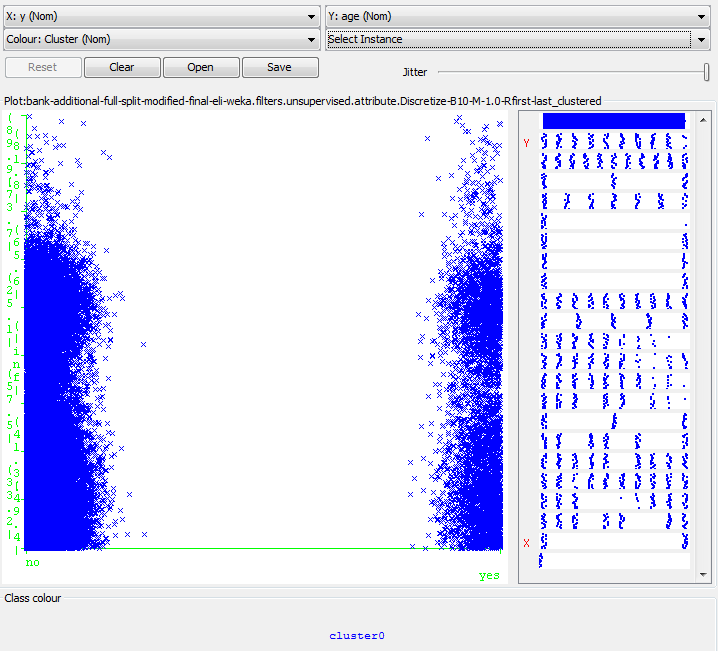
מאחר וישנם כל כך הרבה מאפיינים החלטתי בשלב הראשון להתחיל עם אשכול אחד ולקבל מושג כללי לגבי איך מפוזר המידע שלנו - אולי משם אוכל להסיק לכמה אשכולות לחלק.



בריצה נוודא שכל האיברים שייכים לאותו הCLUSTER:

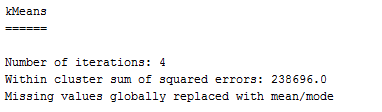


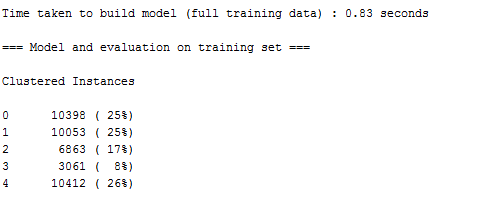
בלחיצה על Visualize Cluster Assignments נפתח, אני מכוון את ה-JITTER עד הסוף כדי לקבל תמונה יותר אמיתית לגבי הפיזורים. ישר אני רואה את הביטוי של הדיסקרטיזציה בצד ימין של החלון - החלוקה ל-10 bins בא לידי ביטוי ברור. עכשיו אני יודע גם להעריך שעד 10 אשכולות אמורים להספיק (כמובן שאפשר גם יותר).



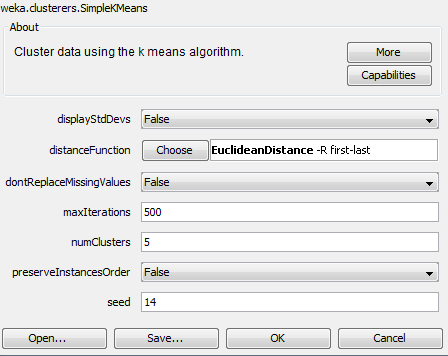
אנסה מ-1 עד 10 אשכולות ואראה מה עובד הכי טוב ובעיקר אחפש פיזור די אחיד בין האשכולות.

ראיתי שב 2 עד 6 אשכולות החלוקות היו די מאוזנות ובחרתי חלוקה ל-5 אשכולות.( kmeans\_seed10.txt)

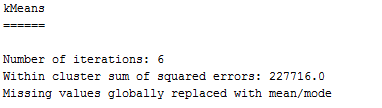


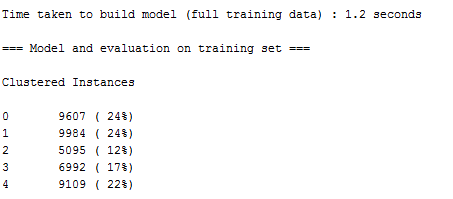


אני רוצה לראות אם אפשר לשפר את הפיזור ולהפחית את השגיאות (Within cluster sum of squared errors), ולכן אשחק עם מספר ה-SEED:



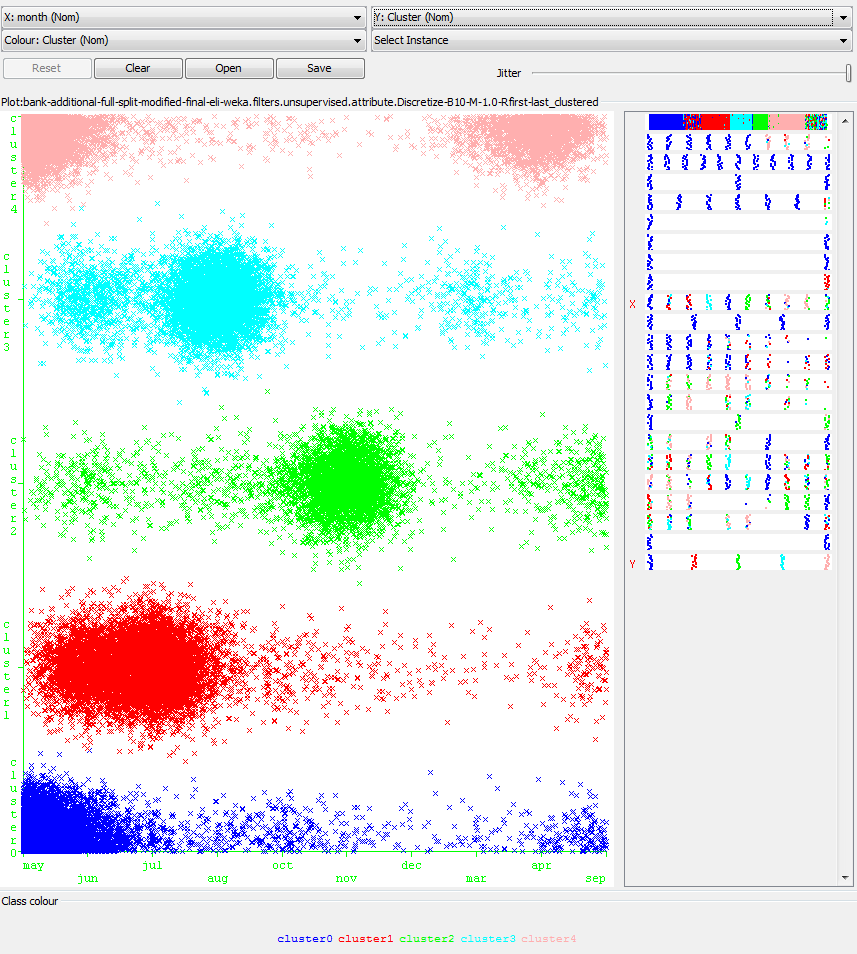
אכן קיבלתי פיזור אחיד יותר SEED 14 וסכום השגיאות המרובעות צנח ב-10980! ( kmeans\_seed14.txt)





מכאן התחלתי לבדוק את הפיזור של האשכולות על גבי כל התכונות, וראיתי שאכן האשכולות עבור כל תכונה בפני עצמה היו מאוזנות. עם זאת מצאתי כמה מדדים מעניינים:

לגבי התכונה month ביחס לכל cluster ניתן לראות במובהק שהיא השפיעה על מבני הclusters ושהחודשים "החזקים" היו מאי,יוני,יולי,אוגוסט, נובמבר, חודשים "בינונים" היו אפריל וספטמבר וחודשים "חלשים" היו מרץ,דצמבר,אוקטובר.



מכאן זה הניע אותי לחפש הצלבה של מידע בין חודש לנתונים אחרים:

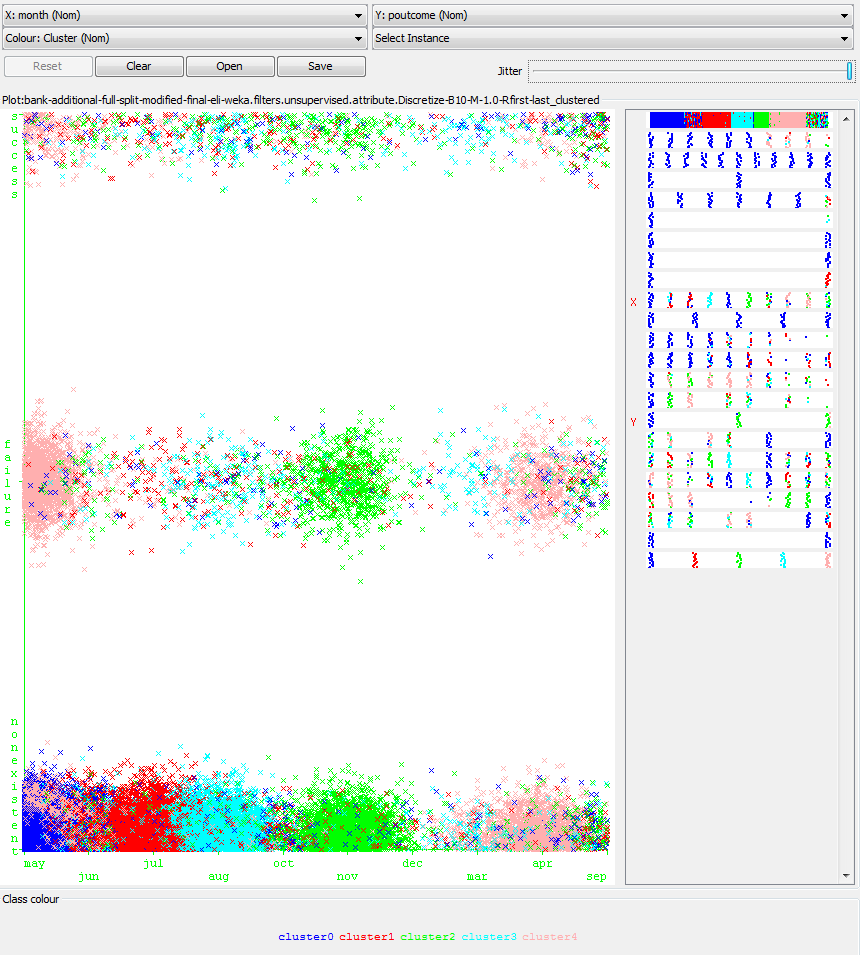
ב-poutcome - שרושם לנו הצלחה\כישלון של הקמפיינג האחרון.

אנחנו רואים שבחודשים מאי ונובמבר וקצת אפריל הקמפיינג הקודם נחל כישלון.

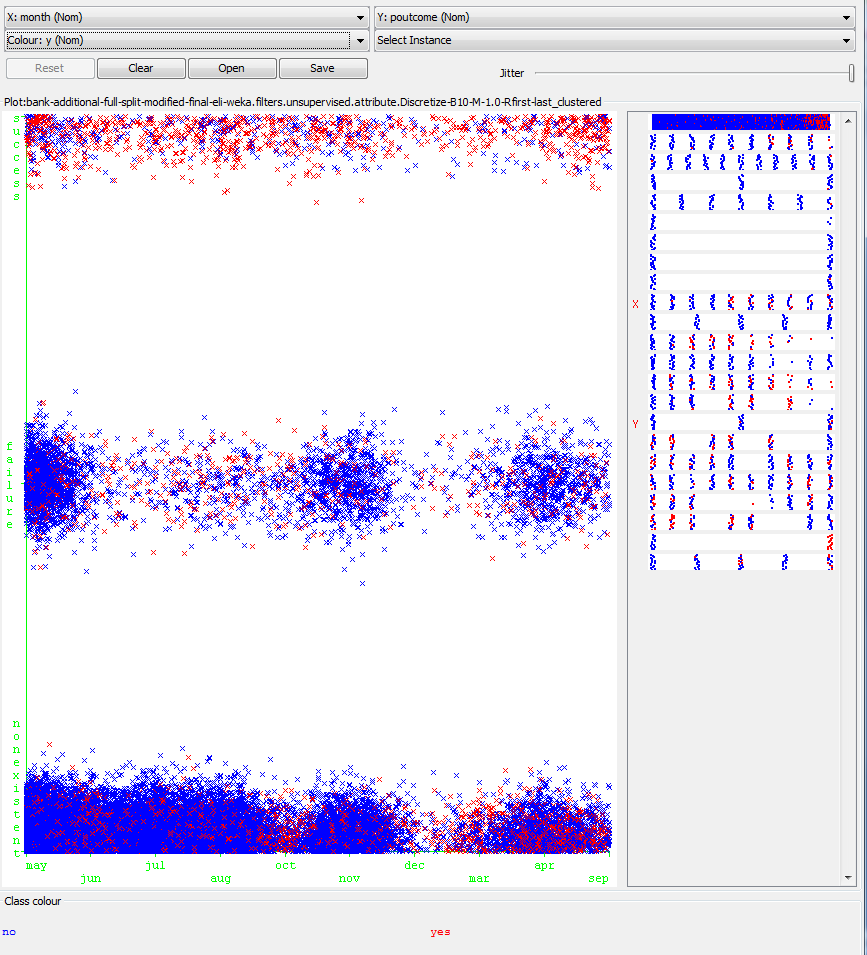
ואילו בחודשים יוני, יולי, אוגוסט היו הרבה הצלחות ביחס לכמות הכישלונות.

למרות שבמאי היו כמות גדולה של הצלחות, כמות הכשילונות באופן יחסי גדולה ומאוד משמעותית.

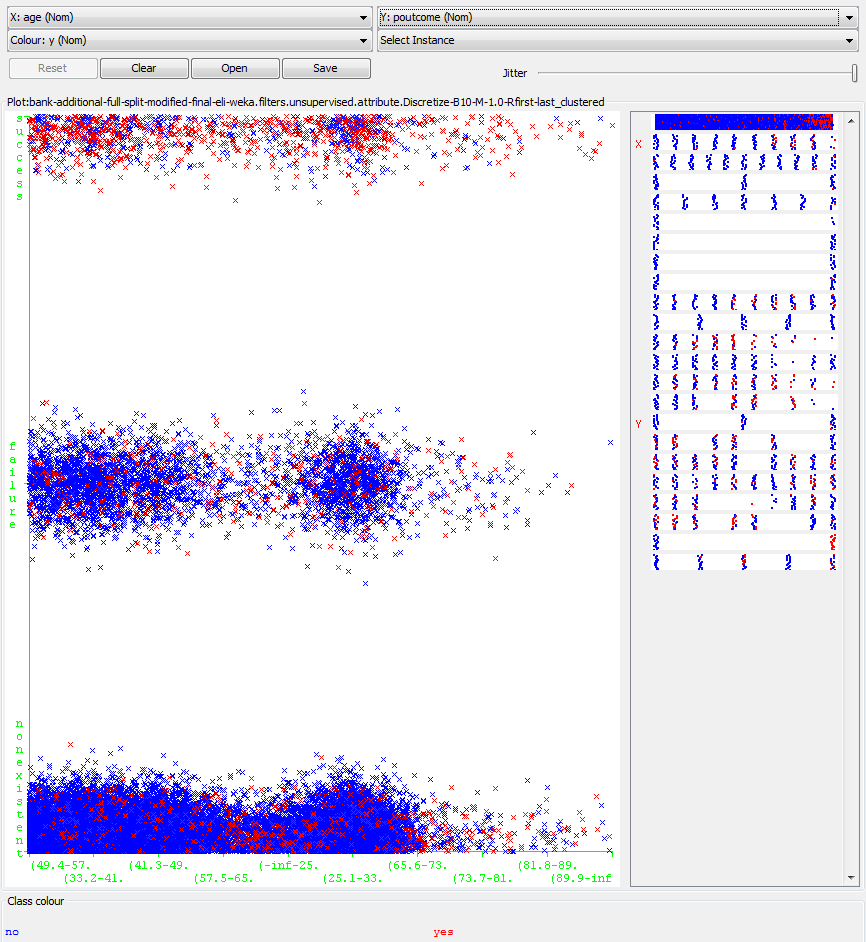
מנתון זה יהיה שמשתלם מאוד לעשות קמפיינגים בחודשי הקיץ שכן יש להם יותר סיכוי להצליח.



באותו הגרף שינוי הצבעים ללהיות מבוססי התכונה y מראה לנו שבאופן מובהק אנשים שנחלו הצלחה בקמפיין הקודם - יקחו פיקדון, ועם זאת אנשים שלא נחלו הצלחה בקמפיין הקודם - כנראה לא יקחו פיקדון.



מבט על poutcome וage (לשים לב להחלפת הצירים) יראה לנו שגילאים 49.4-57.5, 33.2-41.3ו-25.1-33.2 נוחלים יותר הצלחה.



**מסקנה:** אפשר כמובן להמשיך אך המסקנה העיקרית מצפייה בגרפים אלו היא:

שהכי טוב לעשות קמפיין בחודשי הקיץ (יוני,יולי,אוגוסט) עבור גילאים 49.4-57.5, 33.2-41.3ו-25.1-33.2 שכן גם סיכוי ההצלחה בקמפיין יותר טוב וגם לנציג הטלפוני שיתקשר לאחר הקמפיין יהיה סיכוי יותר טוב להחתים לקוח על פיקדון.

**DBSCAN**

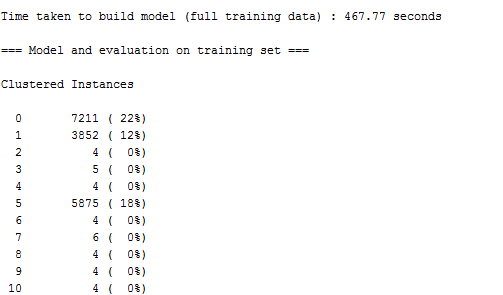
כבר מהריצה הראשונה אתה מבין שמדובר פה באלגוריתם כבד ואיטי, הריצה הראשונה על כל המידע לקחה לי יותר מ-10 דקות, כדי לחסוך זמן, הרצתי את האלגוריתם רק על 4000 שורות ובכך יכלתי לשחק מהר יותר עם המספרים (זמן ריצה היה בממוצע 4 שניות!).

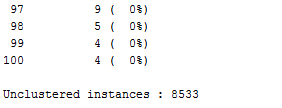
כיוונתי את minPointsל-4 (מינימום נקודות לאשכול) ומרחק Epsilon ל-1.2 (רדיוס בין נקודות), כל אשכול שלא יכיל 4 נקודות יחשב לרעש ויסומן בדו"ח הסופי בתור NOISE.

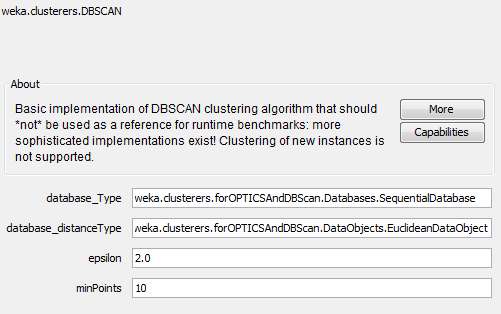
ניתן לראות את הניסיונות שלי בקבצים (DBScan\_1.2.txt, DBScan\_1.5.txt, DBScan\_2.0.txt).

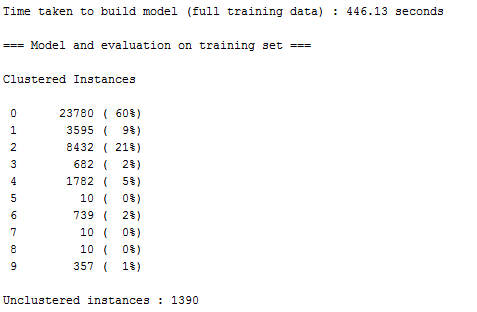
מבנה הדוח מעניין, מעבר לכמות האשכולות שנוצרו, הדו"ח מכיל כל רשומה, ואת מספר האשכול שאליה היא שייכת. וממתחת סיכום של אחוזי הפיזור של האשכולות וכמה רשומות לא שייכות לאף אשכול (שיסומנו כרעש).







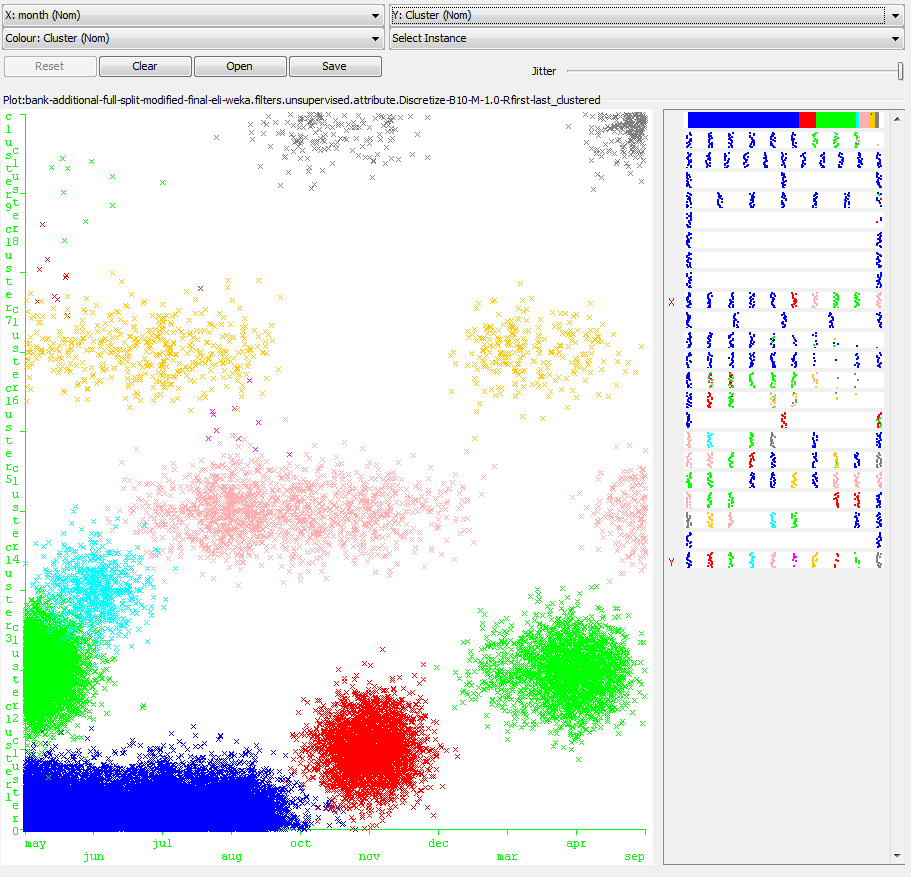
לאחר מספר לא מבוטל של נסיונות הגעתי להגדרת הפרמטרים הרצוייה:  




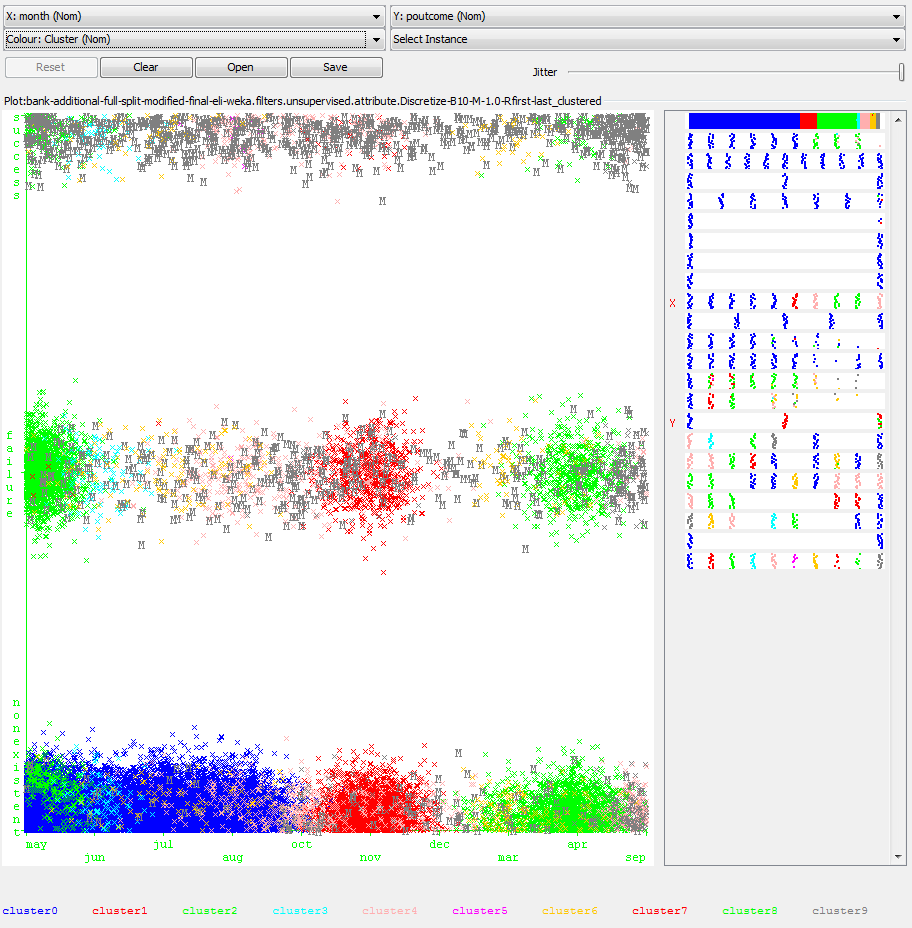
הגענו לעשרה אשכולות בגדלים שונים כאשר רק 1390 איברים נחשבים כרעש כ-3% מקובל מבחינתי.

נתחיל שוב עם בדיקת החודשים, שמתי לב שבנוסף לאשכולות ישנו אשכול אפור המייצג רעש.

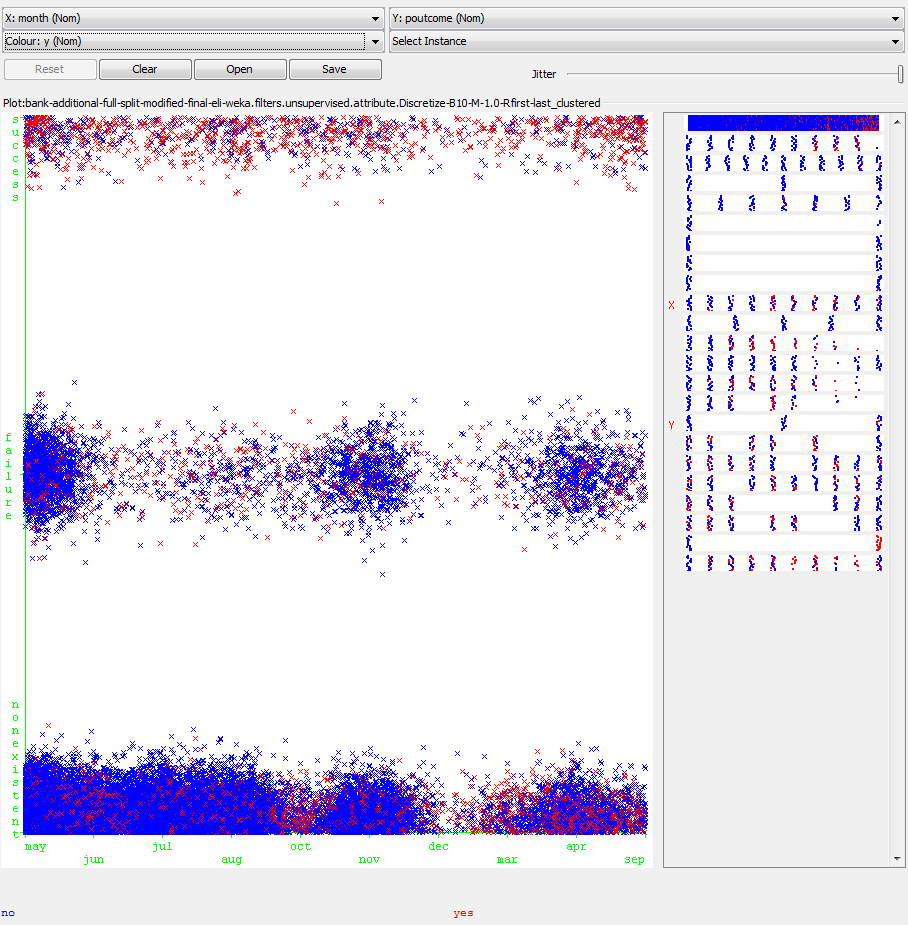
פה נראה שהחודשים "החזקים" הינם:מאי ויוני, החודשים "הבינונים" הינם יולי אוגוסט, נובמבר ואפריל, והחודשים "החלשים" הינם אוקטובר מרץ.



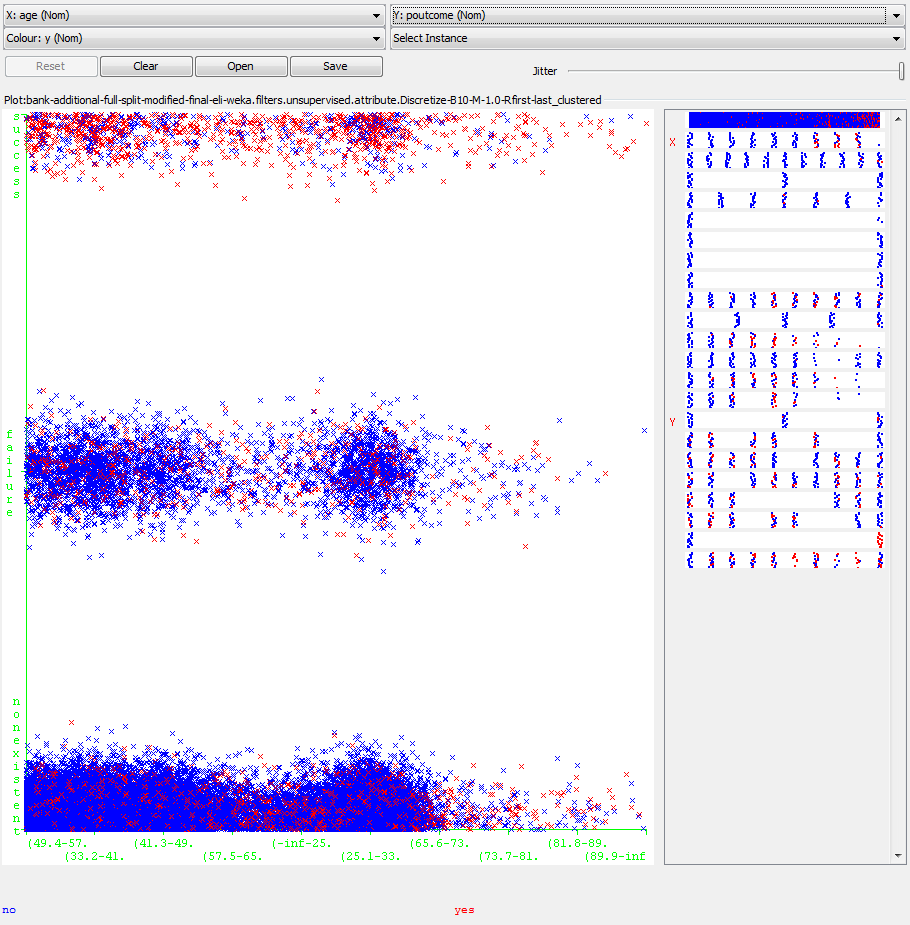
נראה שהחלוקה בשילוב עם POUTCOME זהה כלומר כמות התצפיות היא גדולה מאוד באותם חודשים חזקים ובינונים. כמות ההצלחות היא מאוד אחידה לאורך כל השנה בעוד כמות הכשלון היא בחודשי מאי נובמבר ואפריל ובשאר השנה הכשלון הוא חלש.



מכאן נעבור לראות את צבעי הערך y כדי להצליב את המידע ולהבין את יחס ההצלחה\כשלון בקמפיינג הקודם לעומת לקיחת פיקדון, פה אנו רואים שההצלחה בקמפיינג הקודם גם משפיעה על לקיחת הפיקדון אך עם זאת בחודשים יוני, יולי, אוגוסט, אוקטובר, דצמבר ואפילו קצת מרץ יש לנו אחוזי הצלחה גבוהים יחסית לאחוזי הכישלון בלקיחת פיקדון (y).



נמשיך עם הצלבה של poutcome ו-age יראה לנו שגילאים 49.4-57, 33.2-41.3ו-25.1-33.2 נוחלים יותר הצלחה לאחר קמפיינג שהיה מוצלח.



**מסקנה:** בחודשים יוני, יולי, אוגוסט, אוקטובר, דצמבר ואפילו קצת מרץ ישתלם לנו לעשות קמפיינג ולהתמקד בגילאים 49.4-57, 33.2-41.3ו-25.1-33.2 ובכך נגדיל את אחוזי ההצלחה שלנו באופן יחסי.

**סעיף ו)**

ניתן לראות בבירור ששני האלגוריתמים פועלים בשיטות שונות לחלוטין, לשם כך אציג זאת בטבלה:

|  |  |  |
| --- | --- | --- |
| קטגוריה | DBSCAN | Kmeans |
| גישה | Density | Partioning |
| מספר אשכולות | מספר האשכולות נוצר דינאמית בזמן ריצה ואינו ידוע מראש - במקרה שלנו זה היה 16. | ניתן לקבוע מראש את מספר האשכולות בעזרת פרמטר numClusters - במקרה שלנו 5. |
| מהירות | איטי מאוד על סטים גדולים - זמן חישוב מעל ל-10 דקות לריצה. | מהיר מאוד גם בסטים בגדולים - זמן חישוב הסתכם בשניות בודדות. |
| רעש | היה כ-3% רעש | לא היו רשומות עם רעש |
| אשכולות מאוזנים | לא | כן |

**מסקנה:** מעבר לכך ראינו שלמרות שהמסקנה לגבי גילאים בשני האלגוריתמים היתה זהה, החודשים בהם נרצה לעשות את הקמפיינג שונים בין שני האלגוריתמים. מאחר ובKmeans היו אשכולות מאוזנים יותר ללא רעש, והוא מהיר יותר - כנראה שנבחר בו על גבי DBSCAN.

## שאלה 3.

**סיכום:**

בממ"ן 21 התרכזנו בהגדרת הבעיה והכנת הנתונים - שזה כלל תיקון והשלמת נתונים חסרים, ביצוע דיסקרטיזציה ועוד... את הבעיה ניסינו לפתור באמצעות שיטות סיווג וחיזוי - רוב העיסוק שלנו היה חישוב מדדים והתרכזות בהכנת כל תכונה ותכונה .

בממ"ן 22 התרכזנו בלפתור את הבעיה באמצעות חוקי הקשר וניתוח אשכולות, למדנו את ההבדלים בין הביצועים של אלגוריתמים שונים וכמו כן איך ליצור קשר לוגי כללי של הנתונים ואיחודם לחוקי הקשר ואשכולות.

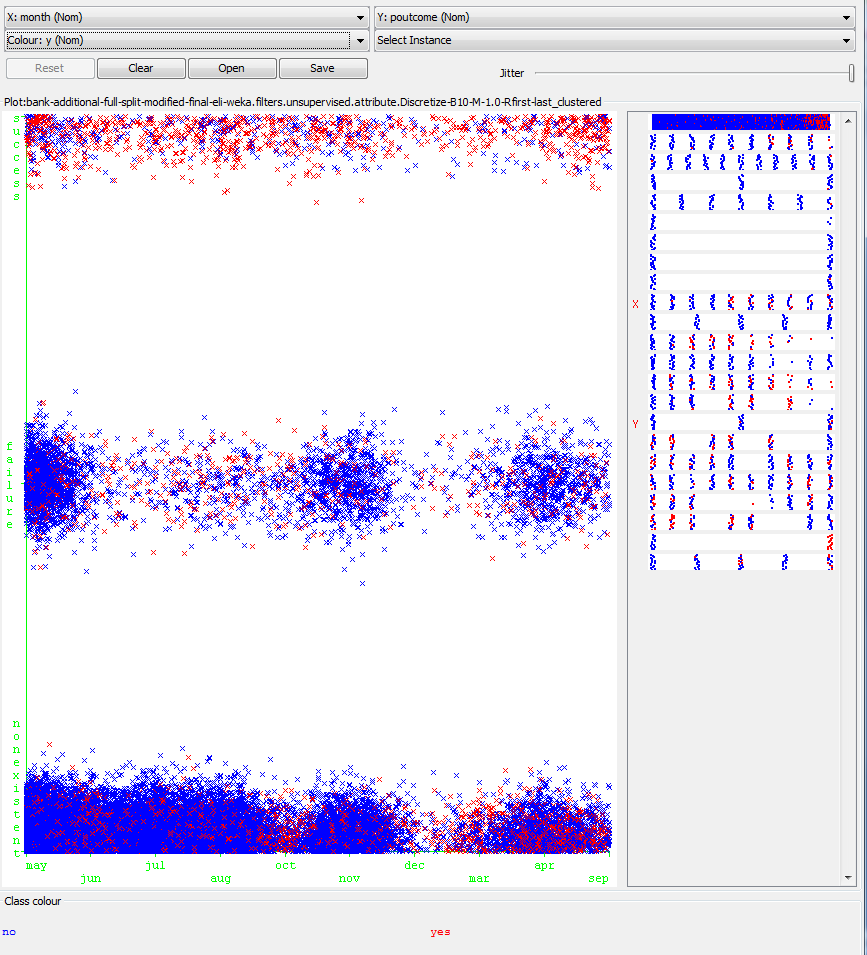
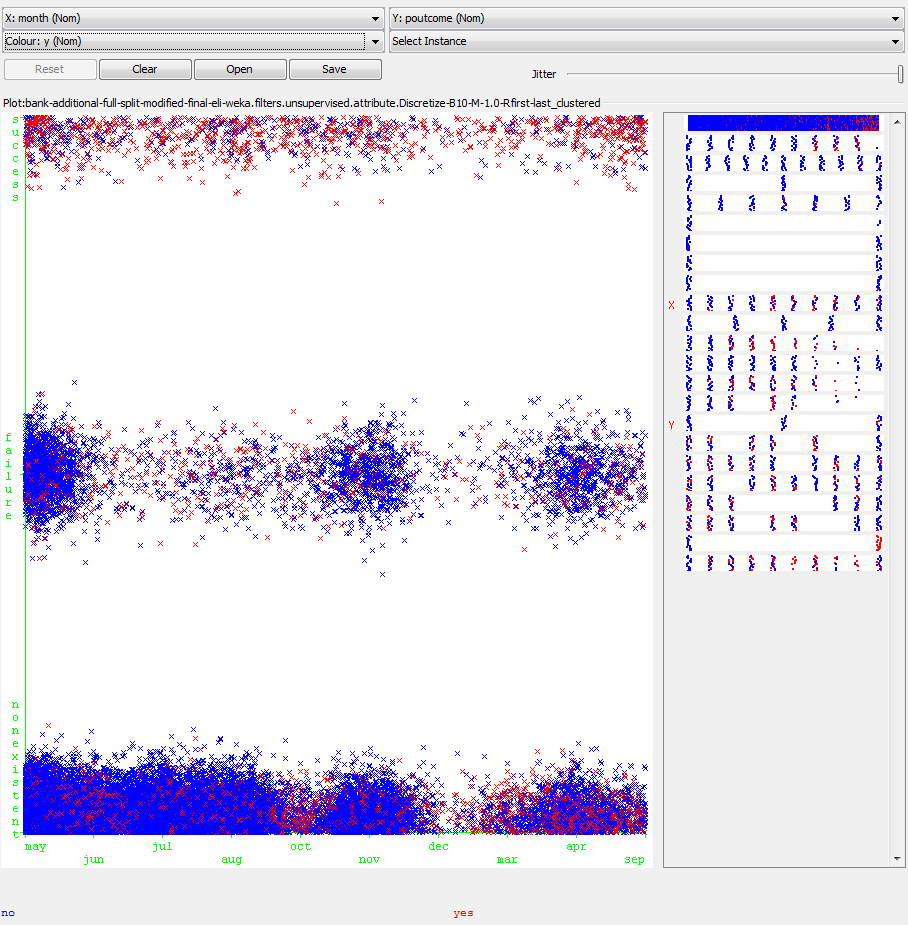
לאחר מכן לקחנו את המידע שקיבלנו (בפרט את המידע של האשכולות) הצלבנו אותו והגענו למסקנה שע"פ אנחנו יכולים לצפות מה יהיה פרופיל הלקוח שיחתום על פיקדון בשיחת טלפון תוך כדי שהתרכזנו בתכונות עיקריות על מנת להמנע מOVERFITTING.

דבר מעניין נוסף היה שכל אלגוריתם בין אם לחוקי הקשר או לאשכול מוציא סט של פלטי מידע משלו ובעל יתרונות וחסרונות שונים.

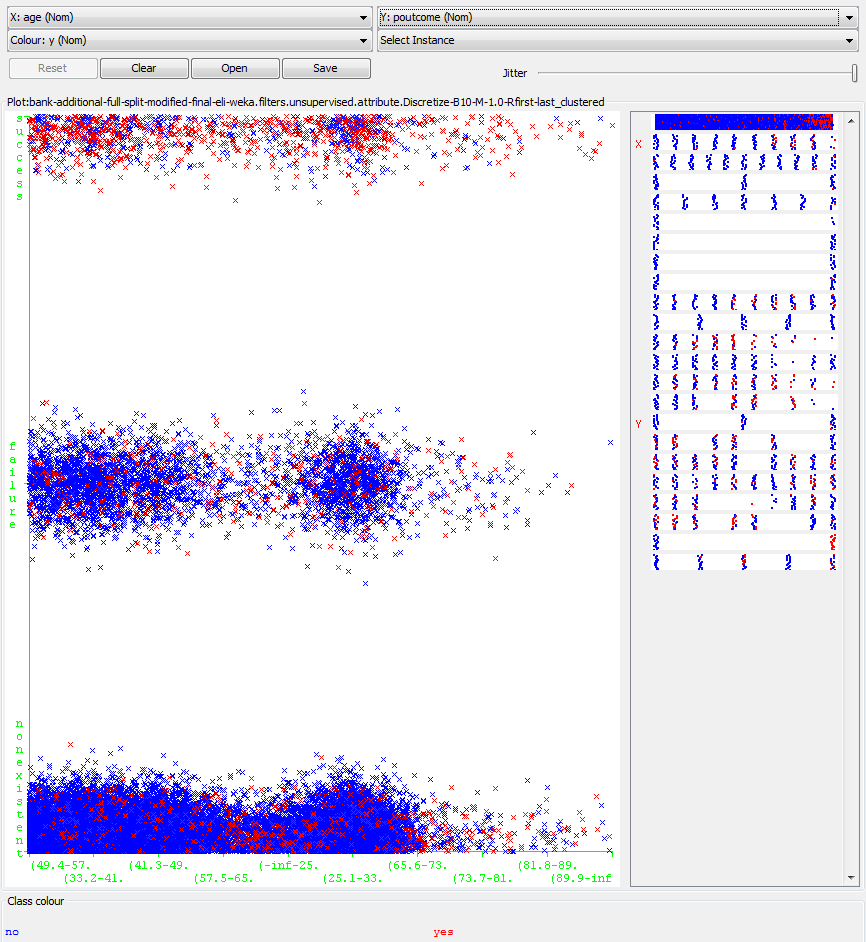
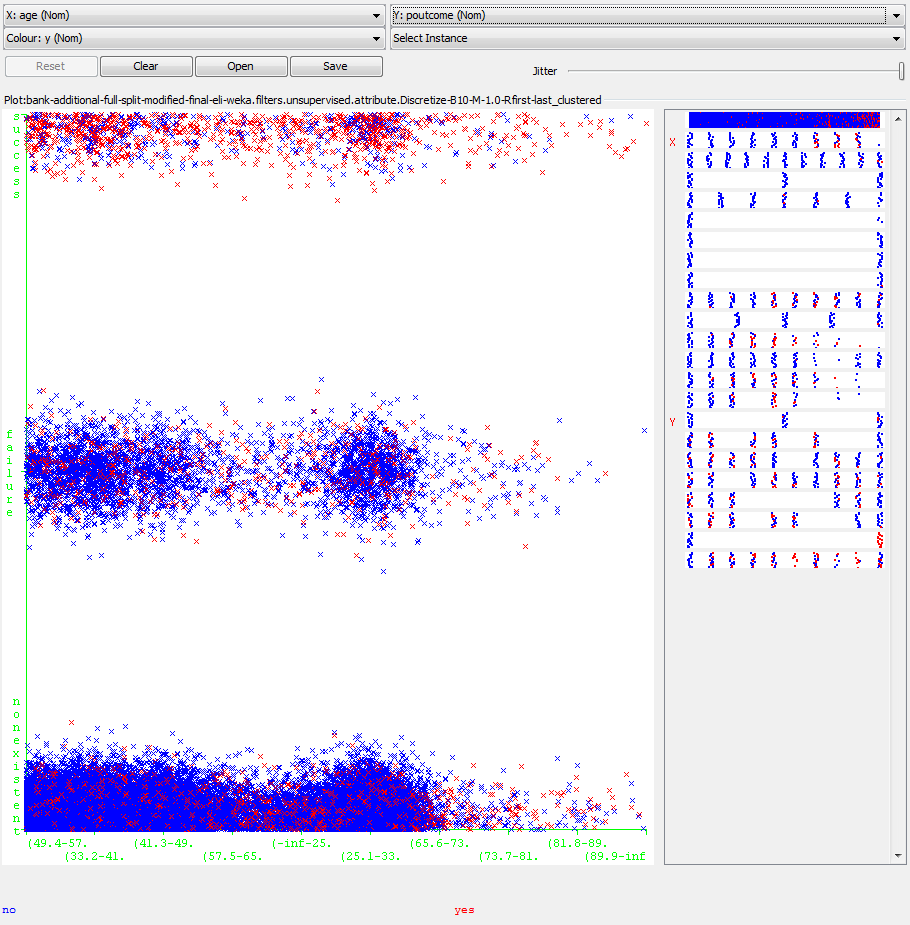
בסופו של דבר הגענו למסקנה שהלקוח הפוטנציאלי שלנו הוא בשכבות גיל 49.4-57, 33.2-41.3ו-25.1-33.2,

שהחליט לחתום על קמפיינג בעבר ושעלינו לרכז את מאמצנו בחודשי הקיץ ואולי בחודשים נוספים בהתאם לאלגוריתם שבחרנו לאישכול.

חודש-תוצאת קמפיינג:

גיל-תוצאת קמפיינג:



היה מעניין ומשכיל, תודה רבה.