א.<u>אלגוריתם A-priori:</u>

האלגוריתם הומצא בשנת 1994 על ידי: Agrawal ו- Srikant .

האלגוריתם מוצא את הפריטים השכיחים ובעזרת תכונת האפריורית(תכונה שדורשת שעבור קבוצת פרטים, כל תת-קבוצה שלה גם כן שכיחה),מרחיב אותם לקבוצות פריטים שכיחות.

האלגוריתם הוא איטרטיבי המשתמש בסריקת BFS ועץ גיבוב למנות את הסטים המועמדים, כאשר הוא מרכיב סט מועמד באורך k מסטים באורך k-1 ואז גוזם את המועמדים שיש להם תת-קבוצות לא שכיחות, כלומר באיטרציה ה k נמצא את כל קבוצת הפרטים השכיחות בגודל k.

בכל איטרציה מתבצעים שלושה מעברים עקרים:

a. 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.
b. גיזום מועמדים שלא מקיימים את התכונה האפריורית.
c. בדיקת התמיכה של קבוצות המועמדים שלא נגזמו בשלב 2 על ידי מניית מספר התנועות שבהן הם מופיעים בתוך בסיס הנתונים עצמו, כלומר בדיקה האם הקבוצות המועמדות באמת שכיחות. צעד זה מתבצע באמצעות עץ גיבוב.

אלגוריתם זה נבחר עקב פשטותו, תכונת האפריורית שלו טריוויאלית והשלבי הביצוע יחסית פשוטים, כמו כן יש עליו המון חומר והוא נלמד בצו רה רחבה כך שהוא מהווה בסיס טוב להשוואה לשאר אלגוריתמים.

<u>אלגוריתם FP-growth:</u>

האלגוריתם הוצע על ידי: Jiawei Han, Jian Pei, and Yiwen Yin.

שיטה יעילה וסקלבילית עבור כריית סט שלם של דפוסים שכיחים על ידי הצמחת חתיכות של תבניות, באמצעות מבנה של עץ תחיליות לאחסון מידע דחוס וחיוני על דפוסי שכיחים, המבנה נקרא- עץ דפוסים שכיחים(FP tree).

האלגוריתם מוצא קבוצות שכיחות ללא יצירת מועמדים ולשם כך משתמש בDFS ובשיטת הפרד ומשול. הוא דוחס את מסד הנתונים לכדי יצירת FP tree שמייצג קבוצות שכיחות, מחלק את מסד הנתונים לבסיסי נתונים מותנים שכל אחד משוייך לדפוס שכיח וכל מסד נתונים נכרה באופן עצמאי

שלבי פעולת האלגוריתם:

a. סופר את המופעים של פריטים במסד הנתונים.
b. פרטים שאינם עוברים את הסף הדרוש(שכיחות) נזרקים.
c. ממיינים את הפריטים.
d. בונה FP tree, עבור כל טרנזקציה מוסיפים את הפריטים לפי הסדר שהם הופיעו בשלב c, כמו כן אם פריט עבר הופיע אז מגדילים את המופע שלו ב1, כך שאם לשני טרנזקציות יש ענף תחילית משותפת כלשהי אז הפריטים בתחיליות יגדילו את כמותם ב1.
e. עוברים על כל ענף של העץ וכוללים רק את הפריטים שהמספר שלהם עובר את הסף ,אלו הקבוצות השכיחות שמהם יבנו חוקי הקשר.

אלגוריתם זה נבחר עקב יעילותו, בסריקה יש לקרוא את הנתונים רק פעמיים, עלות החיפוש עבור דפוסים קצרים באופן רקורסיבי ואז חיבורים לדפוסים שכיחים ארוכים קטנה והמידע מאוחסן בצורה קומפקטית.

ב.לפני מציאת קבוצות הנתונים השכיחות, צריך לבצע דיסקרטיזציה לנתונים, כיוון שאלגוריתמי חוקי ההקשר דורשים נתונים דיסקרטיים.

בהמשך לעבודה על ממן 21, נלקחו הנתונים מגיליון אקסל "ניקוי הנתונים" של ממן 21 תחת

לשונית "המרה סופי - עץ החלטה" וכעת נמצאים בגיליון אקסל "ממן 22 אקסל" תחת לשונית "ניקוי הנתונים ממן 21",הנתונים האלו עברו מילוי של ערכים חסרים ודיסקטיזציה של טווח מחירי הרכבים.

עבור האלגוריתם הא-פריורי, יש צורך לבצע דיסקרטיזציה גסה למספר קטן של bins עקב הכדאיות להשתמש בכמה שיותר תכונות לגילוי חוקי הקשר(במקום להשמיט תכונות) ועקב הדרישה לתמיכה גבוהה(40%). מספר הbins, שווי עומק, שיחולקו התכונות יהיו 2 – מעל החציון ומתחת לו.
ביצוע הדיסקרטיזציה יהיה רק לתכונות הquantitative ולא לתכונות הנומינליות כיוון שהשמת ערכים שבוודאות יודעים שהם שונים זה מזה(למשל מערכת דלק) יהיה שגוי למרות הרצון לא להשמיט תכונות.
הדיסקרטיזציה:

תכונה	סוג	bins
symboling	מספר בדיד אינטרוולי	החציון: 1 [-2,1],[1,+3]:bins
normalized-losses	מספר רציף אינטרוולי	החציון: 119 [56,119],[119,256]:bins
wheel-base	מספר רציף אינטרוולי	החציון: 97 [88.6,97],[97,120.9]:bins
length	מספר רציף אינטרוולי	החציון: 173.2 [141.1,173.2],[173.2,208.1]:bins
width	מספר רציף אינטרוולי	החציון: 65.5 [60.3,65.5],[65.5,72.3]:bins
height	מספר רציף אינטרוולי	החציון: 54.1 [47.8,54.1],[54.1,59.8]:bins
curb-weight	מספר רציף אינטרוולי	החציון: 2414 [1488,2414],[2414,2555.566]:bins
engine-size	מספר רציף אינטרוולי	החציון: 120 [61,120],[120,326]:bins
bore	מספר רציף אינטרוולי	החציון: 3.31 [2.54,3.31],[3.31,3.94]:bins
stroke	מספר רציף אינטרוולי	החציון: 3.29 [2.07,3.29],[3.29,4.17]:bins
compression-ratio	מספר רציף אינטרוולי	החציון: 9 [7,9],[9,23]:bins
horsepower	מספר רציף אינטרוולי	החציון: 95 [48,95],[95,288]:bins
peak-rpm	מספר רציף אינטרוולי	החציון: 5200 [4150,5200],[5200,6600]:bins
City-mpg	מספר רציף אינטרוולי	החציון: 24 [13,24],[24,49]:bins
highway-mpg	מספר רציף אינטרוולי	החציון: 30 [16,30], [30,54]:bins
price:תכונת החיזוי	מספר בדיד אינטרוולי	החציון:10345 [5118,10345], [10345,45400]:bins

טבלה 1: הכנת הנתונים לחוקי ההקשר על פי האלגוריתם הא-פריורי.
לכל נתון מצוין החציון והערכים הנכנסים לכל bin

כאשר לכל הערכים הקטנים לחציון יכנסו לbin בשם "A" וכל הערכים השווים\גדולים מהחציון יכנסו לbin בשם "B".

עבור האלגוריתם fp-Growth, יש צורך לבצע שכל התכונות יהיו בינאריות, לכן תכונות fuel-system,num-of-cylinders,engine-type,drive-wheels,body-style,make הוסרו עקב אי הידיעה איך לפצל את התכונות לשני bins בינארים.

ג.+ ד.

רציתי למצוא את כל החוקים, לשם כך בוצעו מספר הרצות, שבכל הרצה הוגדל פרמטר מספר החוקים המבוקש עד שלא היה שינוי במספר החוקים המתקבל, כמו כן פרמטר האינטרוול(delta) נקבע ל0.5 ושונה עד ל 5×10^{-5}, במכפלות של 10^{-1} לוודא שאכן קיבלנו את כל חוקי ההקשר.

בהרצות נבדקו ערכים נוספים מלבד אלו שנדרשו על מנת לראות את השוני שמשרה כל מדד. נבדקו ערכים שונים של תמיכה, סוגי מדד שונים ליצירת החוקים- confidence ו- lift(הקורלציה של המאורעות, כמה החוק ההקשר מצליח בניבוי יחסית לאחוז באוכלוסייה) וערכים שונים למדדים.

<u>אלגוריתם A-priori:</u>

להלן טבלה עם הפרמטרים השונים שהוזנו והתוצאות שהתקבלו:

מספר הרצה	סוג מדד	מינימום מהמדד	תמיכה	מספר קבוצות תדירות	מספק חוקים חזקים	דוגמא לקבוצות תדירות	דוגמא לחוקים חזקים	זמן ריצה בשניות בדלתא 5×10^{-5}
1	confidence	0.6	0.4	973	10053	דוגמה 1		19
2	confidence	0.6	0.5	109	418	דוגמה 2		4.66
3	confidence	0.6	0.6	23	52	דוגמה 3		3
4	confidence	0.7	0.4	973	8905	דוגמה 4		18.20
5	confidence	0.8	0.4	973	7043	דוגמה 5		17.50
6	confidence	0.5	0.4	973	11301	דוגמה 6		19.19
7	lift	2	0.4	973	16	דוגמה 7		15.27
8	lift	0.5	0.4	973	12086	דוגמה 8		20.30
9	lift	1.5	0.4	973	5656	דוגמה 9		16.29

טבלה 2: הרצות שונות של האלגוריתם האפריורי עם מדדים שונים וזמן הריצה לכל הרצה. מספר ההרצה מקביל למספרי מסמכי ההרצות שבתיקיית "הרצת חוקי הקשר-A-PRIORI<".

- דוגמאות להרצות -

דוגמה 1:

קבוצות תדירות: aspiration=std,engine-location=front,body-style=sedan

חוקי הקשר חזקים:

num-of-cylinders=four 161 ==> engine-location=front 161 <conf:(1)> lift:(1.02) lev:(0.01) [3] conv:(3.1(

engine-type=ohc 150 ==> engine-location=front 150 <conf:(1)> lift:(1.02) lev:(0.01) [2] conv:(2.88(

דוגמה 2:

קבוצות תדירות:,city-mpg=B,fuel-type=gas,engine-location=front

חוקי הקשר חזקים:

fuel-type=gas city-mpg=B 108 ==> engine-location=front 108 <conf:(1)> lift:(1.02) lev:(0.01) [2] conv:(2.08(

engine-type=ohc city-mpg=B 107 ==> engine-location=front 107 <conf:(1)> lift:(1.02) lev:(0.01) [2] conv:(2.06(

fuel-type=gas num-of-cylinders=four city-mpg=B 107 ==> engine-location=front 107 <conf:(1)> lift:(1.02) lev:(0.01) [2] conv:(2.06(

דוגמה 3:

קבוצות תדירות: fuel-type=gas,num-of-cylinders=four,engine-type=ohc

חוקי הקשר חזקים:
engine-location=front 204 ==> fuel-type=gas aspiration=std 158 <conf:(0.77)> lift:(1) lev:(0) [0] conv:(0.98(
engine-location=front 204 ==> engine-type=ohc 150 <conf:(0.74)> lift:(1.02) lev:(0.01) [2] conv:(1.03(
fuel-type=gas engine-location=front 183 ==> engine-type=ohc 134 <conf:(0.73)> lift:(1.02) lev:(0.01) [2] conv:(1.02 (

דוגמה 4:
קבוצות תדירות: fuel-type=gas, drive-wheels=fwd,height=B
חוקי הקשר חזקים:
width=B 106 ==> engine-location=front 106 <conf:(1)> lift:(1.02) lev:(0.01) [2] conv:(2.04(
height=B 106 ==> engine-location=front 106 <conf:(1)> lift:(1.02) lev:(0.01) [2] conv:(2.04(
engine-type=ohc num-of-cylinders=four city-mpg=B 106 ==> engine-location=front 106 <conf:(1)> lift:(1.02) lev:(0.01) [2] conv:(2.04(

דוגמה 5:
קבוצות תדירות: city-mpg=B, city-mpg=B,engine-location=front
חוקי הקשר חזקים:
drive-wheels=fwd city-mpg=B 99 ==> engine-location=front 99 <conf:(1)> lift:(1.02) lev:(0.01) [1] conv:(1.9(
engine-type=ohc compression-ratio=B 99 ==> engine-location=front 99 <conf:(1)> lift:(1.02) lev:(0.01) [1] conv:(1.9(
fuel-type=gas drive-wheels=fwd engine-type=ohc 99 ==> engine-location=front 99 <conf:(1)> lift:(1.02) lev:(0.01) [1] conv:(1.9(

דוגמה 6:
קבוצות תדירות: peak-rpm=A, engine-type=ohc,length=B
חוקי הקשר חזקים:
aspiration=std highway-mpg=B 103 ==> engine-location=front 103 <conf:(1)> lift:(1.02) lev:(0.01) [1] conv:(1.98)
41. fuel-type=gas drive-wheels=fwd num-of-cylinders=four 103 ==> engine-location=front 103 <conf:(1)> lift:(1.02) lev:(0.01) [1] conv:(1.98)
peak-rpm=A 102 ==> engine-location=front 102 <conf:(1)> lift:(1.02) lev:(0.01) [1] conv:(1.96)

דוגמה 7:
קבוצות תדירות: city-mpg=B, engine-location=front,price=A
חוקי הקשר חזקים:
engine-location=front horsepower=A highway-mpg=B 93 ==> aspiration=std drive-wheels=fwd city-mpg=B 83 conf:(0.89) < lift:(2.04)> lev:(0.2) [42] conv:(4.76)
engine-location=front curb-weight=A 102 ==> length=A price=A 83 conf:(0.81) < lift:(2.01)> lev:(0.2) [41] conv:(3.04)
length=A price=A 84 ==> engine-location=front curb-weight=A 83 conf:(0.99) < lift:(2.01)> lev:(0.2) [41] conv:(21.4)

דוגמה 8:
קבוצות תדירות: curb-weight=A, drive-wheels=fwd,length=A
חוקי הקשר חזקים:
drive-wheels=fwd curb-weight=A 89 ==> engine-location=front highway-mpg=B price=A 83 conf:(0.93) < lift:(2)> lev:(0.2) [41] conv:(6.79)
drive-wheels=fwd engine-location=front curb-weight=A 89 ==> highway-mpg=B price=A 83 conf:(0.93) < lift:(2)> lev:(0.2) [41] conv:(6.79)
highway-mpg=B price=A 97 ==> drive-wheels=fwd engine-location=front curb-weight=A 83 conf:(0.86) < lift:(2)> lev:(0.2) [41] conv:(3.7)

דוגמה 9:
קבוצות תדירות: horsepower=A, length=B,highway-mpg=B
חוקי הקשר חזקים:
city-mpg=B price=A 97 ==> engine-location=front curb-weight=A num-of-cylinders=four highway-mpg=B 83 conf:(0.86) < lift:(1.98)> lev:(0.2) [41] conv:(3.67)
engine-location=front city-mpg=B price=A 97 ==> curb-weight=A num-of-cylinders=four highway-mpg=B 83 conf:(0.86) < lift:(1.98)> lev:(0.2) [41] conv:(3.67)
engine-location=front curb-weight=A highway-mpg=B 91 ==> num-of-cylinders=four city-mpg=B price=A 83 conf:(0.91) < lift:(1.98)> lev:(0.2) [41] conv:(5.44))

<u>אלגוריתם FP-growth:</u>
להלן טבלה עם הפרמטרים השונים שהוזנו והתוצאות שהתקבלו:

מספר הרצה	סוג מדד	מינימום מהמדד	תמיכה	positive Index	מספק חוקים חזקים	דוגמא לקבוצות תדירות	דוגמא לחוקים חזקים	זמן ריצה בשניות בדלתא 5×10^{-5}
1	confidence	0.6	0.4	2	18	דוגמה 1		1.06
2	confidence	0.6	0.4	1	708	דוגמה 2		3
3	confidence	0.6	0.5	2	-	-		1.19
4	confidence	0.6	0.5	1	80	דוגמה 3		1.19
5	confidence	0.7	0.4	2	18	דוגמה 4		1.07
6	confidence	0.7	0.4	1	650	דוגמה 5		3.09
7	confidence	0.8	0.4	2	17	דוגמה 6		1.31
8	confidence	0.8	0.4	1	543	דוגמה 7		2.73
9	confidence	0.5	0.4	2	18	דוגמה 8		1.04
10	confidence	0.5	0.4	1	843	דוגמה 9		2.79
12	lift	2	0.4	2	-	-		1.15
13	lift	2	0.4	1	-	-		2.8
14	lift	0.5	0.4	2	18	דוגמה 10		1.29
15	lift	0.5	0.4	1	970	דוגמה 11		2.88
16	lift	1.5	0.4	2	18	דוגמה 12		1.25
17	lift	1.5	0.4	1	300	דוגמה 13		2.95

טבלה 3: הרצות שונות של האלגוריתם FP-growth עם מדדים שונים וזמן הריצה לכל הרצה. מספר ההרצה מקביל למספרי מסמכי ההרצות שבתיקיית "הרצת חוקי הקשר-FP-GROWTH>".

- דוגמאות להרצות -

דוגמה 1:
קבוצות תדירות: curb-weight=B, length=A,wheel-base=A
חוקי הקשר חזקים:
[engine-size=B]: 104 ==> [curb-weight=B]: 94 <conf:(0.9)> lift:(1.78) lev:(0.2) conv:(4.66)
[curb-weight=B]: 105 ==> [engine-size=B]: 94 <conf:(0.9)> lift:(1.78) lev:(0.2) conv:(4.35)
[length=A]: 102 ==> [wheel-base=A]: 90 <conf:(0.88)> lift:(1.77) lev:(0.19) conv:(3.94)

דוגמה 2:
קבוצות תדירות: curb-weight=B, length=A,wheel-base=A
חוקי הקשר חזקים:

[compression-ratio=B]: 138 ==> [aspiration=std, engine-size=A]: 83 <conf:(0.6)> lift:(1.37) lev:(0.11) conv (1.38):
[compression-ratio=B]: 138 ==> [engine-location=front, aspiration=std, engine-size=A]: 83 <conf:(0.6)> lift:(1.37) lev:(0.11) conv (1.38):
[engine-location=front, fuel-type=gas, aspiration=std]: 158 ==> [highway-mpg=B]: 95 <conf:(0.6)> lift:(1.07) lev:(0.03) conv(1.09):

דוגמה 3:
קבוצות תדירות: compression-ratio=B ,aspiration=std, engine-location=front
חוקי הקשר חזקים:
[engine-location=front, compression-ratio=B]: 135 ==> [aspiration=std]: 119 <conf:(0.88)> lift:(1.08) lev:(0.04) conv (1.46):
[city-mpg=B]: 125 ==> [aspiration=std]: 110 <conf:(0.88)> lift:(1.08) lev:(0.04) conv (1.43):
[city-mpg=B]: 125 ==> [engine-location=front, aspiration=std]: 110 <conf:(0.88)> lift:(1.1) lev:(0.05) conv(1.55):

דוגמה 4:
קבוצות תדירות: horsepower=A,price=A ,width=A
חוקי הקשר חזקים:
[wheel-base=A]: 103 ==> [width=A]: 86 <conf:(0.83)> lift:(1.71) lev:(0.17) conv (2.93):
[price=A]: 103 ==> [width=A]: 86 <conf:(0.83)> lift:(1.71) lev:(0.17) conv (2.93):
[price=A]: 103 ==> [horsepower=A]: 86 <conf:(0.83)> lift:(1.75) lev:(0.18) conv(2.99):

דוגמה 5:
קבוצות תדירות: engine-location=front,compression-ratio=B , city-mpg=B
חוקי הקשר חזקים:
[city-mpg=B]: 125 ==> [aspiration=std, compression-ratio=B]: 89 <conf:(0.71)> lift:(1.21) lev:(0.07) conv (1.39):
[city-mpg=B]: 125 ==> [engine-location=front, aspiration=std, compression-ratio=B]: 89 <conf:(0.71)> lift:(1.24) lev:(0.08) conv (1.44):
[engine-location=front, city-mpg=B]: 125 ==> [aspiration=std, compression-ratio=B]: 89 <conf:(0.71)> lift:(1.21) lev:(0.07) conv (1.39):
[compression-ratio=B]: 138 ==> [city-mpg=B]: 98 <conf:(0.71)> lift:(1.18) lev:(0.07) conv(1.33):

דוגמה 6:
קבוצות תדירות: length=A,price=A ,width=A
חוקי הקשר חזקים:
[width=A]: 101 ==> [price=A]: 86 <conf:(0.85)> lift:(1.71) lev:(0.17) conv (3.17):
[width=A]: 101 ==> [length=A]: 86 <conf:(0.85)> lift:(1.73) lev:(0.18) conv (3.2):
[length=A]: 102 ==> [width=A]: 86 <conf:(0.84)> lift:(1.73) lev:(0.18) conv(3.07):

דוגמה 7:
קבוצות תדירות: length=A,price=A ,width=A
חוקי הקשר חזקים:

[engine-location=front, city-mpg=B]: 125 ==> [aspiration=std, highway-mpg=B]: 100 <conf:(0.8)> lift:(1.61) lev:(0.18) conv:(2.42)
[aspiration=std, city-mpg=B]: 110 ==> [engine-location=front, curb-weight=A]: 88 <conf:(0.8)> lift:(1.62) lev:(0.16) conv:(2.43)
[engine-location=front, aspiration=std, city-mpg=B]: 110 ==> [curb-weight=A]: 88 <conf:(0.8)> lift:(1.62) lev:(0.16) conv:(2.43)

דוגמה 8:
קבוצות תדירות: curb-weight=B, length=A,wheel-base=A
חוקי הקשר חזקים:
[engine-size=B]: 104 ==> [curb-weight=B]: 94 <conf:(0.9)> lift:(1.78) lev:(0.2) conv:(4.66)
[curb-weight=B]: 105 ==> [engine-size=B]: 94 <conf:(0.9)> lift:(1.78) lev:(0.2) conv:(4.35)
[length=A]: 102 ==> [wheel-base=A]: 90 <conf:(0.88)> lift:(1.77) lev:(0.19) conv:(3.94)

דוגמה 9:
קבוצות תדירות: fuel-type=gas,highway-mpg=B ,engine-location=front
חוקי הקשר חזקים:
[engine-location=front, fuel-type=gas]: 183 ==> [aspiration=std, city-mpg=B, highway-mpg=B]: 92 <conf:(0.5)> lift:(1.04) lev:(0.02) conv:(1.03)
[engine-location=front]: 204 ==> [peak-rpm=B]: 102 <conf:(0.5)> lift:(0.99) lev:(-0.01) conv:(0.98)
[engine-location=front]: 204 ==> [curb-weight=A]: 102 <conf:(0.5)> lift:(1.01) lev:(0.01) conv:(1)

דוגמה 10:
קבוצות תדירות: engine-size=B,curb-weight=B, bore=B
חוקי הקשר חזקים:
[engine-size=B]: 104 ==> [bore=B]: 90 conf:(0.87) <lift:(1.63)> lev:(0.17) conv:(3.25)
[bore=B]: 110 ==> [curb-weight=B]: 86 conf:(0.78) <lift:(1.54)> lev:(0.15) conv:(2.17)
[curb-weight=B]: 105 ==> [bore=B]: 86 conf:(0.82) <lift:(1.54)> lev:(0.15) conv:(2.46)

דוגמה 11:
קבוצות תדירות: engine-location=front,fuel-type=gas, stroke=B
חוקי הקשר חזקים:
[stroke=B]: 108 ==> [engine-location=front, fuel-type=gas]: 87 conf:(0.81) <lift:(0.91)> lev:(-0.04) conv:(0.57)
[fuel-type=gas]: 186 ==> [stroke=B]: 87 conf:(0.47) <lift:(0.9)> lev:(-0.05) conv:(0.89)
[stroke=B]: 108 ==> [fuel-type=gas]: 87 conf:(0.81) <lift:(0.9)> lev:(-0.05) conv:(0.5)

דוגמה 12:
קבוצות תדירות: width=A, length=A,price=A
חוקי הקשר חזקים:
[price=A]: 103 ==> [width=A]: 86 conf:(0.83) <lift:(1.71)> lev:(0.17) conv:(2.93)
[length=A]: 102 ==> [price=A]: 84 conf:(0.82) <lift:(1.66)> lev:(0.16) conv:(2.7)
[price=A]: 103 ==> [length=A]: 84 conf:(0.82) <lift:(1.66)> lev:(0.16) conv:(2.61)

דוגמה 13:
קבוצות תדירות: city-mpg=B,engine-location=front, aspiration=std
חוקי הקשר חזקים:

[city-mpg=B]: 125 ==> [engine-location=front, aspiration=std, engine-size=A]: 83 conf:(0.66) <lift:(1.51)> lev:(0.14) conv:(1.63)
[engine-location=front, city-mpg=B]: 125 ==> [aspiration=std, engine-size=A]: 83 conf:(0.66) <lift:(1.51)> lev:(0.14) conv:(1.63)
[aspiration=std, engine-size=A]: 91 ==> [city-mpg=B]: 83 conf:(0.91) <lift:(1.51)> lev:(0.14) conv:(4.01)

ה. עבור 2 האלגוריתמים למרות שהתבקשנו להריץ על תמיכה 0.4 וביטחון של 0.6, נעשתה בדיקה עבור ערכים מגוונים עבורם וכמו כן עבור מדד בדיקה שונה(גם lift).
בנוסף עבור שניהם מספר חוקי ההקשר עולה ככל שמורידים את התמיכה ומקטינים את הביטחון.

ניתן לראות שיש פרמטר נוסף הנקרא positiveIndex, שאומר, לגבי שדות בינריים, איזה משני הערכים לקחת כערך שמציין הופעה או קיום של הדבר שהשדה מתאר, ואיזה מתאר העדר. כיוון שחילקנו את הנתונים הרציפים לפי חציונים, נרצה לבדוק את שני המצבים- ערך מעל החציון מציין(ערך 2)\מתחת לחציון(ערך 1) מציין קיום. ניתן לראות שהתקבלו יותר חוקי הקשר עבור ערכים מתחת לחציון.

מספר חוקי ההקשר עבור האלגוריתם A-priori גדול יותר מאשר אלגוריתם FP-growth, מה שמתיישב עם העבודה שהאלגוריתם A-priori מחשב יותר סטים של פריטים שכיחים ומכך מקבל יותר קבוצות שכיחות.
ניתן לראות שזמן הריצה של האלגוריתם A-priori גדול בצורה משמעותי מאשר של FP-growth, ממוצע זמן הרצה עבור הראשון הוא 14.82 ועבור השני הוא 1.92, הבדל זה משמעותי, זה אומר שכל הרצה של A-priori תיקח פי 7.7 זמן מאשר FP-growth.
בנוסף ניתן לראות שעבור lift=2 לא התקבלו כלל תוצאות באלגוריתם FP-growth, בעוד שעבור אלגוריתם A-priori התקבלו 16 תוצאות, עובדה זו מתיישבת עם הידיעה שהאלגוריתם FP-growth לא מייצר את כל הקבוצות השכיחות האפשריות.

נוכל להסיק מכך שלמרות שהאלגוריתם A-priori, נותן הרבה יותר מקרים מאשר אלגוריתם FP-growth, עדיין יהיה עדיף ברוב המקרים להשתמש באלגוריתם FP-growth כיוון שהוא הרבה יותר מהיר.

<u>שאלה 2</u>

א. אשכול הוא קבוצה של אובייקטים היותר דומים אחד לשני מאשר מאובייקטים הנמצאים באשכול אחר על פי קריטריון מסוים.
ניתוח אשכולות היא הפעולה של הפעלת אלגוריתם כלשהו המחלק את האובייקטים לאשכולות, זה למעשה unsupervided learning כאשר אין אנו יודעים בדיוק לאיזה אשכול שייך כל אובייקט מלכתחילה.

ב. לפני הגדרת ממד איכותי, נצטרך לדעת משהו חלוקה לאשכולות איכותית.
אשכול איכותי הוא אשכול שהאובייקטים בו דומים מאוד אחד לשני ופחות לאובייקטים באשכולות אחרים, למעשה ככל שהדמיון בין אובייקטים באותו אשכול גדל והדמיון בין אובייקטים באשכולות אחרים קטן, כך נחשבת החלוקה איכותית יותר.
מדדים איכותיים יהיו מציאת מרחק בין אובייקטים(מרחק מטרי), כך שככל שהם קרובים יותר אחד לשני האשכול יותר איכותי ובדיקה האם אחרי הסיווג נלמד מידע חדש שלא היה אפשר לחזות לפני.

ג. בחרתי בגישה של k-means ובשיטה של DBSCAN.

<u>K-means:</u>

גישה של חלוקה של n האובייקטים לk קלסטרים, אשר כל אובייקט קרוב לקלסטר עם הmean הקרוב ביותר. למצוא את המינימום הגלובלי כאשר k לא נתון היא בעיה NP-HARD אך בהינתן k מראש ומספר המיימדים היא מתכנסת למינימום לוקאלי.
בשיטה, כל אשכול מיוצג על ידי ממוצע של האובייקטים השייכים אליו וכל אובייקט חדש נבדק כנגד ממוצע זה.
שלבי האלגוריתם:

a. לכל אובייקט נקבע אשכול רנדומלי ומבוצע ממוצע על האשכול כדי לקבל את הmean שלו(שיטת random partition).
b. בודקים לאיזה אשכול שייכים האובייקטים ע"י בדיקה של מרחק מממרכזי האשכולות(הממוצעים). אובייקט משתייך למרכז הקרוב אליו ביותר.
c. אם לפחות אובייקט אחד שינה את שיוכו, מבצעים ממוצע על האשכולות וחוזרים לשלב הקודם אחרת מסתיימת הריצה.

- כיוון שיש הרבה אפשרויות לפי ויקיפדיה לבצע את האלגוריתם, נבדק המימוש לפי המימוש של התוכנה. למשל, בבחירת החלוקה ההתחלתית, השיטות הנפוצות הן שיטת Forgy -בחירת k נקודות באופן רנדומלי והם יהיו המרכזים ההתחלתיים או שיטת random partition, שבWEKA נבחרה האפשרות השניה.

נבחר השימוש באלגוריתם זה, כיוון שהוא קל לבנה ונלמד פעמים רבות במהלך הקורס, כמו כן זמן ביצועו מהיר מהרבה אלגוריתמי קלסטרים אחרים.

<u>DBSCAN:</u>
גישה שמקבצת ביחד סט של נקודות באותו מקום, כלומר אזורים צפופים ישתייכו לאותו אשכול, האלגוריתם מלבד סט הנקודות מקבל כקלט את הרדיוס(ε) שמהווה את הסביבה הנבדקת שיש יותר ממינימום מספר של נקודות, פרמטר קלט נוסף הנקרא minpts, ואם תנאי זה מתקיים יצורף אזור זה לאשכול.
שלבי האלגוריתם:

a. מתחילים עם נקודת שעדיין לא סומנה "visited".
b. מוצא את השכנים בסביבת ε ומזהים את הנקודות סביבה, אם יש יותר מminpts נקודות, אז תהליך האשכולות מתחיל והנקודה מסומנות "visited" ושייכת לאותו אשכול, אחרת הנקודה מסומנת כרעש(בהמשך הריצה הנקודה יכולה להפוך לחלק מאשכול).
c. אם נקודה נמצא שהיא חלק מאשכול אז גם הנקודות בסביבת ε שלה יהיו חלק מאותו אשכול והתהליך החל משלב b מתבצע שוב לכל הנקודות בסביבת ε.
d. חוזרים על שלב a עד שכל הנקודות מסומנות(כשייכות לאשכול או כרעש).

- יש עוד ואריאציות של האלגוריתם הזה, כמו למשל למצוא את כל הנקודות שמקיימים שאת תנאי סביבת האפסילון ומינימום נקודות בה, אך עקב אי מציאת פירוט על השיטה הספציפית של WEKA וכיוון שזמן הריצה ותהליך הוואריאציות השונות שקול אז נבחרה הוריאציה הזו.

נבחר השימוש באלגוריתם זה, כיוון שאין צורך להגדיר מראש את מספר האשכולות, הוא יכול לזהות אשכולות בעלי צורות שונות(לא רק צורות קמורות כמו באלגוריתם הקודם) ומזהה רעש.

ד. +ה.
ראשית בהכנת הנתונים נדרש להפוך את התכונות השונות לייצוגים מספריים(כיוון ששיטת המדידה מתבסס על מרחק) ולנרמל אותם(לתת משקל שווה לכל תכונה).
סט הנתונים ילקח ממן 21, אקסל "הכנת הנתונים", גיליון "המרה סופי-רגרסיה לינארית"
כיוון ששם ניתנו ערכים מספריים לתכונות הנומינליות וכל הערכים המספריים נורמלו, כמו כן שמתי את הנתונים המנורמלים באקסל "ממן 22 אקסל" גיליון "נתונים לאשכולות".
תכונה "body-style" נמחקה עקב אי האפשרות לתת מספר לערכים הנומינלי.

K-means:

בוצעו מספר הרצות עם פונקציות מרחק שונות ומספר אשכולות שונה, להלן התוצאות:

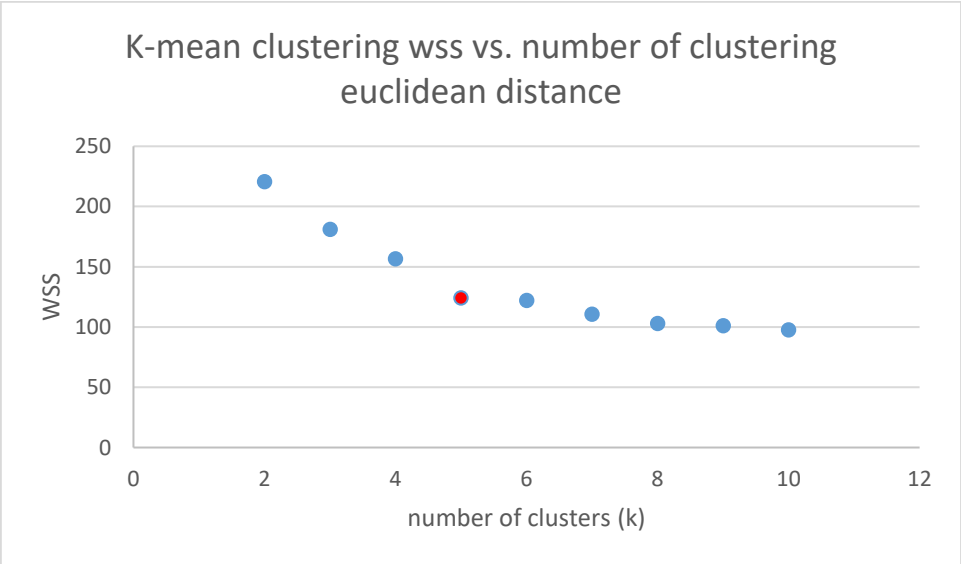
מספר הרצה	פונקציית מרחק	WSS	1#	2#	3#	4#	5#	6#	7#	8#	9#	10#
1	אוקלידי	220.5	40%	60%	-	-	-	-	-	-	-	-
2	אוקלידי	181.1	29%	45%	26%	-	-	-	-	-	-	-
3	אוקלידי	156.5	25%	29%	34%	12%	-	-	-	-	-	-
4	אוקלידי	124	28%	23%	20%	9%	20%	-	-	-	-	-
5	אוקלידי	122	17%	14%	17%	7%	19%	24%	-	-	-	-
6	אוקלידי	110.5	15%	15%	17%	7%	16%	21%	9%	-	-	-
7	אוקלידי	103	5%	13%	14%	8%	16%	19%	7%	18%	-	-
8	אוקלידי	101	5%	13%	14%	8%	16%	18%	7%	18%	1%	-
9	אוקלידי	97.5	5%	12%	14%	8%	16%	18%	5%	17%	1%	3%
10	מנהטן	616.4	43%	57%	-	-	-	-	-	-	-	-
11	מנהטן	540.4	31%	38%	31%	-	-	-	-	-	-	-
12	מנהטן	474.9	31%	24%	21%	24%	-	-	-	-	-	-
13	מנהטן	463.3	29%	24%	4%	23%	21%	-	-	-	-	-
14	מנהטן	444	18%	14%	6%	22%	20%	20%	-	-	-	-
15	מנהטן	426.5	18%	14%	6%	21%	18%	19%	4%	-	-	-
16	מנהטן	417.7	7%	10%	4%	21%	18%	19%	4%	17%	-	-

טבלה 4: טבלה של הרצות עם פונקציות מרחק שונות ועם מספר אשכולות שונה לk-mean. מספר ההרצה מקביל למספרי מסמכי ההרצות שבתקיית "הרצת אשכולות->K-mean".

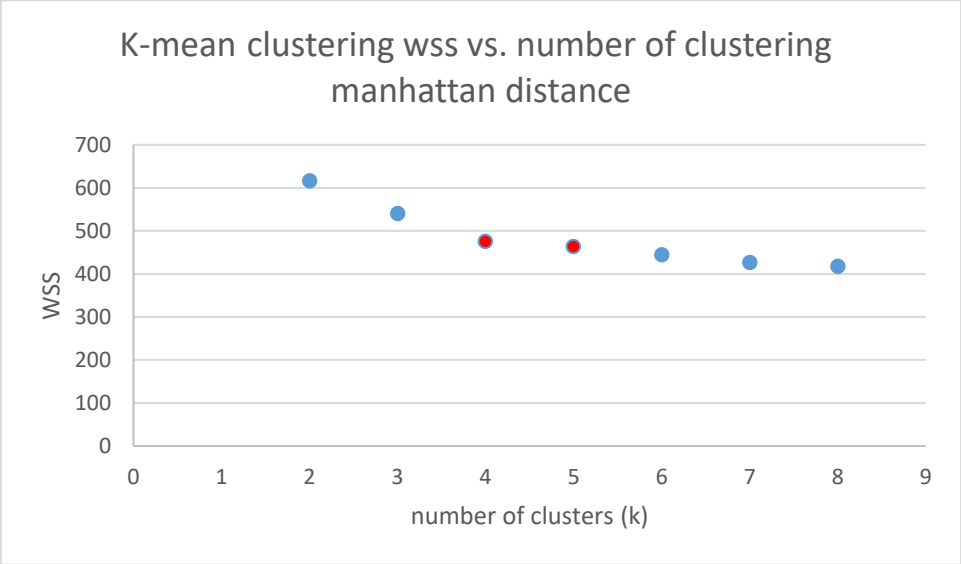
מקרא לטבלה:

- WSS הוא קיצור לwithin-cluster sum of square – סכום ריבועי המרחקים של הנקודות בכל אשכול ממרכזו.
- #number – מציין את מספר האשכול

לכל פונקציית מרחק שורטט גרף "מרפק"(סכום ריבועי המרחקים של הנקודות בכל אשכול ממרכזו כפונקציה של מספר האשכולות), לפי שיטת elbow method, השרטוט נמצא באקסל "ממן 22 אקסל-> גרף מרפק לאלגוריתם האפריורי" גיליון "גרף מרפק".



גרף 1: סכום ריבועי המרחקים של הנקודות בכל אשכול ממרכזו כפונקציה של מספר האשכולות כשפונקציית המרחק היא אוקלידי. מסומנת באדום נקודת המפנה של הגרף.



גרף 2: סכום ריבועי המרחקים של הנקודות בכל אשכול ממרכזו כפונקציה של מספר האשכולות כשפונקציית היא מנהטן. מסומנות באדום נקודות המפנה של הגרף.

כמו כן עבור כל הרצה (הרצה 1- 16) נבדק הפיזור של הנקודות לאשכולות בכל מיימד, גם עבור מספר נמוך של אשכולות התקבלה חלוקה גסה בכל מיימד לאשכולות השונים.
דוגמה מייצגת עבור פיזור נלקח מגרף של "normalized-losses" כפונקציה של "stroke" בהרצה מספר 12 ונמצא ב"הרצת אשכולות->דוגמה מייצגת" .

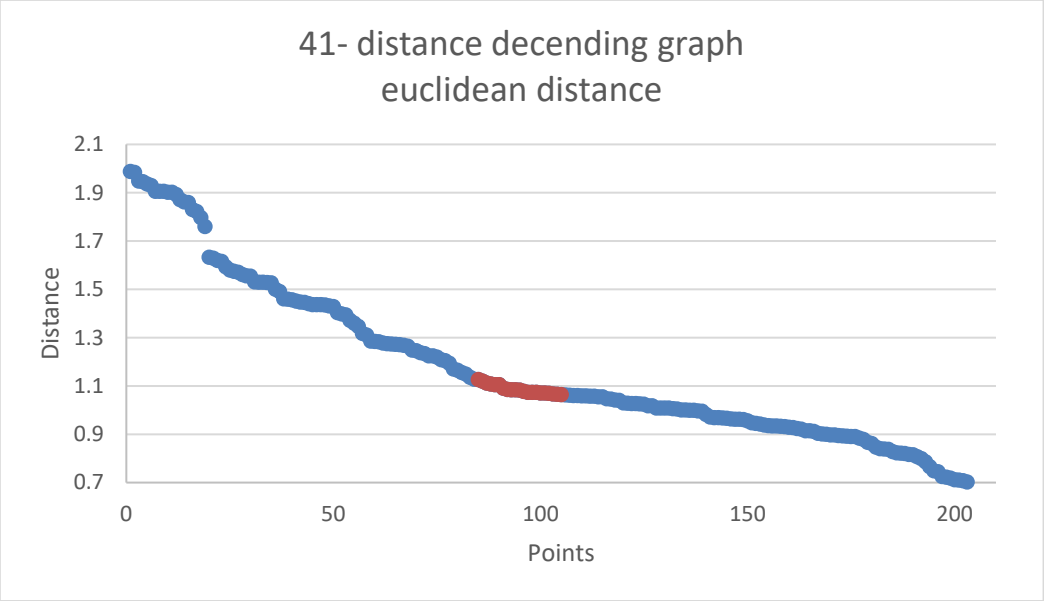
<u>:DBSCAN</u>

כדי למצוא את התוצאות המיטביות, נצטרך למצוא את הפרמטרים המיטביים להכניס לגרף.
לפי הערכת הפרמטרים בויקיפדיה(קישור נמצא בבליוגרפיה), נעריך את הפרמטרים הבאים:

- MinPts- לפי חוק אצבע נותנים ערך בדיקה התחלתי בגודל פעמיים מספר המיימדים((MinPts=2x(Dimensions number).
- Ɛ- שורטט גרף של המרחק לשכן הk הכי קרוב בסדר יורד עבור כל נקודה(k-distance graph), כאשר k= MinPts-1, ערכים טובים לƐ יהיו בסביבות ה"מרפק"(בדומה לגרפים ששורטטו קודם). לשם מציאת הk המרחקים הכי קרובים, יצרתי את קוד המתלב הנמצא בתיקיית "קוד המתלב".
הרעיון נלקח גם מהמאמר "*A density-based algorithm for discovering clusters in large spatial databases with noise*", שמצורף אליו קישור ביבליוגרפיה.
- פונקציית המרחק- אין שיטה חד-משמעית להעריך זאת, לכן נבדקו 2 פונקציות מרחק: אוקלידי ומנהטן(זהה לפונקציית המרחק של Ɛ).

שורטטו מספר גרפים של k שונים עבור פונקציות מרחק שונות(אוקילידי ומנהטן), כאשר המטרה הייתה למצוא את הk המינימלי אשר בו רואים את המרפק הגבוה ביותר ובצורה הברורה ביותר, כאשר נקודת החיפוש ההתחלתית הייתה k=45(כיוון שיש 23 מיימדים).

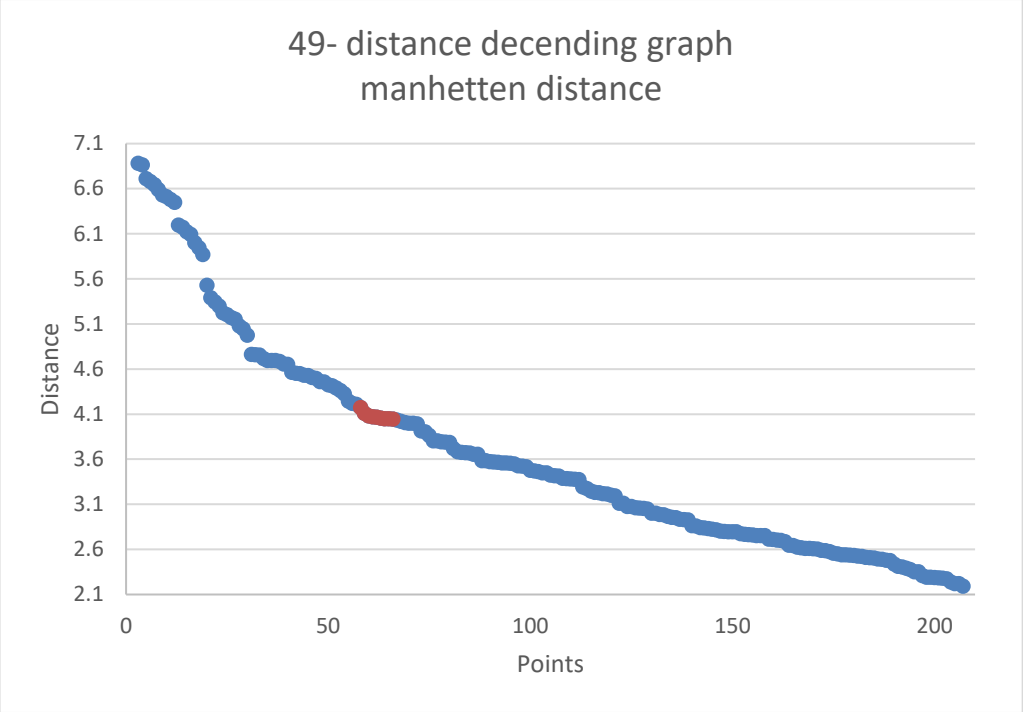
מרחק אוקלידי:
עבור k=41 (MinPts=42) התקבל המרפק הגבוה והברור ביותר, עיבוד התוצאות נמצא בגיליון אקסל "ממן 22 אקסל->הערכת פרמטרים לFPGrowth אוקלידי".



גרף 3: המרחקים של השכן ה41 הכי קרוב לכל נקודה מסודרים בסדר יורד כשפונקציית המרחק היא אוקלידית. באדום מסומן ה"מרפק"

היה קשה לזהות בדיוק את נקודת המפנה, לכן נלקח המרפק מממוצע של טווח נקודות(צבוע בצבע אדום בגרף) וערכו הוא $\varepsilon \approx 1.086$.

מרחק מנהטן:
כפי שנראה בגרף היה קשה לזהות את ה"מרפק", אך אחרי בדיקת הרבה אפשרויות לk התגלתה מגמה שבה עבור האזור המסומן באדום בגרף, לא השתנתה המגמה בצורה ניכרת בין הk השונים ,לעומת זאת שאר ה"זיזים" הפכו להיות חלקים יותר בk יותר גבוהים.
ה k נבחר להיות k=49 (MinPts=50), למרות שבגרף יש עדיין "זיזים" קטנים, זה הערך שעבורו אזור המפנה הגיע לראשונה למקסימום ונראה בצורה ברורה יחסית לשאר הk הנמוכים יותר. עיבוד התוצאות נמצא בגיליון אקסל "ממן 2 אקסל-> הערכת פרמטרים לFPGrowth מנהטן".



גרף 4: המרחקים של השכן ה49 הכי קרוב לכל נקודה מסודרים בסדר יורד
כשפונקציית המרחק היא מנהטן באדום מסומן ה"מרפק"

היה קשה לזהות בדיוק את נקודת המפנה, לכן נלקח המרפק מממוצע של טווח נקודות(צבוע בצבע אדום בגרף) וערכו הוא $\varepsilon \approx 4.078$.

בוצעו מספר הרצות כאשר הערכים של MinPts ו-ε נעים סביב הערכים שנמצאו מקודם.
להלן הטבלה עם ההרצות השונות:

מספר הרצה	פונקציית מרחק	ε	MinPts	noise	#1	#2	#3
1	אוקלידי	1.086	42	46	161	-	-
2	אוקלידי	1.086	40	46	161	-	-

-	-	163	44	35	1.086	אוקלידי	3
-	-	168	39	24	1.086	אוקלידי	4
-	-	61	146	42	0.8	אוקלידי	5
-	-	-	207	42	0.6	אוקלידי	6
-	84	65	58	30	1	אוקלידי	7
-	63	58	86	25	0.8	אוקלידי	8
-	45	28	134	24	0.6	אוקלידי	9
-	-	191	16	49	4.078	מנהטן	10
-	-	145	62	49	3	מנהטן	11
-	-	130	77	49	2.8	מנהטן	12
-	-	198	9	34	4.078	מנהטן	13
-	-	201	6	24	4.078	מנהטן	14
-	-	165	42	30	3	מנהטן	15
-	26	19	162	26	1.5	מנהטן	16
-	26	19	162	24	1.5	מנהטן	17
-	-	25	182	24	1.4	מנהטן	18

טבלה 5: טבלה של הרצות עם פונקציות מרחק שונות ועם מספר אשכולות שונה לDBSCAN. מספר ההרצה מקביל למספרי מסמכי ההרצות שבתיקיית "הרצת אשכולות->DBSCAN".

ו. עבור 2 השיטות, לא היה ידוע את מספר האשכולות מראש והיה ניסיון למצוא אותו ואת הפרמטרים שצריך להזין להרצה ע"י שיטת ה"מרפק".
עבור כל הגרפים היה קשה לקבוע במדויק את מיקום ה"מרפק", לכן כשהייתה אי-ודאות ויזואלית גדולה נלקחו מספר נקודות.
בk-mean, נמצא מספר האשכולות עפ"י הזנה של מספר k (לפי שיטת מרחק) ואז שימוש בשיטת ה"מרפק", לעומת זאת בdbscan נמצאו הערכים המתאימים של הפרמטרים להזנה לאלגוריתם לפי שיטת ה"מרפק" ואז הורץ האלגוריתם בסביבת ערכים אלו כולל ניסיונות נוספים.

עבור k-mean – שהשגיאה באשכולות ירדה ככל שמספר האשכולות עלה(עבור שני שיטות המדידה) והיה קל ויזואלית לקבוע את אזור נקודת ה"מרפק", אך שנבדקו החלוקות לאשכולות בנתונים עצמם, היה קשה לראות הפרדה ברורה ויזאולית – עקב מציאת מרפק ברור, אפשר להניח שיש חלוקה לאשכולות נאותה אך כאשר מנסים לעשות ויזואליזציה של 23 מיימדים ל2 מיימדים לא תמיד תהא הפרדה ברורה.

עבור dbscan – היה קושי לקבוע את נקודת ה"מרפק" ונלקח ממוצע על טווח ערכים אפשריים, כמו כן בהרצות האלגוריתם סביב ערכי הפרמטרים שהתקבלו, קיבלנו רק אשכול אחד. נבדקו ערכים רחוקים מערכי הפרמטרים שקיבלנו בניסיון ליצור יותר מאשכול אחד והמקסימום שהתקבל היה 2, אך שוב כיוון שערכים אלו היו רחוקים מאוד מהפרמטרים שחושבו זה היה יותר לקבל תחושה לגבי האלגוריתם. למרות יתרונו של אלגוריתם זה למציאת נקודות חריגות, לא קיבלנו תוצאות מעניינות, נשייך זאת לעבודה שיש מספר גבוה של מיימדים לעומת מספר קטן של רשומות(יחס של 9:1).

נסיק מכך ששיטת k-mean נתנה תוצאות טובות, למרות שקשה להיווכח בכך ויזואלית וש dbscan נכשלה בחלוקה לאשכולות עקב מספר המיימדים הרב שמשפיע על הפרמטר MinPts.

<u>שאלה 3</u>

בממן 21 , נדרשנו לייצור מודל לניבוי מחיר של רכב לפי שאר התכונות הנתונות.

תוארו שלבי הKDD, הכנו את הנתונים והוצגו שני מודלים לניבוי מחיר הרכב- המודלים שנבחרו הם רגרסיה לינארית ועץ החלטה C4.5.
הרצנו את אותם על הנתונים שהכנו וקיבלנו שעץ ההחלטה חזה את המחירים בצורה מוצלחת יותר מאשר הרגרסיה.
במטלה הנוכחית התבקשנו להסיק מסקנות על הנתונים ועל מגמות בהם ולא על משתנה המטרה.
בחנו כלים שונים ,חוקי הקשר ואשכולות, וקיבלנו תוצאות גם כאשר לא הייתה ידועה הנטייה מראש.
קיבלנו שהאלגוריתם הא-פריורי נותן סט יותר גדול של חוקים, אך רץ הרבה יותר לאט מאשר האלגוריתם FP-growth, בנוסף ראינו שהאלגוריתם k-mean מצליח לחלק לאשכולות, למרות שקשה להיווכח בכך ויזואלית ושהאלגוריתם dbscan נכשל עקב מספר המיימדים הרב.

<u>מסקנות מכריית המידע:</u>

1. כריית מידע היא תהליך דינמי, הרבה פעמים שיניתי את שיטות העבודה. כששיטות אחרות הראו על נטייה מסוימת או קושי בחיזוי הצלחתי להבין יותר טוב את התמונה הרב-מיימדית שהסט מייצג וללכת לגישה אחרת.
2. עיבוד הנתונים עבור גישות שונות מצריך עיבוד שונה, עובדה זו חשובה כיוון שמתקבלות תוצאות, צריך להביא בחשבון את העיבוד הנתונים ההתחלתי בהשוואתן.
3. אין פתרון מוחלט, כל פתרון "תוקף" את הבעיה מכיוון אחר ונושא עמו חסרונות ויתרונות.
4. כדאי לפני שנגשים לנסות שיטה כלשהי לבדוק באינטרנט מאמרים בנושא, כיוון שזה תחום שעדיין מתפתח ואולי יש פיתוחים שיעזרו בתהליך כריית המידע(למשל בממן הנוכחי נעזרתי ב2 מאמרים אקדמיים הנמצאים בביבליוגרפיה).

<u>בבליוגרפיה:</u>

- https://en.wikipedia.org/wiki/Apriori_algorithm
- https://en.wikipedia.org/wiki/Association_rule_learning#FP-growth_algorithm
- https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Frequent_Pattern_Mining/The_FP-Growth_Algorithm
- https://www.singularities.com/blog/2015/08/apriori-vs-fpgrowth-for-frequent-item-set-mining
- https://en.wikipedia.org/wiki/Cluster_analysis
- https://pdfs.semanticscholar.org/759a/988f38b4ea3465f5747442e6c642fc672598.pdf
- https://en.wikipedia.org/wiki/DBSCAN
- https://sites.google.com/site/dataclusteringalgorithms/density-based-clustering-algorithm
- http://www.sthda.com/english/articles/29-cluster-validation-essentials/96-determining-the-optimal-number-of-clusters-3-must-know-methods/
- https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf