2020

השוואת מסווגים

מרצה : מר הדר יניב

מחברים : טל אלפי, זיו פריזה, אולג בלושיצקי.

תוכן עניינים

הקדמה

במסמך זה נתבונן במספר ניתוחים שביצענו על הקבצים הנתונים לנו, על המסווגים השונים אשר למדנו במהלך הקורס. ביצענו מספר ניסויים ובכל סוף ניסוי ניסחנו **מסקנה** שנבעה מאותו הניסוי. העדפנו לפעול כל ולא לכתוב מסקנה אחת מרוכזת בסוף המסמך כדי למנוע "דפדוף יתר" בין הפלוטים השונים שיצרנו. העבודה בוצעה במידה שווה על ידי חברי הצוות ואנו מאחלים לבודק קריאה מהנה, ומקווים שקלענו לדרישות.

בברכה,

טל, זיו ואולג.

הסבר על המימוש שלנו בעבור דיסקריטיזציה לפי אנטרופיה

תיאור הבעיה:

בעבור data set גדול מאוד – מציאת חלוקה ל-bin's , כך שכל bin מכיל info gain מקסימאלי יכולה לקחת המון זמן (המון זמן = מספר ימים ברצף) מכיוון שהאלגוריתם הנאיבי יעבור על כל החלוקות האפשרויות ל K bins.
המחשבה ההתחלתית הייתה לפתור את הבעיה עם אלגוריתם חמדן אשר בכל פעם מפצל ומוסיף bin עם info gain מקסימאלי , אך הוא לא בהכרח ייתן את הפתרון האופטימאלי.

אנו ניסינו למצוא פתרון יצירתי לבעיות אלה.

הרעיון המרכזי של האלגוריתם:

אנו נרצה K bins.
נבצע מיון לפי עמודה אנו רוצים לבצע בה דיסקריטיזציה.
בגלל שיודע שחלוקה ל pure sets תיתן את ה- info gain המקסימלי אנו נחלק אותה ל P בינים.
ואז יתכנו 3 מצבים:

- מצב ראשון P=K :

כלומר כמות ה - bins שהם pure היא כמות ה- bins שאנו צריכים ולכן קיבלנו את החלוקה ל- bins עם מקסימום info gain, כנדרש.

- מצב שני K > P:

כלומר כמות ה - bins שאנו רוצים יותר גדולה מקמות ה - bins שהם pure .
במצב זה אנו נחלק את ה bin הכי רחב באמצע ל 2 ונקבל bin נוסף.
נחזור על הפעולה הזו עד שיהיה לנו K bins.
בסוף נקבל bins שהם pure מהסיבה שאם דלי הוא pure אז גם שני החצאים שלו pure.

- מצב שלישי K < P :

כלומר כמות ה - bins שהם pure גדולה מכמות הדליים שאנחנו צריכים.
במצב זה אנחנו נבצע מיזוג בין bins שהם שכנים במקומות שבהם הנזק ל info gain קטן ביותר.
אופן המיזוג:

אם יש דליים שכנים בעלי אותו רוב לסיווג מסוים מזג אותם.
אחרת מזג bins שכנים אם יחס בין גדול לקטן בגודל ביניהם הכי גדול.
חזור על הפעולה הזו על שנקבל את כמות - bins המבוקשת.

אליָה וקוץ בה

יתכן מצב שבו בעבור ערך נומרי בעמודה של דיסקריטיזציה יחזור על עצמו מספר רב של פעמים עם סיווגים שונים, למרות שמדובר במספרים ממשים יתכן מצב של collision .

מצב זה הוא החיסרון באלגוריתם שאנו פיתחנו.

פתרנו את התקלות במצב זה על ידי "חוק הרוב" שקובע את הסיווג ע"פ סיווג של הערך שחוזר על עצמו מספר רב יותר של פעמים.

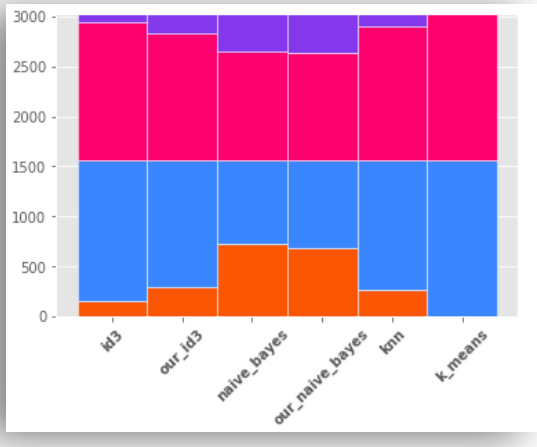
ניתוח גרפים ומסקנות סופיות לפי שני מדדים – split 80: 20 , train-test

בעבור כל אחד מהמסווגים השונים נתבונן בביצוע העבודה שלו על הקבצים באופן הבא :

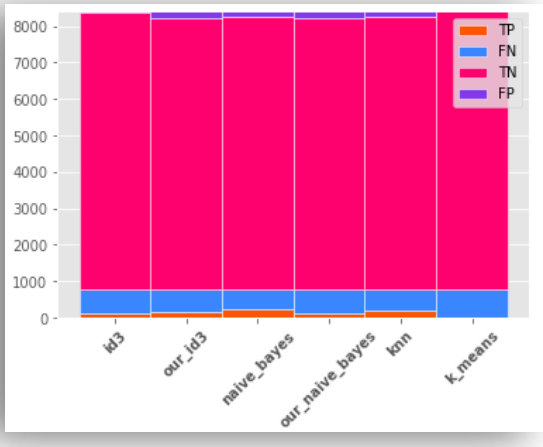
1. חלוקת 80: 20 על קובץ ה train.
2. למידה על קובץ ה train וחיזוי על קובץ ה test.

תחילה ננתח את ה confusion matrix שנתקבלה בעבור כל אחד מהמסווגים השונים.

Split 80: 20



Test - train



בעבור החלוקה של 80: 20 , קיבלנו חיזוי מאוד טוב בעבור כל המסווגים. מעניין לשים לב כי בעבור חיזוי לסיווג 'NO' , ה K-Means ו ID3 נתנו תוצאה כמעט מושלמת. מצד שני בעבור חיזוי ל 'YES' שניהם נתנו תוצאות מאוד חלשות ביחס לשאר המסווגים.

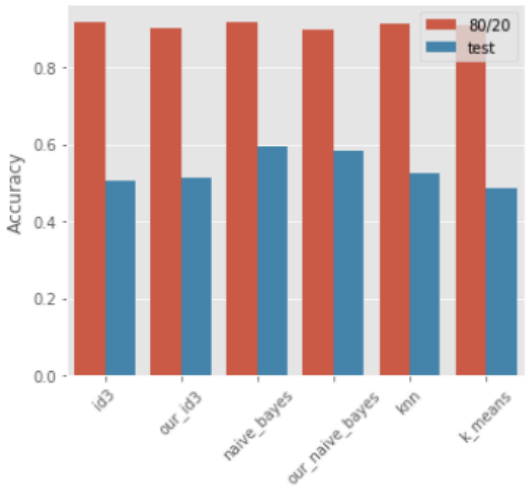
תוצאות מאוד דומות (דומות מבחינה פרופורציונלית) קיבלנו בעבור הניסוי של test - train . גם כאן המסווגים הבולטים בעבור חיזוי ל 'NO' שוב היו ה K-Means ו ID3. וגם כאן , בעבור חיזוי ל 'YES' הם הניבו תוצאות חלשות יותר.

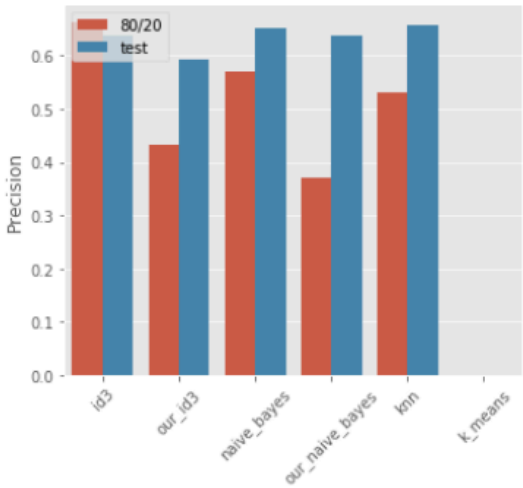
המסקנה : בעבור DATA SET שאינו מאוזן, האלגוריתמים K means ו- ID3 נתנו תוצאות טובות בעבור חיזוי לטובת הערך הנפוץ יותר מעמדות ה 'class' (במקרה של ה dataset שלנו זה 'NO') וסיווגו פחות טוב בעבור הערך הפחות נפוץ - 'YES'. לכן ניתן לומר שהם יותר רגישים בעבור data sets שאינם מאוזנים.
naive bayes ו- knn נתנו תוצאות פחות טובות לעמודה בעבור חיזוי לטובת הערך הנפוץ יותר מעמדות ה 'class' , אך תוצאות מעט טובות יותר בעבור החיזוי ל 'YES'.
כעת נתבונן בגרפים שמשווים בין תוצאות ה Accuracy, Precision , Recall, F1 score.

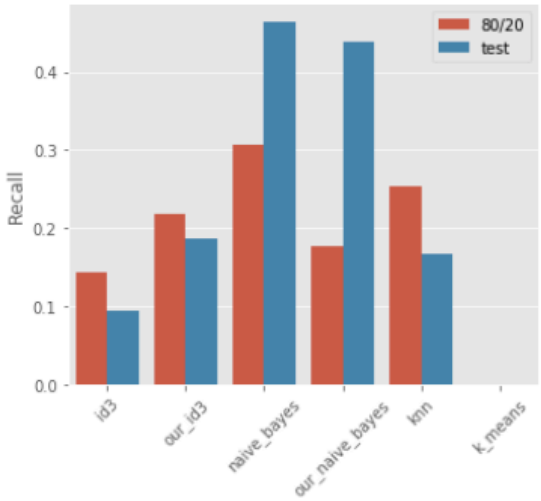
השוואה בין תוצאות ה-Accuracy, Precision , Recall , F1-score

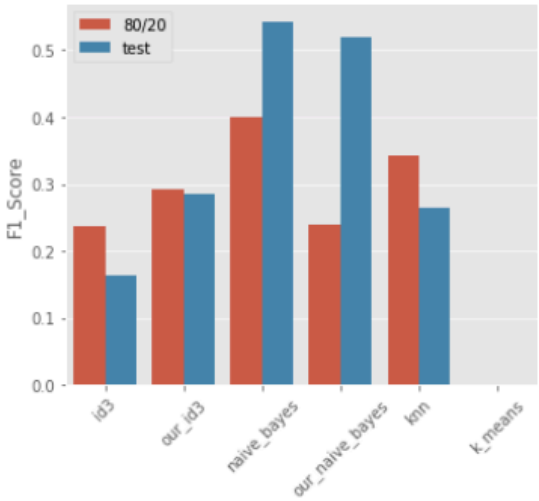
גם כאן, נבחן את תוצאות הריצה של האלגוריתמים על :

1. חלוקת 80:20 על קובץ ה train.
2. למידה על קובץ ה train וחיזוי על קובץ ה test.





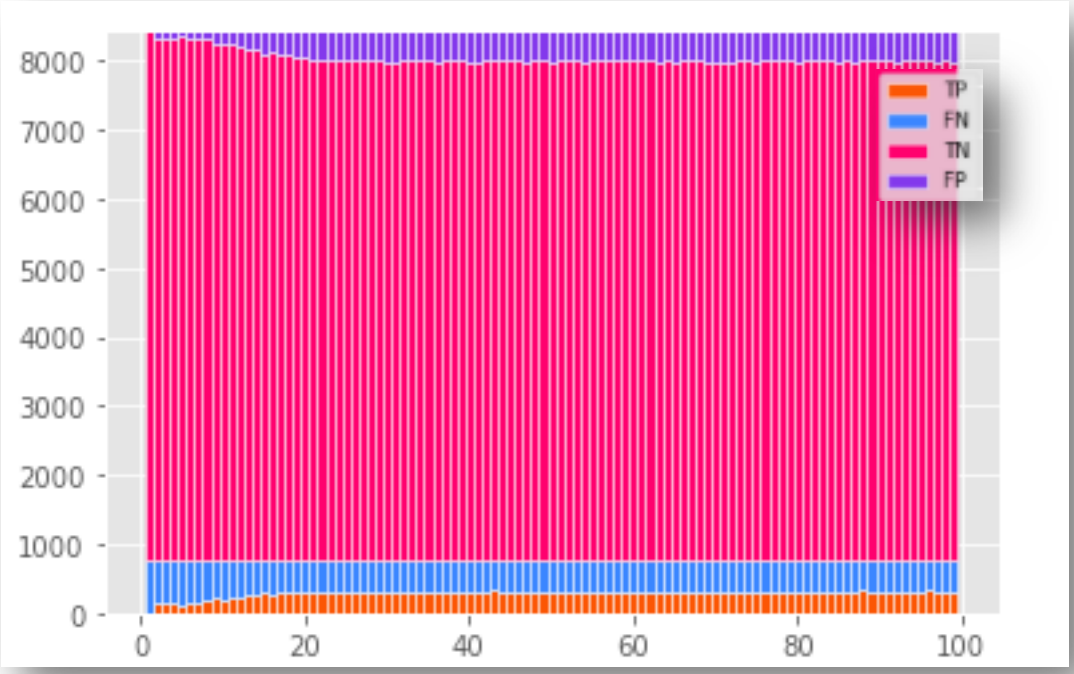




נשים לב, כי על פניו , לפי תוצאות ה Accuracy נראה כי יכולות החיזוי בעבור הפיצול של 80:20 גבוה משמעותית ביחס לחיזוי בעבור קובץ הטסט לבדו. מצד שני, מבט מהיר על דירוג ה F1-score חושף תמונה אחרת , שמראה כי הדירוגים אמנם שונים , אך הפער הוא משמעותית קטן יותר (הבדל של 50%) בין כל המסווגים השונים. כלומר, אם נתבונן למשל במסווג ID3 בגרף ה Accuracy : בעבור הפיצול של 80:20 – הבר האדום – נקבל תוצאה של כ 91% דיוק, ובעבור החיזוי ל- test נקבל חיזוי של 51%. כעת, נתבונן בתוצאות ה F1 score – שם נראה כי הפער קטן משמעותית. כידוע, F1 score היא שיטת מדידה מועדפת בעבור unbalanced data set, ולכן כדאי להסתמך עליה במדידות שמתבצעות בעבור ה – dataset הזה.

מבט ממוקד ב Id3 תחת אילוץ משתנה – עומק העץ.

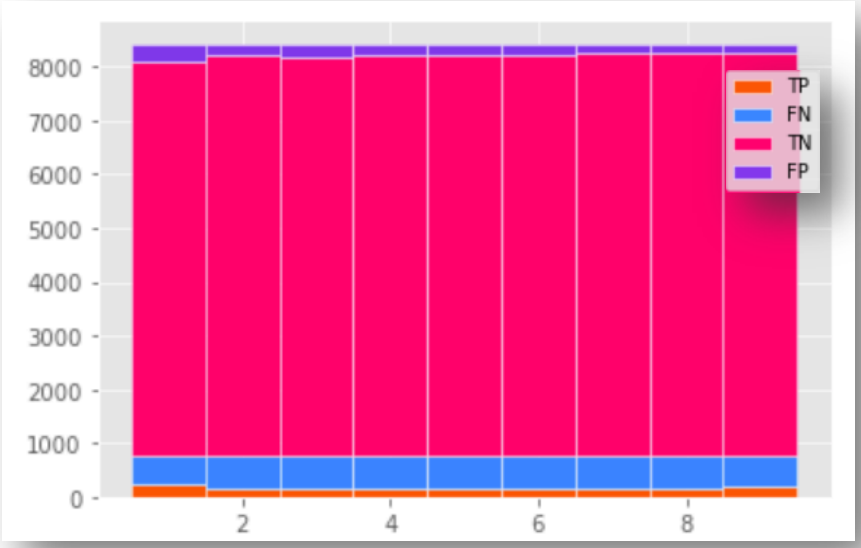
נתבונן בניסוי הבא שביצענו על האלגוריתם ID3 בלבד. עניין אותנו לראות מה תהיה ההשפעה של עומק העץ, על ה confusion matrix. הניסוי ארך זמן רב יחסית – מעל לשמונה שעות ריצה רצופות, אבל אפשר לנו להתבונן במשהו מעניין ביחד ל dataset שלנו :



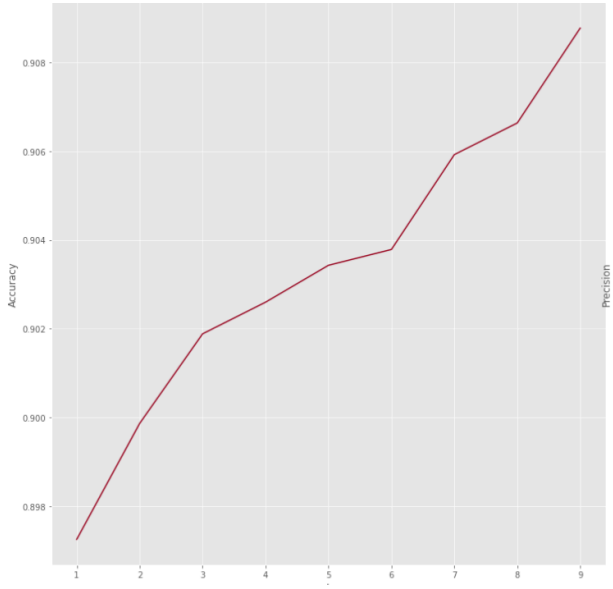
ציר ה - X מייצג את עומק העץ , ציר ה Y מייצג את הדיוק.

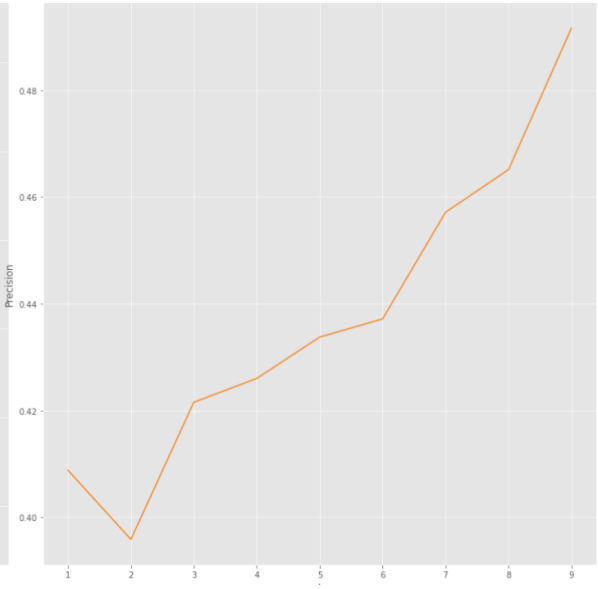
באופן מיידי ניתן לשים לב למגמה הברורה : כבר החל מעומק 20, התוצאות כמעט לא משתנות (המגמה מאוד ברורה משם ועד הערך 100). המסקנה מכאן דיי ברורה : החל מעומק 20, כבר אין תפוקה משמעותית של מידע ביחס לפיצולים שמתבצעים לפי פיצ'רים.

מבט ממוקד ב Id3 שאנחנו מימשנו - תחת אילוץ משתנה – tolerance.

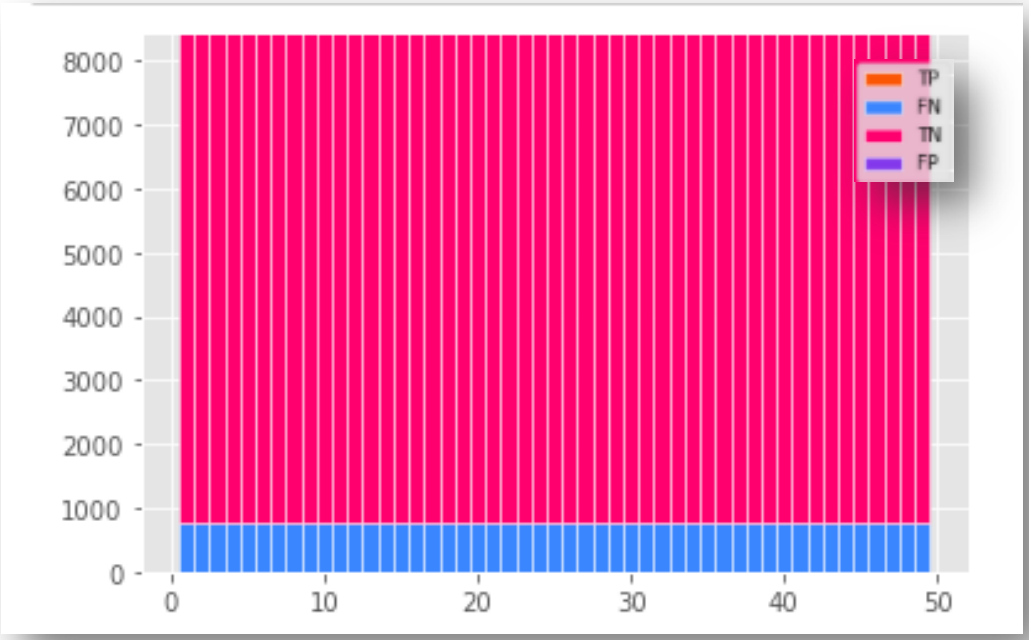


ככל שהעלנו את כמות הרגישות – ניכר שישנו שיפור בכמות הדיוק בחיזוי. בנוסף ניתן לקבל אישוש להשערה ולתצפית על סמך המדדים הבאים:





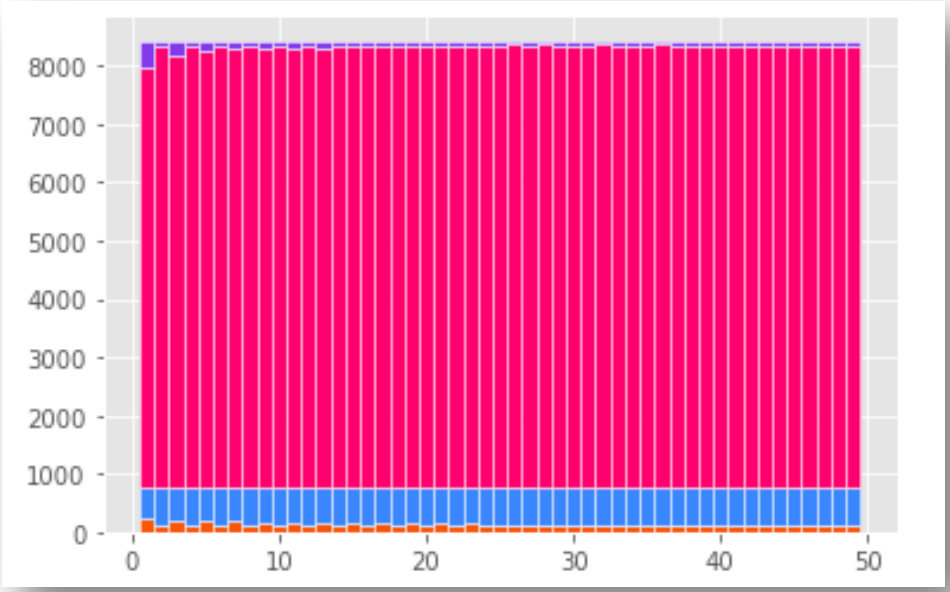
מבט ממוקד ב K means תחת אילוץ משתנה – K.



מהגרף ניתן לראות שכבר החל מ k=1 עד k=50 , כל cluster מסווג להיות no. ייתכן ואיזון* של הנתונים לפני הרצת האלגוריתם על ה data set , הייתה נותנת תוצאת מעניינת יותר עם מגמתיות ברורה יותר.

*איזון – יצירת תת dataset כך שכמות הסיווגים ל yes ו- no ישאפו להיות שווים ככל האפשר, תוך בחירת שורות רנדומאליות מה- dataset המקורי.

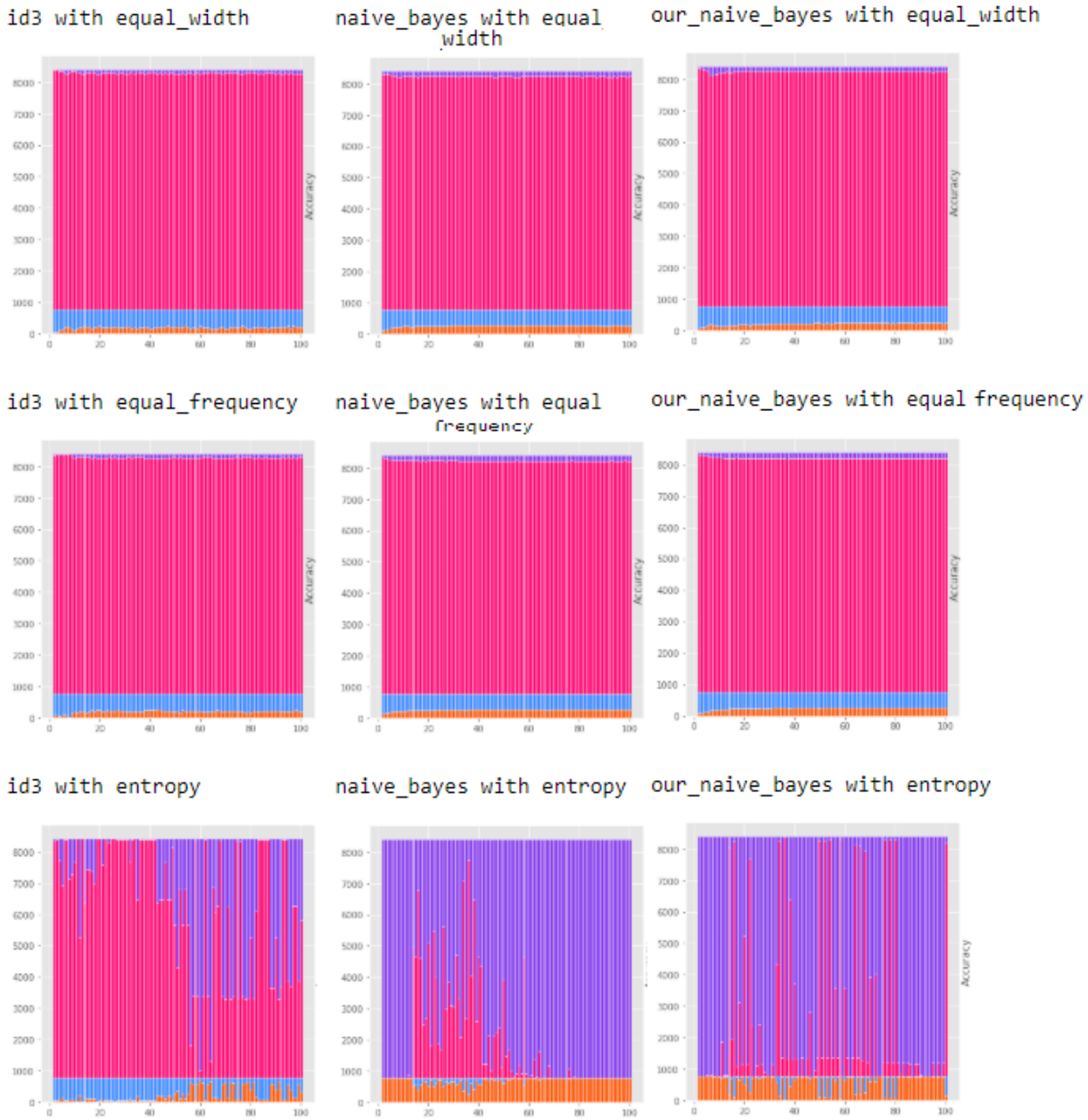
מבט ממוקד ב KNN תחת אילוץ משתנה – K.



כפי שניתן לראות חיזוי המסווגים הנכון ברובו היה TN וזאת מכיוון שקובץ ה-train היה לא מאוזן באופן הבא : 91.16 % לטובת ערכי no בעמודת ה class , ורק 8.83 לערכי yes. לפיכך, החיזויים של המסווגים היו 'no' ולכן מטבע הדברים רוב החיזויים הנכונים הם 'no' .

יכולות הסיווג של האלגוריתמים- בהתאם להגדלת כמות ה bin's.

כעת נבדוק את יכולות הסיווג של האלגוריתמים תחת הגדלת כמות ה bins. בנוסף להגדלת כמות ב bin, נבצע חלוקה באמצעות equal width, equal frequency, entropy . תחילה נתבונן ב plots שנבעו מהם:



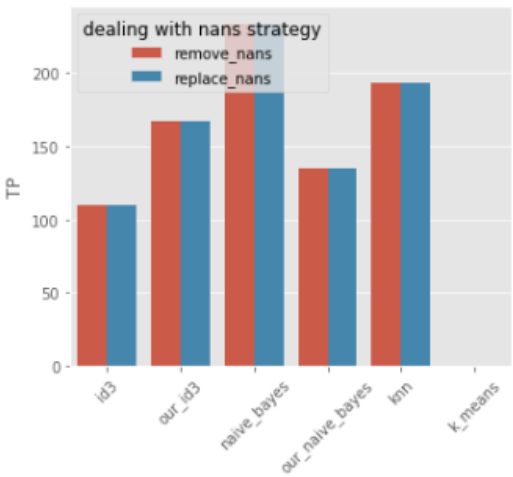
באשר ל equal width, equal frequency , אנו רואים יתרון קל לטובת equal frequency, עם יתרון של כחצי אחוז בממוצע ביחס ל equal width. בנוגע ל entropy שפותח על ידינו , אנו רואים תוצאות שאינן יציבות בכל שלושת המסווגים. ההשערה שלנו היא : מאחר והאלגוריתם זה הוא supervised, הוא מבצע בדיקה בכל צעד וצעד, ולכן יתכן וצעד מושפע מקודמו, מה שעשוי לגרור אי סדר.

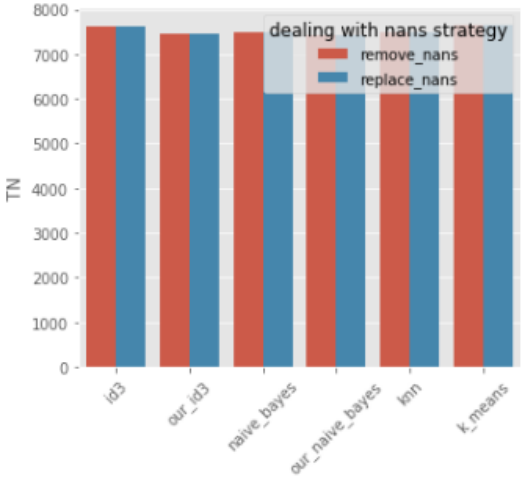
תפקוד האלגוריתמים – לאחר ביצוע פעולות על הערכים חסרים

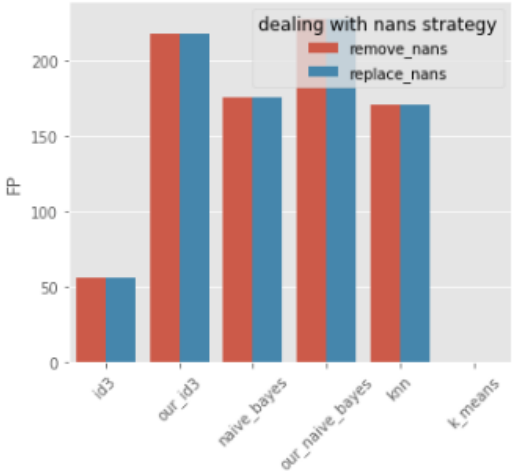
נתבונן בהתמודדות של האלגוריתמים, לאחר ביצוע שתי פעולות על הערכים החסרים:

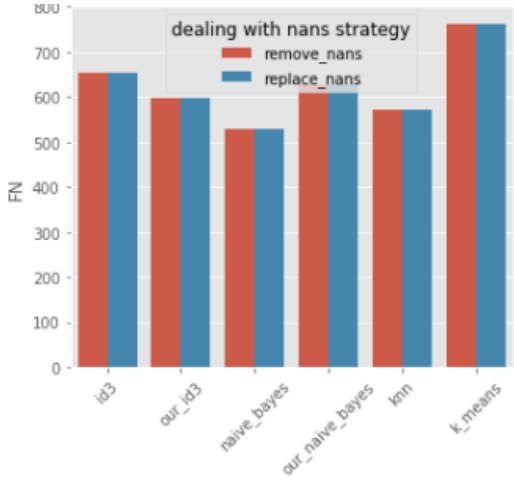
1. Remove nans.
2. Replace nans.

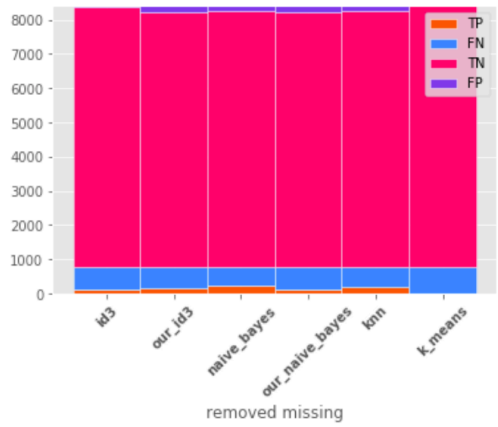
תחילה נסתכל על ערכי ה confusion matrix לאחר ביצוע הפעולות המתוארות לעיל:

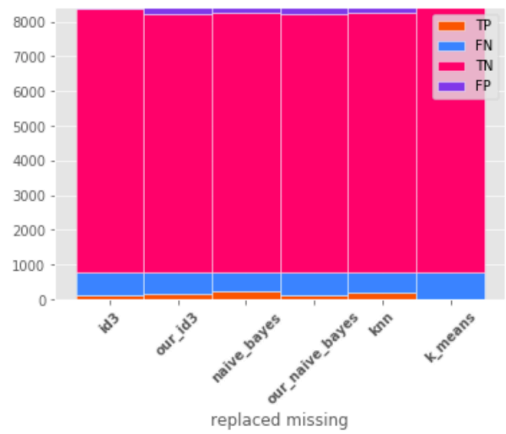




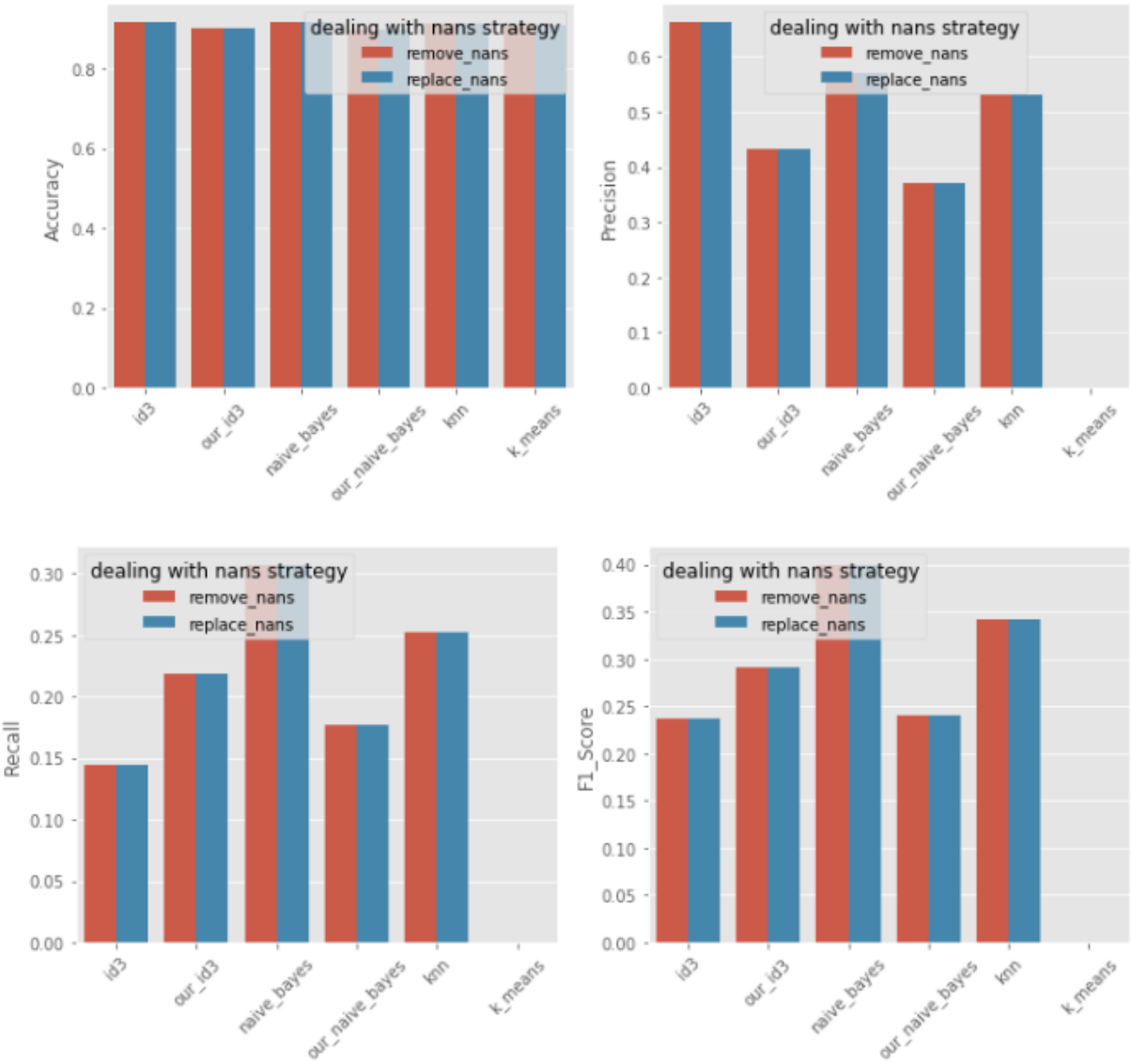








כמו כן, נתבונן גם על תוצאות ה score השונות :



כפי שניתן לראות , ההבדל בין השיטות, בעבור כל האלגוריתמים – זניח, עד כדי לא קיים כלל.

הסיבה לכך , היא הכמות הזניחה של ערכי ה nans בקובץ כולו. ניתן לקרוא בפירוט על כך בקובץ ה EDA שלנו.

-fin-