

תרגיל בית 3 – מבוא ללמידה

הנחיות כלליות:

- תאריך ההגשה: 26/01/2021 ב23:59
- התרגיל הוא להגשה **ביחידים**.
- המתרגל האחראי על התרגיל: רפאל גד, נא לפנות בשאלות אך ורק למייל ai.technion@gmail.com.
- הקוד שלכם ייבדק אוטומטית וגם ידנית, רמאות כלשהי תוביל לועדת משמעת. התרגיל מהווה חלק משמעותי מהציון הסופי שלכם בקורס, כל תקשורת בין סטודנטים בכל פלטפורמה שהיא בנוגע לתרגיל אסורה ומהווה רמאות. קבוצת הקורס תפעיל אמצעי בינה מלאכותית מתקדמים לעלות על הרמאים.
- אנא הקפידו על ההנחיות, כל הפרה תגרום להורדת ניקוד לרבות אי עמידה בציפיות פורמט הפלט של כל קובץ הנדרש מכם.
- בכל יום חמישי בשעה 14:30 רפאל יקיים כיתה הפוכה בה תוכלו לשאול שאלות לגבי התרגיל.
- בחלק מהסעיפים קיימת הגבלת שורות לפתרון שלכם. אם אתם חושבים שבסעיף מסוים הגבלה זו לא ריאלית אנא שילחו לרפאל נימוק על כך במייל ואם הנימוק יתקבל נשנה את ההגבלה לכלל הסטודנטים.
- הקוד שלכם צריך להתייחס לקבצי הדאטה כנמצאים בתיקייה הנוכחית ולא בתת תיקייה. אין לשנות את שמות קבצי הדאטה.
- בסעיפים הרטובים בתרגיל מסופקות לכן מספר מועט יחסית של הנחיות ביחס לתרגילים הקודמים. כל מימוש העונה על הדרישות יתקבל. יש לכם יד חופשית בתכנון הקוד שלכם. אנו ממליצים לממש כל מסווג כ- class שמממש פונקציות $fit(X, Y)$ שמבצעת אימון, ו- $predict(X)$ שמבצעת את הפרדיקציה.
- בחלק מהשאלות תתבקשו להמציא שיפור לאלגוריתם למידה, כתוצאה מכך יש בהן דרגות חופש, הציון בשאלות אלו יינתן בהתאם לטיב הפתרון, הן מבחינת יצירתיות והן מבחינת תוצאות אמפיריות. יש לתאר בצורה ברורה, פורמלית ומדויקת את הפתרון.
- מותר להשתמש בספריות `sklearn, pandas, numpy, random, matplotlib`, `argparse, abc, typing, all the built in packages in python` אך כמובן שאין להשתמש באלגוריתמי הלמידה, או בכל אלגוריתם או מבנה נתונים אחר המהווה חלק מאלגוריתם למידה אותו תתבקשו לממש.

רקע:

לתרגיל מצורף קובץ נתונים על מחלה מסוימת, כאשר כל שורה מתארת אדם. העמודה הראשונה מציינת האם האדם חולה (M) או בריא (B). שאר העמודות מציינות כל מיני תכונות רפואיות של אותו אדם (התכונות קצת מורכבות ואינכם צריכים להתייחס למשמעות שלהן כלל). בשאלות הבאות תתבקשו לממש אלגוריתמי למידה שונים מנת לאפשר חיזוי מדויק ככל האפשר של היות אדם חולה במחלה.

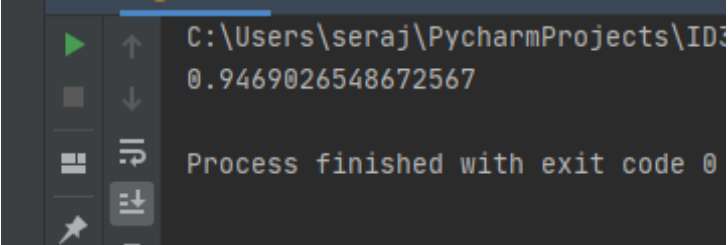
בהצלחה!

1. (15 נק') אלגוריתם ID3:

1.1. ממשו את אלגוריתם ID3 כפי שנלמד בהרצאה.

שימו לב שכל התכונות רציפות. אתם מתבקשים להשתמש בשיטה של חלוקה דינמית המתוארת בהרצאה. כאשר בוחנים ערך סף לפיצול של תכונה רציפה, דוגמאות עם ערך השווה לערך הסף משתייכות לקבוצה עם הערכים הגדולים מערך הסף. במקרה שיש כמה תכונות אופטימליות בצומת מסוים בחרו את התכונה בעלת האינדקס המקסימלי. המימוש צריך להופיע בקובץ בשם ID3.py אשר בהרצתו מאמן עץ על קבוצת האימון בעזרת המימוש שלכם ומדפיס את הדיוק שלו על קבוצת המבחן ללא כל מלל או הדפסות אחרות (ככה יהיו גם כל שאר ההדפסות שלכם בתרגיל).

1.2. אמנו את האלגוריתם על קבוצת האימון ובדקו אותו על קבוצת המבחן. צרפו צילום מסך של תוצאת הדיוק בדו"ח.



```
C:\Users\seraj\PycharmProjects\ID3
0.9469026548672567
Process finished with exit code 0
```

2. (4 נק') הוכח/הפוך: בהינתן דאטה כלשהו עם תכונות רציפות ותיגים בינאריים המחולק לקבוצת אימון ומבחן, הפעלה של פונקציית נירמול MinMax הנלמד בתרגול על הדאטה אינה משפיעה על דיוק של מסווג ID3 הנלמד על קבוצת האימון והנבחן על קבוצת המבחן. [אורך התשובה מוגבל ל-20 שורות]

הוכחה: נוכיח את התכונה הבאה: כאשר האלגוריתם משתמש במשתנים בדידים (נומינליים או סדירים -עם סדר), וחישבו ה-gain ratio בעצי החלטה לא מושפעות בנרמול המספרים. נוכיח באינדוקציה שעבור כל צומת פנימי:

$$IG_{normalized\ data} = IG_{NOT\ normalized\ data}$$

בסיס: נוכיח עבור עלה: נחלק למקרים לפי סוג העלה:

- i. עבור עלה שהוא consistent: במקרה זה נניח בה"כ שכל הדוגמאות בעלה בעל ערך חיובי, אז גם לאחר נרמול כל הדוגמאות, ה-IG בעלה זה לא ישתנה ויהיה שווה ל-0.
- ii. עבור עלה בעל X דוגמאות חיוביות, ו-Y דוגמאות שליליות, ונניח בה"כ ש: $X > Y$, אז לפי האלגוריתם שלנו אנו נסווג את העלה כחיובי, ולכן גם לאחר נרמול הערכים, נקבל אותו יחס של דוגמאות חיוביות/שליליות, וגם כאן ה-IG שווה לאפס באופן ריק.

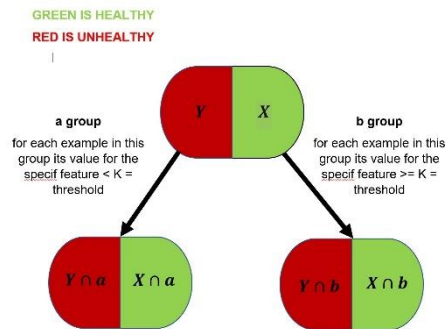
צעד: עבור צומת בעל X דוגמאות חיוביות, ו-Y דוגמאות שליליות, ונניח בה"כ ש: $X > Y$, ובנוסף נניח כי $threshold = K$, ומתקיים שניתן לחלק את הדוגמאות לשתי קבוצות a ו-b, כך שכל איברי a קטנים מ-K וכל איברי b גדולים/שווים ל-K אזי:

$$IG(node) = H(node) - \frac{|a|}{|X+Y|} * H(a) - \frac{|b|}{|X+Y|} * H(b)$$

$$H(node) = -\frac{X}{X+Y} * \log_2\left(\frac{X}{X+Y}\right) - \frac{Y}{X+Y} * \log_2\left(\frac{Y}{X+Y}\right)$$

$$H(a) = -\frac{|a \cap X|}{|a|} * \log_2\left(\frac{|a \cap X|}{|a|}\right) - \frac{|a \cap Y|}{|a|} * \log_2\left(\frac{|a \cap Y|}{|a|}\right)$$

$$H(b) = -\frac{|b \cap X|}{|b|} * \log_2\left(\frac{|b \cap X|}{|b|}\right) - \frac{|b \cap Y|}{|b|} * \log_2\left(\frac{|b \cap Y|}{|b|}\right)$$



עתה נוכיח את הטענה הבאה: לכל תכונה שנמצאת ב-features, ועבור threshold כלשהו, הקבוצות a, ו-b הם אותם קבוצות בדיוק. הסבר: נסתכל על סידור לוקטור ערכים של תכונה כלשהי, נרמול הערכים בווקטור מהתחום הראשוני לתחום $[0,1]$ לא משנה את הסדר בין הערכים של הווקטור, כלומר לכל שני איברים u, v שמקיימים יחס R כלשהו, לאחר הנרמול הם יקיימו אותו יחס R. ולכן כמסקנה נקבל שהקבוצות a, b הם אותם קבוצות עם נרמול או בלי.

ז"א, ההסתברויות $\frac{|a \cap X|}{|a|}$, $\frac{|a \cap Y|}{|a|}$, $\frac{|b \cap X|}{|b|}$, $\frac{|b \cap Y|}{|b|}$ לא משתנות, אי לכך גם

האנטרופיה של שתי הקבוצות לא משתנה, וז"א שה- $IG_{without\ normalization} = IG_{with\ normalization}$.

לסיכום, עבור כל צומת בעץ ההחלטה, ועבור כל ערך סף שנבחר, ותכונה כלשהי, ה-IG של התכונה בהתאם לדוגמאות בצומת ולערך הסף לא יושפע מהנרמול, ולכן גם כן התכונה בעלת ה-IG המקסימלי לא תשתנה, מה שישפיע בסוף על צורת העץ. כלומר מאופן הבנייה (רקורסיה) ומהטענה שהוכחנו באינדוקציה נקבל את אותו עץ בדיוק בשני המקרים (עם/בלי נרמול) ולכן איכות החישוב דומה.

3. (12 נק') גיזום מוקדם.

3.1. הסבירו מה החשיבות של הגיזום באופן כללי ואיזה תופעה הוא מנסה למנוע? [אורך התשובה מוגבל ל3

שורות]

פתרון:

החשיבות הכללית של גיזום עצי החלטה היא להקטין את העץ ולהחליש את אפקט התאמת היתר (מצבו שבו מקבלים שגיאת אימון קטנה ושגיאת מבחן גדולה וזה נובע בד"כ בגלל דוגמאות רועשות), העץ הגזום יגדיל את שגיאת האימון בתקוה להקטנה בשגיאת המבחן, בנוסף נקבל על הדרך אלגוריתם מהיר יותר.

3.2. ממשו את הגיזום המוקדם כפי שהוגדר בהרצאה. הפרמטר M מציין את מספר המינימלי בעלה לקבלת החלטה. על המימוש של הגיזום המוקדם להימצא גם כן בקובץ ID3.py.

3.3. בצעו כיוון לפרמטר M על קבוצת האימון:

i. בחרו לפחות חמישה ערכים שונים לפרמטר M .

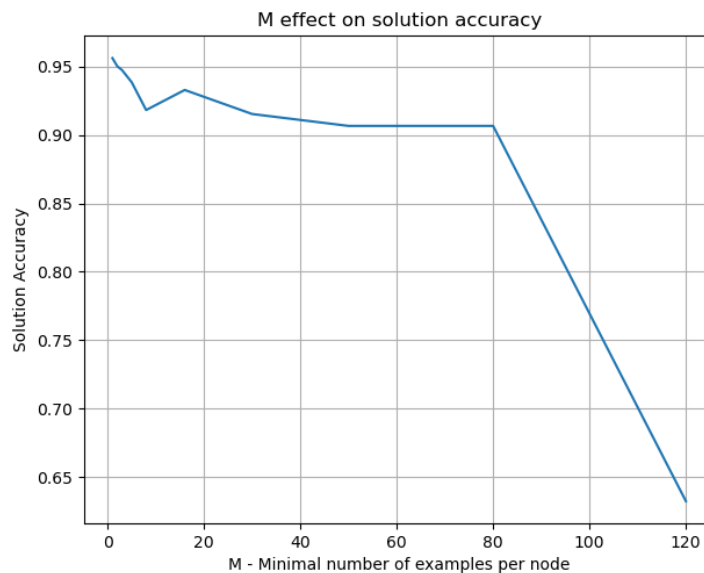
ii. עבור כל ערך, חשבו את הדיוק של האלגוריתם על ידי K-fold cross validation על קבוצת האימון

בלבד. כדי לבצע את חלוקת קבוצת האימון ל- K קבוצות יש להשתמש בפונקציה

[sklearn.model_selection.KFold](#) עם הפרמטרים $n_split=5$, $shuffle=True$ ו- $random_state$ שווה למספר תעודת הזהות שלכם. (כל כיוון פרמטרים בתרגיל יעשה בצורה דומה).

iii. השתמשו בתוצאות שקיבלתם כדי ליצור גרף המציג את השפעת הפרמטר M על הדיוק. צרפו את

הגרף בדו"ח.



iv. הסבירו את הגרף שקיבלתם. לאיזה גיזום קיבלתם התוצאה הטובה ביותר ומהי תוצאה זו? על מימוש כיוון הפרמטר להימצא בפונקציה בשם experiment בקובץ ID3.py. הוסיפו הערה בתחילת פונקציה זו לגבי איך להריץ את הפונקציה שלכם לקבלת הגרף. שימו לב שבריצה של הקובץ ID3.py עדיין אך ורק הדיוק של ID3 ללא גיזום צריך להיות מודפס.

הגרף שקבלתי מתאר את השפעת המשתנה M (מספר הדוגמאות המינימלי בעלה לקבלת ההחלטה) על איכות הפתרון. בגרף לעיל, ציר Y מתאר את איכות הפתרון, כלומר היחס במספר הדוגמאות שהוא צדק בהם באלגוריתם אימות מוצלב (K-fold cross validation). ובציר X יש את הערכים השונים שבדקנו למשתנה M – הערכים הם 1, 2, 3, 5, 8, 16, 30, 50, 80, 120. ניתן לראות שהערך האופטימלי מתקבל באלגוריתם אימות מוצלב בערך $M=1$, וככל שאנחנו מגדילים את גודל המשתנה נקבל אחוז דיוק קטן יותר, כלומר הגיזום הכי טוב שקבלתי הוא שמפסיק את כאשר העלה הוא בעל דוגמה אחת לפחות, וקבלתי דיוק של 0.965.

3.4. השתמשו באלגוריתם ID3 עם הגיזום המוקדם כדי ללמוד מסווג מתוך כל קבוצת האימון ולבצע חיזוי על קבוצת המבחן. השתמשו בערך ה-M האופטימלי שמצאתם בסעיף 3.3. ציינו בדו"ח את הדיוק שקיבלתם? האם הגיזום שיפר את הביצועים ביחס להרצה ללא גיזום בשאלה 1?

ערך ה-M האופטימלי שמצאתי בסעיף 3.3 הוא $M=1$.

בהרצה ללא גיזום קבלנו דיוק: 0.9469.

בהרצה עם גיזום קבלנו דיוק: 0.9469.

ניתן לראות שעבור M האופטימלי שמצאנו בסעיף הקודם, אחוז הדיוק לא משתנה, ולכן הגיזום לא השפיע על הביצועים במקרה הזה.

4. (20 נק') למידה מכוונת מחיר.

במקרה שלנו, בשל אופי הבעיה, סיווג אדם חולה כבריא חמורה פי 10 מסיווג של אדם בריא כחולה. לפיכך, הדיוק שהצגתם בשאלות 1 ו-3 אינו משקף היטב את טיב המסווג שלכם. בעזרת משקול סוגי השגיאות בהתאם למה שלמדתם בתרגול, נגדיר פונקציה חדשה שתייצג את טיב ביצועי המסווגים:

$$loss(C) := \frac{0.1FP + FN}{n}$$

כשאר FP הינו מספר הסיווגים השגויים של בריאים כחולים על ידי המסווג C, FN הינו מספר הסיווגים השגויים של חולים כבריאים על ידי המסווג C ו- n הוא מספר הדוגמאות בקבוצת המבחן.

בשאלה זו תתאימו את אלגוריתם ID3 ללמידה מכוונת מחיר. (אם כבר פתרתם עם KNN התשובות גם כן מתקבלות)

4.1. מדדו את ערך ה-loss של ID3: חזרו על סעיף 3.4 ובמקום הדיוק תמדדו את ערך ה-loss. ה-loss שקבלי הוא 0.02123.

4.2. תארו דרך לגרום לID3 ללמוד מסווג אשר ממזער את פונקציית ה-loss שהוצגה כאן בצורה טובה יותר מאשר האלגוריתם הרגיל..

ניסיתי מספר דרכים שונים, כמו פונקציות אנטרופיה שונות, שימוש במשקלים בפונקציית האנטרופיה, אלגוריתם אחר לבחירת התכונה לבניית העץ ועוד מלא ניסיונות אחרים.

הדרך שאני אציג כאן היא שקבלה loss מינימלי, או לפחות שבאמת שפרה את ה-loss גם בהרצת K-fold וגם בקבוצת המבחן, ולהלן השינויים שבצעתי.

נגדיר Loss(state) להיות ה-Loss המינימלי שיכול להתקבל בצומת נתון כלשהו מבין שני המקרים: כאשר צומת זה מוגדר להיות חיובי (בריאים) וצומת זה מוגדר להיות שלילי (חולים), עבור כל אחד מהמקרים הללו נחשב את ה-loss ונחזיר את המינימום מביניהם.

הרעיון הכללי מאחר השיפור: נרצה להעדיף תכונות אשר מניבות הפרש חיובי מקסימלי מבחינת ערכי ה-loss, כלומר אם בצומת כלשהו יש לנו ערך loss מסוים, ושימוש בתכונה כלשהי עם ערך סף כלשהו מניבים הפרש מקסימלי מבחינת ערכי ה-loss אז אנו נבחר בתכונה זה עם ערך הסף המתאים. ובאופן אינטואיטיבי אפשר לראות שככל שערך זה הוא גדול יותר אז אנו בעצם ממזערים את ה-Loss. ההסבר לכך הוא שאנו מתחילים עם ערך loss לשורש שנתייחס אליו כערך קבוע, וכל שאנחנו נתקדם בעץ, נצפה שערכי ה-loss לצמתיים ירדו.

גישה שונה הייתה שגם כן ממזרעת את ה-loss היא במקום לחשב את ה-loss המינימלי מבין שתי האפשרויות עבור צומת כלשהו, אני אחשב את ה-loss הגרוע ביותר שמתקבל עבור צומת זה, וגם במקרה זה הרעיון הכללי לא משתנה. בהמשך אציג השוואה בין שתי התוצאות.

פונקציית האנטרופיה החדשה:

הערה: במקום לחלק את ה-loss במספר הדוגמאות בפונקציה זו, עשיתי זאת בפונקציה */G*.

```
def lossEntropy(examples):
    # leaf case
    if len(examples) == 0:
        return 0

    # otherwise calculate probabilities
    prob_healthy_example = len([person for person in examples if person[0] == 'B']) / len(examples)
    prob_unhealthy_example = len([person for person in examples if person[0] == 'M']) / len(examples)
    if prob_healthy_example == 0 or prob_unhealthy_example == 0:
        return 0

    # calculate loss without deviding by n
    FP = 0
    FN = 0
    c = calc_loss_for_sub_tree(examples)
    for example in examples:
        if c != example[0] and c == 'B':
            FN += 1
        if c != example[0] and c == 'M':
            FP += 1
    loss = (0.1 * FP + FN)
    return loss
```

הסבר קוד:

קודם כל, אני בודק את מקרה קצה (במקרה שארצה לחשב את ה-loss) לקבוצה ריקה אז אני פשוט אחזיר 0. לאחר מכן גם חישבתי את ההסתברויות לקבלת בן אדם חולה ולקבת בן אדם בריא, בדיקה זו שימשה כדי שצמתיים שמכילים רק סוג אחד של אנשים (חולים או בריאים) אפשר לקבל בהם loss ששווה לאפס ולכן האלגוריתם יעשה זאת. ולבסוף חישבתי את ה-loss הטוב ביותר שמתקבל עם דוגמאות אלו וכל אחד מהסיווגים, ומחזירים בסוף את הערך הבא:

return min (loss assuming this is a 'B' leaf, loss assuming this is a 'M' leaf)

הסבר הפונקציה *calc_loss_for_sub_tree(examples)* יופיע בהמשך, אך בכללי פונקציה זו היא האחראית על סיווג צומת זה כ-'M' או 'B' לפי מי ממזער את ה-loss יותר כפי שהוסבר לעיל ברעיון הכללי.

פונקציית */G*:

```
def IG(f, examples):
    subTree1 = []
    subTree2 = []

    for example in examples:
        if f.classify_example(example):
            subTree1.append(example)
        else:
            subTree2.append(example)

    subTree1_value = (len(subTree1) / len(examples)) * lossEntropy(subTree1)
    subTree2_value = (len(subTree2) / len(examples)) * lossEntropy(subTree2)

    return lossEntropy(examples)/len(examples) - subTree1_value - subTree2_value
```

כאן אני מחשב את הביטוי בדומה לחישוב */G* באלגוריתם הרגיל. ההבדל היחידי הוא שאני משתמש בפונקציית אנטרופיה שונה.

פונקציית `calc_loss_for_sub_tree(examples)`:

```
def calc_loss_for_sub_tree(examples):
    FP = 0
    FN = 0
    # calculate loss (sick people)
    for example in examples:
        if 'B' != example[0]: # sick person is healthy according to our decision
            FN += 1
    loss1 = FN / len(examples)

    for example in examples:
        if 'M' != example[0]: # healthy person is sick according to our decision
            FP += 1
    loss2 = 0.1 * FP / len(examples)

    if loss1 < loss2:
        return 'B'
    else:
        return 'M'
```

הפונקציה מחולקת לשני מקרים:

1. מניחים כי צומת זה מסווג כ-'B' כלומר כולם בריאים, ומחשבים את ה-loss בהתאם.
2. מניחים כי צומת זה מסווג כ-'M' כלומר כולם חולים, ומחשבים את ה-loss בהתאם.

מחזירים את המינימום מבין 1 ו-2 אם נרצה להשתמש בגישה הראשונה, כלומר להסתמך על ה-loss הכי קטן שמתאפשר, או נחליף את הערכים אם נרצה להשתמש בגישה השנייה, כלומר להסתמך על ה-loss הכי גרוע שייתכן.

לפי רעיון זה, כאשר ממקסמים את הפרש ה-loss-ים בין האב לבנים, אנחנו בעצם מקטינים את ה-loss-ים של הילדים, כיוון שאנו מתייחסים שלאב יש loss קבוע כאשר מגיעים למצב שבו נרצה לפתח אותו בעץ.

השוואת תוצאות:

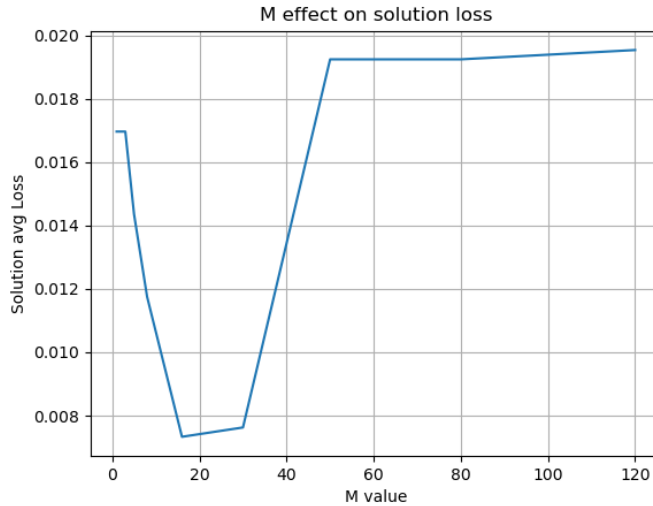
גישה 1 (<i>min loss possible</i>) תוצאות אלה עבור $M=1$ עבור ערכים שונים קבלתי תוצאות טובות יותר	גישה 2 (<i>max loss possible</i>) תוצאות אלה עבור $M=1$ עבור $M=1$ קבלתי את התוצאה הטובה ביותר	
0.0	0.01884057	<i>K-fold test 1</i>
0.00144927	0.0	<i>K-fold test 2</i>
0.02898550	0.01449275	<i>K-fold test 3</i>
0.03088235	0.0	<i>K-fold test 4</i>
0.02352941	0.01911764	<i>K-fold test 5</i>
0.01696930	0.01049019	Avg K-fold loss
0.00733		Avg K-fold loss (M=16)

בנוסף כווננתי את הפרמטר M , שאחראי על הגיזום המוקדם בעץ, אותו פרמטר של שאלה 3, ולהלן הגרף שמתאר את התלות של M על ה-loss ב-K-fold.

אפשר לראות שבשימוש בגיזום מוקדם יחד עם השיטה הראשונה מניבים loss מאוד נמוך ביחס לאלגוריתם המקורי.

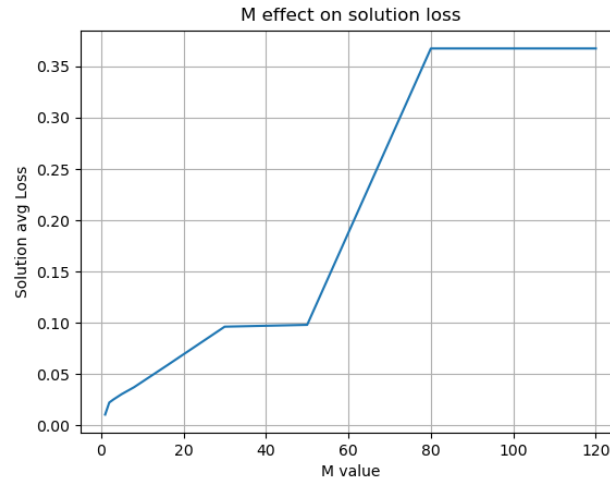
גרף עבור גישה 1:

עבור $M=16$ קבלתי את ה-loss הכי נמוך שהוא 0.00733.



גרף עבור גישה 2:

עבור $M=1$ קבלתי את ה-loss הנמוך ביותר, אבל אפשר לראות שלפי הגישה הראשונה מתקבלים ערכים טובים יותר.



מסקנה: נעדיף להשתמש בגישה 1.

4.3. ממשו הצעתכם בקובץ CostSensitiveID3.py. על קובץ זה להדפיס בהרצתו את ה-loss שקיבלתם מהרצת האלגוריתם המשופר כאשר הוא לומד על קבוצת האימון ונבחן על קבוצת המבחן. אם ביצעתם ניסויים לקביעת פרמטרים לאלגוריתם שלכם, פרטו זאת. כמו כן צרפו בדו"ח את ה-loss שקיבלתם. שימו לב, אינכם צריכים לדאוג מכך שהשיפורים שלכם יפגעו בדיוק ובלבד שהם ישפרו את ה-loss.

ה-loss שהתקבל בקבוצת המבחן הוא : 0.00088, פירוט על כוונן הפרמטר M מופיע לעיל.

```
C:\Users\seraj\anaconda3\envs\ID3.py\python.exe
0.0008849557522123895
```

```
Process finished with exit code 0
```


5. (9 נק') נגדיר דטה סט $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ שבו n דוגמאות מתויגות עם סיווג בינארי $y_i \in \{0,1\}$. כל דוגמה היא וקטור תכונות המורכב משתי תכונות רציפות $x_i = (v_{1,i}, v_{2,i})$. הניחו כי קיים מסווג מטרה $f(x): R^2 \rightarrow \{0,1\}$ שאותו אנו מעוניינים ללמוד (הוא אינו ידוע לנו) וכן שהדוגמאות ב- D עקביות עם מסווג המטרה (כלומר שאין דוגמאות רועשות ב- D).

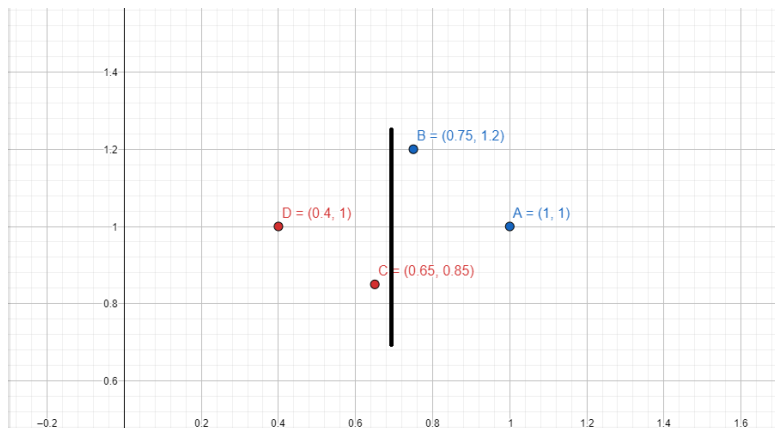
בסעיפים הבאים, עבור KNN, הניחו פונק' מרחק אוקלידי. כמו כן, הניחו שאם קיימות נקודות במרחב כך שעבורן יש מספר דוגמאות במרחק זהה, קודם מתחשבים בדוגמאות עם ערך v_1 מקסימלי ובמקרה של שוויון בערך של v_1 , מתחשבים קודם בדוגמאות עם ערך v_2 מקסימלי. הניחו כי אין דוגמאות זהות לחלוטין (כלומר גם עם ערך v_1 זהה וגם עם ערך v_2 זהה). מסווג ID3 כאן צריך לתאם לכל תנאי קצה שהצגנו למימוש שלכם.

בכל סעיף, הציגו מקרה המקיים את התנאים המוצגים בסעיף, הסבירו במילים, וצרפו תיאור גרפי (ציור) המתאר את המקרה (הכולל לפחות תיאור מסווג המטרה והדוגמאות שבחרתם). סמנו דוגמאות חיוביות בסימן '+' (פלוס) ודוגמאות שליליות בסימן '-' (מינוס). בכל אחת מתתי הסעיפים הבאים אסור להציג מסווג מטרה טריוויאלי, דהיינו שמסווג כל הדוגמאות כחיוביים או כל הדוגמאות כשליליים. [2 שורות לכל סעיף, אין הגבלה על הגרפים, יש להימנע ממלל ופתרון שאינו מוגדר היטב כמתבקש לא מקבל ניקוד] סעיף (א)

הציגו מסווג מטרה $f(x): R^2 \rightarrow \{0,1\}$ וקבוצת אימון בעלת לכל היותר 10 דוגמאות כך שלמידת עץ ID3 תניב מסווג אשר עונה נכון עבור כל דוגמת מבחן אפשרית (כלומר יתקבל מסווג המטרה), אך למידת KNN תניב מסווג שעבורו קיימת לפחות דוגמת מבחן אחת עליה הוא יטעה, לכל ערך K שייבחר. (2 נק')

מסווג המטרה בדוגמה זו הוא:

$$f(x) = \begin{cases} 0 & \text{if } v_{1,i} < 0.7 \text{ (red point)} \\ 1 & \text{otherwise (blue point)} \end{cases}$$



לפי הדרך שבונים בה עץ id3, ה-threshold שיניב את ה-IG האופטימלי הוא 0.7, ואנו נקבל אותו כאשר נחשב את ה-threshold בין C ל-B על ה-feature הראשון, ואז נקבל דיוק 100% ולכן העץ יובל למסווג המטרה. נשים לב כי אין אף ערך סף אחר עבור תכונה כלשהי שיניב IG יותר טוב או שווה לזה, ניתן לחשב את ה-IG עבור כל אחד מערכי הסף של התכונה הראשונה, ונקבל שה-IG האופטימלי יתקבל כאשר $threshold = 0.7$. כנ"ל עבור התכונה השנייה, לא נקבל ערך סף שיניב IG יותר טוב או שווה לזה שתיארנו לעיל. ולכן העץ שמתקבל הוא בדיוק מסווג המטרה.

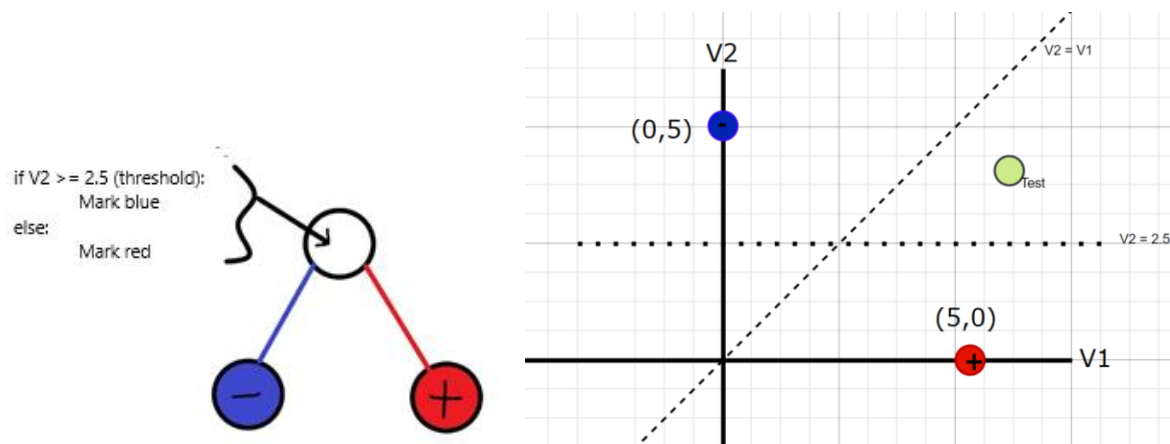
לעומת זאת, באלגוריתם KNN ניתן לראות שעבור כל ערך K , כאשר ניקח את הנקודה (0.7, 0.85), הוא יסווג אותה כאדומה, בניגוד לכך שלפי מסווג המטרה היא צריכה להיות כחולה.

סעיף (ב)

הציגו מסווג מטרסה $f(x): R^2 \rightarrow \{0,1\}$ וקבוצת אימון בעלת לכל היותר 10 דוגמאות כך שלמידת מסווג KNN עבור ערך K מסוים תניב מסווג אשר עונה נכון עבור כל דוגמת מבחן אפשרית (כלומר יתקבל מסווג המטרסה), אך למידת עץ ID3 תניב מסווג אשר עבורו קיימת לפחות דוגמת מבחן אפשרית אחת עליה הוא יטעה. (2 נק')

מסווג המטרסה בדוגמה זו הוא:

$$f(x) = \begin{cases} 0 & \text{if } v_{2,i} > v_{1,i} \text{ (blue point)} \\ 1 & \text{if } v_{1,i} \geq v_{2,i} \text{ (red point)} \end{cases}$$



כפי שניתן לראות באיור הימני, יש לנו שתי דוגמאות, אחת שמוגדרת כחיובית ("+") והשנייה כמינוס (" -"), לפי מימוש עץ id3 עם ערכים רציפים, אנו עוברים על תכונה אחר תכונה, ועבור כל תכונה אנו עוברים על כל ערכי הסף ובודקים אם תכונה זו עם ערך הסף המתאים מניבה IG מקסימלי. במקרה שלנו, אנחנו קודם כל בודקים עבור תכונה v_1 ומחשבים את הממוצע של שני הערכים של v_1 , נקבל כי ערך הסף שווה ל-: $v_1 = \frac{0+5}{2} = 2.5$, ונקבל IG מקסימלי, אבל לפי ההנחיות בתרגיל זה אנו לוקחים את התכונה בעלת האנדקס המקסימלי, כלומר במקרה זה ניקח את התכונה השנייה (v_2) כי גם היא מניבה IG מקסימלי עבור ערך הסף $v_2 = \frac{0+5}{2} = 2.5$. ז"א המסווג המתקבל הוא בדיוק העץ שנמצא באיור השמאלי, ואנו לא נמשיך בתהליך כי קבלנו בכל עלה דוגמאות עם סיווג יחיד לפי קבוצת האימון. אבל ניתן לראות שאנו לא מקבלים מסווג המטרסה ולכן קיימות הרבה דוגמאות שסיווגן שגוי כמו למשל הדוגמה הירוקה באיור הימני שלפי עץ זה היא תסווג כחולה לעומת שלפי מסווג המטרסה היא אמורה להיות מסווגת כאדומה.

ביחס לאלגוריתם KNN, לפי חוקים גיאומטריים, אפשר לראות שכל דוגמה שמקיימת $v_{1,i} > v_{2,i}$ היא תהיה קרובה יותר לדוגמה (5,0), ולכן תסווג כאדום, וכנ"ל כל דוגמה שמקיימת $v_{1,i} < v_{2,i}$ היא תהיה קרובה יותר לדוגמה (0,5), ולכן תסווג כחולה, וכאשר $v_{1,i} = v_{2,i}$ (כלומר כל הדוגמאות שנפלות על הישר $y=x$), יסווגו באדום, כי לפי הנחיות השאלה כאשר קיימות נקודות במרחב כך שעבורן יש מספר דוגמאות במרחק זהה, קודם מתחשבים בדוגמאות עם ערך v_1 מקסימלי, כלומר במקרה שלנו KNN יתחשב בדוגמה החיובית קודם ולכן יסווג אותה כאדומה (סיווג נכון בדומה למסווג המטרסה), ולכן גם עבור מקרה קצה זה נקבל סיווג נכון לפי אלגוריתם KNN.

קבלנו ש-KNN תמיד נותן ערך נכון (מסווג נכון) לעומת שהעץ יחזיר סיווגים שגויים עבור הרבה דוגמאות.

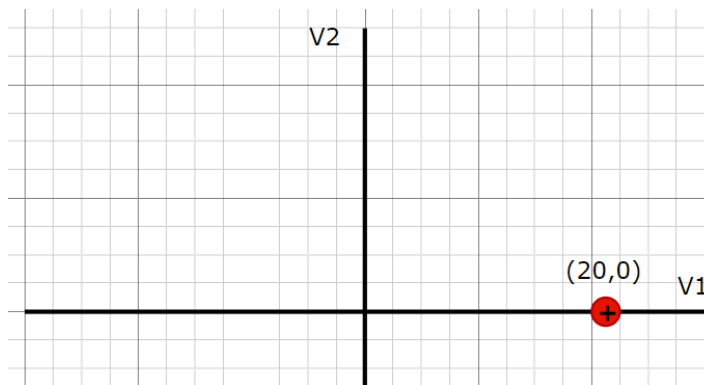
סעיף (ג)

הציגו מסווג מטרסה $f(x): R^2 \rightarrow \{0,1\}$ וקבוצת אימון בעלת לכל היותר 10 דוגמאות כך שלמידת מסווג KNN עבור ערך K מסוים תניב מסווג אשר עבורו קיימת לפחות דוגמת מבחן אפשרית אחת עליה הוא יטעה, וגם למידת עץ ID3 תניב מסווג אשר עבורו קיימת לפחות דוגמת מבחן אחת אפשרית עליה הוא יטעה. (2 נק')

מסווג המטרסה בדוגמה זו הוא:

$$f(x) = \begin{cases} 0 & \text{if } v_{1,i} \geq 15 \text{ (red point)} \\ 1 & \text{otherwise (blue point)} \end{cases}$$

קבוצת האימון אצלנו מורכבת מדוגמה אחת בלבד, שמסומנת כחיובית.



לפי האיור הימני, יש לנו דוגמה אחת שמגודרת כחיובית, ולכן לפי מימוש אלגוריתם של בניית עץ id3, אנחנו נקבל עץ שמורכב מצומת יחיד בלבד שהוא יסווג כאדום, ולכן הוא יראה כמו באיור השמאלי.
עץ id3 המתקבל, יסווג כל דוגמת מבחן כאדומה (חיובית), ולכן עבור כל הדוגמאות שערך התכונה הראשונה אצלם קטן ממש מ-15 העץ יחזיר תשובה שגויה, וזאת כי לפי מסווג המטרסה דוגמאות אלו צריכות להיות מסווגות ככחולות.

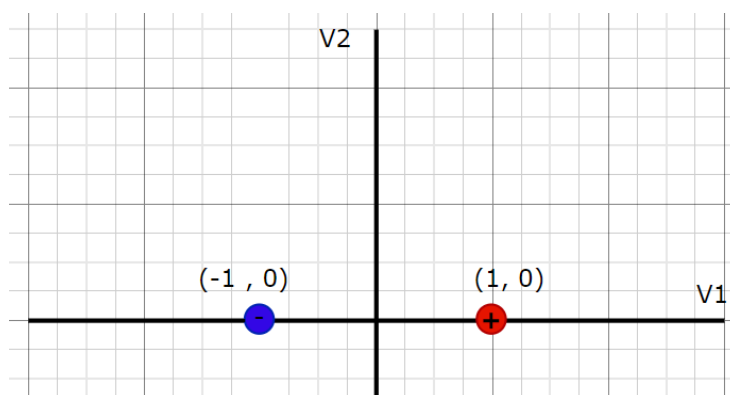
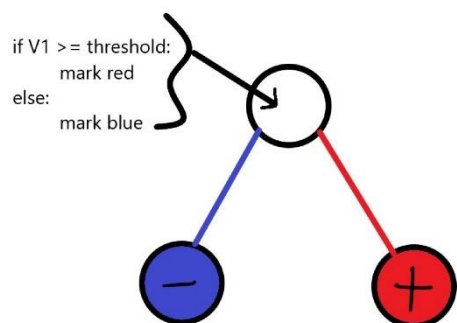
ביחס לאלגוריתם KNN ($1NN$) כי יש לנו רק דוגמה אחת ולכן ערך K לא ישנה את אופי הסיווג, האלגוריתם גם כן יחזיר תשובה שגויה עבור כל דוגמה שבה ערך התכונה הראשונה קטן ממש מ-15, האלגוריתם יסווג את כל הדוגמאות באדום (חיוביות), לעומת שלפי מסווג המטרסה, כל דוגמה בעלת ערך קטן ממש מ-15 בתכונה הראשונה שלו צריך להיות מסווג ככחול. ולכן בשני המקרים ועבור כל K , קיימים דוגמאות מבחן ששני האלגוריתמים (KNN & $id3$) יסווגו אותם בצורה לא נכונה.

סעיף (ד)

הציגו מסווג מטרסה $f(x): R^2 \rightarrow \{0,1\}$ וקבוצת אימון בעלת לכל היותר 10 דוגמאות כך שלמידת מסווג KNN עבור ערך K מסוים תניב מסווג אשר עונה נכון עבור כל דוגמת מבחן אפשרית (כלומר יתקבל מסווג המטרסה), וגם למידת עץ ID3 תניב מסווג עונה נכון עבור כל דוגמת מבחן אפשרית (כלומר יתקבל מסווג המטרסה). (2 נק')

מסווג המטרסה בדוגמה זו הוא:

$$f(x) = \begin{cases} 0 & \text{if } v_{1,i} \geq 0 \text{ (red point)} \\ 1 & \text{otherwise (blue point)} \end{cases}$$



כפי שניתן לראות באיור הימני, יש לנו שתי דוגמאות, אחת שמוגדרת כחיובית ("+") והשנייה כמינוס (" -"), לפי מימוש עץ id3 עם ערכים רציפים, אנו עוברים על תכונה אחר תכונה, ועבור כל תכונה אנו עוברים על כל ערכי הסף ובודקים אם תכונה זו עם ערך הסף המתאים מניבה IG מקסימלי. במקרה שלנו, אנחנו קודם כל בודקים עבור תכונה v_1 ומחשבים את הממוצע של שני הערכים של v_1 , נקבל כי ערך הסף שווה ל-: $\frac{-1+1}{2} = 0$, ולכן נקבל IG מקסימלי, ובעצם אנו נקבל את מסווג המטרסה, כי לפי האלגוריתם שלנו, לכל ערך של התכונה v_1 שהוא גדול או שווה ל-0 אנו נסמן אותו כפלוס, ואחרת נסמן אותו כמינוס. בסה"כ העץ שבנינו יראה כמו באיור השמאלי, ולכן הוא יסווג כל דוגמה בצורה הנכונה כלומר עונה נכון על כל דוגמת מבחן.

ביחס לאלגוריתם KNN ($2NN$), לפי חוקים גיאומטריים, אפשר לראות שכל דוגמה שערך v_1 אצלה גדול ממש מ-0 היא תהיה קרובה יותר לדוגמה $(1,0)$, ולכן תסווג כאדום, וכנ"ל כל דוגמה שערך v_1 אצלה קטן ממש מ-0 היא תהיה קרובה יותר לדוגמה $(-1,0)$, ולכן תסווג ככחולה, וכאשר ערך v_1 שווה ממש לאפס, לפי הנחיות השאלה כאשר קיימות נקודות במרחב כך שעבורן יש מספר דוגמאות במרחק זהה, קודם מתחשבים בדוגמאות עם ערך v_1 מקסימלי, כלומר במקרה שלנו KNN יתחשב בדוגמה החיובית קודם ולכן יסווג אותה כאדומה (סיווג נכון בדומה למסווג המטרסה), ולכן גם עבור מקרה קצה זה נקבל סיווג נכון לפי אלגוריתם KNN .

קבלנו ששני האלגוריתמים יסווגו את דוגמאות המבחן נכון, ולכן הם מסווגי מטרסה.

6. (20 נק') יער.

ננסה עכשיו לשפר את הביצועים שלנו על ידי שימוש בועדות כפי שלמדנו בהרצאה. נציע כאן סוג חדש של יער שתממשו בעצמכם בשם KNN-decision-tree.

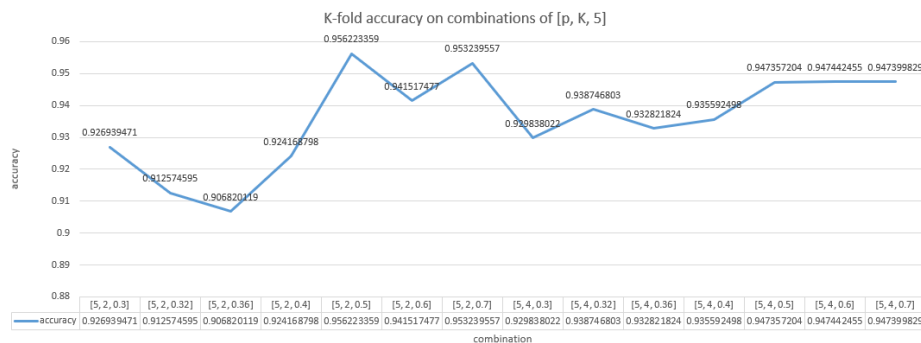
האלגוריתם תחילה ילמד N (פרמטר) עצי החלטה, כל אחד עם תת קבוצה שונה של דוגמאות מקבוצת האימון. בעת סיווג דוגמא חדשה האלגוריתם יבחר K (פרמטר) עצים מתוך ה- N שנלמדו, יסווג את הדוגמא לפי כל אחד מ- K העצים ויחזיר את הסיווג הנפוץ ביותר.

בחירת דוגמאות לכל עץ מתוך N העצים שנלמד: גודל קבוצת האימון יסומן ב- n . עבור כל עץ מ- N העצים הגרילו (באופן רנדומלי) $n \cdot p$ דוגמאות מקבוצת האימון, כאשר p הוא פרמטר לאלגוריתם וערכו בין 0.3 ל-0.7. בחירת K עצים בהם נשתמש לסיווג דוגמא: עבור כל עץ, נחשב centroid: וקטור ממוצע של כל הדוגמאות שבהם השתמשנו לבניית העץ (למשל, אם $p=0.5, n=200$ השתמשנו ב-100 דוגמאות לבניית עץ מסוים. כל "פיצ'ר" בcentroid של עץ זה יהיה ממוצע 100 הפיצ'רים המתאימים ב-100 הדוגמאות בהם השתמשנו). לאחר שחישבנו centroid לכל עץ (תוכלו לעשות זאת כבר בזמן האימון) כאשר נרצה לסווג דוגמא חדשה נבחר את K העצים שהcentroid שלהם קרוב ביותר לדוגמא שאנו רוצים לסווג. למדידת מרחק נשתמש במרחק אוקלידי.

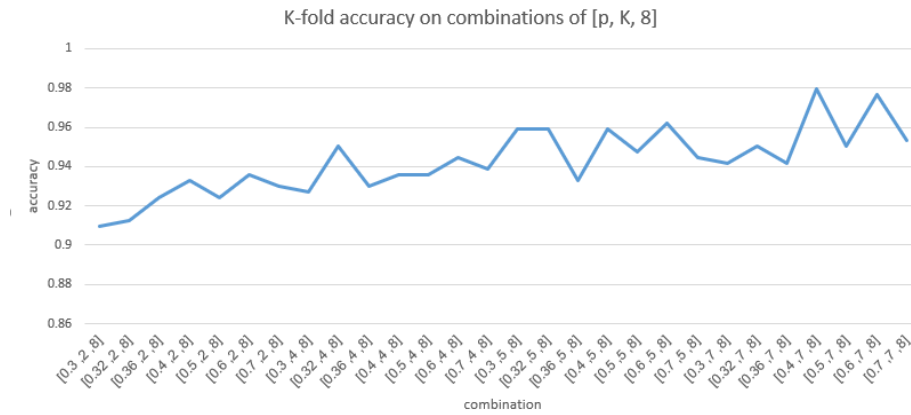
6.1. עליכם לממש בקובץ KNNForest.py את אלגוריתם הלמידה KNN-decision-tree. בהרצת הקובץ האלגוריתם צריך להתאמן על כל קבוצת האימון, ולהדפיס את הדיוק שלו על קבוצת המבחן ללא כל מלל או הדפסות. עליכם לבחור בעצמכם את כל הפרמטרים השונים לאלגוריתם. מומלץ לבצע ניסויים כדי למצוא ערכי פרמטרים טובים. הסבירו אילו ניסויים ביצעתם, וציינו מהו הדיוק המקסימלי שקיבלתם על קבוצת המבחן. שימו לב שמטרת האלגוריתם הינה להשיג דיוק (רגיל) מקסימלי ולא למזער את פונקציית ה-loss.

להלן מספר גרפים המתארים את הדיוק של האלגוריתם עבור קומבינציות שונות של פרמטרי על, כאשר כל הניסויים בוצעו על קבוצת האימון בלבד, כלומר גרפים אלה מתארים את תוצאות KFOLD על קבוצת האימון ללא כל תלות בקבוצת המבחן.

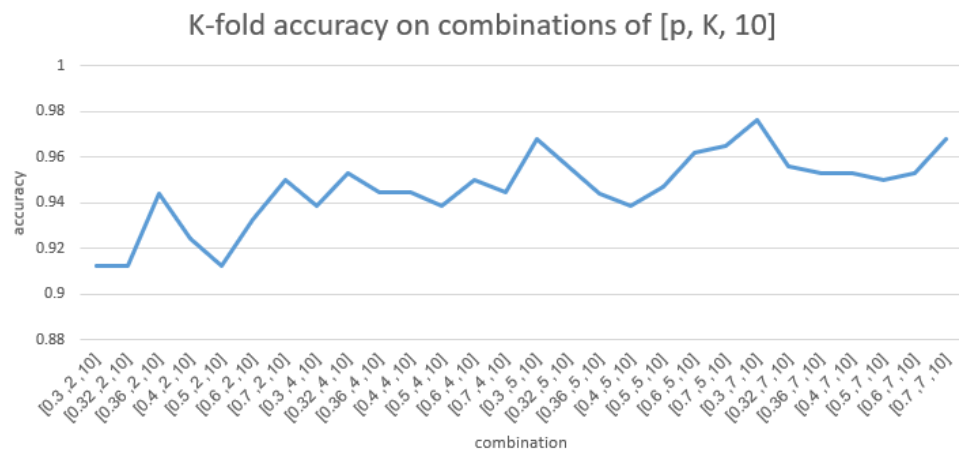
:5=N



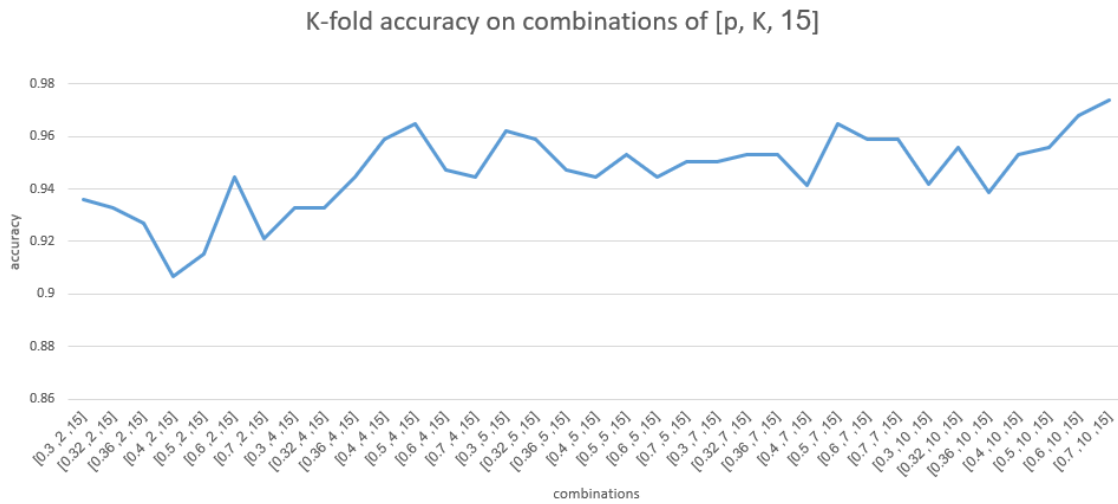
:8=N



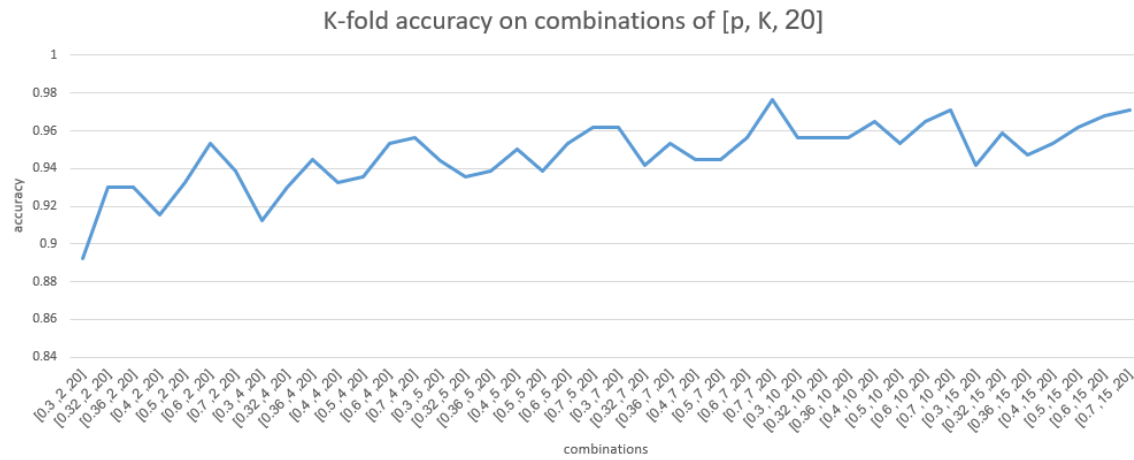
:10=N



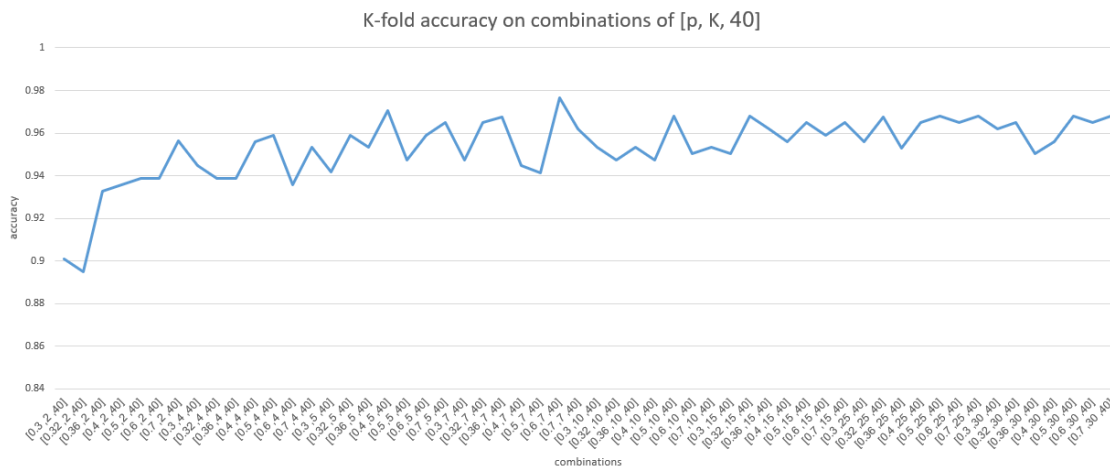
:15=N



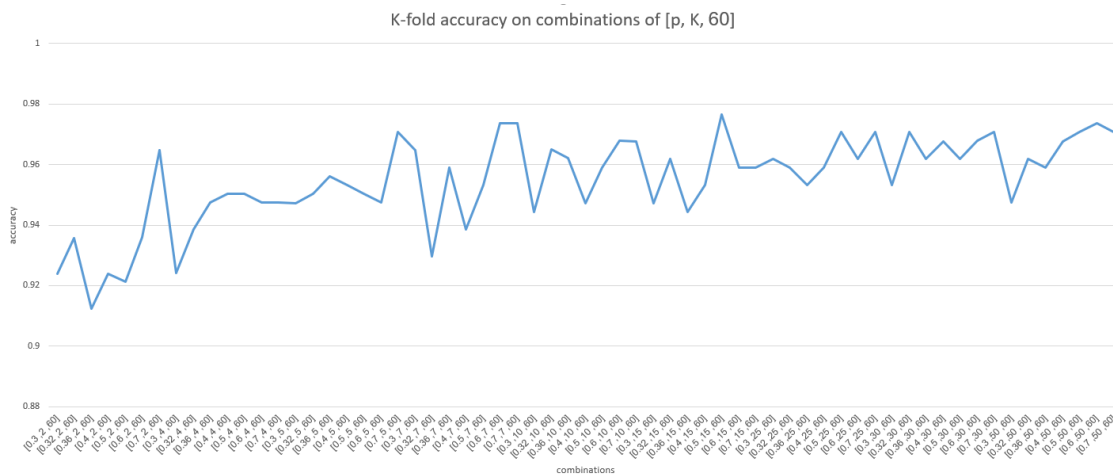
:20=N



:40=N



:60=N



הקומבינציות שנותנת את הדיוק הכי מושלם הם:

דיוק ב-K-FOLD בקבוצה הדוגמאות	קומבינציה [p, K, N]
0.9694117	[0.4, 7, 8]
0.966555	[0.6, 7, 8]
0.966555	[0.3, 7, 10]
0.963657	[0.7, 10, 15]
0.966598	[0.7, 7, 20]
0.966555	[0.6, 7, 40]
0.966555	[0.5, 15, 60]

אפשר לראות שהקומבינציה שמניבה התוצאה הטובה ביותר היא הראשונה.

בבדיקת 20 הרצה עם פרמטרים אלה ([0.4,7,8]) על קבוצת המבחן קבלתי דיוק בין 0.955 ל- 0.8632, כאשר בממוצע הדיוק שמתקבל הוא 0.9625

7. (20 נק') שיפור ליער.

7.1. תארו שיפור לאלגוריתם KNN-decision-tree. השיפור צריך להתייחס למקרה הכללי, ולא ללמידה מוכוונת מחיר בלבד.

תיאור השיפור:

השיפור מורכב משני תתי שיפורים, הראשון חישוב משקלים לתכונות שלנו ב-data-set והשני מתן משקלים לעצים ביער KNN.

שיפור ראשון: משקלים לתכונות (אלגוריתם RFWKNN – Random Forest Weighted KNN):

האלגוריתם שלי, עובר על כל תכונה ותכונה, ומחשב משקל לתכונה באופן הבא:

קודם כל נגדיר ערך שגיא לתכונה בעץ מסוים:

$$Error(feature\ i, Tree\ j) = error\ with\ feature\ i\ in\ Tree\ j - error\ without\ feature\ i\ in\ Tree\ j$$

בנוסף נגדיר ערך ממוצע לשגיאה שמניבה תכונה i באופן הבא:

$$Average\ Error(feature\ i) = \frac{\sum_{k \in K_{Trees}} error(i, k)}{number\ of\ trees = K}$$

לכיוון נגדיר את ה-feature importance:

$$feature\ importance(i) = average\ Error(i)$$

חישוב זה אנו עושים עבור כל תכונה מה-30 תכונות שנתונות לנו ב-train.csv.

המשקלים שקבלתי בהרצה האחרונה הם:

```
weights = [0, 0.0004201680672268907, 0.0008220679576178298, 0.0008281573498964817, 6.938893903907229e-19, 7.930164461608261e-19, 0.00041407867494823957, -6.089392278650196e-06, 0.0037327974668128127, -0.0012544148094020215, -0.0012665935939593232, 9.912705577010326e-20, 0.0008342467421751307, -1.9825411154020652e-19, 0.00041407867494824027, 0.00040798928266958846, 0.0008342467421751318, -0.0004201680672268911, -7.930164461608261e-19, -8.921435019309293e-19, -1.7347234759768072e-19, 0.000840336134453782, -0.009499451954694921, -0.002874193155523078, 0.0016563146997929609, 0.0016684934843502626, -0.002904640116916332, -0.0029472658628668857, 0.002447935696017538, 0.004146876141761052, 0.00462793813177445]
```

אחרי חישוב המשקלים, עדיף לנרמל אותם לטווח בין 0 ל-1 אבל מחוסר זמן השתמשתי בפתרון הבא:

```
for i in range(len(weights)):
    weights[i] = weights[i]+1
```

ז"א עבור כל משקל אני מוסיף אליו 1, כך שכל המשקלים השליליים יעלו מעל 0, זה חשוב כי במקרה ואנחנו מכפילים במינוס זה יפריע לרעה על תוצאת החישוב בהמשך.

משקלים אלה אני מכפיל ב-information gain של אותה תכונה, כפי שניתן לראות כאן:


```

for i in range(len(f_values) - 1):
    threshold = (f_values[i] + f_values[i + 1]) / 2
    new_feature = feature(f_id, threshold)
    ig_f = IG(new_feature, examples) * weights[f_id]
    if ig_f > tmp_best_gain:
        tmp_best_gain = ig_f
        best_threshold_for_curr_f = new_feature

```

כלומר במקרים בהם משקל זה גדול מאוד, הוא יגרום לכך שהמידע המתווסף להיות גדול יותר מהרצה רגילה בלי שיפור זה, במקרים בהם משקל זה הוא קטן מאוד וגדול מאפס, נקבל שמשקל זה ישפיע לרעה על ה-information ולכן זה עוזר לנו.

תיאור מפורט לחישוב המשקלים:

הפונקציה improved_K_fold_experiment היא האחראית על חישוב המשקלים, עבור כל למידה ובחינה בהרצה ב-Kfold אני מחשב את המשקלים מחדש ושומר אותם, ובסוף אני מחשב את ממוצע המשקלים של 5 הרצות ה-Kfold.

בחישוב המשקלים בריצה בודדת, אני עובר כל התכונות, ועבור כל תכונה אני מחשב את השגיאה לכל אחד מה-K העצים שמניבים את המרחק המינימלי מדוגמת הטסט, אבל דאגתי לכך שבכל העצים יש אותם דוגמאות בדיוק כדי לבטל את השפעת ה-random על תהליך חישוב המשקלים. לבסוף אחרי חישוב השגיאות עבור כל אחד מ-K העצים אני מחשב את הממוצע שלהם (כלומר סכום השגיאות חלקי מספר העצים) כפי שתואר לעיל, ועובר לתכונה הבאה... כך בסופו של התהליך מקבל רשימה של משקלים לכל קבוצת לימוד ומבחן שונים ב-K-fold ומחזיר את הממוצע של 5 הרשימות לכל תכונה.

שיפור שני: משקלים לעצים:

הוספת שיפור קטן זה, פותר מקרי קצה בהם אלגוריתם KNN רגיל טועה בסיווג של דוגמאות ספציפיות. השיפור הוא במקרים בהם מספר העצים שמסווגים דוגמה X כלשהיא כחיובית שווה או קרוב מאוד למספר העצים שמסווגים את אותה תכונה כמינוס, במקרה כזה האלגוריתם לא מסווג את הדוגמה באחד מהאפשרויות באופן אקראי, אלא מבצעים צעד נוסף, שהוא חישוב ממוצע המרחקים של העצים מכל סוג וסיווג הדוגמה לפי הממוצע הקרוב ביותר. לדוגמה, כאשר $K=10$ ובמקרה שקיימים 5 עצים שסווגו את הדוגמה X כחיובית, ו-5 עצים שסווגו את הדוגמה כשלילי, האלגוריתם לא מסווג את הדוגמה X באופן אקראי, אלא, מחשב את ממוצע מרחקי 5 העצים שסווגו את הדוגמה כפלוס, ומחשב את ממוצע מרחקי העצים שסווגו את הדוגמה כמינוס, ובוחר לסווג את הדוגמה X לפי הממוצע הקרוב ביותר לדוגמה, (כלומר לפי הממוצע הקטן ביותר, כי אני ממזערים את ההפרש בין הממוצע לדוגמה).

```

# sort the distances
res = {i: dist[i] for i in range(len(dist))}
sorted_res = dict(sorted(res.items(), key=lambda item: item[1]))
# calculate avg distance for positive trees and negative trees.
pos_trees_distances = []
neg_trees_distances = []

Pos = 0
Neg = 0

```

```

Pos = 0
Neg = 0
# choose nearest K
for i in range(K):
    curr_id = list(sorted_res.items())[i][0]
    result = N_trees[curr_id].predict(test)
    if result == 'M':
        Neg += 1
        neg_trees_distances.append(list(sorted_res.items())[i][1])
    else:
        Pos += 1
        pos_trees_distances.append(list(sorted_res.items())[i][1])

if Neg >= 1.2 * Pos:
    results.append('M')
elif Pos >= 1.2 * Neg:
    results.append('B')
else:
    pos_avg_dist = sum(pos_trees_distances[i] for i in range(len(pos_trees_distances))) / len(pos_trees_distances)
    neg_avg_dist = sum(neg_trees_distances[i] for i in range(len(neg_trees_distances))) / len(neg_trees_distances)
    if neg_avg_dist >= pos_avg_dist:
        results.append('M')
    else:
        results.append('B')

```

כאשר אני מגדיר לבצע שיפור זה רק במקרה ומספר העצים שמסווגים את הדוגמה חיובית קרוב מאוד למספר העצים שמסווגים את הדוגמה כשלילית. (קרוב עד כדי יחס 1.2, כלומר במקרה שמספר העצים שמסווגים דוגמה זו כחיובית קטן שווה ל-1.2 ממספר העצים שמסווגים דוגמה זו כשלילית או להפך). שיפור זה ממזער את השגיאה במקרים בהם אין "החלטה ברורה" לסיווג הדוגמה.

בקשר לכוונון פרמטרים, לא היה לי זמן לכוון את הפרמטרים P,N,K,M בחזרה כדי לקבל תוצאות מקסימליות, הרצה כזו לוקחת יום שלם והיא כבדה מדי, לכן נאלצתי להשתמש באותם ערכים שמצאתי בשאלה 6. בנוסף לכך, היה ניתן לכוון את הערך (יחס) שבו אני מבצע תת שיפור 2, כלומר במקום 1.2 להשתמש ביחס אחר, אבל מספר זה מאוד תלוי ב-K ולכן בחרתי אותו בהתאם ל-K=7 שאני משתמש בו בהרצה, ולכן שימוש ביחס אחר דורש כיוונון פרמטרים מחדש מה שלא יכולתי לבצע מחוסר זמן.

הערה כללית: ביחס לנרמול ערכים, בקבוצת המבחן האמיתית נרמול ערכים דווקא פגע באיכות הפתרון, לעומת ב-K-fold זה עזר.

7.2. ממשו הצעתכם בקובץ ImprovedKNNForest.py. על קובץ זה להדפיס בהרצתו את הדיוק שקיבלתם מהרצת האלגוריתם המשופר כאשר הוא לומד על קבוצת האימון ונבחן על קבוצת המבחן. אם ביצעתם ניסויים לקביעת פרמטרים לאלגוריתם שלכם, פרטו זאת. כמו כן צרפו בדו"ח את הדיוק שקיבלתם. הסתמכתי על ערכי הפרמטרים משאלה 6, אולי שיפור לאלגוריתם המשופר שלי, הוא לבצע ניסויים נוספים לקביעת אותם פרמטרים של שאלה 6, ייתכן שעבור פרמטרים אחר במימוש המשופר נקבל תוצאות הרבה יותר טובות מבחינת דיוק.

קבלתי את התוצאות הבאות ב-20 הרצות על קבוצת המבחן (בלי אף נרמול):

```

min accuracy ==> 0.9646017699115044
max accuracy ==> 0.9911504424778761
avg accuracy ==> 0.9840707964601771

```

ניתן לראות שהממוצע עלה בהרבה, גם המינימום.

קבלתי את התוצאות הבאות ב-20 הרצה על אותו מבחן אך עם נרמול:

```

min accuracy ==> 0.9380530973451328
max accuracy ==> 0.9734513274336283
avg accuracy ==> 0.9623893805309731

```

אבל לפי fold-K קבלתי:

בלי נרמול:

```
avg result ==> 0.9503410059676046
```

עם נרמול:

```
avg result ==> 0.9473572037510657
```

אפשר לראות שבלי נרמול מקבלים את התוצאות האופטימליות.