

מבוא לבינה מלאכותית תרגיל בית 3

יובל גושן 205810179

זמני ריצה

נא לשים לב לזמן הריצה של התרגיל האחרון!
זמני הריצה הצפויים:

ID3 – כשנייה

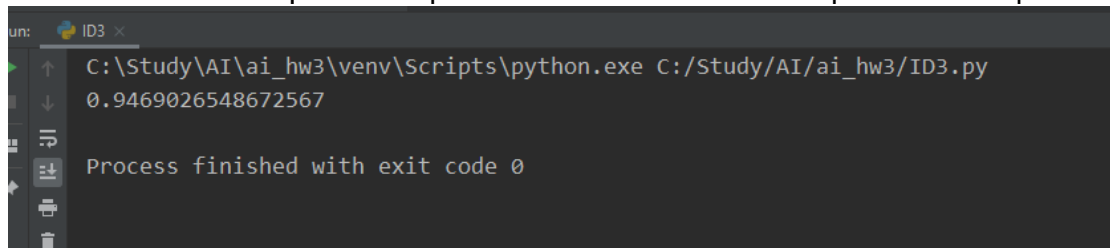
CostSensetiveID3 – כשנייה

KNNForest – כ7 שניות

ImprovedKNNForest – כדקה

שאלה 1

1.2 התקבלה תוצאת דיוק של 0.9469026548672567 על קבוצת המבחן.



שאלה 2

הטענה נכונה. נראה זאת בשני שלבים:

1. העץ שנבנה בשלב האימון מתפצל לפי תכונות זהות אבל עם ערכי threshold המתאימים לנרמול.

נראה שנרמול MinMax מונוטוני, כלומר משמר יחס סדר, נניח שיש 2 דוגמאות ופיצ'ר f עבור דוגמא 1: f_1 ועבור דוגמא 2: f_2 וערכי מקסימום עבור אותו פיצ'ר בכל סט הדוגמאות f_{min}, f_{max} כך שמתקיים $f_1 < f_2$. נפעיל סדרה של פעולות מונוטוניות השומרות על אי השוויון:

$$f_1 - f_{min} < f_2 - f_{min} \Rightarrow f'_1 = \frac{f_1 - f_{min}}{f_{max} - f_{min}} < \frac{f_2 - f_{min}}{f_{max} - f_{min}} = f'_2$$

לכן בכל פעם שנרצה לפצל עץ, ונבחר פיצ'ר Threshold שהוא ממוצע בין כל 2 ערכים כפי שנלמד בהרצאה הפיצול עבור הממוצעים המנורמלים יהיה זהה, ולכן אותה כמות דוגמאות תעבור לכל בן של אותה הצומת, ערך האנטרופיה יהיה זהה עבור כל התכונות ולכן האנטרופיה המינימלית תהיה עבור אותן תכונות ואותם ערכי Threshold מנורמלים, ולכן כל צומת תתפצל באופן זהה כך ש:

$$threshold'_f = \frac{threshold_f - f_{min}}{f_{max} - f_{min}}$$

2. בעץ המנורמל, דוגמאות מנורמלות יעברו באותו מסלול.

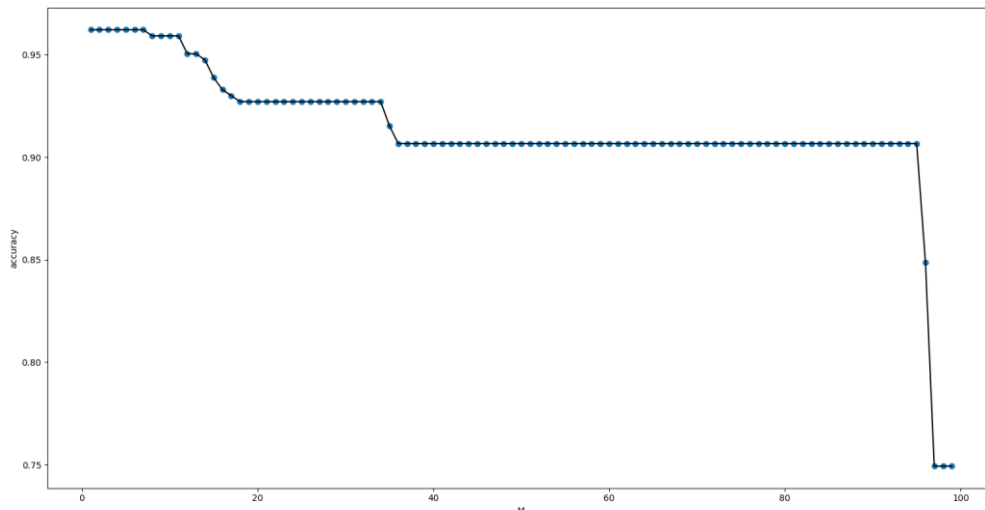
נניח שיש דוגמת מבחן וצומת כלשהי עם פיצ'ר f Threshold T . ערך הפיצ'ר עבור דוגמת המבחן הוא f_1 ונניח בה"כ $f_1 < T$, כלומר הדוגמא תעבור לצומת הבן שמתאימה לערכים קטנים מה-Threshold. מתכונת שימור הסדר של הנרמול שהראנו קודם, מתקיים גם $f'_1 < T'$ ולכן גם הדוגמא המנורמלת תעבור לאותו בן של אותה צומת בעץ המנורמל. זה נכון עבור כל פיצול בעץ ולכן בסופו של דבר דוגמת המבחן תעבור באותו מסלול, תגיע לאותו עלה ותסווג באופן זהה.

הראנו שהעץ שיבנה יהיה זהה רק עם ערכים מנורמלים של Threshold ומצד שני דוגמאות המבחן יגיעו לאותם עלים ולכן הסיווגים יהיו זהים במקרה של נרמול ולא תהיה השפעה על התוצאה עבור קבוצת המבחן (כל עוד גם היא מנורמלת באופן זהה)

שאלה 3

3.1. הגיזום מנסה למנוע את תופעת התאמת היתר (overfitting) בה האלגוריתם מאומן על קבוצת אימון אך לא יודע להכליל ולקבל תוצאות מספיק טובות על קבוצות אחרות. החשיבות שלו הוא שהוא מוריד עלים שהתקבלו ככל הנראה מדוגמאות שהן "רעש" ולא מייצגות את העולם באופן כללי.

3.3 מצורף הגרף, אמנם נתבקשנו לבחור לפחות 5 ערכי M שונים אבל בגלל נדיבות יתר שלי תקבלו אפילו 100 :



למעשה התוצאה הטובה ביותר היא עבור $M=1$, והיא יורדת ככל שמעלים את M (בחלק מהנקודות היא לא משתנה כי אין עלה שמתפצל לפי אותה כמות דגימות). כלומר ללא גיזום מוקדם כלל התקבלה התוצאה הטובה ביותר. התוצאה הממוצעת בין folds על validation set היא 96 אחוז.

3.4 מכיוון שהתוצאה הטובה ביותר היא ללא גיזום בכלל בסעיף זה קיבלנו שוב תוצאה של 0.9469026548672567 ואין שיפור בעץ.

שאלה 4

4.1 קיבלתי את ערך הloss הבא:

```
04
05 delta_diagnosis = predicted_diagnosis - test_set_labels
06 false_negative = sum(delta_diagnosis == -1)
07 false_positive = sum(delta_diagnosis == 1)
08 # print("loss = ", (0.1*false_positive + false_negative)/len(test_set_labels))
09
main()
ID3 x
C:\ProgramData\Anaconda3\envs\ai_hw3\python.exe C:/Study/AI/ai_hw3/ID3.py
0.9469026548672567
loss = 0.021238938053097345
```

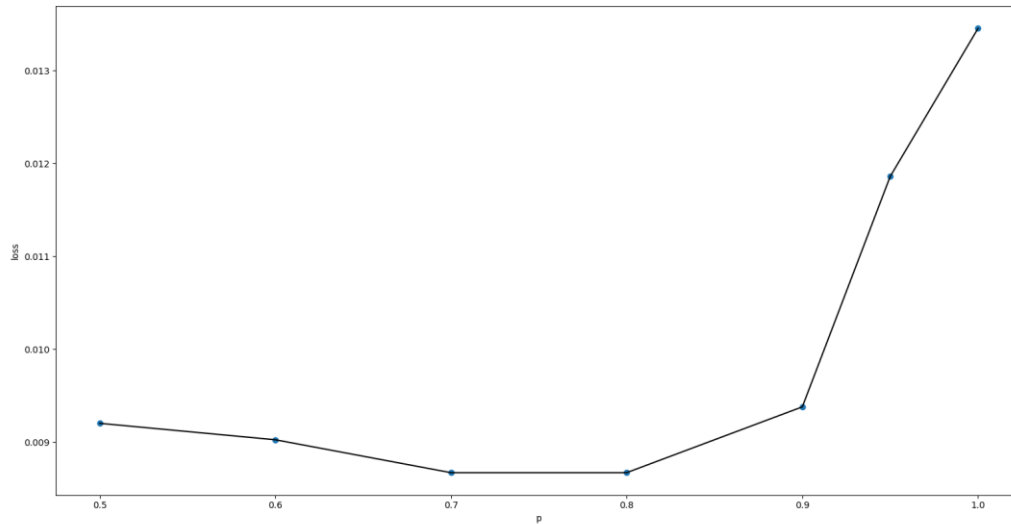
0.021238938053097345

הוא נכון גם עבור "הגיזום הכי טוב" כי הגיזום הכי טוב הוא עבור $M=1$

4.2 נבצע החמרה בכל פיצול במקרים בהם רוב הדוגמאות הם של חולים. נגדיר פרמטר $p \in [0.5, 1]$ ובכל פעם שאנחנו מפצלים צומת, נבדוק אם החלק של הדגימות שערכן 1 (חולה) מכל הדגימות גדול מ-p, אם כן הצומת הרלוונטי יהפוך לעלה שנותן חיזוי של חולה. כך בעצם האלגוריתם מחמיר ונותן

חיזוי של חולה לכל קבוצה בה יש רוב משמעותי של חולים ולא ממשיך לפצל אותה מתוך הנחה שיש הסתברות שהפיצול יתבסס על דגימות בודדות שנובעות מרעש.

4.3 ביצעתי תהליך cross validation עם $k\text{-fold}=5$ בדומה לשאלה הקודמת על הערך p שהוגדר. תוצאות הניסוי:



שהק הטוב ביותר הוא 0.7 או 0.8, בחרתי להשתמש ב 0.8 ושיפרתי משמעותית את loss.

קיבלנו:

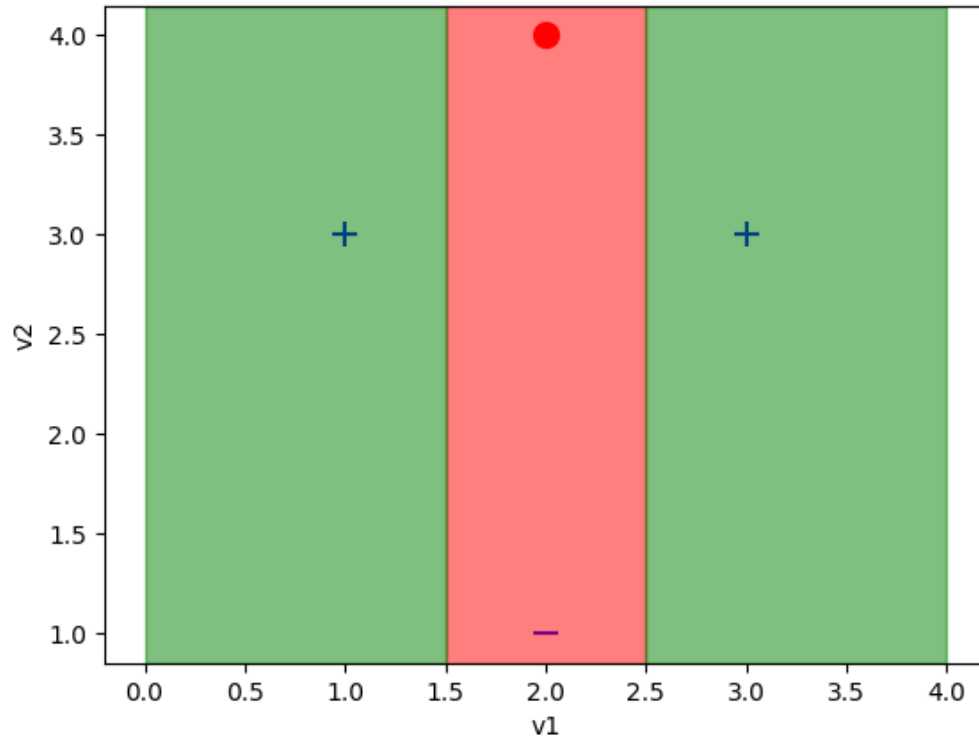
```
Run: CostSensetiveID3
C:\ProgramData\Anaconda3\envs\ai_hw3\python.exe C:/Study/AI/ai_hw3/CostSensitiveID3.py
loss = 0.003539823008849558
Process finished with exit code 0
```

$loss = 0.003539823008849558$

שאלה 5

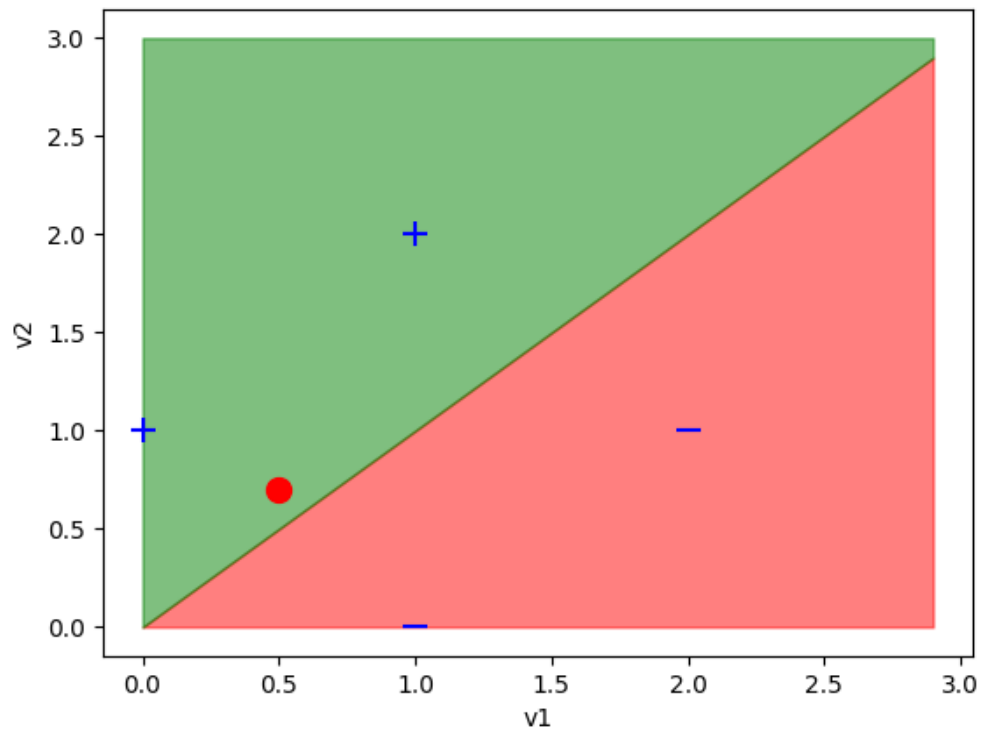
בכל הגרפים הבאים צבע הרקע מסמן את מסווג המטרה $f(x)$, כלומר הסיווג הנכון. רקע אדום מסמן דוגמא שלילית ורקע ירוק מסמן דוגמא חיובית.

סעיף א'



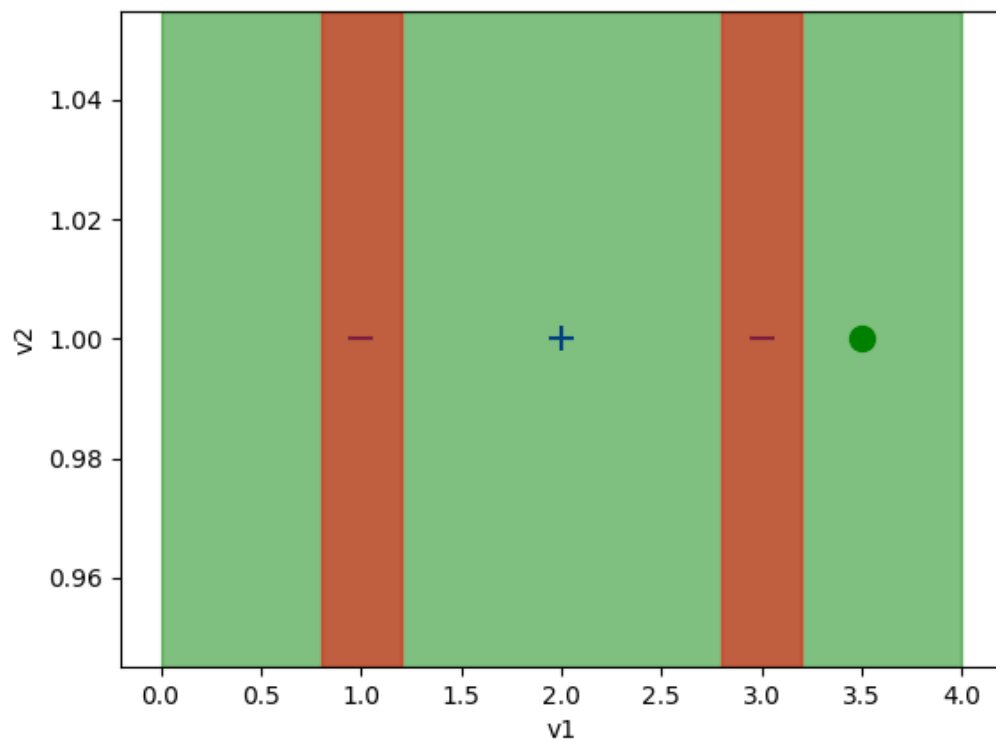
עבור דוגמת המבחן המסומנת בעיגול אדום, מסווג KNN יטעה עבור כל K כיוון שתמיד יהיו יותר פלוסים קרובים. לכן יהיה קל ללמוד - פשוט 2 פיצולים לפי v_1 .

סעיף ב'



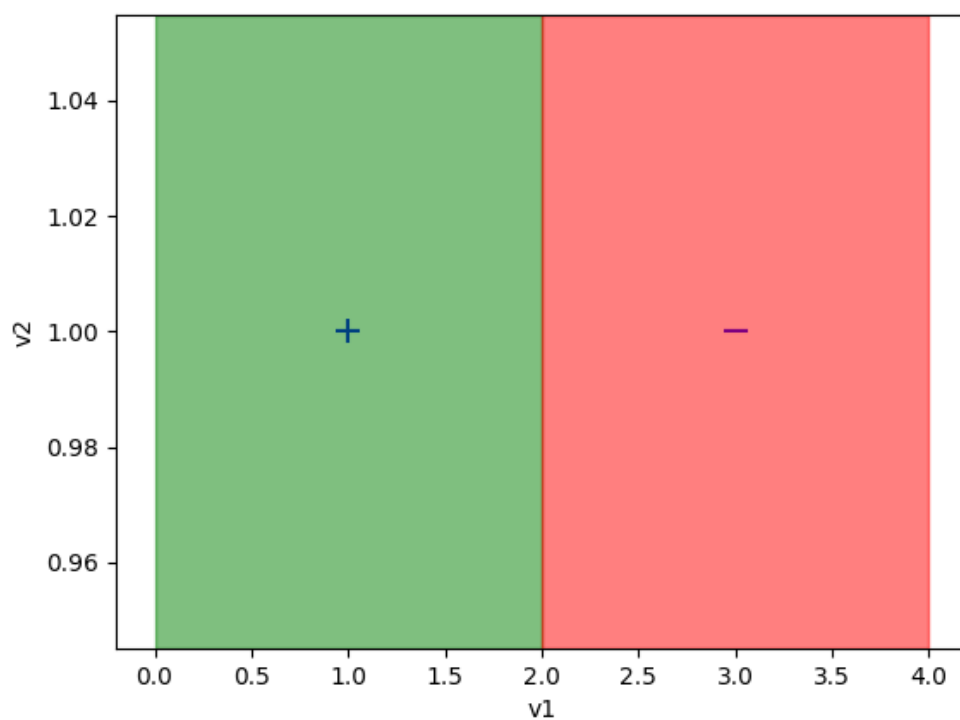
כאן KNN עם $K=1$ תמיד יהיה צודק כי דוגמאות באזור הירוק תמיד יהיו קרובות יותר לפלוס בעוד שדוגמאות באזור האדום יהיו קרובות יותר למינוס אבל ID3 לא ידע להפריד טוב לפי קווים אלכסוניים.

סעיף ג'



כאן גם KNN וגם ID3 יטעו עבור דוגמת המבחן שמשומנת בעיגול ירוק. עבור KNN עם $K=1$ הכי קרוב הוא מינוס, ID3 יפצל את המרחב לפי דוגמת המינוס הימנית כך שכל מה שמימינה יקבל סיווג שלילי.

סעיף ד'



כאן KNN עם $K=1$ תמיד יצדק כי משמאל לקו ההפרדה תמיד יהיה קרוב לדוגמא החיובית ולהיפך מצד ימין. ID3 מפצל לפי ממוצע ערכי הפיצ'ר לכן הוא ילמד פיצול אחד בדיוק בין 2 הדוגמאות.

שאלה 6

ביצעתי ניסויים עבור 3 פרמטרים: M, K, k . אם היינו רוצים לבצע ניסויים על כל הקומבינציות, והיינו בוחרים 10 ערכים לכל פרמטר (K, M רציתי לבחור הרבה יותר) היינו מקבלים שצריך לעשות $10^3 = 1000$ ניסויים כפול 5 K-folds והזמן יהיה לא סביר (סדר גודל של 15 שעות), לכן ביצעתי ניסוי עבור כל פרמטר בנפרד. בכל ניסוי ביצעתי 5-fold cross validation כך שעבור כל fold ביצעתי 5 ניסויים ועשיתי עליהם ממוצע כדי לנקות רעש שנובע מהאקראיות של היער של היער. ניסיתי ערכי M מתוך: $\{5, 10, 15, \dots, 100\}$ וקיבלתי שהשונויות נמוכה יחסית אבל התוצאה הטובה ביותר היא $M=75$.

לאחר מכן ניסיתי ערכי K אי זוגיים מתוך $\{3, 5, 7, \dots, 73\}$ וקיבלתי תוצאה טובה ביותר עבור $k=23$. לאחר מכן ניסיתי ערכי k מתוך $\{0.3, 0.4, 0.5, 0.6, 0.7\}$ וקיבלתי תוצאה טובה ביותר עבור 0.3. הציון על קבוצת המבחן נע בין 97-100 אחוזי הצלחה כשבדרך כלל הוא בסביבות ה-98 אחוז. דוגמא להרצה:

```
C:\ProgramData\Anaconda3\envs\ai_hw3\python.exe C:/Study/AI/ai_hw3/KNNForest.py
0.9823008849557522
```

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


שאלה 7

7.1 שני שיפורים אפשריים:

1) ראינו בהרצאה שאת אלגוריתם KNN אפשר לשפר באמצעות מתן משקל גדול יותר לשכנים יותר קרובים, כלומר, במקום לספור שכנים מכל מחלקה, נוסיף גם את המרחק לחישוב ושכנים קרובים יותר יקבלו משקל גדול יותר.

לכל עץ מהא הכי קרובים נחשב משקל :

$$w_j = \frac{\frac{1}{\text{distance}(j)}}{\sum_{i \in K} \frac{1}{\text{distance}(i)}}$$

כאשר K היא קבוצת כל השכנים הקרובים. סכום המשקלים יהיה 1 ולעצים קרובים יותר יהיה משקל גדול יותר.

לאחר מכן נסכום את כל הסיווגים עם המשקלים שלהם כאשר סיווג בריא הוא 1- (לא אפס) וסיווג חולה הוא 1, אם נקבל תוצאה גדולה מאפס סימן שבממוצע ממושקל הועדה סיווגה את הדוגמא כחולה, אחרת כבריא.

לאחר מספר ניסויים, גיליתי שהעלאת המרחק במכנה בריבוע משפר עוד יותר את האלגוריתם מכיוון שהוא נותן עוד יותר משקל לשכנים שיותר קרובים, ולכן שיניתי את הגדרת המשקל ל:

$$w_j = \frac{\frac{1}{\text{distance}(j)^2}}{\sum_{i \in K} \frac{1}{\text{distance}(i)^2}}$$

2) שיפור נוסף הוא הוספת פיצ'רים מסדר 2, כלומר כפל של כל פיצ'ר בכל דוגמא בכל הפיצ'רים האחרים, כך במקום 30 תכונות יש 900 ואפשר לייצר מודל יותר "עשיר".

7.2 לאחר הכנסת השיפורים, גיליתי שמשקול הKNN בפועל לא משפר את התוצאות גם עם העלאה בריבוע ולכן עדיף להשתמש בKNN רגיל. הוספת פיצ'רים לעומת זאת משפרת את המודל (למרות שמאריכה במעט את זמן הריצה). השארתי את המימוש עם משקול בקובץ ImprovedKNNForest אבל בפועל השתמשתי במימוש ללא משקלים אבל עם השיפור של הוספת פיצ'רים מסדר 2.

ביצעתי ניסויים זהים לשאלה 6 וקיבלתי את הערכים:
M=45, K=17, p=0.5 והדיוק יוצר לרוב מעל 98 אחוז, בערך 99.

```
C:\ProgramData\Anaconda3\envs\ai_hw3\python.exe C:/Study/AI/ai_hw3/ImprovedKNNForest.py  
0.9911504424778761
```

(יש לשים לב שזמן הריצה כעת הוא כדקה)