# תרגיל בית 3 – MDP ומבוא ללמידה

עברו על כלל ההנחיות לפני תחילת התרגיל.

## הנחיות כלליות:

- תאריך ההגשה: 06/07/23 ב23:59
- את המטלה יש להגיש **בזוגות בלבד.**
- יש להגיש מטלות מוקלדות בלבד. פתרונות בכתב יד לא ייבדקו.
- ניתן לשלוח שאלות בנוגע לתרגיל בפיאצה בלבד.
- המתרגל האחראי על תרגיל זה: **אור רפאל בידוסה.**
- בקשות דחיה מוצדקות (מילואים, אשפוז וכו') יש לשלוח למתרגל האחראי (**ספיר טובול**) בלבד.
- במהלך התרגיל ייתכן שנעלה עדכונים, למסמך הנ"ל – תפורסם הודעה בהתאם.
- העדכונים הינם מחייבים, ועליכם להתעדכן עד מועד הגשת התרגיל.
- שימו לב, התרגיל מהווה כ- 15% מהציון הסופי במקצוע ולכן העתקות תטופלנה בחומרה.
- התשובות לסעיפים בהם מופיע הסימון ✍ צריכים להופיע בדוח.
- לחלק הרטוב מסופק שלד של הקוד.
- אנחנו קשובים לפניות שלכם במהלך התרגיל ומעדכנים את המסמך הזה בהתאם. גרסאות עדכניות של המסמך יועלו לאתר. הבהרות ועדכונים שנוספים אחרי הפרסום הראשוני יסומנו כאן בצהוב. ייתכן שתפורסמנה גרסאות רבות – אל תיבהלו מכך. השינויים בכל גרסה יכולים להיות קטנים.

שימו לב שאתם משתמשים רק בספריות הפייתון המאושרות בתרגיל (מצוינות בתחילת כל חלק רטוב)
לא יתקבל קוד עם ספריות נוספות

מומלץ לחזור על שקפי ההרצאות והתרגולים הרלוונטיים לפני תחילת העבודה על התרגיל.

# חלק א' – MDP (60 נק')

## רקע

בחלק זה נעסוק בתהליכי החלטה מרקובים, נתעניין בתהליך עם **אופק אינסופי** (מדיניות סטציונרית).

## חלק א' - חלק היבש ✍️

1. בתרגול ראינו את משוואת בלמן כאשר התגמול ניתן עבור המצב הנוכחי בלבד, כלומר $R: S \to \mathbb{R}$, למתן תגמול זה נקרא "תגמול על הצמתים" מכיוון שהוא תלוי בצומת שהסוכן נמצא בו.
בהתאם להגדרה זו הצגנו בתרגול את האלגוריתמים Value iteration ו-Policy Iteration למציאת המדיניות האופטימלית.

בעת, נרחיב את ההגדרה הזו, לתגמול המקבל את המצב הנוכחי והפעולה לביצוע שבה בחר הסוכן, כלומר: $R: S \times A \to \mathbb{R}$, למתן תגמול זה נקרא "תגמול על פעולה".

**א.** (2 נק') התאימו את הנוסחה של התוחלת של התועלת מהתרגול, עבור התוחלת של התועלת המתקבלת במקרה של "תגמול על פעולה", אין צורך לנמק.

הנוסחה החדשה שמתקבלת הינה: $U^{\pi}(s) = \mathrm{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s\right]$

**ב.** (2 נק') כתבו מחדש את נוסחת משוואת בלמן עבור המקרה של "תגמול על פעולה", אין צורך לנמק.

משוואת בלמן עבור המקרה החדש: $U(s) = \max_{a \in A(s)} \left[ R(s,a) + \gamma \sum_{s'} P(s' | (s,a)) \cdot U(s') \right]$

**ג.** (4 נק') נסחו את אלגוריתם Value Iteration עבור המקרה של "תגמול על פעולה".

השינוי באלגוריתם אינו גדול וניתן לראותו בפסודו קוד הבא:

```
Repeat
      U ← U', δ ← 0
      For each state s in S do:
            U'[s] ← max_{a∈A(s)} [ R(s,a) + γ Σ_{s'} P( s'|(s,a) ) · U(s') ]
            δ ← max ( δ, |U'[s] − U[s]| )
Until δ < ϵ(1 − γ)/γ
```

במקרה בו $\gamma = 1$ נשנה את תנאי העצירה כך שהאלגוריתם הופך להיות:

```
Repeat
      U ← U′, δ ← 0
      For each state s in S do:
            U′[s] ← max_{a∈A(s)} [ R(s,a) + γ Σ_{s′} P( s′|(s,a) ) · U(s′)]
            δ ← max ( δ, |U′[s] − U[s]| )
Until δ = 0
```

כך שבעצם עכשיו עוצרים כאשר אין שינוי ממש בין ערכי תוחלת התועלת בין איטירציות, ועשינו את השינוי בתנאי העצירה כי עכשיו אין התכנסות כמו קודם שנבעה מזה שהמקדם קטן מאחד ו קיבלנו סכום גאומטרי. נציין שכאן צריך להיות זהירים עם הערך שנותנים ל *Rewards* כי למשל אם נותנים ערכים חיוביים נקבל שתוחלת התועלת הינה אינסופית) לא נעצור).

**ד.** (4 נק') נסחו את אלגוריתם Policy Iteration עבור המקרה של "תגמול על פעולה".

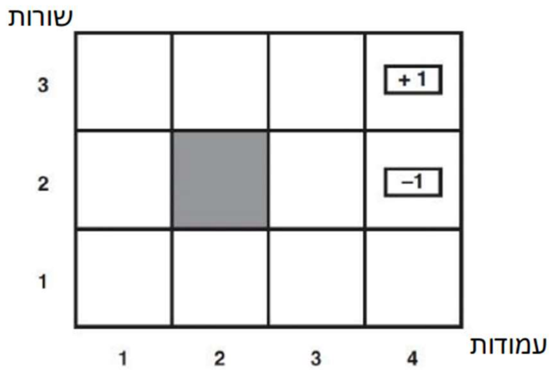
השינוי באלגוריתם אינו גדול וניתן לראותו בפסודו קוד הבא וניתן לראותו בעיקר בפונקציית getSum() שבה משתמשים בתועלת שהגדרתה שינינו להיות כפי שמפורט למטה:

```
def getSum( s, a, γ):
      return Σ_{s′} P( s′|(s,a) ) · U(s′)
such that: U[s] ← max_{a∈A(s)} [ R(s,a) + γ Σ_{s′} P( s′|(s,a) ) · U(s′)]
Repeat
      U ← Policy − Evaluation(π, U, mdp) , Unchanged ← True
      For each state s in S do:
            if max_{a∈A(s)} [ getSum(s,a,γ)] > [ getSum(s,π(s),γ)]:
                  π(s) ← arg max_{a∈A(s)} [ getSum(s,a,γ)]
                  Unchanged ← False
            δ ← max ( δ, |U′[s] − U[s]| )
Until Unchanged == True
```

במקרה בו $\gamma = 1$ האלגוריתם לא יתכנס כי תמיד יהיה הפרש בין המקסימום של getSum על כל הפעולות האפשריות ו getSum על הפוליסה שלנו ולכן תמיד יהיה Unchanged == False מה שגורם להרצה שלא נגמרת. אלא אם כן התחלנו בהתחלה עם הפוליסה האופטימלית ואז נסיים באיטרציה אחת כי לא נמצא הפרש כמו קודם ו Unchanged == True כתוצאה מכך מה שגורם לסיום האלגוריתם באיטרציה הראשונה.

הערה: בסעיפים ג' וד' התייחסו גם למקרה בו $\gamma = 1$, והסבירו מה לדעתכם התנאים שצריכים להתקיים על הסביבה\mdp על מנת שתמיד נצליח למצוא את המדיניות האופטימלית.

2. נתון ה־MDP הבא $< S, A, P, R, \gamma >$, אופק אינסופי:




<u>מצבים:</u>

$$S = \{(1,1),(1,2),(1,3),(1,4),(2,1),(2,3),(2,4),(3,1),(3,2),(3,3),(3,4)\}$$
$$S_G = \{(2,4),(3,4)\}$$

<u>פעולות</u>

$$\forall S \backslash S_G : A(s) = \{Up, Down, Left, Right\}$$

<u>תגמולים:</u>

נתונים התגמולים של המצבים הסופיים בלבד: $R\big((2,4)\big) = -1, R\big((3,4)\big) = +1$.

שימו לב, התגמולים הינם תגמולים על המצבים.

ישנם תגמולים עבור שאר המצבים, הם פשוט לא נתונים כחלק מהשאלה.

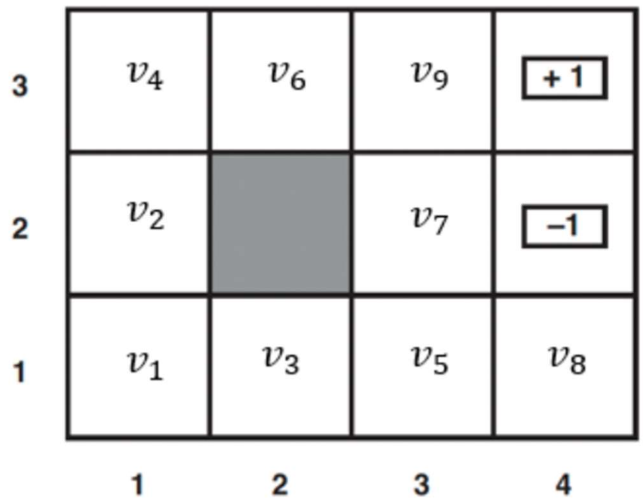
<u>מודל מעבר:</u>

כל פעולה "מצליחה" בהסתברות 0.8, ואם היא לא מצליחה אז בהסתברות שווה מתבצעת אחת הפעולות המאונכות לפעולה המתבקשת. כאשר הסוכן הולך לכיוון הקיר או מחוץ ללוח הוא נשאר במקום.

<u>מקדם דעיכה:</u> $0 < \gamma < 1$.

הרצתם את האלגוריתם $value\ iteration$ עם $\varepsilon \to 0$ וקיבלתם את הפלט הבא:

(משמעות הדבר ש־$\varepsilon \to 0$ היא שתנאי העצירה קיֵים שנורמה האינסוף בין ווקטורי התועלת הייתה אפסית, כלומר לאחר הריצה ערכי התועלת שהתקבלו מקיימים את משוואת בלמן).




כאשר $v_i$ הוא ערך התועלת למצב ה־$i$ כפי שניתן לראות בתרשים. בנוסף נסמן את התגמול למצב ה־$i$ ב־$r_i$.

ענו נכון \ לא נכון, וספקו הסבר קצר או דוגמה נגדית <u>מפורטת</u>.

א. (3 נק') אם $v_9 > 1$, אז בהכרח מתקיים ש־$r_9 > 1$. נכון \ לא נכון.

נימוק \ דוגמה נגדית: ניקח למשל המקרה הבא שבו, $\gamma = 0.96$ ונבחר $r_7 = 2, r_9 = 0.5 < 1$ כאשר מתחילים במצב (3,3) נקבל שמתקיים $[\sum_{s'} P(s'|(s,a)) \cdot U(s')]$ $v_9 = r_9 + \gamma \max_{a \in A(s)}$ נבצע החישוב הזה עבור הפעולה

$a = 'RIGHT'$ ובטח שמקסימום גדול / שווה לערך שנקבל:

$$v_9 \geq r_9 + \gamma \sum_{s'} P\big(s'\big|(s,'RIGHT')\big) \cdot U(s')$$

$$= r_9 + \gamma\big(0.8 * R\big((3,4)\big) + 0.1 * r_7 + 0.1 * r_9\big)$$

$$= r_9 + \gamma(0.8 + 0.1 * r_7 + 0.1 * r_9) = 0.5 + \gamma(0.8 + 0.2 + 0.05)$$

$$= 0.5 + 1.05 \cdot 0.96 = 1.508$$

מזה נובע ש: $v_9 \geq 1.508 > 1$ אך ראינו שבהנחה שלנו $r_9 = 0.5 < 1$ לכן הטענה שגויה.

ב. (3 נק') אם $\forall i \in [9]: v_i > 0$, אז בהכרח $\exists i \in [9]: r_i > 0$. נכון \ לא נכון.

נימוק \ דוגמה נגדית: ניקח את הדוגמה הבאה: אשר בה $\forall i \in [9]: r_i = 0$ אך $\forall i \in [9]: v_i > 0$ שראינו אותה בתרגול 8:

| | | | |
|---|---|---|---|
| 0.645 | 0.744 | 0.848 | +1 |
| 0.566 | | 0.572 | -1 |
| 0.491 | 0.431 | 0.475 | 0.277 |

זאת דוגמה אשר מהווה דוגמה נגדית והכל בגלל התגמול $+1$ על המצב הסופי $(3,4)$.

ג. (3 נק') אם $r_1 = r_2 = \cdots = r_9 < 0$, אז בהכרח $v_1 = \min\{v_i | i \in [9]\}$. נכון \ לא נכון.

נימוק \ דוגמה נגדית: ניקח המקרה בו $\gamma = 0.9$ , $r_7 = -100, \forall i \neq 7 \in [9]: r_i = -2$ במקרה כזה
למשל $v_7 = r_7 + \gamma \max_{a \in A(s)} [\sum_{s'} P(s'|(s,a)) \cdot U(s')] = -100 + \gamma \cdot x$ כאשר $x < 0$ כיוון שכל
התגמולים שלנו הן שליליים וקטנים/שווים ל $-2$, לכן $v_7 < -100$. לעומת זאת, מתקיים שהערך
המתקבל עבור $v_1 = -14.4$ לפי אלגוריתם Value-Iteration כלומר הערך המינימלי הוא לא של $v_1$
אלא של $v_7$ לכן הטענה שגויה.

ד. (3 נק') אם $v_1 > v_2 > v_3 > 0$, אז בהכרח $\pi^*((1,1)) = Up$. נכון \ לא נכון.

נימוק \ דוגמה נגדית: ניקח למשל המקרה בו $v_1 = 10 > v_2 = 2 > v_3 = 1 > 0$
מתקיים שה utility המתקבלת עבור ביצוע $'UP'$ במשבצת $(1,1)$:

$$v_{UP} = r_1 + \gamma(0.8 * v_2 + 0.1 * v_1 + 0.1 * v_3)$$

$$v_{RIGHT} = r_1 + \gamma(0.8 * v_3 + 0.1 * v_2 + 0.1 * v_1)$$

$$v_{UP} - v_{RIGHT} = \gamma(0.8(v_2 - v_3) + 0.1(v_3 - v_1))$$

נקבל שלפי הנתונים $(v_2 - v_3) > 0, (v_3 - v_1) < 0$ וכאשר מתקיים התנאיהתנאי הבא:

$$v_{UP} < v_{RIGHT} \leftrightarrow 0.8(v_2 - v_3) + 0.1(v_3 - v_1) < 0 \leftrightarrow 0.8(v_2 - v_3) < 0.1(v_1 - v_3)$$

$$\leftrightarrow 8 < \frac{v_1 - v_3}{v_2 - v_3}$$

הפוליסה תבחר ללכת ימינה $(RIGHT)$ ולא למעלה לפי החישוב הנ"ל.

במקרה שלנו מתקיים: $\frac{v_1 - v_3}{v_2 - v_3} = \frac{10-1}{2-1} = \frac{9}{1} = 9 > 8$ לכן הבאנו דוגמה למקרה שעונה על התנאים אך הפוליסה
בוחרת ללכת ימינה ולא למעלה לכן הטענה שגויה.

ה. (2 נק') אם $\gamma = 0$, מה מספר המדיניות האופטימליות הקיימות? נמקו.

אם $\gamma = 0$ נקבל: $v_i = r_i + \gamma \max_{a\in A(s)} \left[\sum_{s'} P\left(s' \middle| (s,a)\right) \cdot U(s')\right] = r_i$, נשים לב שלכל משבצת יש לנו 4 פעולות אפשרויות שנוכל לבצע, ללכת לכיוון קיר הינה פעולה שמשאירה אותנו במקום, וכיוון שלכל פעולה מתקבל ערך $v_i = r_i$ נקבל שכל מצב שאינו מצב סופי יש לו 4 פעולות אפשריות שכולן טובות באותה מידה. ולכן יש סך הכל: $4^{|S\setminus S_G|} = 4^9$ מדיניות אופטימליות.

ו. (2 נק') לסעיף זה בלבד נתון כי $v_5 = -1, r_8 = 0$. מהו $\pi^*\left((1,4)\right)$? ציינו את כל האפשרויות ונמקו.

לפי הנתונים הנ"ל נקבל:

$$v_8 = r_8 + \gamma \max_{a\in A(s)} \left[\sum_{s'} P\left(s' \middle| (s,a)\right) \cdot U(s')\right] = \gamma \max_{a\in A(s)} \left[\sum_{s'} P\left(s' \middle| (s,a)\right) \cdot U(s')\right]$$

עבור $a = 'LEFT'$: $\sum_{s'} P\left(s' \middle| (s,a)\right) \cdot U(s') = 0.8 * v_5 + 0.1 * v_8 + 0.1 * (-1) = 0.1 * v_8 - 0.9$

עבור $a = 'RIGHT'$: $\sum_{s'} P\left(s' \middle| (s,a)\right) \cdot U(s') = 0.8 * v_8 + 0.1 * v_8 + 0.1 * (-1) = 0.9 * v_8 - 0.1$

עבור $a = 'UP'$: $\sum_{s'} P\left(s' \middle| (s,a)\right) \cdot U(s') = 0.8 * (-1) + 0.1 * v_5 + 0.1 * v_8 = 0.1 * v_8 - 0.9$

עבור $a = 'DOWN'$: $\sum_{s'} P\left(s' \middle| (s,a)\right) \cdot U(s') = 0.8 * v_8 + 0.1 * v_5 + 0.1 * v_8 = 0.9 * v_8 - 0.1$

קיבלנו שני ערכים אפשריים, נבחן את האפשרויות השונות:

<u>ערך ראשון:</u> $v_8 = \gamma(0.1 * v_8 - 0.9) \leftrightarrow v_8 * (1 - 0.1 * \gamma) = -0.9 * \gamma \leftrightarrow v_8 = -\frac{9\gamma}{10-\gamma}$

<u>ערך שני:</u> $v_8 = \gamma(0.9 * v_8 - 0.1) \leftrightarrow v_8 * (1 - 0.9 * \gamma) = -0.1 * \gamma \leftrightarrow v_8 = -\frac{\gamma}{10-9\gamma}$

1) $0.9 * v_8 - 0.1 > 0.1 * v_8 - 0.9 \leftrightarrow 0.8 * v_8 > -0.8 \leftrightarrow v_8 > -1$

במקרה הזה $v_8$ שווה לערך השני ולכן צריך להתקיים:

$$v_8 = -\frac{\gamma}{10 - 9\gamma} > -1 \leftrightarrow -\gamma > 9\gamma - 10 \leftrightarrow 10 > 10\gamma \leftrightarrow 1 > \gamma$$

ואכן זה מקרה חוקי לכן הפעולות האפשריות הן $'RIGHT'\, or\, 'DOWN'$ כי לשניהם יש אותו ערך מקסימלי מבין הערכים האפשריים ונבצע אחד מהם.

2) $0.9 * v_8 - 0.1 < 0.1 * v_8 - 0.9 \leftrightarrow 0.8 * v_8 < -0.8 \leftrightarrow v_8 < -1$

במקרה הזה $v_8$ שווה לערך הראשון ולכן צריך להתקיים:

$$v_8 = -\frac{9\gamma}{10 - \gamma} < -1 \leftrightarrow -9\gamma < \gamma - 10 \leftrightarrow 10 < 10\gamma \leftrightarrow 1 < \gamma$$

ובעצם נקבל סתירה לנתון בשאלה שאומר $0 < \gamma < 1$. לכן זה מקרה שאינו אפשרי.

3) $0.9 * v_8 - 0.1 = 0.1 * v_8 - 0.9 \leftrightarrow 0.8 * v_8 = -0.8 \leftrightarrow v_8 = -1$

במקרה הזה מתקיים שוויון בין הערך הראשון והערך השני לכן צריך להתקיים:

$$-\frac{9\gamma}{10-\gamma} = -\frac{\gamma}{10-9\gamma} \leftrightarrow \frac{9}{10-\gamma} = \frac{1}{10-9\gamma} \leftrightarrow 90 - 81\gamma = 10 - \gamma \leftrightarrow 80 = 80\gamma \leftrightarrow 1 = \gamma$$

ובעצם נקבל סתירה לנתון בשאלה שאומר $0 < \gamma < 1$. לכן זה מקרה שאינו אפשרי.

ז. (2 נק') נתון כי $v_1 > v_2 > v_3 > 0$, מצאו חסמים צמודים, עליון ותחתון ל־$r_1$ כפונקציה של $v_i$ (ולא כפונקציה של $\gamma$).

במקרה הזה מתקבל:

$$v_{UP} = r_1 + \gamma(0.8 * v_2 + 0.1 * v_1 + 0.1 * v_3)$$

$$v_{RIGHT} = r_1 + \gamma(0.8 * v_3 + 0.1 * v_2 + 0.1 * v_1)$$

$$v_{DOWN} = r_1 + \gamma(0.8 * v_1 + 0.1 * v_1 + 0.1 * v_3) = r_1 + \gamma(0.9 * v_1 + 0.1 * v_3)$$

$$v_{LEFT} = r_1 + \gamma(0.8 * v_1 + 0.1 * v_2 + 0.1 * v_1) = r_1 + \gamma(0.9 * v_1 + 0.1 * v_2)$$

אנחנו יודעים שמתקיים: $v_{DOWN} < v_{LEFT}$ כי $v_2 > v_3$.

בנוסף לכך, $v_{LEFT} > v_{UP}$, $v_{LEFT} > v_{RIGHT}$ בגלל ש $v_1 > v_2 > v_3$:

$$v_{LEFT} = r_1 + \gamma(0.8 * v_1 + 0.1 * v_1 + 0.1 * v_2) > v_{RIGHT} = r_1 + \gamma(0.8 * v_3 + 0.1 * v_2 + 0.1 * v_1)$$

$$v_{LEFT} = r_1 + \gamma(0.8 * v_1 + 0.1 * v_1 + 0.1 * v_2) > v_{Up} = r_1 + \gamma(0.8 * v_2 + 0.1 * v_1 + 0.1 * v_3)$$

מזה נקבל שהמקסימום מביניהם הינו $v_{LEFT}$ ולכן:

$$v_1 = \mathrm{v_{LEFT}} = r_1 + \gamma(0.9 * v_1 + 0.1 * v_2) \overset{\gamma\in(0,1)}{\lessgtr} r_1 + 0.9 * v_1 + 0.1 * v_2$$

ומזה נקבל: $r_1 > 0.1 * (v_1 - v_2)$.

בנוסף לכך,

$$v_1 = r_1 + \gamma \cdot \max_{a\in A(s)} \sum_{s'} P(s'|(s,a)) \cdot U(s')$$

$$r_1 = v_1 - \gamma \cdot \max_{a\in A(s)} \sum_{s'} P(s'|(s,a)) \cdot U(s') < v_1$$

האי-שוויון הנ"ל התקבל כתוצאה מזה שהעלמנו את הצד הימני במשוואה אשר נמצא בסימן מינוס, ניתן להגיע לשם על ידי השאפה של $\gamma$ לאפס.

ולכן נקבל החסמים הבאים על $r_1$:

$$0.1 * (v_1 - v_2) < r_1 < v_1$$

## חלק ב׳ - היכרות עם הקוד

חלק זה הוא רק עבור היכרות הקוד, עבורו עליו במלואו ווודאו כי הינכם מבינים את הקוד.

mdp.py – אתם לא צריכים לערוך כלל את הקובץ הזה.

בקובץ זה ממומשת הסביבה של ה-mdp בתוך מחלקת MDP. הבנאי מקבל:

- board - המגדיר את המצבים האפשריים במרחב ואת התגמול לכל מצב, תגמול על הצמתים בלבד.
- terminal_states – קבוצה של המצבים הסופיים (בהכרח יש לפחות מצב אחד סופי).
- transition_function – מודל המעבר בהינתן פעולה, מה ההסתברות לכל אחת מארבע הפעולות האחרות. ההסתברויות מסודרות לפי סדר הפעולות.
- gamma – discount factor המקבל ערכים - $\gamma \in (0,1)$.
  בתרגיל זה לא נבדוק את המקרה בו $\gamma = 1$.

הערה: קבוצת הפעולות מוגדרת בבנאי והיא קבועה לכל לוח שיבחר.

למחלקת MDP יש מספר פונקציות שעשויות לשמש אתכם בתרגיל.

- print_rewards() – מדפיסה את הלוח עם ערך התגמול בכל מצב.
- print_utility(U) – מדפיסה את הלוח עם ערך התועלת U לכל מצב.
- print_policy(policy) – מדפיסה את הלוח עם הפעולה שהמדיניות policy נתנה לכל מצב שהוא לא מצב סופי.
- step(state, action) – בהינתן מצב נוכחי state ופעולה action מחזיר את המצב הבא באופן דטרמיניסטי. עבור הליכה לכיוון קיר או יציאה מהלוח הפונקציה תחזיר את המצב הנוכחי state.

## חלק ג' – רטוב

כל הקוד צריך להיכתב בקובץ mdp_implementation.py

מותר להשתמש בספריות:

All the built-in packages in python, numpy, matplotlib, argparse, os, copy, typing, termcolor, random

עליכם לממש את הפונקציות הבאות:

- (רטוב 10 נק'): value_iteration(mdp, U_init, epsilon) – בהינתן ה-mdp, ערך התועלת ההתחלתי U_init, וחסם העליון לשגיאה מהתוחלת של התועלת האופטמילי epsilon מריץ את האלגוריתם value iteration ומחזיר את U המתקבל בסוף ריצת האלגוריתם. **TODO**

- (רטוב 5 נק'): get_policy(mdp, U) – בהינתן ה-mdp וערך התועלת U (המקיים את משוואת בלמן) מחזיר את המדיניות (במידה וקיימת יותר מאחת, מחזיר אחת מהן). **TODO**

- (רטוב 5 נק'): policy_evaluation(mdp, policy) – בהינתן ה-mdp, ומדיניות policy מחזיר את ערכי התועלת לכל מצב. **TODO**

- (רטוב 10 נק'): policy_iteration(mdp, policy_init) - בהינתן ה-mdp, ומדיניות התחלתית policy_init, מריץ את האלגוריתם policy iteration ומחזיר מדיניות אופטימלית. **TODO**

עבור מצבים סופיים וקירות (WALL), הערך שצריך לחזור בתאים אלו עבור טבלאות המדיניות הוא None. כל ערך אחר לא יתקבל בתשובה.

עבור קירות הערך שצריך עבור טבלאות התועלת הוא None. כל ערך אחר לא יתקבל בתשובה.

main.py – דוגמת הרצה לשימוש בכל הפונקציות.

בתחילת הקובץ אנו טוענים את הסביבה משלושה קבצים:
board, terminal_states, transition_function
ויוצרים מופע של הסביבה (mdp).

- שימו לב, שברגע הקוד ב-main לא יוכל לרוץ מכיוון שאתם צריכים להשלים את הפונקציות הרלוונטיות ב-mdp_implementation.py .
- בנוסף, על מנת לראות את הלוח עם הצבעים עליכם להריץ את הקוד בIDE לדוגמה PyCharm.

# חלק ב' - מבוא ללמידה (40 נק')

## ✍ חלק א' – חלק היבש (20 נק')

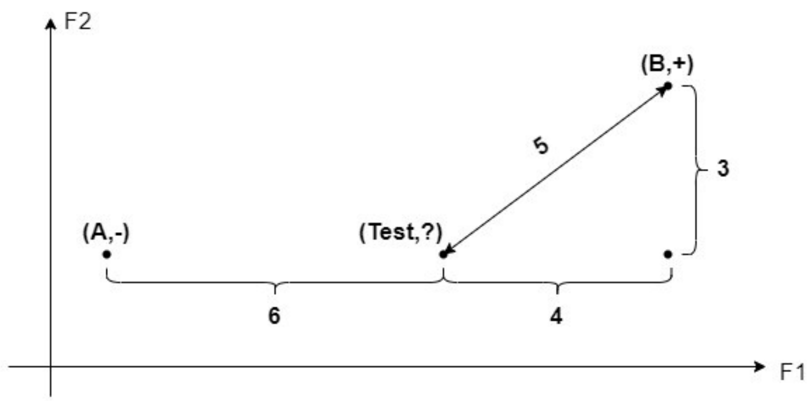
### kNN – נעים להכיר

בחלק זה תכירו אלגוריתם למידה בשם kNN, או בשמו המלא k-Nearest Neighbors, כאשר ה־k הוא למעשה פרמטר!
יהי סט אימון עם $n$־דוגמות, $D=\{(x_1,y_1),\ldots,(x_n,y_n)\}$, כאשר $\forall i: x_i\in\mathbb{R}^d, y_i\in\mathcal{Y}$.
כלומר הדוגמות הינן וקטורים $d$־ממדיים והתגיות הינן מדומיין כלשהו, הבעיה היא בעיית קלסיפיקציה (סיווג).
אם לא נאמר אחרת, הקלסיפיקציה תהיה בינארית, כלומר $\mathcal{Y}=\{-,+\}$.
עבור כל דוגמה בסט האימון, ניתן להסתכל על הכניסה ה־$i$ בווקטור כעל הfeature ה־$i$ של הדוגמה, קרי כל דוגמה $x_i$ מיוצגת על ידי $d$־ערכים: $f_1(x_i), f_2(x_i), \ldots, f_d(x_i)$.
תהליך ה"אימון" של האלגוריתם הוא טריוויאלי – פשוט שומרים את סט האימון במלואו.
תהליך הסיווג הוא גם פשוט למדי – כאשר רוצים לסווג דוגמה מסט המבחן מסתכלים על $k$ השכנים הקרובים ביותר שלה במישור ה־$d$־ממדי מבין הדוגמות בסט האימון, ומסווגים את הדוגמה על פי הסיווג הנפוץ ביותר בקרב $k$ השכנים.
על מנת להימנע משוויון בין הסיווגים, נניח בדרך כלל כי $k$־אי זוגי, או שנגדיר היטב שובר שוויון.
אם לא נאמר אחרת, במקרה של שוויון בקלסיפיקציה בינארית, נסווג את הדוגמה כחיובית $+$.

שאלות הבנה

א. (3 נק') כאמור, בתהליך הסיווג אנו בוחרים עבור הדוגמה את הסיווג הנפוץ ביותר של $k$ השכנים הקרובים ביותר, אולם עלינו להגדיר את פונקציית המרחק עבור קביעת סט שכנים זה.
שתי פונקציות מרחק נפוצות הינן מרחק אוקלידי ומרחק מנהטן.
עבור בעיית קלסיפיקציה בינארית תנו דוגמה פשוטה לערכי $d, k$, סט אימון ודוגמת מבחן בה השימוש בכל אחת מפונקציות המרחק הנ"ל משנה את סיווג דוגמה המבחן.

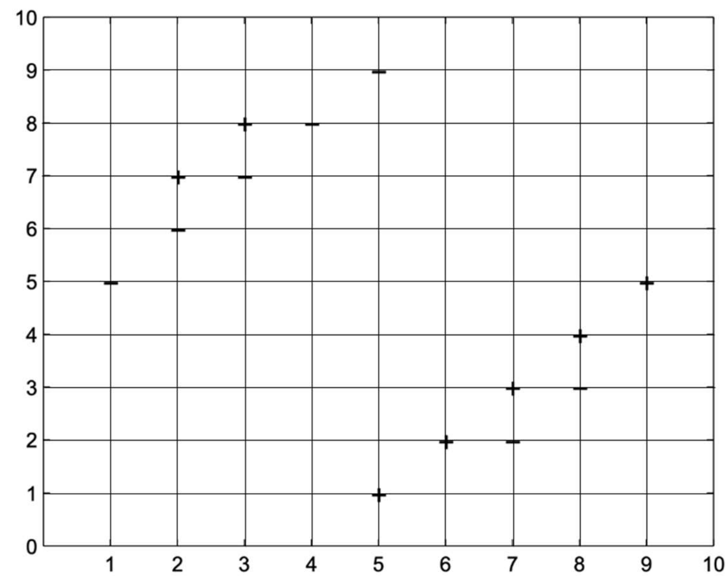
- עבור k = 1, d = 2 , נסתכל על סט אימון שמכיל שתי נקודות: B,A ודוגמת מבחן (Test) כמתואר בציור הבא:-



- במקרה בו משתמשים במרחק מנהטן בתור מטריקת המרחק שלנו: אז דוגמת המבחן תהייה יותר קרובה לנקודה A ולכן תסווג בתור (+) , אמנם אם נשתמש במרחק האוקלידי אז דוגמת המבחן תהייה יותר קרובה מנקודה B כי אז המרחק ממנה הוא 5 והמרחק מ-A לא השתנה , עדיין 6 -> ולכן תסווג בתור (-).

מעתה, אלא אם כן צוין אחרת, נשתמש במרחק אוקלידי.

נתונה קבוצת האימון הבאה, כאשר $d = 2$:




**ב.** (1 נק') איזה ערך של $k$ עלינו לבחור על מנת לקבל את הדיוק המרבי <u>על קבוצת האימון</u>? מה יהיה ערך זה?

- נבחר k = 1, שיחזיר לכל דוגמה מקבוצת האימון את עצמה התור השכן הקרוב ביותר ולכן נקבל דיוק 100% על קבוצת האימון.

**ג.** (1 נק') עבור איזה ערך של $k$ נקבל מסווג $majority$ של קבוצת האימון? קרי כל דוגמת מבחן תקבל את הסיווג הנפוץ של כלל קבוצת האימון?

- יש לנו 14 דוגמאות בסט האימון ולכן עבור k = 14, כל הדוגמאות בסט ייחשבו בתור 14 השכנים הקרובים ביותר לכל דוגמת מבחן, ואז מהם נחזיר את ה-majority.

**ד.** (2 נק') נמקו מדוע שימוש בערכי $k$ גדולים או קטנים מדי יכול להיות גרוע עבור קבוצת הדגימות הנ"ל.

- עבור K קטן , כמו למשל 1NN: לרוב החיזויים יהיו שגויים, כי לפי התבנית של ה-data , דוגמת מבחן ששייכת ל- (-) היא תהייה בסיכוי גדול בנקודה: (9,5) או (6,1) .. כלומר תמשיך עם אותו קו ליניארי, אמנם שם היא יותר קרובה לנקודות (+) מאשר לנקודות (-) וכנ"ל לגבי דוגמה שהתיוג שלה הוא (+). ואפשר להגיד שנקבל תופעה של overfitting.

- מצד שני, עבור K-ים גדולים, אם דוגמת המבחן שהתיוג שלה נופל באזור שלמטה היא יכולה לקבל סיווג יחובי כי דווקא שם יש לנו מספר יותר גדול של דוגמאות אימון חיוביות שגם יהיו קרובים אליה. ואותו דבר לגבי דוגמת מבחן ען תיוג חיובי שנופלת למעלה בגרף ואז עלולה לקבל סיווג שלילי כי שם ה-Majority הוא דווקא השליליים.

וריאציה נוספת של אלגוריתם הלמידה $kNN$ מקבלת במקום $k$ את הפרמטר $r$ – רדיוס.
בעת סיווג של דוגמת מבחן יתבצע על ידי הסיווג הנפוץ ביותר של דוגמות הנמצאות במרחק לכל היותר $r$ מדוגמת המבחן, כלומר "ברדיוס הסיווג".
במקרה של שוויון, גם אם ריק, הסיווג יהיה חיובי.

למען הפשטות, בסעיפים הבאים יש להזניח מקרים בהם קבוצת $k$ השכנים הקרובים ביותר אינה מוגדרת היטב, כלומר מצב בו יש יותר מ$k$ שכנים קרובים ביותר בגלל שוויון במרחק לדוגמת המבחן.

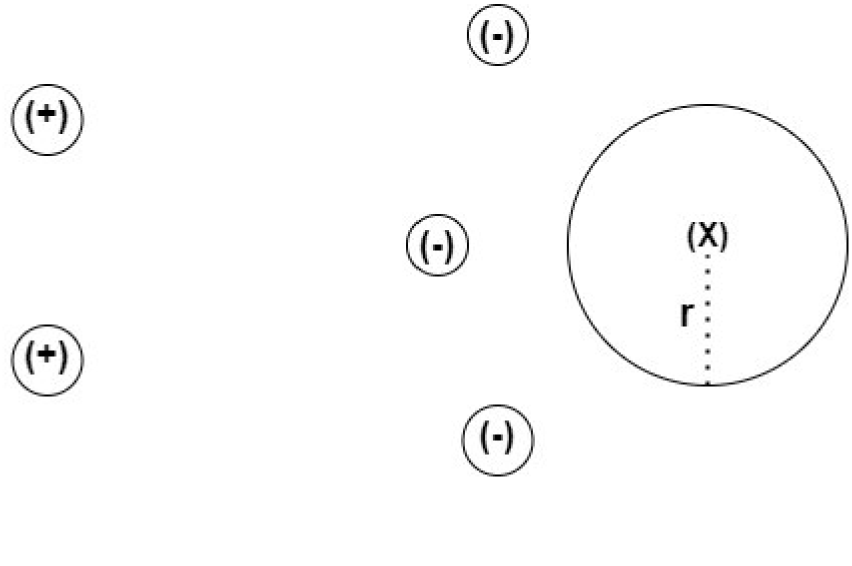
הוכיחו או הפריכו.

ה. (3 נק') קיימים ערכי $d, k$, סט אימון ודוגמת מבחן כך שלא קיים $r$, עבורו סיווג דוגמת המבחן בווריאציה החדשה יהיה זהה לסיווג בגרסה המקורית של האלגוריתם.

- זה נכון, יהיו d,K סט אימון ודוגמת מבחן כלשהם, נתון שתמיד קבוצת $k$ השכנים הקרובים ביותר מוגדרת היטב, ולכן נסמן $d^{Max}$ = המרחק המקסימלי מנקודת המבחן לשכן הרחוק ביותר ממנה מבין K השכנים הקרובים ביותר שאותם מצא האלגוריתם,
אזי עבור בחירת $r = d^{Max}$ תחת הווריאציה החדשה: תחת ההנחה שקבוצת K השכנים הקרובים ביותר מוגדרת היטב, אז קבוצת הנקודות מתוך סט האימון שיהיו בתוך רדיוס הסיווג הם בדיוק K השכנים הקרובים ביותר שמצא הרלגוריתם בווריאציה הראשונה ולכן גם עכשיו נחזיר אותו סיווג.

ו. (3 נק') קיימים ערכי $d, r$, סט אימון ודוגמת מבחן כך שלא קיים $k$, עבורו סיווג דוגמת המבחן בגרסה המקורית של האלגוריתם יהיה זהה לסיווג בווריאציה החדשה.

- לא נכון, נסתכל על הדוגמה הבא בה d=2 ויש לנו דוגמת מבחן מסומנת ב-X ו-5 דוגאמאות אימון: 3 שליליות ו-2 חיוביות:-



- תחת הווריאציה החדשה נקבל סיווג חיובי (אין נקודות בתוך רדיוס הסיווג), אמנם ניתן לראות שלכל ערך של K תמיד הנקודה X תקבל סיווג שלילי כי הנקודות הכי קרובות יהיו עם תיוג שלילי.

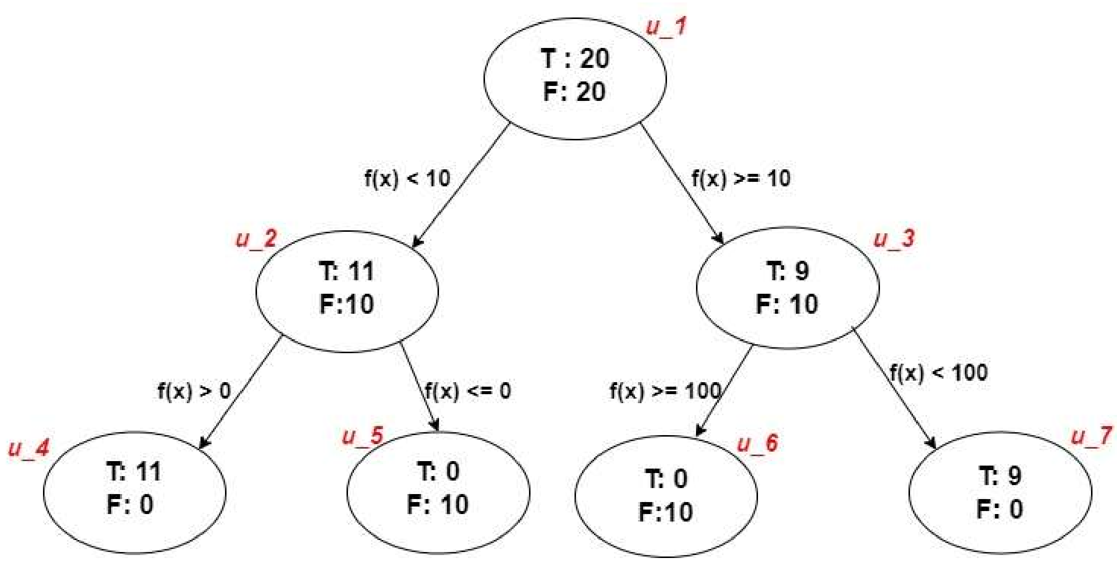
## מתפצלים ונהנים

(7 נק') כידוע, בעת סיווג של דוגמת מבחן על ידי עץ החלטה, בכל צומת בעץ אנו מחליטים לאיזה צומת בן להעביר את דוגמת המבחן על ידי ערך סף $v$ שמושווה לfeature של הדוגמה. לפעמים ערך הסף קרוב מאוד לערך הfeature של דוגמת המבחן. היינו רוצים להתחשב בערכים "קרובים" לערך הסף בעת סיווג דוגמת מבחן, ולא לחרוץ את גורלה של הדוגמה לתת־עץ אחד בלבד; לצורך כך נציג את האלגוריתם הבא:

יהיו עץ החלטה $T$, דוגמת מבחן $x \in \mathbb{R}^d$, ווקטור $\varepsilon \in \mathbb{R}^d$ המקיים $\forall i \in [1, d]: \varepsilon_i > 0$.
כלל אפסילון־החלטה שונה מכלל ההחלטה הרגיל שנלמד בכיתה באופן הבא:
נניח שמגיעים לצומת בעץ המפצל לפי ערכי התכונה $i$, עם ערך הסף $v_i$.
אם מתקיים $|x_i - v_i| \leq \varepsilon_i$ אזי ממשיכים **בשני** המסלולים היוצאים מצומת זה, ואחרת ממשיכי לבן המתאים בדומה לכלל ההחלטה הרגיל. לבסוף, מסווגים את הדוגמה $x$ בהתאם לסיווג הנפוץ ביותר של הדוגמאות הנמצאות בכל העלים אליהם הגענו במהלך הסיור על העץ (במקרה של שוויון – הסיווג ייקבע להיות $True$).

יהא $T$ עץ החלטה לא גזום, ויהא $T'$ העץ המתקבל מ־$T$ באמצעות גיזום מאוחר שבו הוסרה הרמה התחתונה של $T$ (כלומר כל הדוגמות השייכות לזוג עלים אחים הועברו לצומת האב שלהם).
הוכיחו\הפריכו: **בהכרח** קיים ווקטור $\varepsilon$ כך שהעץ $T$ עם כלל אפסילון־החלטה והעץ $T'$ עם כלל ההחלטה הרגיל יסווגו כל דוגמת מבחן ב$\mathbb{R}^d$ בצורה זהה.

- הטענה לא נכונה:- נציג דוגמה לעץ T עם 40 דוגמאות בסט האימון , 20 מהם עם סיווג True ו-20 False , ונניח שיש לנו רק feature אחד רציף , נקרא לו f, כך שבבניית עץ ההחלטה פיצלנו לפי הערכים המופעים בציור:-



- נניח בשלילה כי קיים $\varepsilon > 0$ כך שלכל דוגמת מבחן: T אחרי גיזום כל הרמה התחתונה ייתן אותו סיווג כמו T עם כלל אפסילון-החלטה,
אזי בפרט עבור דוגמת המבחן x שמקיימת $f(x) = 50$ הטענה צריכה להתקיים.

- סיווג T הגזום על X יהייה באופן הבא: נתחיל מהשורש $u_1$ ואז נרד ימינה ל-$u\_3$ והיות שאחרי הגיזום $u_3$ הופך לעלה שהסיווג שלו הוא False – שזה רוב הדוגמאות בתת העץ המקורי -> אז נקבל שסיווג X הוא: false.
- במקרה של כלל האפסילון:
  - אז מצד אחד, אם $\varepsilon < 40$ אז במהלך הסיווג של X על T אף פעם לא נהייה קרובים מסף הפיצול והסיווג ילך כך: $u_1 \to u_3 \to u_7 = True$.
  - מצד שני, אם $\varepsilon \geq 40$:
    - אז אם $40 \leq \varepsilon < 50$:- נצטרך במהלך הסיווג לרדת בשני הצמתים $u_2, u_3$ , אך ברגע שירדנו ב-$u_2$: לא נהייה קרובים מערך הסף ונרד רק בעלה $u_4$ ונפעפע עלפי מעלה שראינו 11 דוגמאות T ו-0 דוגמאות F, באותו אופן כשנרד ב-$u_3$ לא נהייה קרובים מהסף ונרד רק בעלה $u_7$ ונחזיר למעלה שראינו 9 דוגמאות T ואפס F, בסך הכל כשנחזור לשורש ונשלב את המידע משני תתי העצים נקבל 20 T מול 0 F ולכן הסיווג יהייה True
    - אחרת אם $50 \leq \varepsilon$ : אז בכל צומת שנבקר בו נהייה קרובים לסף ונצטרך לרדת ימינה ושמאלה כך שבסוף נבקר ונאסוף את מספר הדוגמאות מכל העלים שזה שווה למספר הדוגמאות הכולל ולכן נקבל 20 T מול 20 F -> לפי מה שנתון במקרה הזה הסיווג צריך להיות True
- אז בכל מקרה נקבל סתירה להנחת השלילה , ולכן לא קיים אפסילון כזה.

## חלק ב' - היכרות עם הקוד

### רקע

חלק זה הוא רק עבור היכרות הקוד, עבורו עליו במלואו ווודאו כי הינכם מבינים את הקוד.
בחלק של הלמידה, נעזר ב $dataset$, הדאטה חולק עבורכם לשתי קבוצות: קבוצת אימון train.csv וקבוצת מבחן test.csv.
בכלל, קבוצת האימון תשמש אותנו לבניית המסווגים, וקבוצת המבחן תשמש להערכת ביצועיהם.

בקובץ utils.py תוכלו למצוא את הפונקציות הבאות לשימושכם:
load_data_set, create_train_validation_split, get_dataset_split
אשר טוענות/מחלקת את הדאטה בקבצי ה־csv למערכי np.array (קראו את תיעוד הפונקציות).

הדאטה של ID3 עבור התרגיל מכיל מדדים שנאספו מצילומים שנועדו להבחין בין גידול שפיר לגידול ממאיר. כל דוגמה מכילה 30 מדדים כאלה, ותווית בינארית diagnosis הקובעת את סוג הגידול (0=שפיר, 1=ממאיר). כל התכונות (מדדים) רציפות . העמודה הראשונה מציינת האם האדם חולה (M) או בריא (B). שאר העמודות מציינות כל תכונות רפואיות שונות של אותו אדם (התכונות מורכבות ואינכם צריכים להתייחס למשמעות שלהן כלל).

<u>תיקיית $ID3-dataset$:</u>

- תיקיה זו אלו מכילה את קבצי הנתונים עבור $ID3$.

<u>קובץ $utils.py$:</u>

- קובץ זה מכיל פונקציות עזר שימושיות לאורך התרגיל, כמו טעינה של $dataset$ וחישוב הדיוק.
- בחלק הבא יהיה עליכם לממש את הפונקציה $accuracy$. קראו את תיעוד הפונקציות ואת ההערות הנמצאות תחת התיאור **TODO**.

<u>קובץ $unit\_test.py$:</u>

- קובץ בדיקה בסיסי שיכול לעזור לכם לבדוק את המימוש.

<u>קובץ $DecisionTree.py$:</u>

- קובץ זה מכיל 3 מחלקות שימושית לבניית עץ $ID3$ שלנו.
  - <u>המחלקה $Question$:</u> מחלקה זו מממשת הסתעפות של צומת בעץ. היא שומרת את התכונה ואת הערך שלפיהם מפצלים את הדאטה שלנו.
  - <u>המחלקה $DecisionNode$:</u> מחלקה זו מממשת צומת בעץ ההחלטה.
    הצומת מכיל שאלה $Question$ ואת שני הבנים $true\_branch, false\_branch$ כאשר $true\_branch$ הוא הענף בחלק של הדאטה שעונה $True$ על שאלת הצומת (הפונקציה $match$ של ה$Question$ מחזירה $True$).
    ו־ $false\_branch$ הוא הענף בחלק של הדאטה שעונה $False$ על שאלת הצומת (הפונקציה $match$ של ה$Question$ מחזירה $False$).
  - <u>המחלקה $Leaf$:</u> מחלקה זו מממשת צומת שהוא עלה בעץ ההחלטה. העלה מכיל לכל אחד מהמחלקות בדאטה את מספר הדוגמאות בעלה עבור כל מחלקה (למשל: $\{‘B’ : 5, ‘M’ : 6\}$).

<u>קובץ $ID3.py$:</u>

- קובץ זה מכיל את המחלקה של $ID3$ שתצטרכו לממש חלקים ממנה, עיינו בהערות ותיעוד המתודות.

<u>קובץ $ID3\_experiments.py$:</u>

- קובץ הרצת הניסויים של ID3, הקובץ מכיל את הניסויים הבאים, שיוסברו בהמשך:
  $cross\_validation\_experiment, basic\_experiment$

## חלק ג' – חלק רטוב ID3 (20 נק')

עבור חלק זה מותר לכם להשתמש בספריות הבאות:

All the built in packages in python, sklearn, pandas ,numpy, random, matplotlib, argparse, abc, typing.

<u>אך כמובן שאין להשתמש באלגוריתמי הלמידה, או בכל אלגוריתם או מבנה נתונים אחר המהווה חלק מאלגוריתם למידה אותו תתבקשו לממש.</u>

**1.** (3 נק') השלימו את הקובץ utils.py ע"י מימוש הפונקציה $accuracy$.
קראו את תיעוד הפונקציה ואת ההערות הנמצאות תחת התיאור **TODO**.
(הריצו את הטסטים המתאימים בקובץ $unit\_test.py$ לוודא שהמימוש שלכם נכון).
שימו לב! בתיעוד ישנן הגבלות על הקוד עצמו, אי־עמידה בהגבלות אלו תגרור הורדת נקודות.
בנוסף, שנו את ערך הID בתחילת הקובץ מ־123456789 למספר תעודת הזהות של אחד מהמגישים.

**2.** (10 נק') אלגוריתם ID3:

**a.** השלימו את הקובץ ID3.py ובכך ממשו את אלגוריתם $ID3$ כפי שנלמד בהרצאה. **TODO**
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 <u>האינדקס המקסימלי</u>.
כלל המימוש הנ"ל צריך להופיע בקובץ בשם $ID3.py$, באזורים המוקצים לכך.
(השלימו את הקוד החסר אחרי שעיינתם והפנמתם את הקובץ $DecisionTree.py$ ואת המחלקות שהוא מכיל).

**b.** ממשו את $basic\_experiment$ שנמצאת ב $ID3\_experiments.py$ **TODO**
והריצו את החלק המתאים ב $main$ ציינו בדו"ח את הדיוק שקיבלתם. ✍

- בחלק זה קיבלנו דיוק של 94.69% .

**3.** גיזום מוקדם.
פיצול צומת מתקיים כל עוד יש בו יותר דוגמאות מחסם המינימום $m$, כלומר בתהליך בניית העץ מבוצע "גיזום מוקדם" כפי שלמדתם בהרצאות. שימו לב כי פירוש הדבר הינו שהעצים הנלמדים אינם בהכרח עקביים עם הדוגמאות. לאחר סיום הלמידה (של עץ יחיד), הסיווג של אובייקט חדש באמצעות העץ שנלמד מתבצע לפי רוב הדוגמאות בעלה המתאים.

✍ **a.** (2 נק') הסבירו מה החשיבות של הגיזום באופן כללי ואיזה תופעה הוא מנסה למנוע?

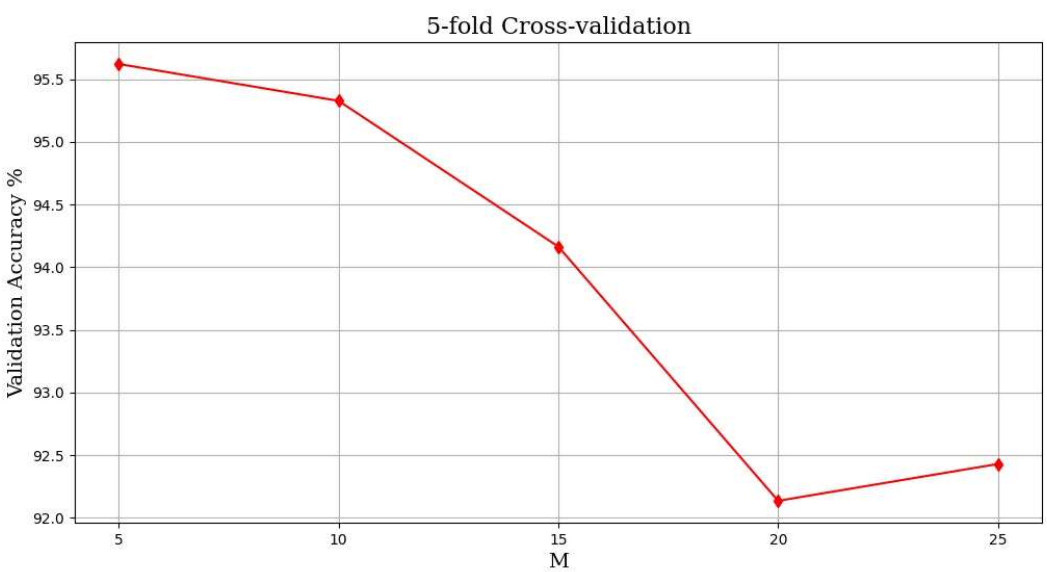
- הגיזום חשוב כדי שנמנע את תופעת ה- overfitting שבה המודל מנסה להיות נכון ומדוייק כמה שיותר על קבוצת האימון עצמה (דבר שמתבטא בעץ החלטות גדול מאוד ומלא בפיצולים לפי קבוצת האימון) עד כדי כך שזה מתחיל להשפיע על יכולת ההכללה של המודל על קבוצות מבחן אחרות.

**b.** (3 נק') <u>עדכנו</u> את המימוש בקובץ $ID3.py$ כך שיבצע גיזום מוקדם כפי שהוגדר בהרצאה.
הפרמטר $min\_for\_pruning$ מציין את המספר המינימלי בעלה לקבלת החלטה, קרי יבוצע גיזום מוקדם אם ורק אם מספר הדוגמות בצומת קטן שווה לפרמטר הנ"ל. **TODO**

**c.** <u>סעיף זה בונוס (5 נקודה לציון התרגיל):</u>
שימו לב, זהו סעיף יבש ואין צורך להגיש את הקוד שכתבתם עבורו.
בצעו כיוונון לפרמטר $\mathrm{M}$ על קבוצת האימון:
1. בחרו לפחות חמישה ערכים שונים לפרמטר $\mathrm{M}$.
2. עבור כל ערך, חשבו את הדיוק של האלגוריתם על ידי $\mathrm{K-fold\ cross\ validation}$ על קבוצת האימון בלבד.
כדי לבצע את חלוקת קבוצת האימון ל- $\mathrm{K}$ קבוצות יש להשתמש בפונקציה
[sklearn.model_selection.KFold](sklearn.model_selection.KFold) עם הפרמטרים $\mathrm{n\_split} = 5$, $\mathrm{shuffle} = \mathrm{True}$
ו־$\mathrm{random\_state}$ אשר שווה למספר תעודת הזהות של אחד מהשותפים.

✍ i. השתמשו בתוצאות שקיבלתם כדי ליצור גרף המציג את השפעת הפרמטר $\mathrm{M}$ על הדיוק.
צרפו את הגרף בדו"ח. (לשימושכם הפונקציה $util\_plot\_graph$ בתוך הקובץ $utils.py$).

- בדקנו את ערכי M הבאים :- {5, 10, 15, 20, 25}, וקיבלנו את התוצאות הבאות:




✍ ii. הסבירו את הגרף שקיבלתם. לאיזה גיזום קיבלתם התוצאה הטובה ביותר ומהי תוצאה זו?

- הגיזום הטוב ביותר אצלנו היה עבור M = 5, נשים לב כי הערך 25 הניב תוצאות טובות יותר מ-20 - > וזה בדיוק מראה איך הגיזום יכול להיות טוב לפעמים בלפתור את בעיית ה-overfitting .

תם סעיף הבונוס, הסעיף הבא הינו סעיף חובה.

✍ **d.** (2 נק') 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 c. (ממשו $best\_m\_test$ שנמצאת ב $ID3\_experiments.py$ והריצו את החלק המתאים ב $main$). ציינו בדו"ח את הדיוק שקיבלתם. האם הגיזום שיפר את הביצועים ביחס להרצה ללא גיזום?
הערה: בסעיף זה אם לא מימשתם את סעיף c השתמשו בערך $M = 50$.

- כשהצבנו M = 5, דווקא קיבלנו אחוז דיוק של 94.75% שזה היה רק טיפה יותר טוב מאחוז הדיוק המקורי, -> זה יכול להיות בגלל שעכשיו אנחנו בודקים על test_set אחר לגמרי מזה שביצענו עליו את ה-K Fold Cross Validation .

## הוראות הגשה

- ✓ הגשת התרגיל תתבצע אלקטרונית בזוגות בלבד.
- ✓ הקוד שלכם ייבדק (גם) באופן אוטומטי ולכן יש להקפיד על הפורמט המבוקש. הגשה שלא עומדת בפורמט לא תיבדק (ציון 0).
- ✓ המצאת נתונים לצורך בניית הגרפים אסורה ומהווה עבירת משמעת.
- ✓ הקפידו על קוד קריא ומתועד. התשובות בדוח צריכות להופיע לפי הסדר.
- ✓ יש להגיש קובץ zip יחיד בשם AI3_<id1>_<id2>.zip (ללא סוגריים משולשים) שמכיל:

- קובץ בשם AI_HW3.PDF המכיל את תשובותיכם לשאלות היבשות.
- קבצי הקוד שנדרשתם לממש בתרגיל <u>**ואף קובץ אחר:**</u>
  - קובץ – utils.py
  - בחלק של עצי החלטה – ID3_experiments.py, ID3.py
  - בחלק של mdp - mdp_implementation.py

<u>אין להכיל תיקיות בקובץ ההגשה, הגשה שלא עומדת בפורמט לא תיבדק.</u>