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 $R: S \rightarrow \mathbb{R}$, 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: $R: S \times A \rightarrow \mathbb{R}$, 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.

הנוסחא המותאמת עבור "תגמול על פעולה" היא:

$$U^{\pi}(s) = E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^{t} R\left(S_{t},, \pi(s)\right) | S_0 = s\right]$$

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

הנוסחא המותאמת עבור "תגמול על פעולה" היא:

$$U(s) = \max_{a \in \mathbb{A}(s)} \sum_{s'} P(s'|a, s)[R(s, a) + \gamma U(s')]$$

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

האלגוריתם המתאים:

```
function VALUE-ITERATION(mdp, ε) returns a utility function
  inputs: mdp, an MDP with states S, actions A(s), transition model P(s' | s, a),
          rewards R(s), discount γ
          ε, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
                   δ, the maximum change in the utility of any state in an iteration

  repeat
     U ← U'; δ ← 0
     for each state s in S do
        U'[s] ← max_{a∈A(s)} Σ_{s'} P(s'|a,s)[R(s,a) + γU(s')]
        if |U'[s] − U[s]| > δ then δ ← |U'[s] − U[s]|
  until δ < ε(1 − γ)/γ
  return U
```

במצב שבו γ הוא 1, נקבל כי תנאי העצירה החדש הוא כאשר δ הוא 0, כלומר נעצור כאשר התכנסנו לערך האופטימלי.

נשים לב כי האלגוריתם ייתכנס בוודאות עבור מרחב מצבים סופי, עבור מצב שבו התועלת לא תשאר קבועה אחרי איטרציות שונות (כי אז נתקע בלולאה אינסופית) וגם עבור כל מצב שבו יש חסם על התגמול.

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

לצורך ניסוח האלגוריתם נגדיר את:

$$U^{\pi}(s) = \sum_{s'} P(s'|\pi(s),s)[R(s,\pi(s)) + \gamma U^{\pi}(s')]$$

כעת ננסח את האלגוריתם המותאם:

```
function POLICY-ITERATION(mdp) returns a policy
  inputs: mdp, an MDP with states S, actions A(s), transition model P(s' | s, a)
  local variables: U, a vector of utilities for states in S, initially zero
                   π, a policy vector indexed by state, initially random

  repeat
     U ← POLICY-EVALUATION(π, U, mdp)
     unchanged? ← true
     for each state s in S do
        if max_{a∈A(s)} Σ_{s'} P(s'|a,s)[R(s,a) + γU(s')] > Σ_{s'} P(s'|π(s),s)[R(s,π(s)) + γU^π(s')] then do
           π[s] ← argmax_{a ∈ A(s)} Σ_{s'} P(s' | s, a) U[s']
           unchanged? ← false
  until unchanged?
  return π
```

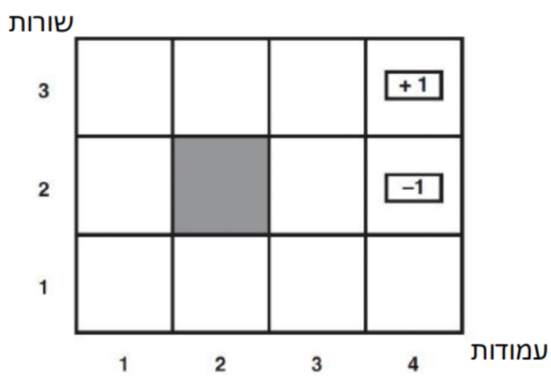
כפי שנלמד בתרגול ובהרצאה עבור Iteration Policy, כאשר מרחב המצבים והפעולות סופי, ופונקציית התגמולים חסומה, קיים מצב סופי ללא מעגל תגמולים חיוביים.

בשונה מ- Value Iteration, האלגוריתם יעצור כאשר לא יהיה שינוי במדיניות הפעולות בהשוואה לאיטרציה הקודמת, וההתכנסות לפתרון תיהיה מהירה יותר.

לכן גם כאשר $\gamma = 1$, אין צורך להוסיף תנאי מיוחד כדי שהאלגוריתם יתכנס לפתרון בזמן סופי.

הערה: בסעיפים ג' ו־ד' 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: $R\big((3,4)\big) = +1$, $R\big((2,4)\big) = -1$.

שימו לב, התגמולים הינם תגמולים על המצבים.

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

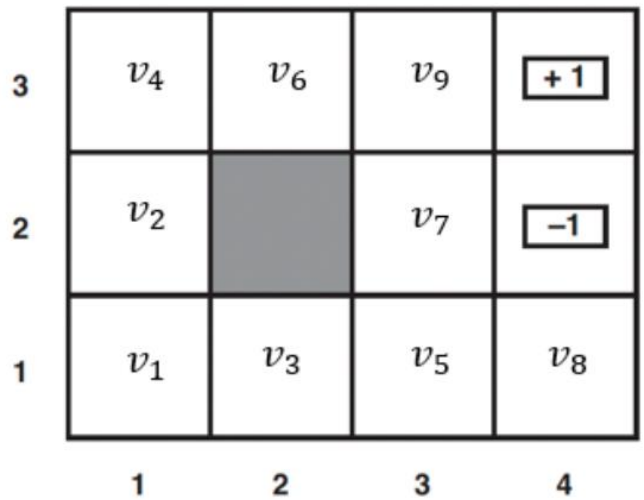
<u>מודל מעבר:</u>

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 קֻיַּם 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: במקרה שבו התגמול מאוד גבוה עבור מצב כלשהו ועבור מצב אחר הוא קטן מ-1, נקבל כי עבור מקדם דעיכה מתאים התועלת של שני המצבים תהיה מעל 1 שכן עבור מצב אחד כבר יש תגמול גבוה מאוד ועבור המצב השני מתקיים כי ההסצברות ללכת בכיוון השני היא קטנה מידי.

לדוגמא עבור מצבים 6 ו-9 ומקדם דעיכה 0.8, אם נגדיר תגמול גבוה מאוד למצב 6 ומספיק תגמול קטן מ-1 למצב 9 ואז תתקיים הסיטואציה.

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: הטענה שגויה, כפי שראינו בתרגול – אם התגמול הוא תמיד 0 אזי התועלת עדיין יוצא חיובית לכל המצבים.

ניתן לראות ממשוואת בלמן:

$$v_i = R_i + \gamma \cdot \max \sum_{s'} P(s'|a,s)U(s')$$

קיימת אפשרות ש: $v_i > 0$, כאשר: $R_i < \gamma \cdot \max \sum_{s'} P(s'|a,s)U(s')$

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:

ניתן לראות דוגמה נגדית מהחלק הרטוב אשר $v_1 > v_8$:

```
Final utility:
| 0.509 | 0.65  | 0.795 | 1.0  |
| 0.399 | WALL  | 0.486 | -1.0 |
| 0.296 | 0.254 | 0.345 | 0.13 |
```

כאשר כל התגמולים עבור כל המצבים בלוח שליליים:

```
@@@@@@ The board and rewards @@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
| -0.04 | -0.04 | -0.04 | +1    |
| -0.04 | WALL  | -0.04 | -1    |
| -0.04 | -0.04 | -0.04 | -0.04 |

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

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

נימוק \ דוגמה נגדית: כאשר נפעיל את משוואת בלמן על אותו המצב, הפעולה שתמקסם את ערך התועלת היא Left:

$$
\begin{aligned}
U(1,1) = \mathrm{R(1,1)} + \gamma \max[\ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && (Up)\\
& \boxed{0.9U(1,1) + 0.1U(1,2),} && \boxed{(Left)}\\
& 0.9U(1,1) + 0.1U(2,1), && (Down)\\
& 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ ]. && (Right)
\end{aligned}
$$

** כאשר: $U(1,2) > U(2,1)$

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

כאשר $\gamma = 0$, התועלת עבור כל אחד מהמצבים תהיה תלויה אך ורק בתגמול של אותו המצב, ללא תלות בפעולה שתבחר. לכן בשל העובדה שניתן לבחור באחת מ 4 פעולות לכל מצב ויש סך הכל 9 מצבים, מספר האופציות האופטימליות הוא 4^9.

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

במקרה המתואר, מתוך כל המצבים האפשריים אליהם הסוכן יוכל להגיע לפי הפעולה שיבחר היא המצב הנוכחי: s_8, לכן המדינות המקסימליות במקרה זה: $\pi^*\big((1,4)\big) = Down\backslash Right$.

עבור פעולות $Down\backslash Right$, התועלת תהיה:

$$U(1,1) = 0 + \gamma * [0.8 * 0 + 0.1 * 0 + 0.1 * (-1)] = -0.1$$

עבור פעולות Up\Left, התועלת תהיה:

$$U(1,1) = 0 + \gamma * [0.8 * (-1) + 0.1 * (-1) + 0.1 * (0)] = -0.9$$

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 γ).

תחילה נציג את משוואת בלמן עבור v_1:

$$\begin{aligned} v_1 = R(1) + \gamma \cdot \max [0.8 \cdot v_2 + 0.1 \cdot v_1 + 0.1 \cdot v_3, \quad & Up \\ 0.9 \cdot v_1 + 0.1 \cdot v_2, \quad & Left \\ 0.9 \cdot v_1 + 0.1 \cdot v_3, \quad & Down \\ 0.8 \cdot v_3 + 0.1 \cdot v_1 + 0.1 \cdot v_2, \quad & Right] \end{aligned}$$

בשל העובדה ש $v_1 > v_2 > v_3 > 0$ הפעולה Left היא זו שתמקסם את התועלת.

מפיתוח של המשוואה נקבל:

$$v_1 = R(1) + \gamma(0.9 \cdot v_1 + 0.1 \cdot v_2)$$

$$R(1) = v_1 - \gamma(0.9 \cdot v_1 + 0.1 \cdot v_2) \rightarrow R(1) = v_1(1 - 0.9\gamma) - 0.1 \cdot \gamma \cdot v_2)$$

כאשר נציב $\gamma = 0$ נקבל חסם עליון: $R(1) = v_1$

כאשר נציב $\gamma = 1$ נקבל חסם תחתון: $R(1) = 0.1(v_1 - v_2)$

ולכן $R(1)$ חסום באופן הבא:

$$0.1(v_1 - v_2) \leq R(1) \leq v_1$$

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 <u>המצבים</u> האפשריים במרחב ואת <u>התגמול</u> 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 <u>דטרמיניסטי</u>. 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 כתשובה.

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

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 יכול 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')

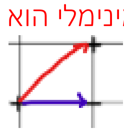
✍️ חלק א' – חלק היבש (20 נק')

kNN – נעים להכיר

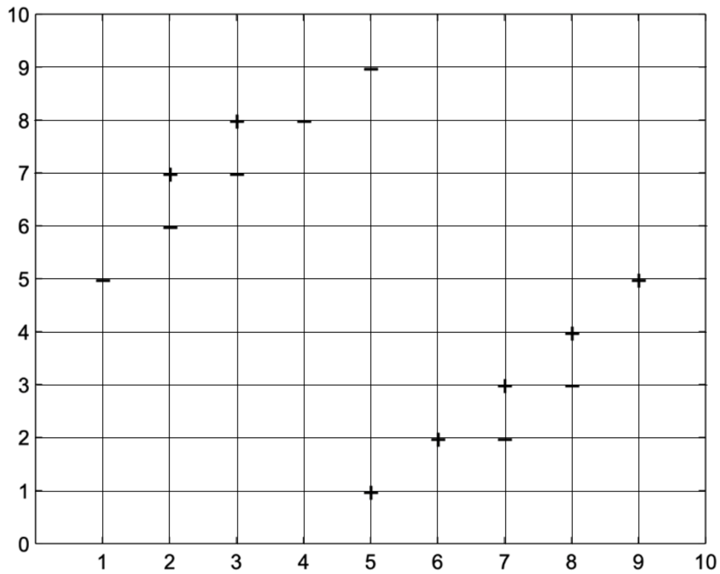
בחלק זה תכירו אלגוריתם למידה בשם kNN, או בשמו המלא k-Nearest Neighbors, כאשר ה־k הוא למעשה פרמטר!
יהי סט אימון עם n־דוגמות, $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$, כאשר $\forall i: x_i \in \mathbb{R}^d, y_i \in \mathcal{Y}$.
כלומר הדוגמות הינן וקטורים d־ממדיים והתגיות הינן מדומיין כלשהו, הבעיה היא בעיית 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: $f_1(x_i), f_2(x_i), \dots, f_d(x_i)$.
תהליך ה"אימון" של האלגוריתם הוא טריוויאלי – פשוט שומרים את סט האימון במלואו.
תהליך הסיווג הוא גם פשוט למדי – כאשר רוצים לסווג דוגמה <u>מסט המבחן</u> מסתכלים על k השכנים הקרובים ביותר שלה במישור הd־ממדי <u>מבין הדוגמות בסט האימון</u>, 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 $+$.

<u>שאלות הבנה</u>

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 <u>פשוטה</u> לערכי d, k, סט אימון ודוגמת מבחן בה השימוש בכל אחת מפונקציות המרחק הנ"ל משנה את סיווג דוגמה המבחן.
עבור הנתונים הבסיסיים (k = 1 ובעולם דו מימדי) נקבל כי המרחק האוקלידי המינימלי הוא האלכסון מהנקודה שבה אנו נמצאים ומרחק מנהטן המינימלי הוא על צלע הריבוע ולכן התוצאה תהיה שונה.
(מרחק אוקלידי באדום ומרחק מנהטן בסגול)



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? נרצה ערך קטן מספיק כדי שלא תהיה גלישה לשתי הקבוצות וכתוצאה מכך סיווג לא נכון, אך באותה מידה נרצה ערך גדול מספיק כדי שלא נקבל מצב שבו צד אחד יהיה רק מינוסים והשני רק פלוסים ולכן הערך המתאים הוא 3 (עבור 2 התרחיש השני מתקיים אם אנחנו נמצאים בפינה).

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? כפי שראינו בנתון, אם יש שיוויון אז הולכים עם הפלוס, לכן נרצה לייצר מצב שבו יש שיוויון, נשים לב כי מספר הפלוסים והמינוסים שווה ולכן אם נבחר k = 14 נקבל כי כל הנתונים שלנו ייבחרו ולכן עקב השיוויון נלך תמיד עם הפלוסים.

ד. (2 נק') נמקו מדוע שימוש בערכי k גדולים או קטנים מדי יכול להיות גרוע עבור קבוצת הדגימות הנ"ל. שימוש בערכי קיצון של k יכול לגרור מצב של בעיות בסיווג – אם קטן מידי נקבל כי בכל צד יש ערכים שונים ולכן לא נוכל להתמודד עם הרעש ולעומת זאת אם הוא גדול מידי נקבל כי תהיה גלישה לשתי הקבוצות ואז ההתחשבות לא תהיה מדוייקת לשכנים הקרובים אלא גם לרחוקים יותר.

וריאציה נוספת של אלגוריתם הלמידה kNN מקבלת במקום k את הפרמטר r – רדיוס.
כעת 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 מk שכנים קרובים ביותר בגלל שוויון במרחק לדוגמת המבחן.

הוכיחו או הפריכו.

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

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

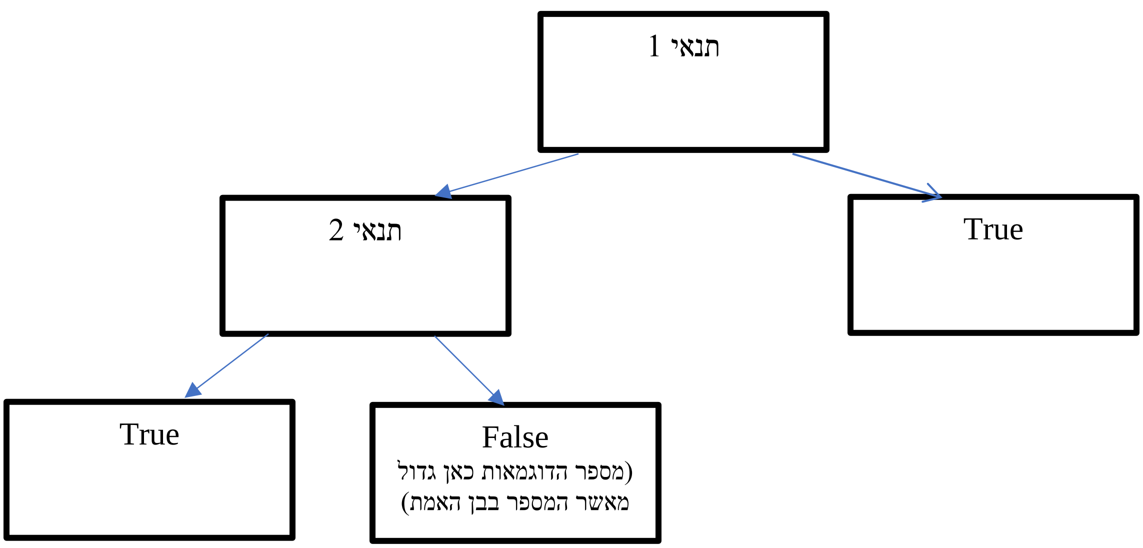
מתפצלים ונהנים

(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 ε כך שהעץ T עם כלל אפסילון־החלטה והעץ T' עם כלל ההחלטה הרגיל יסווגו כל דוגמת מבחן ב$\mathbb{R}^d$ בצורה זהה.

הטענה שגויה!
נגדיר עץ T כך שלאחר גיזום יתקבל עץ G אשר יסווג את x באופן שונה ממה שסיווג אותו T.



במקרה זה נקבל כי לאחר הגיזום הבן השמאלי של תנאי 1 יהיה False (שכן יש לו יותר דוגמאות מבחן) ולכן עבור x מתאים שבו חלק ראשון לא עומד בתנאי 1 וחלק שני עומד בתנאי 2 נקבל כי בעץ T נלך שמאלה ואז שמאלה (כי תנאי 2 מתקיים) ולכן x יסווג כ-True.
לעומת זאת בעץ הגזום נקבל כי נלך שמאלה ואז מכיוון שהבן השמאלי הוא false אזי x יסווג כ-False.

חלק ב' - היכרות עם הקוד
רקע
חלק זה הוא רק עבור היכרות הקוד, עברו עליו במלואו וודאו 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 למערבי 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. ✍

מצ"ב התוצאה שהתקבלה לאחר הרצת הניסוי:

```
"C:\Users\Uriya Hubara\AppData\Loca
Test Accuracy: 96.46%

Process finished with exit code 0
```

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 מייעל לנו את זמן הריצה של התוכנית, שכן נוכל לעצור אותה מבלי לרוץ על כל העץ.
בנוסף לכך הגיזום מאפשר לנו להתעלם מרעש שמופיע וכך מונע overfitting.

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**

.4

.a סעיף זה בונוס (5 נקודה לציון התרגיל):

שימו לב, זהו סעיף יבש ואין צורך להגיש את הקוד שכתבתם עבורו.

בצעו כיוונון לפרמטר M על קבוצת האימון:

1. בחרו לפחות חמישה ערכים שונים לפרמטר 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- K קבוצות יש להשתמש בפונקציה 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 M על הדיוק. צרפו את הגרף בדו"ח. (לשימושכם הפונקציה $util\_plot\_graph$ בתוך הקובץ $utils.py$).

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

תם סעיף הבונוס, הסעיף הבא הינו סעיף חובה.

✍️ **.b** (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 $M = 50$.

מצ"ב התוצאות שהתקבלו לפני ואחרי הגיזום:

```
"C:\Users\Uriya Hubara\AppData\Local\
Test Accuracy: 96.46%
Test Accuracy: 97.35%

Process finished with exit code 0
```

כפי שניתן לראות, לאחר הגיזום ישנו שיפור בדיוק.

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.py, ID3_experiments.py
  - בחלק של mdp - mdp_implementation.py

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