

תרגיל בית 3 – MDP ומבוא ללמידה

עברו על כלל ההנחיות לפני תחילת התרגיל.

הנחיות כלליות:

- תאריך ההגשה: לחלק א' של התרגיל (MDP) – עד ליום האחרון של הסמסטר - 08/04/2024 ב23:59
לחלק ב' של התרגיל (מבוא ללמידה) – עד לסוף מועדי א' - 17/05/2024 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: **דניאל אלגריסי**.
- 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, העתקות טטופלנה 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 הפיתוח 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 (44 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.

כעת, נרחיב את ההגדרה הזו, לתגמול המקבל את המצב הנוכחי והמצב אליו הגיע הסוכן, כלומר:

$R: S \times S \rightarrow \mathbb{R}$, למתן תגמול זה נקרא "תגמול תוצאתי". לצורך שלמות ההגדרה, נגדיר שאם לכל

$$a \in A \text{ מתקיים } - \infty < R(s, a) < \infty \text{ אז } P(s'|s, a) = 0$$

א. (1 נק') התאימו את הנוסחה של התוחלת של התועלת מהתרגול, עבור התוחלת של התועלת

המתקבלת במקרה של "תגמול תוצאתי", אין צורך לנמק.

ב. (1 נק') כתבו מחדש את נוסחת משוואת בלמן עבור המקרה של "תגמול תוצאתי", אין צורך לנמק.

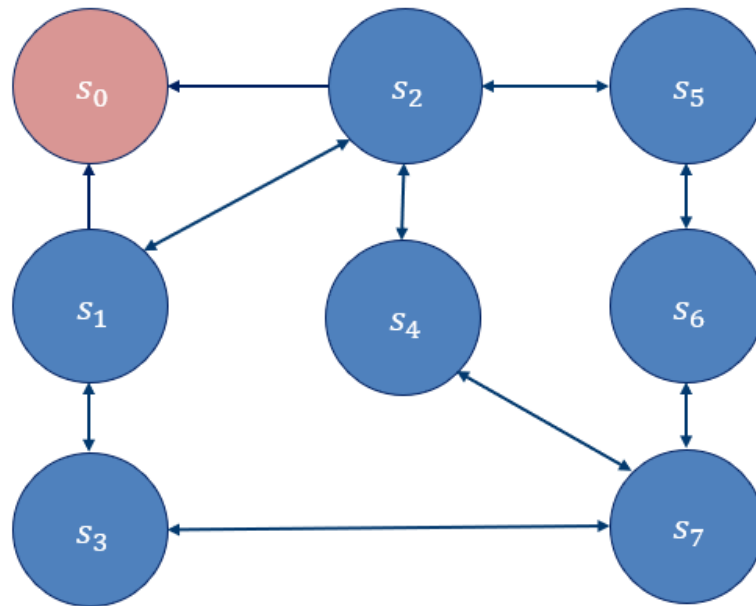
בסעיפים הבאים התייחסו גם למקרה בו $\gamma = 1$, והסבירו מה לדעתכם התנאים שצריכים להתקיים על

הסביבה mdp על מנת שתמיד נצליח למצוא את המדיניות האופטימלית.

ג. (2 נק') נסחו את אלגוריתם Value Iteration עבור המקרה של "תגמול תוצאתי".

ד. (2 נק') נסחו את אלגוריתם Policy Iteration עבור המקרה של "תגמול תוצאתי".

נתון הגרף הבא:



נתונים:

- (Discount factor) $\gamma = 0.5$.
- אופק אינסופי.
- $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$ – קבוצת המצבים – מתארים את מיקום הסוכן בגרף.
- $S_G = \{s_0\}$ – קבוצת המצבים הסופיים.
- קבוצת הפעולות לכל מצב (על פי הגרף), לדוגמא: $A(s_3) = \{\uparrow, \rightarrow\}$.
- תגמולים ("תגמול על פעולה"):
 - $\forall s \in S, s' \in S \setminus S_G: R(s, s') = -1, \quad R(s_1, s_0) = 5, \quad R(s_2, s_0) = 7$
- מודל המעבר הוא דטרמיניסטי, כלומר כל פעולה מצליחה בהסתברות אחת.

ה. (יבש 2 נק') הרץ את האלגוריתם Value iteration שכתבת על הגרף הנתון. ומלא את הערכים בטבלה הבאה, כאשר $\forall s \in S \setminus S_G: U_0(s) = 0$. (ייתכן שלא צריך למלא את כולה).

	$U_0(s_i)$	$U_1(s_i)$	$U_2(s_i)$	$U_3(s_i)$	$U_4(s_i)$	$U_5(s_i)$	$U_6(s_i)$	$U_7(s_i)$	$U_8(s_i)$
s_1	0								
s_2	0								
s_3	0								
s_4	0								
s_5	0								
s_6	0								
s_7	0								

ו. (יבש 2 נק') הרץ את האלגוריתם Policy iteration שכתבת על הגרף הנתון. ומלא את הערכים בטבלה הבאה, כאשר המדיניות ההתחלתית π_0 מופיעה בעמודה הראשונה בטבלה. (ייתכן שלא צריך למלא את כולה).

	$\pi_0(s_i)$	$\pi_1(s_i)$	$\pi_2(s_i)$	$\pi_3(s_i)$	$\pi_4(s_i)$	$\pi_5(s_i)$	$\pi_6(s_i)$	$\pi_7(s_i)$	$\pi_8(s_i)$
s_1	↓								
s_2	↓								
s_3	→								
s_4	↑								
s_5	←								
s_6	↑								
s_7	↑								

ז. (יבש 2 נק') חיזרי על הסעיף הקודם. הפעם עם אופק סופי כאשר $N = 2$ (שימי לב, המדיניות לא חייבת להסתיים במצב מסיים, ישנם מצבים שלא יכולים להגיע למצב מסיים עם אופק זה. ישנם צמתים עם מספר תשובות נכונות, נקבל את כולם).

ח. (1 נק') ללא תלות בשינוי של הסעיף הקודם. אם $\gamma = 0$, מה מספר המדיניות האופטימליות הקיימות? נמקו.

ט. (1 נק') ללא תלות בשינוי של הסעיף הקודם, הסבירי מה היה קורה אם

$$R(s_1, s_2) = R(s_2, s_1) = 2, \quad \gamma = 1$$

בתשובתך, התייחסי גם לערכי התועלות של כל צומת וגם לשינוי במדיניות, אין צורך לחשב.

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 קבוצה 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:

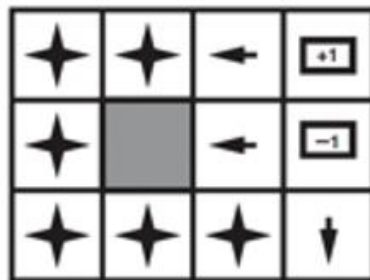
All the built-in packages in python, numpy, matplotlib, argparse, os, copy, typing, termcolor, random

עליכם לממש את הפונקציות הבאות:

- (רטוב 6 נק'): `value_iteration(mdp, U_init, epsilon)` – בהינתן ה-`mdp`, ערך התועלת ההתחלתי `U_init`, וחסם העליון לשגיאה מהתוחלת של התועלת האופטימלי `epsilon` מריץ את האלגוריתם `value iteration` ומחזיר את `U` המתקבל בסוף ריצת האלגוריתם. **TODO**
- (רטוב 4 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 4 נק'): `policy_evaluation(mdp, policy)` – בהינתן ה-`mdp`, ומדיניות `policy` מחזיר את ערכי התועלת לכל מצב. **TODO**
- (רטוב 6 נק'): `policy_iteration(mdp, policy_init)` – בהינתן ה-`mdp`, ומדיניות התחלתית `policy_init`, מריץ את האלגוריתם `policy iteration` ומחזיר מדיניות אופטימלית. **TODO**

למימוש הפונקציות הבאות ניתן להשתמש באיזה ספריות שתמצאו.

- (רטוב 5 נק'): `get_all_policies(mdp, U, ...)` – בהינתן ה-`mdp`, וערך התועלת `U` (המקיים את משוואת בלמן) מדפיס\מציג את כל המדיניות המקיימות ערך זה בלוח בודד (יש לבצע ויזואליזציה להצגת כל המדיניות), לדוגמא:

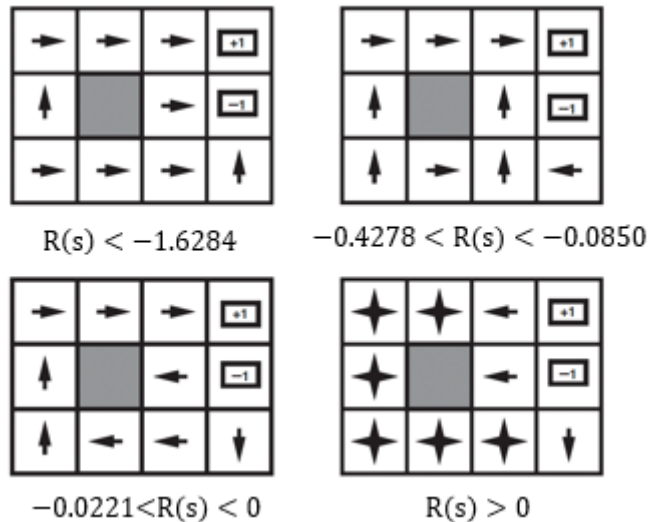


הפונקציה מחזירה את מספר המדיניות (`policies`) השונות הקיימות את `U`. **TODO**

- (רטוב 5 נק'): `get_policy_for_different_rewards(mdp, ...)` – בהינתן ה-mdp מדפיס\מציג את

המדיניות האופטימלית כתלות ב- r (ערכי התגמול לכל מצב שאינו סופי). **TODO**

דוגמא חלקית של פתרון אפשרי:



👉 בנוסף לקוד עליכם לצרף להגשה היבשה את התצוגות של הפונקציות על הסביבה שניתנה בתרגיל.

עבור מצבים סופיים וקירות (WALL), הערך שצריך לחזור בתאים אלו עבור טבלאות המדיניות 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, שכרגע 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 ב-IDE לדוגמה PyCharm.

חלק ב' - מבוא ללמידה (56 נק')

👉 חלק א' – חלק היבש (28 נק')

kNN – נעים להכיר

בחלק זה תכירו אלגוריתם למידה בשם kNN, או בשמו המלא k-Nearest Neighbors, כאשר ה-k הוא למעשה פרמטר!

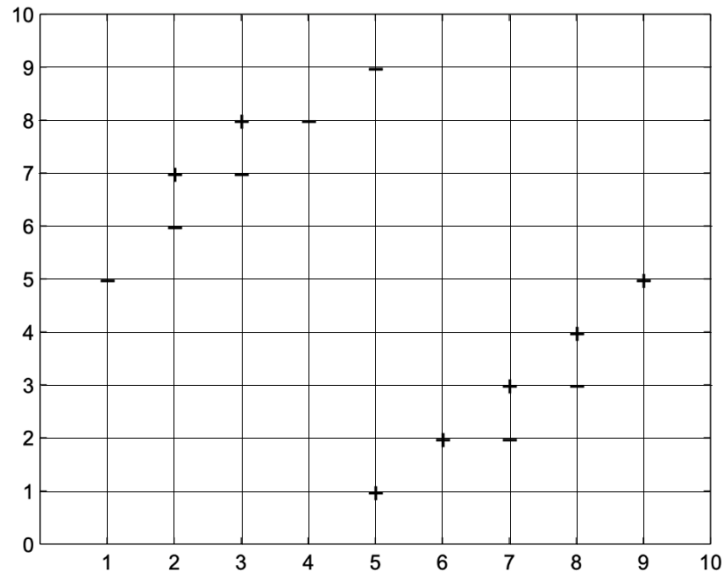
יהי סט אימון עם n דוגמות, $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$, כאשר $\forall i: x_i \in \mathbb{R}^d, y_i \in \mathcal{Y}$. כלומר הדוגמות הינן וקטורים d -ממדיים והתגיות הינן מדומיין כלשהו, הבעיה היא בעיית קלסיפיקציה (סיווג). אם לא נאמר אחרת, הקלסיפיקציה תהיה בינארית, כלומר $\mathcal{Y} = \{-, +\}$. עבור כל דוגמה בסט האימון, ניתן להסתכל על הכניסה ה- i בווקטור כעל ה- i feature 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 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 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.
- 1) עבור איזה ערכים של d, k נקבל שאין תלות בבחירת פונקציית המרחק? (נמקי)
 - 2) 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 דוגמת המבחן.

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

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



- (3) (1 נק') איזה ערך של k עלינו לבחור על מנת לקבל את הדיוק המרבי על קבוצת האימון? מה יהיה ערך זה?
- (4) (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?
- (5) (2 נק') נמקו מדוע שימוש בערכי k גדולים או קטנים מדי יכול להיות גרוע עבור קבוצת הדגימות הנ"ל.
- (6) (2 נק') שרטט את גבול ההחלטה של 1-nearest neighbor עבור הגרף.

השוואה בין מודלי למידה:

- (1) (3 נק') 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) (3 נק') הציגו מסווג מטרה $f(x): R^2 \rightarrow \{0,1\}$ 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 המטרה), אך למידת עץ ID3 תניב מסווג אשר עבורו קיימת לפחות דוגמת מבחן אפשרית אחת עליה הוא יטעה.
- (3) (3 נק') הציגו מסווג מטרה $f(x): R^2 \rightarrow \{0,1\}$ 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.
- (4) (3 נק') הציגו מסווג מטרה $f(x): R^2 \rightarrow \{0,1\}$ 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 המטרה), וגם למידת עץ ID3 תניב מסווג עונה נכון עבור כל דוגמת מבחן אפשרית (כלומר יתקבל מסווג המטרה).

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 τ שמושווה לfeature של הדוגמה. לפעמים ערך הסף קרוב מאוד לערך feature 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 τ_i . אם מתקיים $|x_i - \tau_i| \leq \varepsilon_i$ 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 הסיווג 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.

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*. ככלל, 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/מחלקות 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 באלה, 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).

תיקיית *dataset – ID3*:

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

קובץ *utils.py*:

- 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**.

קובץ *unit test.py*:

- קובץ בדיקה בסיסי שיכול לעזור לכם לבדוק את המימוש.

קובץ *DecisionTree.py*:

- קובץ זה מכיל 3 מחלקות שימושיות לבניית עץ *ID3* שלנו.
 - המחלקה *Question*: מחלקה זו מממשת הסתעפות של צומת בעץ. היא שומרת את התכונה ואת הערך שלפיהם מפצלים את הדאטה שלנו.
 - המחלקה *DecisionNode*: 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*).
 - המחלקה *Leaf*: 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}).

קובץ *ID3.py*:

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

קובץ *ID3 experiments.py*:

- קובץ הרצת הניסויים של ID3, הקובץ מכיל את הניסויים הבאים, שיוסברו בהמשך:
cross_validation_experiment, basic_experiment

חלק ג' – חלק רטוב ID3 (28 נק')

עבור חלק זה מותר לכם להשתמש בספריות הבאות:

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

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

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 ה-ID 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 האינדקס המקסימלי.
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. 🍌

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? 🍌


b. (3 נק') עדכנו 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. (8 נק') 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 $K - \text{fold cross validation}$ על קבוצת האימון בלבד.

כדי לבצע את חלוקת קבוצת האימון ל- K קבוצות יש להשתמש בפונקציה `sklearn.model_selection.KFold` עם הפרמטרים `shuffle = True, n_split = 5` ו-`random_state` אשר שווה למספר תעודת הזהות של אחד מהשותפים. השתמשו בתוצאות שקיבלתם כדי ליצור גרף המציג את השפעת הפרמטר M על הדיוק. i. 

צרפו את הגרף בדו"ח. (לשימושכם הפונקציה `util_plot_graph` בתוך הקובץ `utils.py`). הסבירו את הגרף שקיבלתם. לאיזה גיזום קיבלתם התוצאה הטובה ביותר ומהי תוצאה זו? ii. 

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?

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 ואף קובץ אחר:
 - קובץ `utils.py`
 - בחלק של עצי החלטה – `ID3.py`, `ID3_experiments.py`
 - בחלק של mdp – `mdp_implementation.py`

אין להכיל תיקיות בקובץ ההגשה, הגשה שלא עומדת בפורמט לא תיבדק.

נספח MDP:

דוגמת הרצה (שימו לב שהרצה זו השתמשה במודל הסתברותי שונה משלכם)

יצירת הסביבה:

```
mdp = MDP(board=board_env,  
          terminal_states=terminal_states_env,  
          transition_function=transition_function_env,  
          gamma=1.0)
```

הדפסת הלוח עם התגמולים לכל מצב:

```
print('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@')  
print("##### The board and rewards #####")  
print('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@')  
mdp.print_rewards()
```

פלט:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
##### 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 |
```

Value iteration:

```
print('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@')  
print("##### Value iteration #####")  
print('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@')
```

```
U = [[0, 0, 0, 0],  
     [0, 0, 0, 0],  
     [0, 0, 0, 0]]  
print("\nInitial utility:")  
mdp.print_utility(U)  
print("\nFinal utility:")  
U_new = value_iteration(mdp, U)  
mdp.print_utility(U_new)  
print("\nFinal policy:")  
policy = get_policy(mdp, U_new)  
mdp.print_policy(policy)
```

פלט:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
##### Value iteration #####  
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
  
Initial utility:  
| 0.0 | 0.0 | 0.0 | 0.0 |  
| 0.0 | WALL | 0.0 | 0.0 |  
| 0.0 | 0.0 | 0.0 | 0.0 |  
  
Final utility:  
| 0.812 | 0.868 | 0.918 | 1.0 |  
| 0.762 | WALL | 0.66 | -1.0 |  
| 0.705 | 0.655 | 0.611 | 0.388 |  
  
Final policy:  
| RIGHT | RIGHT | RIGHT | +1 |  
| UP | WALL | UP | -1 |  
| UP | LEFT | LEFT | LEFT |
```


:Policy iteration

```
print('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@')
print("@@@@@@@@ Policy iteration @@@@@@")
print('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@')

print("\nPolicy evaluation:")
U_eval = policy_evaluation(mdp, policy)
mdp.print_utility(U_eval)

policy = [['UP', 'UP', 'UP', 0],
          ['UP', 'WALL', 'UP', 0],
          ['UP', 'UP', 'UP', 'UP']]
print("\nInitial policy:")
mdp.print_policy(policy)
print("\nFinal policy:")
policy_new = policy_iteration(mdp, policy)
mdp.print_policy(policy_new)

print("Done!")
```

פלט:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@ Policy iteration @@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Policy evaluation:
| 0.812 | 0.868 | 0.918 | 1.0 |
| 0.762 | WALL | 0.66 | -1.0 |
| 0.705 | 0.655 | 0.611 | 0.388 |

Initial policy:
| UP | UP | UP | +1 |
| UP | WALL | UP | -1 |
| UP | UP | UP | UP |

Final policy:
| RIGHT | RIGHT | RIGHT | +1 |
| UP | WALL | UP | -1 |
| UP | LEFT | LEFT | LEFT |

Done!
```