

# מבוא לבינה מלאכותית – תרגול 5

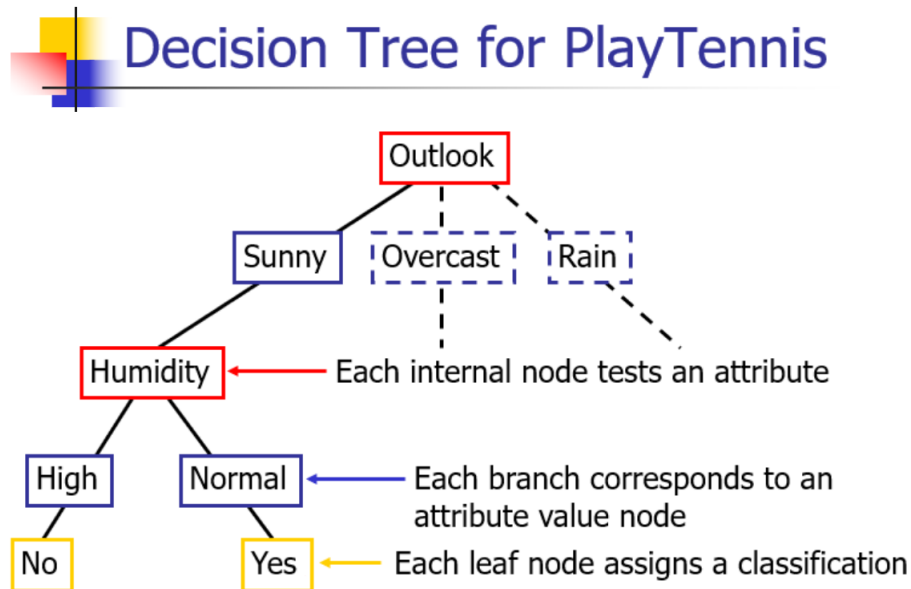
## נושאים:

- עצי החלטה (Decision Trees)
- Random Forest
- XGBoost-ו Boosting

### עצי החלטה – הרעיון הבסיסי:

עצי החלטה הם עצים, במובן של תורת הגרפים, כך שבהינתן קלט מתאים, הפיצ'רים של הקלט קובעים את התיוג של הקלט באמצעות שאלות על הפיצ'רים. העץ וההחלטות עובדים כך: כל נקודות הקלט נכנסות לעץ דרך השורש. לכל קודקוד פנימי (שורש ומה שאינו עלה), הקודקוד מחזיק שאלה על ערך פיצ'ר מסוים, וכל קשת מהקודקוד אל בן שלו בעץ מתאימה לתשובה אפשרית לשאלה זו, כך שלמעשה כל נקודה עוברת מסלול בעץ, בהתבסס על הפיצ'רים שלה. לבסוף, לכל עלה של העץ מתאים ערך, כך שנקודה מהקלט שהגיעה במסלולה אל העלה הזה תקבל את התיוג המוחזק בעלה.

דוגמה - נחליט האם לשחק היום טניס:



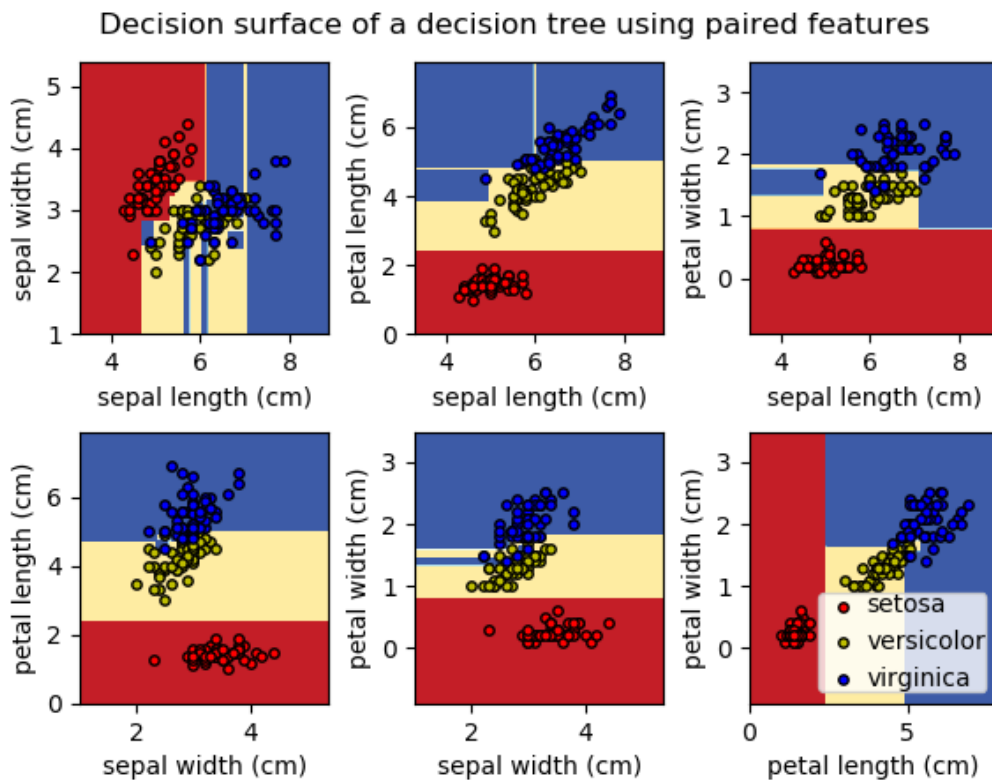
לעץ החלטה יש כל מיני יתרונות כמסווג (או ככלי לרגרסיה), בהם:

- קל להבין מה קורה בו, אפשר גם לראות זאת ממש ולהסיק מסקנות (כמו מה מוביל להחלטה לתייג נקודה באופן שתויגה).

- לא צריך לעשות הרבה פעולות להכנת הדאטא (למרות שאלגוריתם כזה לא מתמודד עם ערכי פיצ'רים חסרים, ולעתים כדאי להפעיל אלגוריתמים להורדת ממד מרחב הפיצ'רים).
- יכול לקבל פיצ'רים מסוגים שונים (רציף או בדיד, מספרי או קטגורי).
- עלות השימוש בעץ (=חיזוי הערך שניתן לנקודה) נמוכה. ליתר דיוק היא לוגריתמית במספר הנקודות שבקבוצת האימון.

כמובן שיש לו גם חסרונות:

- עצי החלטה נוטים לעשות overfitting לדאטא (ראו ציור הממחיש למה).



לכן, נעשה שימוש בשיטות להקטנת עומק העץ (הגבלה של העץ לגודל קבוע או החלטה שלא לחלק אם זה לא משתלם).

- עצי החלטה נוטים להיות לא יציבים – שינויים קטנים בדאטא עלולים להביא לשינויים גדולים בתוצאות שנותן העץ. האפקט הזה פחות חזק כשמשתמשים בעצים רבים להחלטה.
- מציאת עץ ההחלטה האופטימלי היא בעיה שפתרון מלא שלה הוא NP-complete. לכן, מימוש של בניית עצי החלטה נעשה באמצעות אלגוריתמים שלא בהכרח מבטיחי עץ אופטימלי. שוב, שימוש בעצים רבים עוזר לשיפור הלמידה.
- כאשר יש דאטא לא מאוזן מבחינת כמויות, חוסר האיזון מתבטא גם בתוצאות המודל וזה לא רצוי בהרבה מקרים.

## איך בונים עצים כאלה – Entropy, Information Gain, Gini:

כדי לבנות עץ החלטה, נצטרך לקבוע את סדר השאלות שנשאל על הפיצ'רים שלנו. נרצה שהן תהיינה יעילות ככל האפשר, למשל: במשחק "21 שאלות" (חושבים על דמות והמטרה לגלות אותה תוך 21 שאלות), השאלה "האם הדמות היא זזהבה בן?" (לא יודע איך בחרתי דווקא בה) היא שאלה לא טובה, ואילו השאלה "האם הדמות היא זמרת?" היא שאלה טובה יותר.

נלמד עכשיו מדדים לכמה "טובה" השאלה שאנחנו שואלים:

### אנטרופיה:

נניח לשם נוחות שיש לנו שני תיוגים: +, -. האנטרופיה מודדת לנו את אי הודאות שיש לנו לגבי התיוג בקבוצה. היא מוגדרת ע"י:

$$H(S) = -p_+ \log(p_+) - p_- \log(p_-)$$

כאשר  $p_{\pm}$  הם היחסים של התיוגים הרלוונטיים מתוך כלל הקבוצה. ביותר משני תיוגים אפשר לרשום סכום דומה.

נשים לב שככל שיש לנו בקבוצה S יותר מתיוג אחד יחסית לשאר, האנטרופיה תקטן וזה מה שנרצה.

### Information Gain:

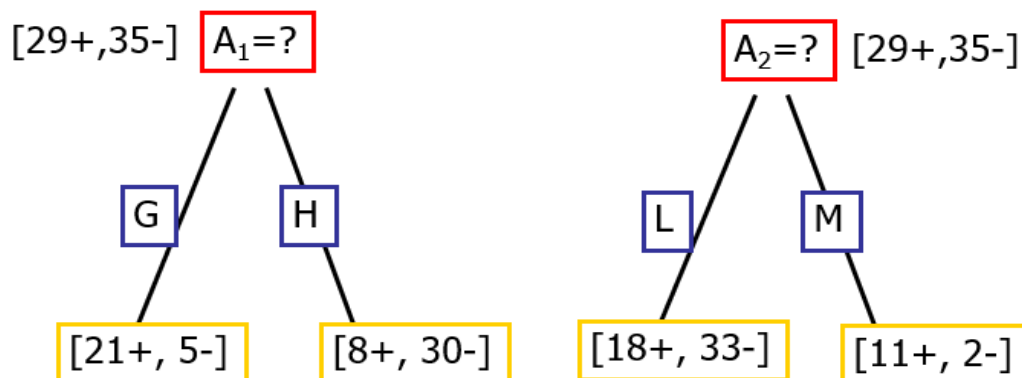
בעזרת האנטרופיה, משתמשים במדד הבא לקביעת הפיצ'ר שיהיה הקריטריון שלנו:

$$Gain(S, a) = H(S) - \sum_{v \text{ in children}} \frac{|S_v|}{|S|} H(S_v)$$

כלומר, ההפרש בין האנטרופיה של הקבוצה לפני החלוקה, לבין הסכום הממושקל (לפי יחסי הגדלים של הקבוצות החדשות) של האנטרופיות בקבוצות החדשות.

ככל שנקטין את האנטרופיה בקבוצות, כך ה-Gain יגדל. לכן, פיצ'ר שייתן לנו Gain גדול יהיה עדיף.

דוגמה:



$$\text{Entropy}([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$$

$$\text{Entropy}([21+,5-]) = 0.71$$

$$\text{Entropy}([8+,30-]) = 0.74$$

$$\text{Gain}(S,A_1) = \text{Entropy}(S)$$

$$-26/64 * \text{Entropy}([21+,5-])$$

$$-38/64 * \text{Entropy}([8+,30-])$$

$$= 0.27$$

$$\text{Entropy}([18+,33-]) = 0.94$$

$$\text{Entropy}([11+,2-]) = 0.62$$

$$\text{Gain}(S,A_2) = \text{Entropy}(S)$$

$$-51/64 * \text{Entropy}([18+,33-])$$

$$-13/64 * \text{Entropy}([11+,2-])$$

$$= 0.12$$

ג'יני:

זה מדד נוסף לכמה "טהור" הדאטא שאנחנו מחלקים. הפעם, עבור K קלאסים שונים, ה- gini impurity בקודקוד הוא:

$$H(S) = \sum_{i=1}^K p_i(1 - p_i)$$

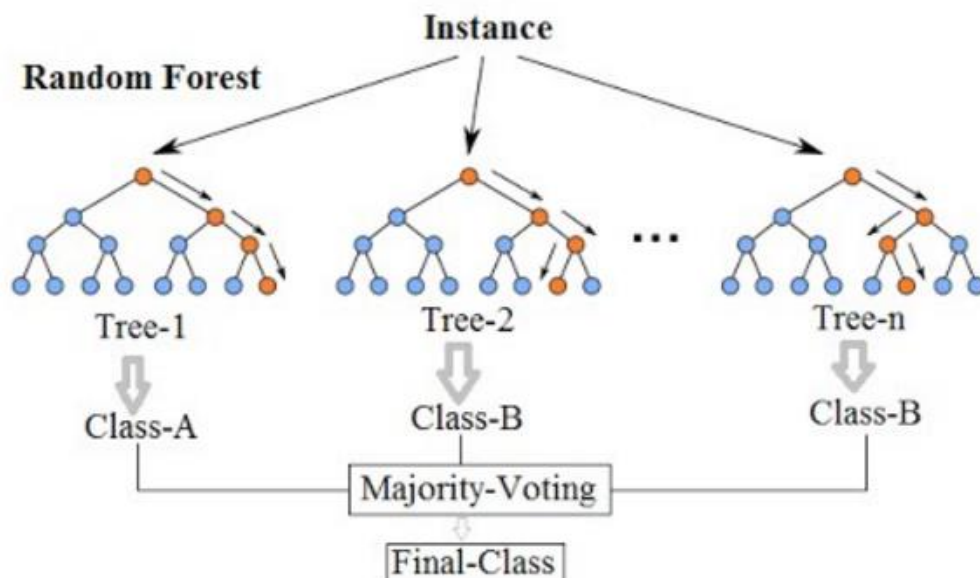
כאן, נרצה ערך נמוך ככל האפשר (אידיאלית, 1 מערך אחד היה נותן 0 בסכום). חלוקה שתיתן סכום ממושקל נמוך של ערכי ג'יני בקבוצות המחולקות תהיה מועדפת (משתמשים באותה נוסחה כמו למעלה, רק שהמשמעות של  $H(S)$  פה שונה).

## אלגוריתמים לעצי החלטה – קצת על ID3, C4.5, C5 ועל Random Forest:

כאמור, בעיית מציאת עץ אידיאלי לא ניתנת לחישוב בזמן סביר, ולכן במהלך השנים ניסו למצוא שיטות לבניית עצי החלטה. רעיונות נפוצים שעלו:

- בניית עץ בהתבסס אך ורק על Gain מקסימלי לכל חלוקה (כלומר, אלגוריתם חמדן). אלגוריתם ID3 (Quinlan, 1986) בונה כך עץ החלטה, כאשר הוא בונה את העץ במלואו (כלומר, עד הגעה לתיג ספציפי), ובגרסה הישנה שלו הוא עובד רק עם פיצ'רים קטגוריים.
- מציאת ערכי סף לפיצ'רים רציפים. למשל, עבור פיצ'ר של גובה, האלגוריתם ימצא שערך הסף הכי מתאים הוא 1.80 מטר. אלגוריתם C4.5 הוא שיפור של הקודם (Quinlan, 1993), בכך שהוא מאפשר גם להשתמש בפיצ'רים כאלה, וכן מתמודד עם ערכים חסרים (פשוט מתעלמים מהם בחישוב ה-Gain).
- "גיזום" של העץ: במקום לבנות את העץ עד החלטה "קשיחה" על תיג, אפשר לקצר את העץ אם חלוקה נוספת לא מוסיפה Gain משמעותי. זה מקטין overfitting. כך עושה C4.5.
- אלגוריתם C5.0 (Quinlan, 1996) משפר את C4.5. ההבדלים מקודמו הם בין היתר: שיפור סיבוכיות זיכרון וזמן ריצה, ייצור עצים קטנים יותר, הכנסת אפשרות למשקל סוגים שונים של אי התאמה (למשל, חשוב יותר שלא לטעות ולסווג אדם חולה בתור בריא מאשר להפך) וגם אפשרות לבצע boosting (נראה מה זה בהמשך התרגול).

### Random Forest:



באלגוריתם זה, אנחנו בונים "יער" (אוסף של עצים). הוא יהיה מורכב מהרבה עצי החלטה שונים, כאשר כל אחד מהעצים יהיה מבוסס על חלק מהפיצ'רים שדגמנו אקראית.

לעתים, כל עץ גם יאמן רק על חלק מקבוצת האימון, או ייקח פיצ'רים מוסטים במעט מהערכים המקוריים שלהם. הסיווג לבסוף יתבצע לפי החלטת רוב העצים או מדדים דומים.

המטרה באלגוריתם הזה לעומת עץ החלטה יחיד היא (בעזרת ריבוי העצים ויצירת השונות בפיצ'רים) מניעת overfitting ושיפור היציבות.

כמה הבדלים עיקריים מעץ החלטה רגיל ויחיד:

- מתוך עץ החלטה יחיד, ניתן להבין מהם השיקולים לסיווג דוגמה באופן שבו היא בוצעה (למשל: לא נלך לשחק טניס כי יש גשם, או כי אין פרטנר למשחק וגם המחבט שבור). לעומת זאת, בריבוי של עצים, אי אפשר לקבל דבר כזה באופן מוחלט. מה שכן אפשר לקבל הוא feature importance, כלומר כמה מכריע פיצ'ר מסוים להחלטה.
- כאמור, אפקט ה-overfitting נמנע (אבל לא תמיד באופן מוחלט). זאת בגלל ריבוי העצים והחלוקה לתת קבוצות של פיצ'רים, מה שגם יוצר עצים קטנים יותר.
- כמות עצים גדולה באלגוריתם random forest עלולה לגרום זמן חישוב ארוך בהרבה משיטות למידה אחרות, ובהן עץ החלטה יחיד.
- לא דיברנו על זה, אבל גם רגרסיה ניתן לבצע בעזרת עצים. רגרסיה נעשית קלה ויעילה יותר לביצוע בעזרת random forest מאשר עץ החלטה (אלגוריתם למשל שמאפשר רגרסיה בעזרת עץ החלטה הוא CART. זה אלגוריתם שבו גם נעשה שימוש בג'ני).

## Boosting:

אינטואיטיבית, הגישה של Boosting היא ללמוד בחלקים: להשתמש באוסף של מודלים נלמדים פשוטים ("Weak Learners"), שהביצועים שלהם לכשעצמם לא טובים, ולשלב אותם. בצורה איטרטיבית, כל weak learner מצליח לסווג חלק מתוך הדאטא בוודאות גבוהה. כאשר כבר סיווגנו נקודות מסוימות בוודאות גבוהה, אין מה לחזור ולסווג אותן ולכן ה-weak learner הבא ייתן עדיפות לסיווג של נקודות שעוד לא סווגו בוודאות גבוהה.

### אלגוריתם:

בהינתן סט אימון  $S = (x_1, y_1), \dots, (x_m, y_m)$  weak learner שבחרנו מראש ומספר סיבובים  $T$ .

1. אתחל את הבעיה כאשר לכל הנקודות אותה חשיבות לסיווג: יש לנו וקטור התפלגות

$$D^{(1)} = \left(\frac{1}{m}, \dots, \frac{1}{m}\right)$$

שמתאר את החשיבות של כל נקודה (ניתן גם לאתחל אחרת).

2. לכל  $t = 1, \dots, T$ :

- הפעל Weak learner ללמוד על הדאטא.

- חשב פונקציית שגיאה של הלומד, ממושקלת לפי חשיבויות הנקודות  $D_i^{(t)}$ .

- חשב משקל מתאים ללומד  $w_t$  כתלות בשגיאה  $\epsilon_t$ . ככל שהשגיאה נמוכה יותר, נרצה

שהמסווג יהיה בעל משקל גבוה יותר.

- עדכן את וקטור החשיבויות  $D_i^{(t+1)} = D_i^{(t)} \cdot something(i)$ , כך שאם הוודאות של הסיווג

של  $i$  גבוהה אז החשיבות תקטן ולהפך.  $something$  יכול להיות למשל  $softmax$ , כך ש-

$$D_i^{(t+1)} = \frac{D_i^{(t)} e^{-w_t y_i h_t(x)}}{\sum_j D_j^{(t)} e^{-w_t y_j h_t(x)}}$$

3. לבסוף נקבל מסווג  $h_{final}(x) = \sum_{t=1}^T w_t h_t(x)$  עד כדי נרמול בסכום המשקלים.

### דוגמה – XGBoost (על קצה המזלג):

זהו אלגוריתם שעושה Boosting ומשתמש בעצים כ-weak learners.

מבלי להיכנס לפרטים, האלגוריתם מבסס את הניחוש שלו על אוסף של עצי החלטה. פונקציית

השגיאה מורכבת משני איברים: loss מבוסס על הפער בין התיוג האמיתי והתיוג שניחשנו,

ורגולריזציה שנועדה לשמור על העצים בגודל סביר.

המאמץ להקטנת פונקציית השגיאה (או, בשפת בני אדם, אימון) מתבצע בצורת Boosting, כאשר

העץ שמתווסף בכל איטרציה יהיה זה שיקטין את השגיאה ככל האפשר (מעשית, באלגוריתם

מבצעים קירוב מסדר שני לשגיאה ובעזרתו מוסיפים את העץ המתאים).

באלגוריתם יש עוד גורמים שמנסים להילחם ב-overfitting מעבר לרגולריזציה:  
מצמצמים את התרומות שעצים אינדיבידואליים מכניסים (דומה ל-learning rate ב-gradient descent) ומשתמשים ב-subsampling (דגימה של חלק בלבד מכלל הפיצ'רים לבניית העצים).  
לא אכנס לשאלה איך המודל בוחר לפצל עלים בעצים.

מוזמנים לקרוא בפירוט את [המאמר](#) שמתאר את האלגוריתם ולראות כיצד כמה רעיונות שראינו בקורס משתלבים באלגוריתם.