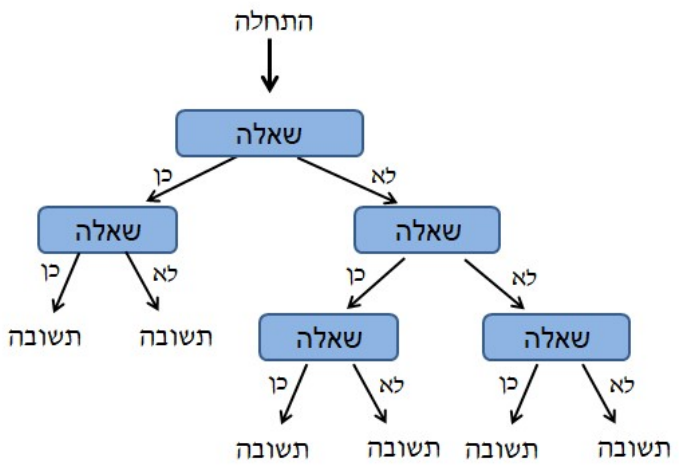
עצי החלטה – Decision Trees

1. הקדמה:

עצי החלטה יכולים לשמש הן לבעיות קלסיפיקציה (סיווג) והן לבעיות רגרסיה כפי שיוסבר בהרחבה בהמשך הפרק. עצי החלטה למעשה לוקחים את המשתנה שברצוננו להסביר (נקרא גם משתנה חזוי, משתנה מוסבר או משתנה מטרה), ומחלקים את המרחב שלו לסגמנטים (קבוצות). כאשר מתקבלת תצפית חדשה ואנחנו צריכים לשייך אותה לאחד מהסגמנטים, אנחנו משתמשים ב- "ממוצע" (mean) או ב- "שכיח" (mode) של ה- training data ששייכים לאותו סגמנט. מאחר וכללי הסיווג לסגמנט כזה או אחר יכולים להצטייר בצורת עץ, השיטה הזו נקראת "עצי החלטה".

הגישות השונות בעצי החלטה פשוטות ואינטואיטיביות להבנה, אולם הם לא מצליחות להתחרות במדדי הדיוק של מודלים אחרים של supervised learning. לכן, בפרקים הבאים נציג שיטות כמו: bagging, random forests and boosting. כל אחת מהגישות הללו מבצעת בנייה של כמה עצי-החלטה שמשולבים בסופו של דבר לניבוי אחד. נראה בפרקים הבאים ששימוש במספר גדול של עצים יכול לשפר דרמטית את מדדי הדיוק של המודל, אך ב- "מחיר" של יכולת הסבר פחות אינטואיטיבית לגורם צד ג' שלא מבין את השיטה לעומק (למשל גורם עסקי בארגון שאנחנו מעוניינים להסביר לו את תוצאות המודל).

המבנה הבסיסי של עץ הוא כזה:



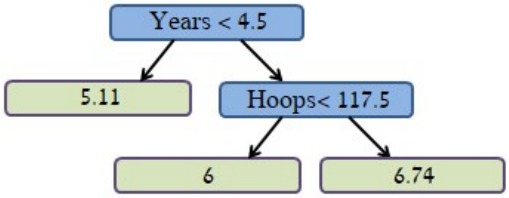
על מנת להבין את מבנה העץ ולייצר שפה משותפת נציג את השמות המקובלים בעבודה עם עצים:

- Root (שורש) – נקודת הכניסה לעץ (חלקו העליון ביותר של העץ).
- Node (צומת) – נקודת ההחלטה/פיצול של העץ – השאלות.
- Leaves (עלים) – הקצוות של העצים – התשובות. נקראים גם terminal nodes.
- Branch (ענף) – חלק מתוך העץ המלא (תת-עץ)
- Depth (עומק) – מספר ה- nodes במסלול הארוך ביותר בעץ.

2. היסודות של עצי החלטה:

עצי החלטה יכולים לשמש הן לבעיות קלסיפיקציה (סיווג) והן לבעיות רגרסיה. נתחיל קודם בבעיות רגרסיה ולאחר מכן נעבור לבעיות קלסיפיקציה.

איור 1:



העץ באיור מעלה מתאר עץ רגרסיה שחוזה את Log השכר של שחקן כדורסל בהינתן מספר השנים שהוא שיחק בליגת העל וכמות הסלים שהוא קלע בשנה הקודמת.

הצד השמאלי של העץ מתייחס ל- $X_j < t_k$ והצד הימני של העץ מתייחס ל- $X_j \geq t_k$.

כך למשל, חלקו העליון של העץ (ה- Root) מתחלק ל-2 ענפים. הענף השמאלי מתייחס לחלק שבו **years<4.5**, והענף הימני מתייחס לחלק שבו **years>=4.5**.

לעץ יש 2 צמתים (שאלות. נקראים גם internal nodes) ו-3 עלים (תשובות. נקראים גם terminal nodes). המספר בכל עלה שבתחתית העץ הוא הממוצע של כל התצפיות שסווגו לעלה הזה. לאחר שהעץ יהיה מוכן, כל תצפית חדשה שתסווג לעלה מסוים תקבל את הערך הממוצע של התצפיות שעל בסיסם נבנה העץ (ה- training data).

1.2.עצי רגרסיה:

כדי להבין מהם עצי רגרסיה נתחיל בדוגמא פשוטה:

חיזוי שכר שחקני כדורסל באמצעות עצי החלטה:

אנחנו נשתמש בנתונים על שחקני כדורסל בכדי לחזות את השכר של כל שחקן בהתבסס על:

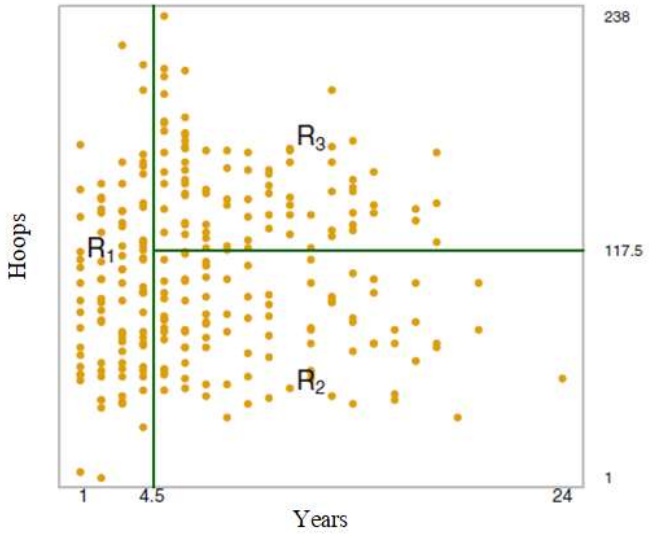
- **שנים** - מספר השנים שאותו ששחקן שיחק בליגת העל.
- **סלים** - מספר הסלים שהוא קלע בשנה הקודמת.

תחילה נסיר תצפיות ללא ערכים עבור המשתנה המוסבר שלנו, הלא הוא "**שכר**" ובנוסף נבצע על אותו משתנה log-transform בכדי שהוא יהיה ככל הניתן בקירוב להתפלגות נורמלית (עקומת גאוס/פעמון), ונזכור שהשכר כעת נמדד באלפי שקלים.

איור 1 לעיל מציג עץ רגרסיה שהותאם לנתונים בדוגמא. נקודות הפיצול והערכים שבהם נקבעו ע"פ סט של כללי-פיצול שנסביר בפירוט בהמשך הפרק.

איור 2 מציג אופן הצגה שונה לעץ הרגרסיה שלנו שמסווג את הנתונים ל-3 קבוצות.

איור 2:



השכר חושב ע"פ log השכר הממוצע של השחקנים ב- training set באזור שבו **years<4.5** (R1). עבור שחקנים אלה, log השכר הממוצע הוא 5.107, כך שהשכר החזוי שלהם שווה ל- $e^{5.107}$ אלפי דולרים, כלומר $165,174. שחקנים עם **years>=4.5** מסווגים לענף הימני (איור 1) ומתחלקים שוב ע"פ כמות הסלים שהם קלעו. בסופו של דבר העץ שלנו מחלק את השחקנים לקבוצות של פרדיקציות (חיזויים):

- שחקנים ששיחקו 4 שנים או פחות
- שחקנים ששיחקו 5 שנים או יותר ושקלעו פחות מ- 118 סלים בשנה שעברה
- שחקנים ששיחקו 5 שנים או יותר ושקלעו יותר מ- 118 סלים בשנה שעברה

3 הקבוצות שקיבלנו מיוצגות כך:

R1 ={X | Years<4.5}

R2 ={X | Years>=4.5, Hoops<117.5}

R3 ={X | Years>=4.5, Hoops >=117.5}

איור 2 ממחיש את הקבוצות כפונקציה של years ו- Hoops.

השכר החזוי ל-3 הקבוצות האלה הוא:

R1 ➡ $\$1,000\times e^{5.107}=\$165,174$

R1 ➡ $\$1,000\times e^{5.999}=\$402,834$

R3 ➡ $\$1,000\times e^{6.740}=\$845,346$

אם נשמור על האנלוגיה של העץ, האזורים R1,R2 ו- R3 ידועים בשם terminal nodes, או "עלים" של העץ.

אפשר לפרש את עץ הרגרסיה שבאיור 1 גם בצורה הבאה: years הוא הפרמטר המרכזי ביותר בקביעה מה יהיה גובה השכר, ושחקנים עם פחות שנות ניסיון ירוויחו פחות משחקנים מנוסים יותר. שחקן שאינו בעל הרבה שנות ניסיון, פחות משנה לנו מה כמות הסלים שהוא קלע בשנה הקודמת, מאחר והם משחקים תפקיד יחסית שולי בקביעת גובה השכר שלו, **פשוט כי שנות הניסיון הוא משתנה דומיננטי יותר בקביעת השכר**. לעומת זאת, בקרב שחקנים עם 4.5 שנות ניסיון או יותר, כמות הסלים שהם קלעו בשנה הקודמת כן משפיעה על השכר, ושחקנים שקלעו הרבה סלים בשנה הקודמת כנראה ירוויחו יותר כסף מאשר שחקנים עם אותו ניסיון אך קלעו פחות סלים.

עץ הרגרסיה שהובא בדוגמא מפשט קצת "יתר על המידה" את הקשר "האמתי" בין years, hoops ו- salary. אולם יש לו יתרון על פני מודלים אחרים מסוג רגרסיה (כמו רגרסיה לינארית וכו') בכך שהוא פשוט יותר להבנה וקל יחסית לייצוג באמצעים גרפיים (כמו שראינו באיורים 1 ו-2).

חיזוי באמצעות ריבוד (Stratification) של מרחב המשתנים:

עד עכשיו נתנו הבנה אינטואיטיבית כיצד העץ נבנה. כעת נדבר על **התהליך** של בניית עץ ונפרוט את השלבים.

באופן כללי, יש שני צעדים בבניית העץ:

1. מחלקים את מרחב הערכים האפשריים של המשתנה המוסבר כתלות ב- X1,X2, . . .Xp אל תוך j אזורים נפרדים (ולא חופפים) R1,R2, . . . Rj.
2. לכל תצפית שנופלת באזור Rj אנחנו נותנים את אותו חיזוי, כלומר: ממוצע הערכים בתצפיות של סט האימון שנפלו באותו Rj.

לדוגמא: נניח ובשלב 1 אנחנו משיגים שני אזורים: R1 ו- R2, ושהממוצע של תצפיות האימון ב- R1 הוא 10, והממוצע של תצפיות האימון ב- R2 הוא 20. אז עבור תצפית חדשה X = x - אם x ∈ R1 אנו ניתן לו תחזית של הערך 10, ואם x ∈ R2 - החיזוי שניתן יהיה 20.

איך "תוחמים" את האזורים R1,R2, . . . Rj (שלב 1)?

תיאורטית, לכל אזור יכולה להיות כל צורה שהיא. אולם לטובת הפשטות, אנו בוחרים לחלק את המשתנה ל- "אזורים מרובעים". המטרה היא למצוא את האזורים שמביאים למינימום את **RSS (residual sum of squares)** שמוגדר כך:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

כאשר:

$\hat{y}_{R_j}$ הוא ממוצע המשתנה המוסבר של תצפיות האימון באזור j.

$y_i - \hat{y}_{R_j}$ הוא המרחק כל תצפית לערך הממוצע באזור j.

המשוואה לעיל מטרתה למצוא את מקבץ התצפיות בכל אזור שבו המרחק (בריבוע) בין הערך הממוצע לבין כל תצפית יהיה מינימלי.

למרבה הצער, כשמחלקים את מרחב המשתנה ל- j אזורים - בלתי אפשרי מבחינה חישובית להתחשב בכל החלוקות האפשריות. לכן, אנו לוקחים גישה "חמדנית" אשר עובדת "מלמעלה למטה". הגישה מכונה "פיצול בינארי רקורסיבי" (*recursive binary splitting*).

נסביר. הגישה הזו עובדת "מלמעלה למטה" כי היא מתחילה מראש העץ (מה- root, היכן **שכל** התצפיות עדיין שייכות לקבוצה אחת גדולה), ואז ברצף היא מפצלת מרחב הערכים של המשתנה המוסבר. כל פיצול מסומן באמצעות שני ענפים חדשים בהמשך העץ.

למקרה שתהיתם למה הגישה נקראת "גישה חמדנית", אז זה נובע מכך שבכל שלב בתהליך בניית העץ - הפיצול הטוב ביותר מתבצע **באותו שלב בדיוק**, במקום להסתכל כמה שלבים קדימה ולבחור את הפיצול שיוביל בהמשך לעץ טוב יותר. דרך טובה להמחיש את זה היא לדמיין שאתם עומדים בפקק תנועה, ובניסיון לעקוף את הפקק אתם בוחרים בפניה הראשונה שנראית לכם פחות פקוקה, מבלי להתחשב בפקק שעשוי לבוא בהמשך. לעומת זאת אם הייתם נעזרים ב- waze, יתכן ובהתחלה נדמה היה שהוא לוקח אתכם בדרך ארוכה יותר, אך בדיעבד, בראי התמונה הכוללת עדיף היה להקשיב לו ולא לקחת את הפניה הקרובה רק כי היא נראית לכם הכי טובה.

נחזור לדוגמא שלנו. כדי לבצע את אותו "פיצול בינארי רקורסיבי", נבחר את המשתנה המסביר Xj ואת נקודת החיתוך s שמחלקת את המרחב של המשתנה לקבוצות {X | Xj < s} ו- {X | Xj ≥ s} שיובילו לצמצום האופטימלי של RSS (הביטוי X | Xj < s משמעותו קבוצת הערכים במשתנה **המוסבר** שנמצאת בתחום שבו המשתנה **המסביר** Xj קטן מ- s). כלומר אנחנו מתחשבים בכל המשתנים המסבירים X1, . . .,Xp ובכל נקודות החיתוך s האפשריות לכל משתנה ומשתנה כך שבסוף נבחר את הנקודה s בכל משתנה כך שלעץ הנבחר יהיה את ה- RSS הכי קטן.

בניסוח יותר פורמלי נגיד כך:

2. עבור כל j וכל s נקבל את ערכי המשתנה **המוסבר** בתוך תחום מסוים:

$$R1(j, s) = \{X \mid Xj < s\}$$

$$R2(j, s) = \{X \mid Xj \geq s\}$$

3. לאחר מכן, נחפש את הערכים של j ו- s שמביאים למינימום את המשוואה הבאה:

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

כאשר:

$\hat{y}_{R_1}$ מבטא את ממוצע הערכים של המשתנה **המוסבר** בתצפיות האימון ב- R1(j, s)

$\hat{y}_{R_2}$ מבטא את ממוצע הערכים של המשתנה **המוסבר** בתצפיות האימון ב- R2(j, s)

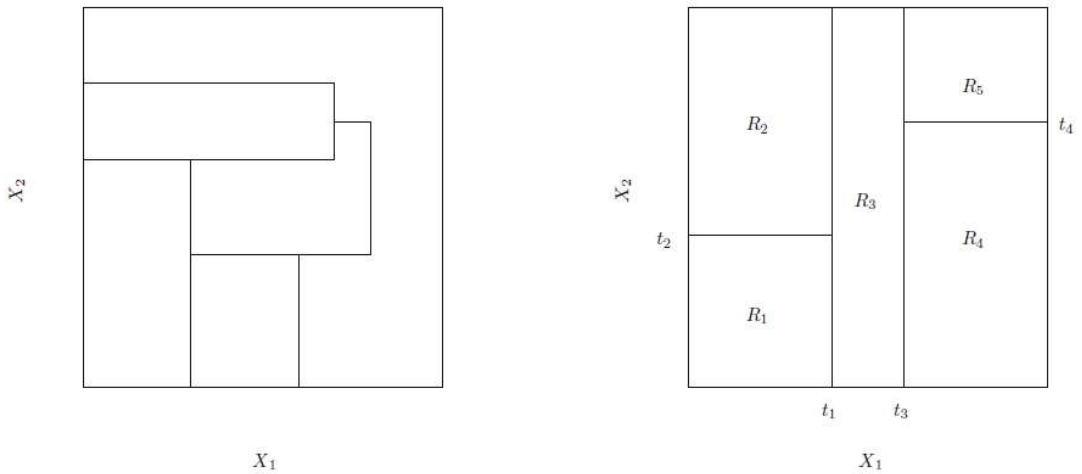
מציאת הערכים של j ו- s שמביאים למינימום את המשוואה יכולה להתבצע די בקלות כשכמות המשתנים המסבירים לא גדולה מידי, אך זה תהליך שעלול להיות די סבוך כשיש הרבה משתנים מסבירים.

קיבלנו מהתהליך הזה ענף אחד שיוצא מה- root, ולו 2 עלים. כעת נחזור על התהליך הזה שוב, במטרה למצוא את המשתנה המסביר **הבא** שמביא למינימום את RSS בכל קבוצה, כך שיהיו לנו כעת 3 קבוצות. ושוב, נרצה לפצל את אחת מאותן 3 קבוצות שכבר יש לנו כדי לצמצם עוד את ה- RSS. התהליך הזה נמשך איטרטיבית (מחזורית) עד שמגיעים "לקריטריון עצירה" מסוים. למשל, אפשר להגדיר קריטריון עצירה שאומר שנמשיך לפצל את העץ לעוד ועוד ענפים/קבוצות עד שאין קבוצה שמכילה יותר מ-5 תצפיות, משהגענו לנקודה כזו אנחנו מפסיקים לנסות לפצל עוד.

אחרי שסיימנו לייצר את הקבוצות R1, . . . , RJ, נשתמש בעץ שלנו כדי לתת חיזוי ל- "נתוני המבחן" (test data). העץ יסווג כל תצפית לקבוצה מסוימת, והחיזוי שהיא תקבל יהא ע"פ ממוצע הערכים של התצפיות ב- training set שנמצאים באותה קבוצה.

באיור מס' 3 מוצגת דוגמא עם 5 קבוצות שחולקו על פי הגישה שתוארה לעיל:

איור 3:

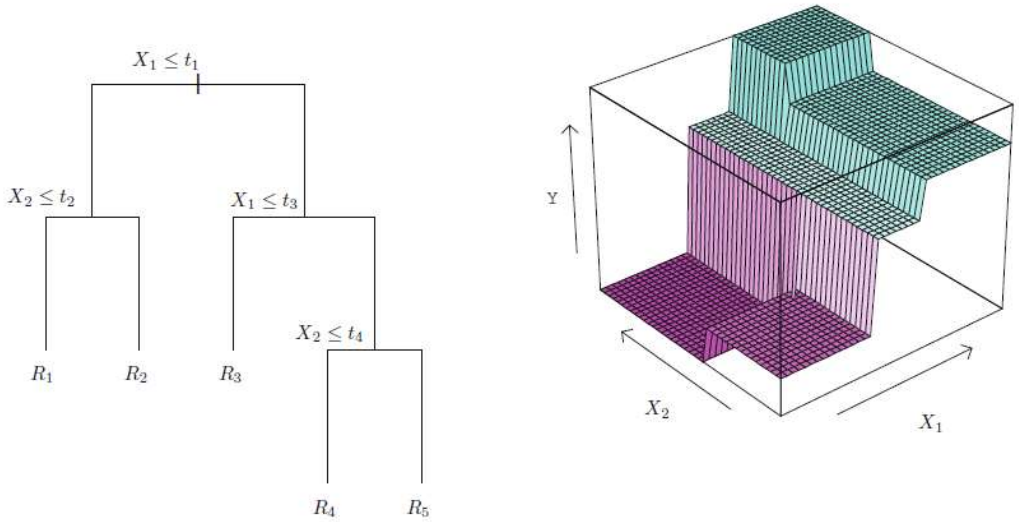


מתוך הספר An Introduction to Statistical Learning

האיור הימני מתאר חלוקה דו-ממדית של שני משתנים באמצעות פיצול בינארי רקורסיבי.

האיור השמאלי מתאר חלוקה דו-ממדית של שני משתנים **שלא יכלה להיווצר** באמצעות פיצול בינארי רקורסיבי (הגישה החמדנית שתיארנו קודם). היא מורכבת מידי ואינה תואמת את הגישה החמדנית שרוצה "חלוקה פשוטה ומהירה".

איור 4:



מתוך הספר An Introduction to Statistical Learning

האיור השמאלי מתאר עץ אשר תואם לחלוקה המוצגת באיור 3 (הימני).
האיור הימני נותן פרספקטיבה רחבה כיצד חיזויים מתבצעים באמצעות העץ.

גיזום (pruning):

מבוא

התהליך שתיארנו למעלה משקף את התהליך הקלאסי של יצירת עץ רגרסיה, והוא יכול לתת חיזויים טובים ב- Training set (RSS נמוך/Low bias), אבל עשוי לעשות התאמת-יתר (overfitting) על הנתונים, מה שיוביל לביצועים גרועים על דאטה חדש (test data). זה קורה בגלל שהעץ שנבנה בתהליך האימון עשוי להיות "מורכב מידי" ובעצם יותר מידי מותאם בצורה כמעט מושלמת לנתוני האימון, אך כשמגיעים נתונים חדשים החיזויים נותנים שגיאה גדולה יותר בין הערך החזוי והערך האמיתי. לעומת זאת, עץ קטן יותר עם פחות פיצולים (כלומר, פחות קבוצות R1, . . ., RJ) ייתן חיזויים עם מעט יותר הטיה (bias) בהשוואה לאלטרנטיבה הראשונה, אך כנראה יוביל לשונות קטנה יותר בין ה- training data וה- test data ולפרשנות טובה יותר.

אז איך בונים נכון יותר את העץ?

אפשרות אחת הינה בניית העץ כך שבכל פיצול הירידה ב- RSS תעלה על סף מהותיות (גבוה) כלשהו. כלומר, פיצול שמוביל לירידה שולית יחסית ב- RSS לא יתבצע. אסטרטגיה זו של קביעת סף מהותיות לפיצול תביא לעצים קטנים יותר (שיעשו פחות overfit).

אך יש גם חיסרון בגישה זו, בכך שהיא קצרת-ראייה. כלומר, יתכן פיצול חסר ערך לכאורה בשלב מוקדם של העץ, אך כזה שעשוי להוביל אחריו לפיצול טוב מאוד - כלומר פיצול שמוביל לירידה גדולה ב- RSS בהמשך. שיטה זו "תדלג" על אותו פיצול "חסר ערך" מאחר והוא לא עובר את סף המהותיות שהוגדר.

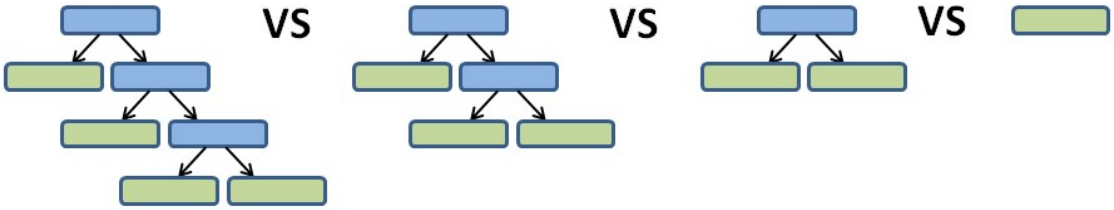
אסטרטגיה טובה יותר תהיה לגדל עץ גדול מאוד T_0, ובשלב שני, כדי להתמודד עם ה- overfit הצפוי לנו מאותו עץ, לגזום כמה מהעלים שלו כך שיתקבל עץ-משנה (subtree) של העץ המקורי. את מקומם של הענפים שהסרנו יתפוס עלה בודד שמקבל ערך ממוצע המתחשב במספר גדול יותר של תצפיות. בניתוח ראשון נראה שב- training-set נקבל עבור אותו עלה שגיאה גדולה יותר, אבל השאיפה היא שזה ישתלם לנו בבדיקת השגיאה ב- test-set.

למעשה כל המטרה של גיזום הוא למנוע Overfitting ב- training data, בכדי שהעץ יעשה עבודה טובה יותר עם ה- test data.

אם כך, כיצד נדע מהי הדרך הטובה ביותר לגזום את העץ?

אינטואיטיבית, המטרה שלנו היא לבחור subtree מתוך העץ הגדול (T_0) שמוביל לשגיאה הקטנה ביותר. לאחר שבחרנו subtree מסוים, אנו יכולים **או** לאמוד את ה- test-error שלו ע"י בדיקת השגיאה **רק** ב- test set, **או** באמצעות *cross validation*.

הערכת השגיאה באמצעות *cross-validation* עבור כל subtree אפשרי תהיה מסורבלת, ארוכה ויקרה מדי (במשאבים) מכיוון שיש המון עצים אפשריים. לכן, ישנה גישה הנקראת *Cost complexity pruning* (ידועה גם בשם *weakest link pruning*). בגישה זו, במקום להתחשב בכל subtree אפשרי, אנו מסתכלים על רצף מסוים של subtrees שהם למעשה חלקים שונים המרכיבים את אותו העץ שבנינו בהתחלה (T_0):



כל עץ ברצף מקבל ציון. אך לפני שנפרט עוד על גישה זו, נפרוט את השלבים ע"פ התובנות שלנו עד כה.

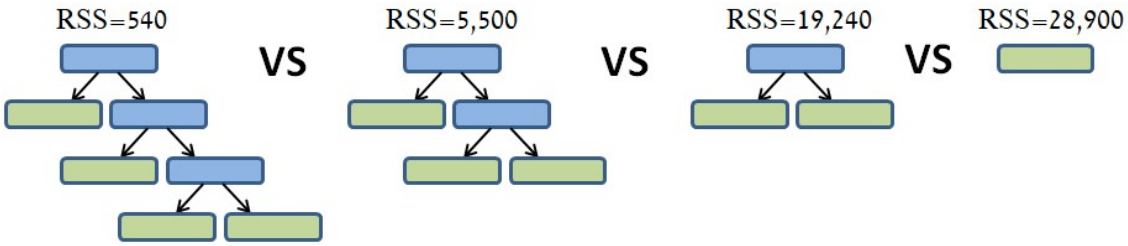
השלבים שנבצע הם כדלקמן:

א. כדי לגזום עץ צריך קודם כל לבנות אחד. אז בשלב ראשון נבנה עץ רגרסיה **הבנוי <u>מכל</u> ה- data** (לא רק מה- training data) בגישת ה- *recursive binary splitting* (הגישה החמדנית שתיארנו קודם) ונעצור רק כאשר לכל עלה (terminal node) יש קצת פחות ממספר מינימום כלשהו של תצפיות (קריטריון עצירה). לעץ הזה נקרא T_0.

ב. כל עלה בעץ מקבל את הערך הממוצע של תצפיות המשתנה המוסבר שבאותו עלה. נחשב את RSS עבור **כל עלה ועלה** באותו עץ.

ג. נסכום את ה- RSS שקיבלנו בכל העלים. הסכום שהתקבל הוא ה- RSS של כל העץ T_0.

ד. כעת נבצע את שלבים א'-ג' ל- subtree הבא ברצף. זהו עץ-משנה שכמעט זהה לעץ המקורי, למעט שני עלים שגזמנו מהעץ הקודם. וכך חוזרים על התהליך בכל ה- subtrees , עד ל- subtree האחרון שמורכב בעצם רק מה- root של העץ המקורי.



- התוצר של השלב הזה הוא RSS לכל subtree ברצף העצים שלנו.
- נשים לב שבכל פעם שהסרנו ענף, קיבלנו RSS גדול יותר לעומת ה- subtree מהשלב הקודם. זה לא מפתיע, שהרי כל העיקרון של הגיזום הוא לקבל עץ שלא יעשה לנו בהמשך overfit, גם אם זה במחיר של שגיאה (bias) גדולה יותר בשלב הנוכחי.

נכון לשלב זה יש לנו כמה subtrees, אבל איך נדע במי מהם לבחור?

ה. לשם כך נשתמש בגישת *Cost complexity pruning* שהזכרנו קודם. היא עובדת כך שהיא מחשבת לכל עץ ברצף העצים שלנו ציון, ע"פ הנוסחה הבאה:

$$\text{Tree score} = \text{RSS} + \alpha T$$

הציון מורכב מ-

- RSS של העץ או של עץ-המשנה.
- עץ מורכב מידי יביא ל- overfitting מחד, אבל יהיה בעל RSS נמוך יותר מאידך. היינו שמחים לקבל את האופטימום משני העולמות אבל יש כאן trade-off שצריך להיעשות. ***α הוא "פרמטר כיוונון" שאנחנו מוצאים באמצעות cross-validation*, והוא שולט בפשרה (trade-off) בין 2 העקרונות הללו**, כלומר בין רמת המורכבות של ה- subtrees ובין ההתאמה לדאטה בשלב האימון.
- T מבטא את מספר העלים (ה- **T**erminal nodes) בעץ או ב- subtree.
- הרכיב αT מכונה Tree Complexity Penalty , ותפקידו "לפצות" על ההפרש במספר העלים שבין ה- subtrees השונים שנבחנים.
- הכוונה היא ש- subtree עם מעט עלים יביא בהכרח ל- RSS גבוה. אך **אנו רוצים שכל עץ יקבל score שמבטא גם את החוזקות שלו ולא רק את ה- RSS שלו**. לשם כך אנו מפצים כל עץ ב- "רכיב" שיאזן קצת את השפעת RSS בחישוב ה- score.

את RSS כבר חישבנו בסעיף הקודם, וכעת נשאר רק לחשב Tree score לכל עץ/עץ-משנה.

נשים לב כי:

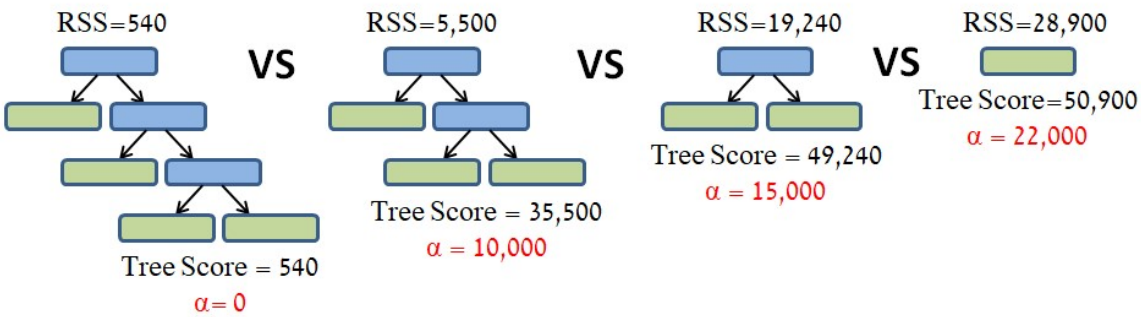
- כאשר α=0, העץ המקורי (T_0) יהיה בהכרח בעל ה- Tree score הנמוך ביותר. זה בגלל שכאשר α=0 אז כל הרכיב αT בנוסחה שווה ל- 0, וה- Tree score יהיה זהה לגמרי למדד ה- RSS:

$$\text{Tree score} = \text{RSS} + \mathbf{0}\, T$$

ממילא שאר שלבי החישוב מיותרים, כי אנחנו כבר יודעים שהעץ T_0 הוא בעל ה- RSS הנמוך ביותר משאר ה- subtrees, וכן בעל ה- Tree Score הנמוך ביותר. **לכן, ל- subtrees הבאים נרצה לקבוע ש- $\alpha > 0$.**

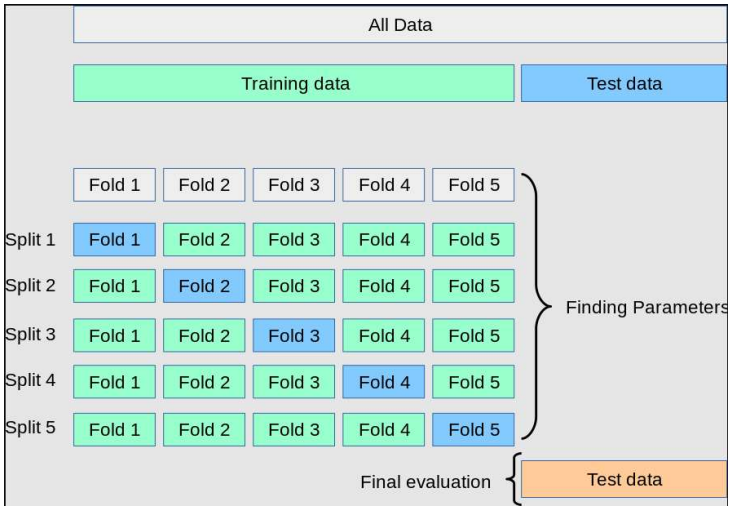
ו. נגזום 2 עלים מתחתית העץ, ובמקביל נגדיל את *α* עד שנקבל Tree Score נמוך יותר ככל האפשר.

ז. נחזור על אותם צעדים עד שאין יותר מה לגזום. בסופו של תהליך ערכים שונים של *α* יתנו לנו רצף של עצים, מעץ מלא ועד לעלה בודד. יוצא, שככל שאנחנו מגדילים את הפרמטר α, ענפים נגזמים מהעץ בצורה מקוננת וצפויה, ולכל ערך של α יהיה subtree מותאם $T \subset T_0$ כך שה- Tree score המתקבל יהיה קטן ככל האפשר.



ח. אחרי שסיימנו את זה, נבחר ב- subtree עם ה- score הנמוך ביותר. אך נזכור מהם ערכי ה- α של ה- subtrees השונים שקיבלנו בסעיף ז' כי אנחנו נשתמש בהם שוב מאוחר יותר.

ט. בשביל לבצע *cross-validation* נחזור ל- data set המלא (שממנו בנינו את העץ הראשון T_0), **והפעם** נחלק אותו ל-training data ול- testing data.



מתוך דוקומנטציה באתר של scikit-learn

- ב- *cross-validation* המודל מתאמן תחילה לפי split 1 על התצפיות שבחלקים הירוקים, ובוחן את החיזויים על סט התצפיות הכחול.
- התהליך הזה נעשה K פעמים, כל פעם על פיצול אחר.
- בכל איטרציה כזו נמדדים הביצועים (RSS), ובסוף לוקחים את ה- RSS הממוצע.
- אחרי מציאת כל הפרמטרים, ההערכה הסופית נעשית על ה- test data.

י. נשתמש רק ב- training data כדי לבנות עץ מלא (הפעם נקרא לו T_1) **ורצף (sequence) חדש** של subtrees שמביאים למינימום את ה- Tree Score (אותו סדר פעולות כמו בסעיפים א'-ז'). אך כדי לבנות את אותם subtrees חדשים נשתמש באותם ערכי α שמצאנו לכל subtree בסעיפים א'-ז'.

- במילים אחרות, ממש כמו קודם כאשר $\alpha = 0$ אנחנו בונים עץ זהה לחלוטין ל- T_1, ולו יהיה ה- Tree Score הנמוך ביותר. לעומת זאת כאשר $\alpha > 0$ נוכל למצוא נקודה שבה ה- Tree Score יהיה נמוך יותר מהעץ הקודם אם נגזום צמד עלים.

יא. נחשב את RSS לכל subtree באמצעות שימוש ב- testing data בלבד. ונרשום לעצמנו מי ה- subtree שקיבל את ה- RSS הנמוך ביותר.

יב. כעת בשביל להשלים את תהליך ה- *cross-validation*, נחזור ל- data set המלא שלנו ונייצר training data ו- testing data **חדשים** (הפעם לפי split 2), ונחזור על סעיפים י'-י"א כדי ליצור רצף חדש של עצים שגם להם נמצא מיהו ה- subtree בעל ה- RSS הנמוך ביותר, ונמשיך ככה עד שנשלים K-fold cross validation.

יג. בכל איטרציה קיבלנו subtree בעל RSS מינימלי וערך α מסוים. אז כעת משסיימנו את כל האיטרציות של ה- *cross-validation* נבדוק מהו ערך ה- α נתן לנו את העץ עם ה- RSS המינימלי, **וזה יהיה הערך הסופי של α.**

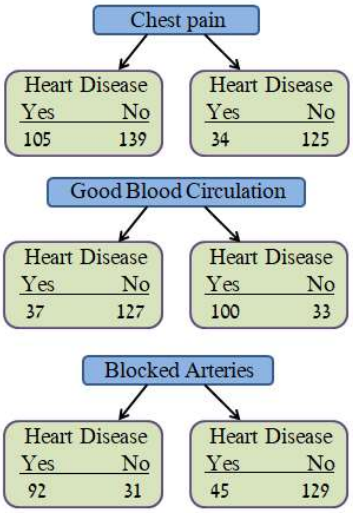
יד. לבסוף, נחזור לעץ המקורי (T_0) וה- subtrees שנבנו מה- data set המלא (שמכיל את training+test data), ונבחר את העץ שתואם לערך ה- α הנבחר. ה- subtree הזה יהיה "העץ הגזום הסופי" שלנו.

לסיכום, מה בעצם קרה כאן?

- אחרי כל החישובים מצאנו שהעץ T_0 הוא בעל ה- RSS הנמוך ביותר, אך חשוד להתגלות בהמשך כ- overfit. ולכן השתמשנו ברכיב ש- "מפצה" את יתר העצים שהינם בעלי RSS גבוה יותר, ו- "מעניש" את העץ המקורי (T_0) בגלל ריבוי העלים שבו.
- התקבל ציון שמאפשר לנו להשוות בין העצים ולבחור את העץ הטוב ביותר. העץ הזה מהווה עבורנו מעין איזון בין הרצון ל- RSS מינימלי לבין שונות נמוכה בין ה- train ל-test .
- ביצענו *cross-validation* כדי למצוא מהו ערך ה- α שנותן לנו עץ בעל RSS אופטימלי שלא עושה (או יעשה כמה שפחות) overfitting.

2.2.עץ קלסיפיקציה (סיווג)

עץ קלסיפיקציה די דומה לעץ רגרסיה, רק שעץ קלסיפיקציה משמש לחיזוי מידע איכותי, ולא מידע כמותי. נזכיר שבעץ רגרסיה שתיארנו קודם, החיזוי שניתן לתצפית מסוימת – ניתן לה ע"י הערך הממוצע של תצפיות האימון ששייכות לאותו terminal node. לעומת זאת, בעץ קלסיפיקציה מסווגים כל תצפית לקבוצה (class) מסוימת. למשל, נניח ואנו מעוניינים לסווג מטופל מסוים האם יש לו מחלת לב או לא. אנו יכולים לבנות עץ החלטה על בסיס מאפיינים של חולים שאובחנו בעבר ואנו יודעים להגיד מי מהם באמת חולה לב ומי לא, ועל בסיס העץ הזה להחליט עבור כל מטופל חדש האם הוא דומה במאפיינים שלו למטופלים שאובחנו בעבר כחולי לב או לא. כך שהתשובה שהעץ נותן היא לא "ערך ממוצע" כמו שראינו בעצי רגרסיה, אלא פשוט החלטה - "כן חולה לב" או "לא חולה לב". המיוחד בעצי קלסיפיקציה הוא שלעתים קרובות אנו מעוניינים לא רק בחיזוי ה-class המתאים (חולה/לא חולה), אלא גם בפרופורציות של ה- class-ים השונים בקרב תצפיות האימון שנופלים באותו אזור.



באיור לעיל אנו רואים 3 משתנים (שבמקרה הזה הם סימפטומים של מטופל) שאמורים לסווג האם למטופל יש מחלת לב או לא.

כפי שניתן לראות אף אחד מהמשתנים אינו מצליח לסווג **לבד** בצורה מושלמת האם למטופל יש מחלת לב או לא. ניתן לראות זאת בכך שבאף אחד מהעלים אין 100% מחלת לב, וגם לא 100% **לא מחלת לב.** לכן, כל המשתנים האלה נחשבים כלא הומוגניים (impure - לא נקיים). אך בכל זאת אנו נדרשים לבחור באחד מהמשתנים להיות המשתנה שמסווג ראשון את העץ (root node), ולכן נצטרך למצוא דרך למדוד ולהשוות את רמת ה- impurity של כל משתנה.

<u>יש כמה מדדים לחוסר הומוגניות שמודדים את רמת ה- impurity בעצי קלסיפיקציה:</u>

- Gini index
- Entropy

---

1. <u>Gini index:</u>

$$Gini = 1 - \sum_j p_j^2$$

כאשר p_j הוא ההסתברות ל- j class

כלומר:

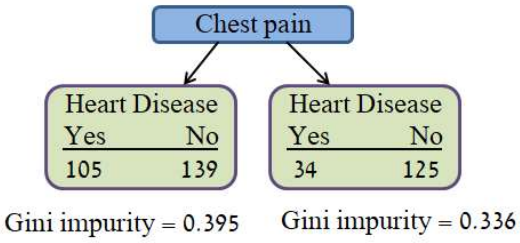
1 – (the probability of "yes")² – (the probability of "no")²

המשמעות האינטואיטיבית של מדד ג'יני היא: אם נבחר תצפית רנדומלית מהדאטה שלנו, ונסווג אותה לפי הפרופורציות של כל class בדאטה, מה הסיכוי שסיווגנו את אותה תצפית באופן שגוי?

נקח למשל את המשתנה Chest Pain, ונחשב את מדד הג'יני כדי להחליט האם להשתמש בו כמשתנה המפצל הראשון, נקבל ב- עלה השמאלי:

$$1-\left(\frac{105}{105+3}\right)^2-\left(\frac{39}{105+39}\right)^2=0.395$$

באותו אופן נחשב את העלה הימני ונקבל:

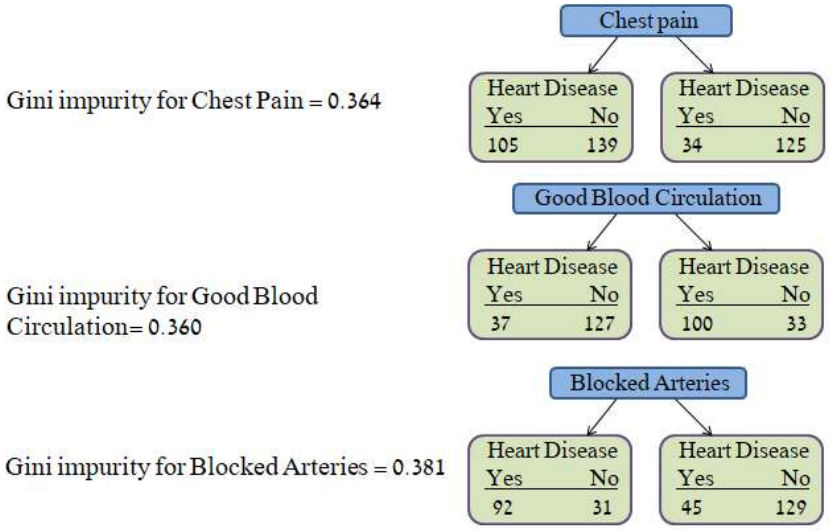


אחרי שחישבנו את ה- gini impurity לשני העלים, נחשב את מדד ג'יני הכולל של כל המשתנה chest pain.

אבל יש לנו בעיה. מאחר והעלה השמאלי מייצג 144 מטופלים והעלה הימני מייצג 159 מטופלים, שני העלים ביחד אינם מייצגים את אותה כמות מטופלים, ולכן כדי לחשב את מדד ג'יני למשתנה הזה, נצטרך לחשב **ממוצע משוקלל של חוסר ההומוגניות בשני העלים.**

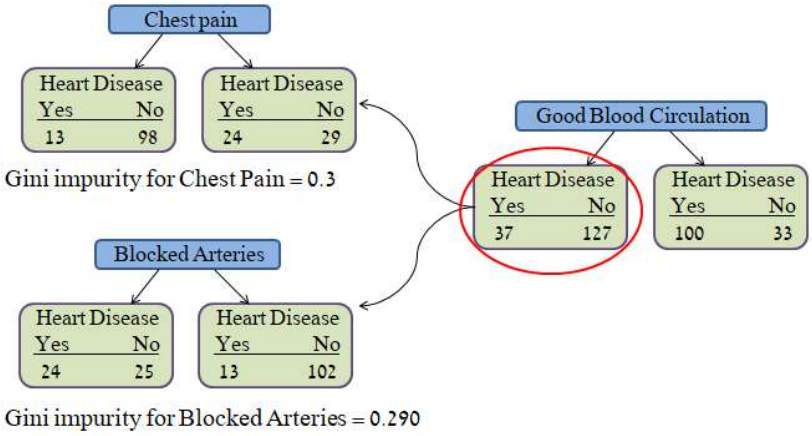
$$\left(\frac{144}{144+159}\right)0.395+\left(\frac{159}{144+1}\right)0.336=0.364$$

נחשב באותו אופן גם ביתר המשתנים:

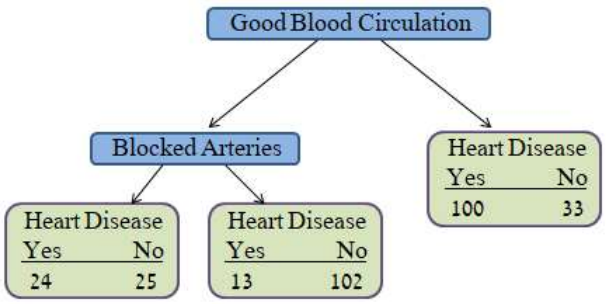


המשתנה Good Blood Circulation יש את הציון הכי נמוך, מה שאומר שהוא מסווג הכי טוב את המטופלים עם ובלי מחלת לב. לכן נשתמש בו כמשתנה המסווג הראשון בראש העץ (the root).

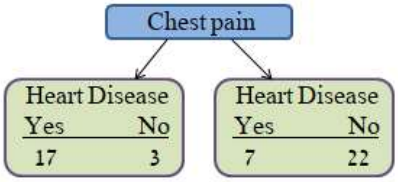
כעת נצטרך לפצל את שני העלים שקיבלנו ע"פ יתר המשתנים. כדי להחליט מי יהיה המשתנה **הבא** שיפצל את העלה השמאלי ומי יהיה המשתנה שיפצל את העלה הימני, נעשה את אותו תהליך כמו קודם.



כמו שרואים באיור למעלה, ל- Blocked Arteries יש ציון ג'יני נמוך יותר ולכן הוא זה שיפצל הבא את הענף השמאלי. וכעת העץ שלנו נראה ככה:

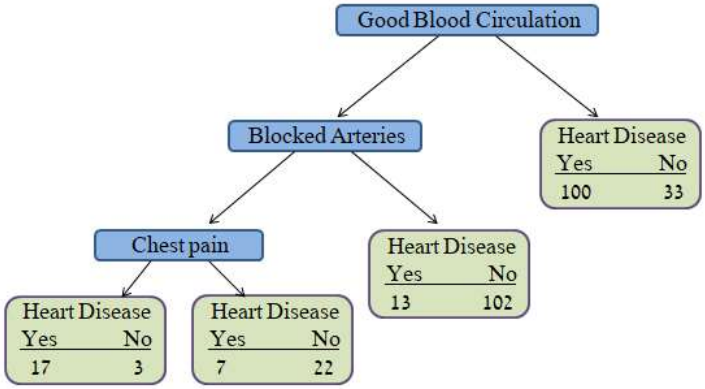


המשתנה שנשאר לנו הוא Chest Pain, אז כעת נבחן איך הוא מסווג את 49 המטופלים שבעלה השמאלי (24/25):

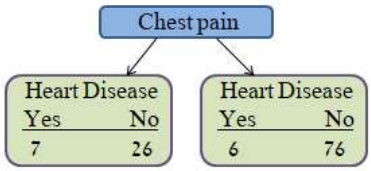


המשתנה Chest Pain עושה עבודה די טובה בסיווג המטופלים.

וכעת העץ נראה ככה:



בנקודה זו אנו מתלבטים בשאלה האם לפצל את העלה 13/102 או לא. ננסה לפצל אותו לפי Chest Pain ולחשב את מדד ג'יני לאחר הפיצול:

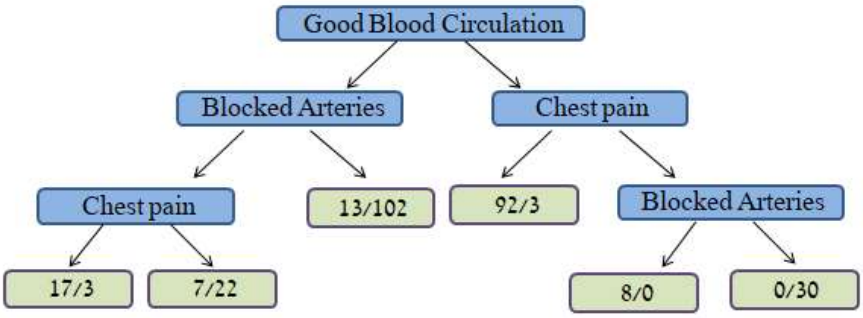


Gini impurity for Chest Pain = 0.29

מדד ג'יני שהתקבל הוא 0.29, בעוד שאם לא היינו מבצעים את הפיצול, מדד הג'יני של העלה 13/102 הוא 0.2, ולכן במקרה הזה עדיף להשאיר אותו כפי שהיה לפני ניסיון הפיצול.

סיימנו לחלק את כל החלק השמאלי של העץ. כעת נחלק את החלק הימני בדיוק באותו האופן:

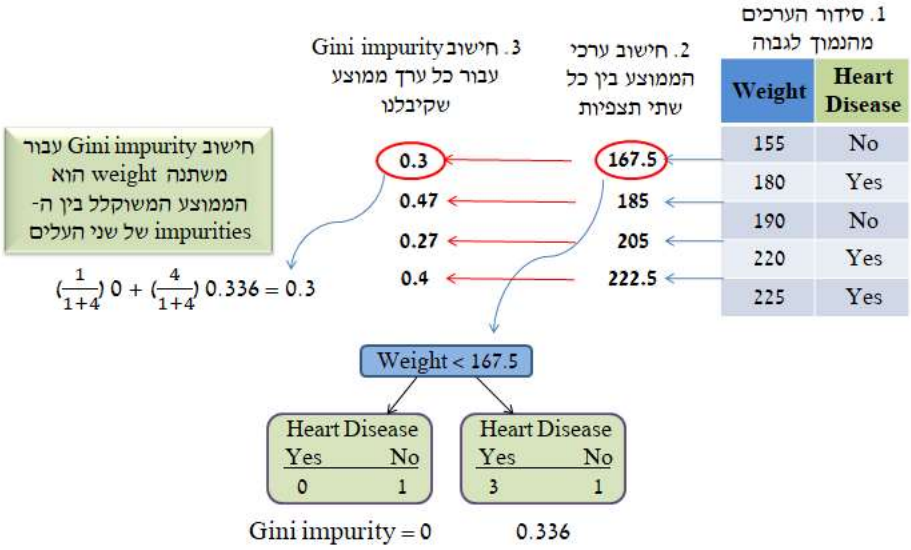
1. נחשב את מדדי ג'יני
2. אם לענף הקיים יש ציון ג'יני נמוך יותר, אז אין טעם לפצל עוד, והוא הופך להיות terminal node.
3. אם פיצול הענף הקיים מביא לשיפור, בוחרים את המשתנה המפצל בעל ציון ג'יני הנמוך ביותר.



עד כאן דיברנו על איך מחלקים משתנים עם שאלות של "כן/לא". אבל מה קורה כשיש לנו משתנים רציפים כמו "משקל"? איך נחליט באיזה נקודה לפצל את המשקל?

כמה שלבים:

1. לסדר את ערכי המשתנה מהערך הנמוך ביותר לערך הגבוה ביותר
2. לחשב את הערך הממוצע בין כל 2 תצפיות
3. לחשב את gini impurity עבור כל ערך ממוצע שקיבלנו בשלב 2



אנחנו מקבלים את הג'יני הנמוך ביותר מתי שאנחנו קובעים Tresh-Hold של weight<205.

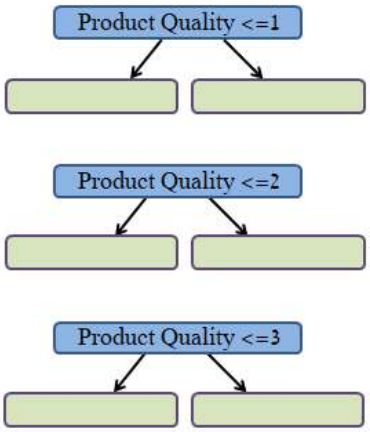
אז זהו ה- cutoff וערך הג'יני שנשתמש בהם כשאנחנו משווים את המשתנה weight ליתר המשתנים.

עד עכשיו דיברנו על איך מחלקים משתנה רציף ואיך מחלקים משתנה בינארי (שאלות כן/לא). כעת נדבר איך מחלקים משתנים **קטגוריאליים**. למשל: משתנה מדורג שמקבל ערכים מ-1 עד 4. למשל: טיב מוצר מ-1-4. או משתנה שמכיל מספר קטגוריות. למשל: צבע מועדף (מכיל ערכים: אדום, ירוק, כחול וכו').

משתנה מדורג מאוד דומה למשתנה רציף, חוץ מזה שאנחנו צריכים לחשב בו את הציון עבור כל חלוקה אפשרית.

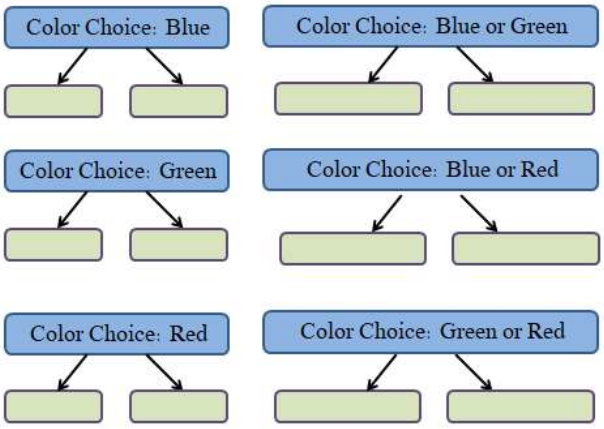
Product Quality	Satisfied
1	No
1	Yes
3	No
1	Yes
וכו'...	וכו'...

שימו לב: אנחנו לא צריכים לחשב את ה-impurity score ל- prod. quality<=4 בגלל שזה יכלול בעצם את כולם



כשיש משתנה קטגוריאלי עם כמה קטגוריות, אפשר לחשב את ציון הג'יני עבור כל קטגוריה, כמו גם עבור כל קומבינציה אפשרית:

Color Choice	Satisfied
Blue	No
Green	Yes
Red	No
Green	Yes
וכו'...	וכו'...



2. Entropy & Information Gain:

גם פה המטרה היא לבחור את המשתנה שמפצל הכי טוב את הנתונים ל- class-ים השונים. ובמינוח מדויק יותר: להוציא את ה- information gain המקסימלי מכל פיצול.

אנטרופי הוא מדד שמצביע על השגיאה של התפלגות המשתנה הנבחן מול משתנה המטרה.

והוא אומר כך: אם ישנן n תוצאות אפשריות, ולכל אחת מהן יש הסתברות p_i, אז מדד ה- entropy מוגדר כ-

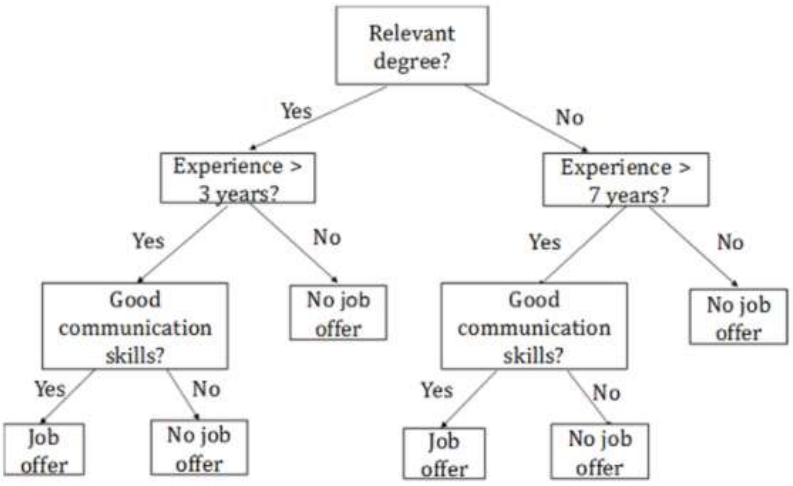
$$Entropy = -\sum_j p_j \log_2 p_j$$

בדומה למדד ג'יני הפיצול האופטימלי נבחר ע"י המשתנה שלו מדד האנטרופי הנמוך ביותר. אם כל התצפיות בעלה מסוים משויכות לאותו class, אזי מדד אנטרופי יהיה 0. מאידך, כאשר בעלה מסוים יש התפלגות שווה בין 2 ה- class של המשתנה המוסבר, מדד אנטרופי יהיה מקסימלי ושווה ל- 1.

מקודם פירטנו שלב אחרי שלב את חישוב מדד ג'יני ל- Use-Case של חולי לב. כעת כשהבנו את העיקרון של impurity, ניקח דוגמא אחרת פשוטה יותר הנוגעת לסיווג מועמדים לתפקיד מסוים לשתי קטגוריות:

1) כאלו שצריכים לקבל הצעת עבודה (job offer)

2) כאלו שצריכים לומר להם "תודה אבל לא תודה" (no job offer)



הדוגמא מתארת את אחד המאפיינים העיקריים של עצי החלטה. ההחלטה מתקבלת על ידי הבאה בחשבון של המאפיינים - אחד בכל פעם במקום כולם בבת אחד. תחילה, נלקח בחשבון המאפיין החשוב ביותר (תואר רלוונטי במקרה שלפנינו), לאחר מכן נלקחים שנות הניסיון, וכך הלאה.

נניח שהסיכוי בקרב כלל העובדים להתקבל לעבודה הוא 20%, והסיכוי לא להתקבל הוא 80%.

אזי:

$Entropy = -[0.2 \ln(0.2) + 0.8 \ln(0.8)] = 0.5004$

עוד נניח:

- 30% מאלו שיש להם תואר רלוונטי מקבלים הצעת עבודה
- 10% מאלה שאין להם תואר רלוונטי מקבלים הצעת עבודה

- אם למועמד מסוים **יש** תואר רלוונטי, מדד האנטרופי הופך ל-

$Entropy = -[0.3 \ln(0.3) + 0.7 \ln(0.7)] = 0.6109$

- אם למועמד מסוים **אין** תואר רלוונטי, מדד האנטרופי הופך ל-

$$Entropy = -[0.1\ln(0.1) + 0.9\ln(0.9)] = 0.3251$$

נניח ול- 50% מהמועמדים יש תואר רלוונטי, הרי שתוחלת האנטרופי לאחר שנקבע האם למועמד מסוים יש תואר רלוונטי היא:

$$0.5 \times 0.6109 + 0.5 \times 0.3251 = 0.4680$$

מדד ה- impurity שמתקבל מהידיעה האם למועמד מסוים יש תואר רלוונטי או לא, הוא תוחלת הירידה באי הוודאות. אם אי הוודאות נמדדת באמצעות מדד האנטרופי, הרי שהרווח מהמידע הוא:

$$Information\ gain = 0.5004 - 0.4680 = 0.0324$$

הן עבור מועמדים בעלי תואר והן עבור כאלה ללא תואר, המאפיין שממקסם את הרווח מהמידע הצפוי (ירידה באנטרופי הצפוי) הוא מספר שנות הניסיון העסקי. כאשר למועמד יש תואר רלוונטי, רמת הסף עבור מאפיין זה הממקסמת את הרווח מהמידע הצפוי היא 3 שנים. ברמה השנייה של העץ, "יש תואר רלוונטי" מתפצלת ל- 2 ענפים: "ניסיון > 3" ו- "ניסיון ≤ 3". עבור הענף התואם למועמד שאין לו תואר רלוונטי רמת הסף הממקסמת את הרווח מהמידע הצפוי היא 7 שנים. לפיכך, שני הענפים הבאים הם: "ניסיון > 7" ו- "ניסיון ≤ 7". אנו עושים שימוש באותה הפרוצדורה לבניית יתר העץ.

Misclassification rate

אז אחרי שבנינו עץ קלסיפיקציה ועמדנו בכל שלב על הדרך למצוא את הפיצול הטוב ביותר. צריך לבדוק את רמת הדיוק של העץ שלנו על דאטה חדש. בעץ רגרסיה מדדנו את RSS. בבעיות סיווג מקובל למדוד את Misclassification rate. הוא מודד את הפרופורציה בין כמות התצפיות שהמודל סיווג באופן שגוי, מתוך כלל התצפיות.

פונקציית ה- loss היא:

$$l(\hat{y}, y) = I\{\hat{y} \neq y\}$$

פונקציית loss זו נקראת פונקציית ה zero-one loss. תחת פונקציה זו חיזויים נכונים יקבלו ציון 0 ושגיאות יקבלו ציון 1 (לא תלות בגודל השגיאה). פונקציית ה misclassification rate נראית כך:

$$R(h) = \mathbb{E}\left[I\{h(\mathbf{x}) \neq \mathrm{y}\}\right]$$

סיכום

עץ החלטה (Decision Tree) הינו אלגוריתם לסיווג או לחיזוי ערכו של משתנה, כאשר המאפיינים מסודרים לפי סדר החשיבות. לצורך סיווג, קיימים שני מדדים אלטרנטיביים לאי וודאות: מדד אנטרופי (Entropy) ומדד ג'יני (Gini). כאשר ערכו של משתנה מסוים נחזה, אי הוודאות נמדדת באמצעות RSS. חשיבותו של מאפיין הינו הרווח מהמידע הצפוי שלו (Expected Information Gain) . הרווח מהמידע הצפוי נמדד על ידי הירידה באי הוודאות הצפויה אשר תתרחש כאשר יתקבל מידע אודות המאפיין.

במקרה של חלוקת **משתנה קטגוריאלי**, המידע המתקבל הינו על פי רוב אודות קטגוריה (Label) של המשתנה למשל: צבע מועדף (מכיל ערכים: אדום, ירוק, כחול וכו'). במקרה של **משתנה רציף** יש לקבוע ערך סף (Threshold)

אחד (או יותר) המגדיר שני טווחים (או יותר) עבור ערכי המשתנה. ערכי סף אלו נקבעים באופן שממקסם את ה-information gain הצפוי.

אלגוריתם עץ ההחלטה קובע תחילה את צומת השורש (Root Node) האופטימלי של העץ באמצעות קריטריון "מקסום הרווח מהמידע" שהוגדר לעיל. לאחר מכן הוא ממשיך לעשות אותו הדבר עבור הצמתים העוקבים. הקצוות של הענפים הסופיים של העץ מכונים צמתי עלים (Leaf Nodes או Terminal Nodes).

כאשר עץ ההחלטה משמש לסיווג, או אז צמתי העלים כוללים בחובם את ההסתברויות של כל אחת מהקטגוריות להיות הקטגוריה הנכונה. כאשר עץ ההחלטה משמש לחיזוי ערך נומרי, או אז צמתי העלים מספקים את ערך התוחלת של היעד. הגיאומטריה של העץ נקבעת באמצעות סט האימון (Training Set), אך הסטטיסטיקה שעוסקת ברמת הדיוק של העץ צריכה כמו תמיד בלמידת מכונה לבוא מתוך סט הבדיקה (Test Set) ולא מתוך סט האימון .

אחרית דבר:

מדדי RSS ומדד misclassification rate שהוצגו בפרק זה הינם רק חלק מהמדדים המקובלים. לכל מדד יתרונות וחסרונות, והמדד הרלוונטי יבחר ע"פ סוג הנתונים והבעיה העסקית.

יתרונות:

- בהשוואה לאלגוריתמים אחרים, עצי החלטה דורשים פחות השקעה בתהליך הכנת הנתונים (pre-processing).
- עץ החלטה לא דורש נרמול של הדאטה.
- ערכים חסרים בדאטה לא משפיעים על תהליך בניית העץ.
- עץ החלטה תואם לאופן שבו מרבית בני האדם חושבים על בעיה מסוימת והוא פשוט להסבר למי שאינם מומחים.
- אין שום דרישה שהקשר בין המשתנה המוסבר והמשתנים המסבירים יהיה לינארי.
- העץ בוחר אוטומטית במשתנים הטובים ביותר על מנת לבצע את החיזוי.
- עץ החלטה רגיש פחות לתצפיות חריגות מאשר רגרסיה.

חסרונות:

- שינוי קטן בנתונים יכול לגרום לשינוי גדול במבנה עץ ההחלטה ולגרום לחוסר יציבות.
- לפעמים החישוב בעצי החלטה יכול להיות מורכב מאוד ביחס לאלגוריתמים אחרים.
- עצי החלטה לעיתים תכופות דורשים יותר זמן הרצה לאימון המודל, ומשכך מדובר באלגוריתם "יקר" במשאבים.
- האלגוריתם של עץ ההחלטה אינו מספיק ליישום רגרסיה ולניבוי ערכים רציפים.
- עץ החלטה נוטה לעיתים תכופות לעשות overfitting.