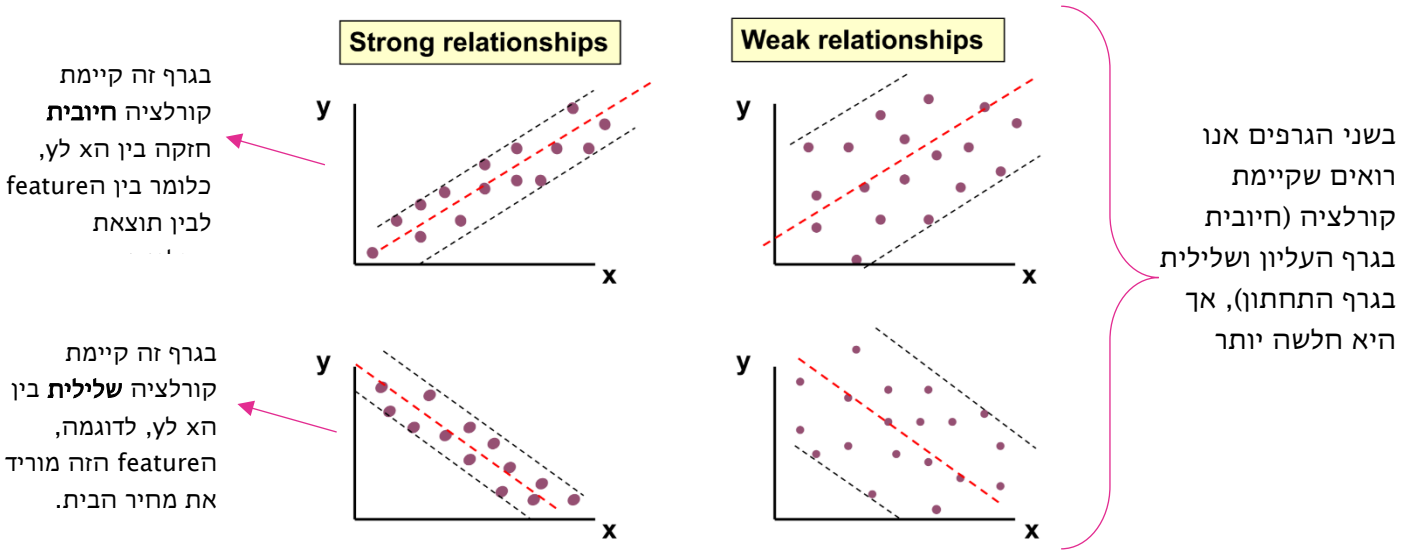
למידה חישובית - שיעור 2

Correlation

אנו מעוניינים למדוד כמה feature כלשהו משפיע על התוצאה של האלגוריתם שלנו. הקורלציה היא היחס בין שני המשתנים (x ו-y בשרטוט שמולנו). המשתנים יכולים להיות קשורים אחד לשני, או לא. בקונטקסט של רגרסיה לינארית- אנו מדברים על הקשר בין הfeature לבין התוצאה של האלגוריתם הלומד.

בציר ה-x נשים את המשתנה ה"מסביר", לדוגמה גודל מ"ר של בית.

בציר ה-y נשים את המשתנה ה"מוסבר", לדוגמה מחיר הבית.



הערה: חשוב לשים לב שלא בהכרח יהיה קשר בין משתנים, או שיהיה קשר שאינו לינארית.

אם הקשר בין המשתנים אינו לינארי, אז הקורלציה לא תהיה מושלמת.

Pearson Correlation

פונקציה המודדת מתי היחס בין 2 משתנים הוא לינארי:

$$r = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{m}(y_i - \bar{y})^2}} = \frac{\partial_{xy}}{\partial_x \partial_y}$$

r = Sample Pearson correlation coefficient
m = Number of samples
x = Value of an explaining variable
y = Value of the dependent variable

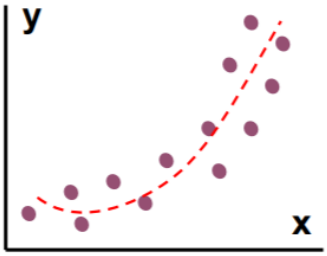
r נקרא מקדם הקורלציה., יכול לקבל ערכים בין מינוס 1 ל-1.

נשים לב שאם ערכי x וגם ערכי y מעל הממוצע או מתחת לממוצע (מעל או מתחת ל$(\bar{x},\bar{y})$, המונה של הביטוי יהיה חיובי ובגלל שהמכנה תמיד חיובי נקבל r חיובי.

ככל ש r יתקרב ל-0, נקבל יחס לינארי חלש יותר.
ככל ש r יתקרב למינוס 1, נקבל יחס לינארי חזק שלילי.
ככל ש r יתקרב ל-0, נקבל יחס לינארי חזק חיובי.

רגרסיה פולינומיאלית

לא נוכל להסביר כל data בצורה לינארית, ופעמים רבות נוכל להסביר את הdata בצורה פולינומיאלית.



בדוגמה זו נוכל להעביר קו לינארי, אבל אם נעשה זאת לא נסביר את הdata בצורה המיטבית.

על מנת להסביר את הdata בצורה המיטבית נשתמש ברגרסיה לינארית (במקרה זה בפונקציה ממעלה שניה).

נוכל להשתמש במנגנון שלמדנו ברגרסיה לינארית, על מנת לייצג מידע ממעלה שניה. אם ברגרסיה לינארית יש לfeature עמודה אחת שמייצגת מעלה אחת, נוסיף עמודה שתייצר את הfeature בריבוע ונפתור את הבעיה כפי שפתרנו רגרסיה לינארית ובכך יצרנו רגרסיה לפולינום ממעלה 2.

נוכל לבצע את התהליך עבור כל מעלה שהיא.

<u>הערה:</u> נלמד בהמשך הקורס שככל שנעלה את מעלת הפולינום, הסיכוי שנמצא את מה שאנו מחפשים גבוה יותר.

נציג כעת טכניקה נוספת לפתרון בעיית הרגרסיה הלינארית, ללא Gradient Decent. בשונה מ Gradient Decent, הטכניקה הזו לא בהכרח תעבור על כל פונקציה.

$$\mathbf{X}\cdot\vec{\boldsymbol{\theta}} = \vec{\mathbf{Y}}$$

$$\begin{pmatrix} x_0^{(1)} & x_1^{(1)} & \dots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ x_0^{(m)} & x_1^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \\ \\ y_m \end{pmatrix}$$

$$\bar{x}\theta - y : \mathbb{R}^n \to \mathbb{R}^m$$

<u>הערה:</u> x וy קבועים וθ הוא המשתנה שלנו. כאשר אנו מכפילים את $\bar{x}$ ב-θ אנו בעצם מעבירים את θ ממימד n למימד m (גדול יותר).

$$Error(\theta) = J(\theta) = ||\bar{x}\theta - y||^2 : \mathbb{R}^n \to \mathbb{R}^+$$

בד"כ לא נוכל להביא את הטעות להיות בדיוק 0 כי אנו מעבירים מרחב קטן למרחב גדול.
בנוסף לרוב יש רעש בניסוי אז גם כשהתופעה לינארית, כשנמדוד את התופעה לא נקבל לינאריות טהורה, ולכן נחפש קירוב. לכן נמדוד את הקירוב ע"י נורמה בריבוע.

$$||v||^2 = \sum_{i=1}^{n} {v_i}^2$$

אנו רוצים להביא את הטעות $J(\theta)$ (שהיא הנורמה בריבוע) למינימום. בפונקציה עם מספר משתנים נביא למינימום על ידי השוואת הגרדיאנט ל-0:

$$\nabla J(\theta) = \left(\frac{\partial J}{\partial \theta_j}(\theta)\right)_{j=1}^{n}$$

אנו רוצים לדעת את הטעות כפונקציה של כל קבוצת משקולות שאני שמה עליהם. נחשב כל אחת מהנגזרות החלקיות

$$\frac{\partial J}{\partial \theta_j}(\theta) = \frac{\partial}{\partial \theta_j}\left(\sum_{i=1}^{m}\left(\underbrace{x\ (i,\ :)\cdot\theta}_{\text{מכפלה של השורה i עם הוקטור } \theta} - y^{(i)}\right)^2\right)$$

$$\frac{\partial}{\partial \theta_j}\sum_{i=1}^{m}\left(\sum_{k=1}^{n} x(i,k)\theta_k - y^{(i)}\right)^2 \xrightarrow[\theta_j \text{ נגזור לפי}]{} \sum_{i=1}^{m} 2\cdot\left[\sum_{k=1}^{n}\left(x(i,k)\theta_k - y^{(i)}\right)\right]x(i,j)$$

$$2\cdot\sum_{i=1}^{m}\underbrace{x(i,j)}_{\text{עמודה}}\left(\underbrace{x(i,:)\cdot\theta - y^{(i)}}_{\text{וקטור ההפרש}}\right) \xrightarrow[\text{מכפלה פנימית}]{} 2\cdot x(:,j)\cdot(x\theta - y)$$

קיבלנו את הנגזרת החלקית של J (הטעות) לפי θ_j. כעת נשווה ל-0 כדי לקבל מינימום:

$$2\cdot x(:,j)\cdot(x\theta - y) = 0 \quad\to\quad x^T\cdot(x\theta - y) = 0 \quad\to\quad x^T\cdot x\theta - x^Ty = 0$$

$$x^Tx\theta = x^Ty \ /\cdot(x^Tx)^{-1} \quad\to\quad \underbrace{(x^Tx)^{-1}(x^Tx)}_{I}\theta = (x^Tx)^{-1}x^Ty \quad\to\quad \theta = (x^Tx)^{-1}x^Ty$$

הפונקציה: $(x^Tx)^{-1}x^Ty$ שהתקבלה נקראת $pinv$ (קיצור של pseudo inverse), שנותנת את הקירוב הטוב ביותר ל-x ההפיכה.

הראנו את הpseudo inverse שפותר את בעיית הרגרסיה הלינארית, אבל רק אותה. אם ניתן לפונקציית $pinv$ רגרסיה פולינומיאלית (בהנחה ולא נעשה לבעיה הפולינומיאלית התאמות כלשהן) ולכן אנו מסיקים שהיתרון הגדול של Gradient Decent הוא שהוא עובד בכל שאלת אופטימיזציה.

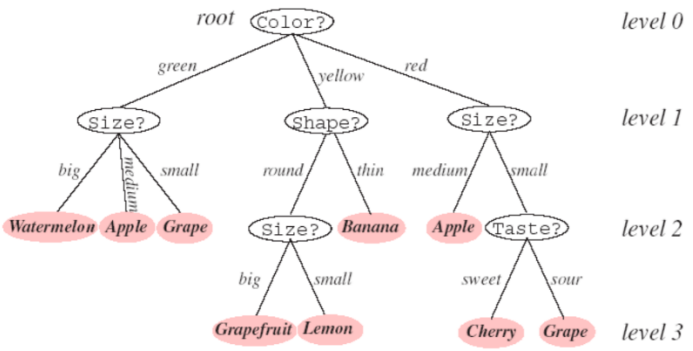
עצי החלטה

עד כה דיברנו על קיום מפריד לינארי, אך לא דיברנו על הדרך ללמוד מפריד הלינארי.

עצי החלטה היא דרך למיין דברים לפי feature שלהם. זהו עץ עם שאלות כך שכל instance שעובר בעץ יישאל שאלה לגבי feature מסוים של הdata. תהליך הסיווג מתבצע כך שכל instance מתחיל משורש העץ ומתפזר לאורך העץ בהתאם לתשובות לשאלות. בכל צומת בעץ מתבצעת החלטה (ומכאן שמו) בנוגע לסיווג הinstance הנוכחי.

תהליך הסיווג נפסק כאשר הinstance הפך להיות עלה. נסיק מכך ש**במקרה הגרוע** כל עלה בעץ הוא instance, לכן מספר העלים שווה למספרים הinstanceים בdata. אנו כמובן יכולים להגיע למצב בו יש מספר instanceים בכל עלה ולכן זה חסם עליון על מספר העלים בעץ. ברוב המוחלט של המקרים לא נגיע אליו.

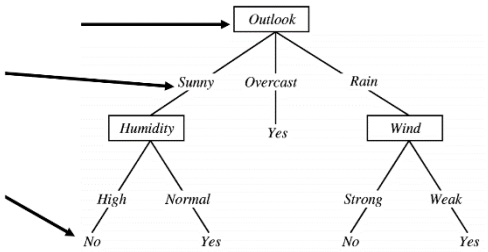
בדוגמה שמולנו, instance של לימון יתחיל בראשית העץ. הfeature הראשון עליו הוא ישאל יהיה ה-color. בהתאם לתשובתו הוא ירד במורד העץ (ימינה או שמאלה). בהמשך ישאל על הfeature של ה-shape ולבסוף על הfeature של הsize.



בתום תשאול כל הinstanceים נקבל עץ בינארי.

<u>דוגמה נוספת ופופולרית של עצי החלטה:</u>

נרצה לדעת האם אדם ישחק טניס או לא.



נבדוק מה האפשרות למשחק טניס בכל מזג אוויר. נתקדם בעץ עד אשר נגיע לעלה. בשלב זה נדע האם התשובה היא yes\no.

<u>הערה</u>: ניתן להפוך כל עץ החלטה שאינו בינארי לעץ החלטה בינארי.

האלגוריתם שיבנה את עץ ההחלטה הוא **אלגוריתם הלמידה**. העץ עצמו הוא אלגוריתם הביצוע.

נשים ♥ שאם התשובה לשאלתנו היא מספרים רציפים, נבחר חתכים לערכים אלה ונעבוד לפיהם.

לדוגמה אם נרצה לחשב הסתברות להתקף לב, ואחד מהפיצ'רים שמעניינים אותנו הוא גיל, נמיר את המספר הרציף שמייצג את הגיל לערך רציף כלשהו שיכול להוות תשובה בינארית, לדוגמה: במקום גיל 55 או 65 נשאלה האם גדול מ65.

על ידי תהליך הקטגוריזציה אנו מאבדים את הדיוק ואת המדרג. דוגמה טובה לאיבוד המדרג הינה סיווג צבעים לפי מדרג כלשהו. ירוק הוא כמובן לא יותר גדול או קטן מאדום ולכן אנו מאבדים משהו מהמדרג של המשתנה.

אנו כמובן יכולים להפוך משתנה קטגוריאלי למשתנה רציף, אבל בפעולה זו אנו מייצרים מדרג שאינו קיים.

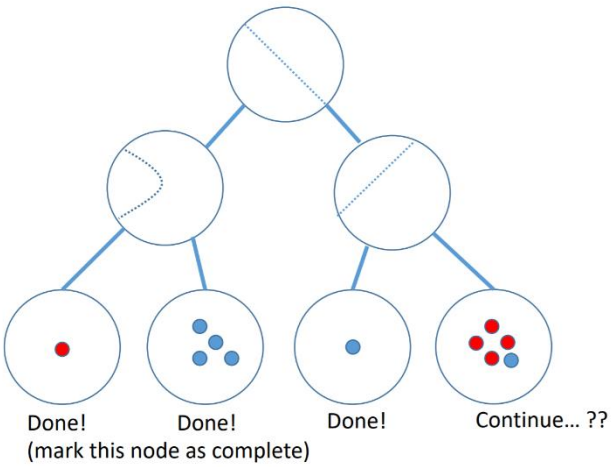
מתי נשתמש בעצי החלטה:

1. כאשר הנתונים שלנו בtraining data הם כאלה שעונים על attributeים מסוימים
2. כאשר פונקציית המטרה שלנו אינה רציפה
3. כאשר נרצה שלהיפותזה שלנו יהיה מבנה של עץ
4. כאשר הdata שיש ברשותנו שאנו רוצים ללמוד אינו מסודר

אלגוריתם ליצירת עץ החלטה - מימוש באמצעות תור

1) ניצור שורש שמכיל את כל ה-training data שאותו אנו מעוניינים ללמוד.
2) נכניס את השורש לתור
3) נוציא את הnode הבא מהתור.
   a. אם הnode מסווג עם סיווג 1 (perfectly classified), סמן את הnode שסיימנו את עבודתנו עמו וחזור לשלב (3), כלומר נוציא את הnode הבא מהתור.
   b. אם הnode לא מסווג 1, נשים ב-A את ה-attribute שמחלק את המידע בnode הנוכחי בצורה הטובה ביותר, ונגדיר שה- node הנוכחי מחולק על ידי A.
   c. ניצור בנים ל-n ככמות החלוקות ש-A משרה ונעביר את כל ה-instances לבנים
   d. נכניס את כל הבנים לתור.
4) אם התור לא ריק, חזור לשלב (3)

איך נדע מתי להפסיק?



ייתכן שנגיע למצב בו הattribute שלנו לא מספיקים על מנת לסווג את העץ בצורה מושלמת, או שאנו לא מעוניינים בסיווג מושלם.

הבחירה האם ומתי להפסיק לסווג את הinstanceים בעץ תלויה בdata שלנו ובהחלטה שלנו לגבי רמת הדיוק. לא נתעמק בנושא בקורס.

עולה השאלה איך נבחר *attribute* (תכונה).

אנו נרצה אלגוריתם שיחליט עבורנו בכל שלב (כלומר באופן לוקלי) באיזה attribute להשתמש על מנת לפצל את הענף הנוכחי, וכן היכן לשים את הסף לפיצול. אנו כמובן מעוניינים לבחור ב-attributeים שישרו לנו חלוקה טובה ככל שניתן.

נשתמש במדד אי וודאות כאשר 1 הוא אי וודאות גבוה ביותר, ו0 הוא ודאות מוחלטת. זוהי תכונה של צומת בעץ. אנו מעוניינים למצוא *attribute* כך שניתן יהיה להוריד את אי הודאות ככל שניתן.

Goodness of Split

אנו מעוניינים לדעת האם הattribute מחלק טוב את הdata. נעשה זאת על ידי שימוש במדד אי הוודאות. נרצה לדעת כמה 1 יש לעומת 0.

Impurity Criteria

פונקציה $\varphi: [0,1]^k \to \mathbb{R}$ המקיימת את התנאים הבאים להסתברויות הבאות $P = (p_1, \dots, p_k) \in [0,1]$

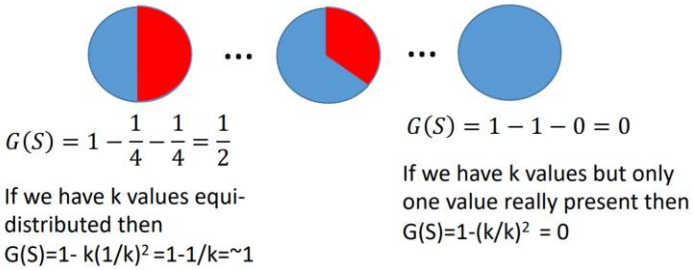
1) $\varphi(P) \geq 0$
2) הערך המינימלי מתקבל כאשר $\exists i \; s.t \; p_i = 1$
3) הערך המקסימלי יתקבל כאשר $1 \leq i \leq k$ $p_i = \frac{1}{k}$
4) סימטרית ביחס למרכיבים של P
5) גזירה אינסוף פעמים בתחום.

Gini Impurity

מדד ג'יני מודד כמה פעמים instance שנבחר באופן אקראי מהdata שלנו, יתוייג בצורה שגויה, אם תויג באופן אקראי.

$$G(S) \equiv 1 - \sum_{i=1}^{c} (p_i)^2 = 1 - \sum_{i=1}^{c} \left(\frac{|S_i|}{|S|}\right)^2$$

תחום הפונקציה הינו [0,1]. אם יתקבלו ערכים נמוכים, נסיק כי יש מעט פיזור. אם יתקבלו ערכים גבוהים נסיק כי יש יותר פיזור.



תהליך השימוש בGINI:

1. עבור כל node שעבר פיצול באמצעות attribute כלשהו, נמדוד את הGINI שהקבל טרם הפיצול - $G(S)$.
2. נמדוד את הGINI שהתקבל לאחר הפיצול- עבור כל הבנים של הnode שפיצלנו.
3. נבצע ממוצע משוקלל לערכים שקיבלנו ב-(2) - $\sum_{v \in Values(A)} \frac{|S_V|}{|S|} G(S_v)$.
4. נבדוק האם חל שיפור בimpurity בעקבות הפיצול באמצעות הattribute הנוכחי.
5. נחזור על התהליך עם כל הattributes.

בתום התהליך המתואר, נבחר את הattribute שהניב לנו את השינוי הגדול ביותר בimpurity.

**הוכחה במודל לכך שבפונקציית GINI יש תמיד שיפור.

<u>כלומר הגדרנו קריטריון לפיצול:</u>

$$GiniGain(S,A) = \Delta G(S,A) = G(S) - \sum_{v \in Values(A)} \frac{|S_V|}{|S|} G(S_v)$$

- A זה ה-attribute שבאמצעותו אנו מחלקים.
- v זה ערך אפשרי של ה-attribute.
- S זאת קבוצת ה-Instances בnode הנוכחי.

לא דיברנו על כך בשיעור, אבל נכיר את הGini Index:

מחשב פיזור. אם נקבל Gini Index נמוך, נדע שהפיזור נמוך גם כן, ולהפך:

$$G(S) = 1 - \sum_{i=1}^{c} p_i^2 = 1 - \sum_{i=1}^{c} \left(\frac{|S_i|}{|S|}\right)^2$$