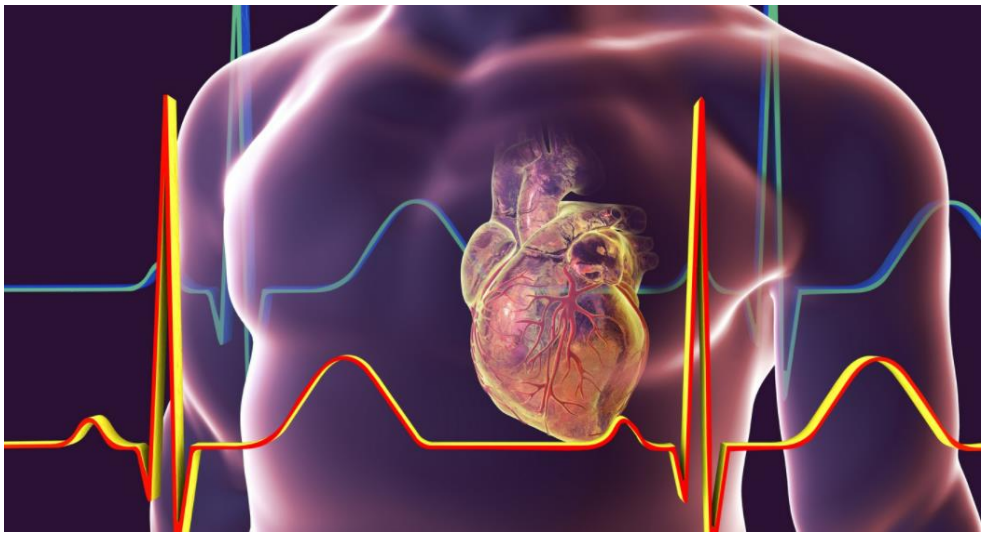
פרויקט סוף
קורס מבוא לבינה מלאכותית

Heart disease prediction

מרצה : ד"ר תמר שרוט

מגישה : ליהיא סבאג

Could our model save lives?



תוכן עניינים

הקדמה

מחלת לב (HD), הנקראת גם מחלת עורקים כליליים (CAD), גורמת לכשליש ממקרי המוות בקרב אנשים מעבר לגיל 35 במדינות המערב .HD מתרחש באשר זרימת הדם ללב מוגבלת עקב היצרות העורקים הכליליים, ולכן נגרם נזק לשריר הלב, המוביל לתוצאות חמורות כמו אוטם שריר הלב (התקף לב) , הפרעות קצב חדריות ואפילו מוות לבבי פתאומי. HD ניתן למניעה, ולכן אבחון מוקדם והתערבות נכונה הם שלבים קריטיים בהפחתת שיעור התמותה. ישנם מס כלים לאבחון פיזי של החשודים ב-HD ,רוב החולים החשודים נאלצים לעבור CT או CAG תוך סיכון של חשיפה מופרזת לקרינה, תגובה אלרגית לחומר ניגוד וכו' יתרה מכך, אבחון התוצאות דורש ניתוח יסודי של גורמים שונים מה שהופך את עבודתו של הרופא לקשה. כלי אבחון נוסף הוא מגנטו קרדיוגרפיה (MCG) המבטיח בהליך לא פולשני לגילוי מוקדם של HD. למרות איכות הפענוח המעולה שלו, הפרשנות של MCG היא אינטנסיבית לעבודה ומסתמכת מאוד על הניסיון של המתורגמן, מה שמגביל את הקבלה שלו במרפאות. לכן, מערכת אוטומטית שיכולה לזהות HD משלב מוקדם תהיה מועילה ביותר לרופאים.

סקירת התחומים לפתרון

הגדרת הבעיה שנבחרה

בהינתן מס מאפיינים רפואיים של אדם (תסמינים ,בדיקות מעבדה וכו') נרצה לומר בסבירות גבוה האם האדם חולה במחלת לב או לא.

הגישה לפתרון הבעיה

הבעיה הינה בעיית Classification (binary classification) המתארת גישה מפוקחת (Supervised) בלמידת מכונה כאשר האלגוריתם למד באמצעות מידע מתויג (Labelled) ואחרי כן הוא משתמש בתהליך הלמידה שהועבר על מנת לפתח מודל כך שבהינתן מידע חדש הוא ידע לסווג אותו (בהסתברות גבוהה) .

במקרה שלנו נאמן את האלגוריתם על סט גדול של מאפיינים בריאותיים אמינים של אנשים החולים במחלת לב וכאלה שלא וע"י ניתוח המאפיינים ובחירת משקלי חשיבות עבור כל תכונה נבנה מודל שיסווג את הדוגמאות החדשות לתגיות מסוג Binary labelling כאשר הסיווגים לבסוף יהיו {1:האדם חולה במחלת לב , 0: האדם לא חולה }.

מדוע הגישות שנבחרו הם אלו בהם בחרנו לפתרון הבעיה ?

מהגדרת הבעיה הבנתי שכדי לאבחן האם אדם חולה או לא אני צריכה למצוא דרך שבה המחשב יוכל ללמוד מנתונים רפואיים ולהבין האם הם מאפיינים אנשים החולים במחלת לב או האם הם מאפיינים אנשים בריאים ולכן בחרתי בגישה לפתרון הבעיה מתחום למידת המכונה:

תחום הבעיה - הבעיה שנבחרה הינה מתחום למידת מכונה מכיוון שלמידת מכונה היא תת תחום בבינה מלאכותית העוסק בפיתוח אלגוריתמים המיועדים לאפשר למחשב ללמוד מתוך דוגמאות , כאשר המטרה המרכזית של למידת המכונה היא טיפול ממוחשב בנתונים מן העולם האמיתי עבור בעיה מסוימת, למשל, בעיית זיהוי שמומחה אנושי מסוגל לפתור, אך לא מסוגל לכתוב את הכללים לזיהוי בצורה מפורשת, או שהם משתנים עם הזמן ולא ניתנים לכתיבה מראש ,כאשר מטרת הלמידה יכולה להיות מידול ,חיזוי או גילוי (דטקציה) של עובדות לגבי העולם האמיתי .

והבעיה שלנו היא בעיה מהעולם האמיתי המושפעת ממימד הזמן (דברים משתנים לאורך זמן וזה משפיע לנו על המערכת) והיא דורשת למידה ממצבים ישנים והתאמה למצבים חדשים כאשר הידע בסביבה משתנה ולכן תחומי בינה מלאכותית שמשיגים את המטרה ע"י פתירת הבעיה רק פעם אחת והסתכלות רק על המידע שבסביבה (כמו gaming,blind-search וכו') לא יהיו יעילים עבורנו לפתרון הבעיה שלנו ואילו שימוש באלגוריתמי למידת מכונה שיאפשרו למחשב ללמוד מתוך הדוגמאות יתנו לנו את הפתרון האופטימלי עבור הבעיה שלנו.

לאחר שהבנתי שאפתור את הבעיה בעזרת למידת מכונה , הייתי צריכה להבין מה תהיה גישת הלמידה הטובה ביותר לפתרון הבעיה , בלמידת מכונה ישנן 3 גישות ללמידה :

Supervised Learning למידה מפוקחת - טכניקת למידה זו מאפשרת לפתח מערכת שלומדת לפתור בעיות על בסיס מאגר גדול של דוגמאות "פתורות". הלמידה עצמה נעשית באמצעות חיפוש היפותזה - פונקציה ממרחב הדוגמאות למרחב התיוגים שמתארת את המידע בצורה הנכונה ביותר .

Unsupervised Learning למידה לא מפוקחת - טכניקת למידה שבה מנסים ללמוד את התכונות והמבנה של אוסף דוגמאות נתונים כאשר הנתונים זמינים כפי שהם ללא תוספת תיוגים. התובנות לגבי התכונות של הנתונים הבלתי מתויגים יכולות לשמש למשל כדי לזהות אנומליות או כדי לחלק את הנתונים לקטגוריות.
המטרה היא פשוט לבצע ניתוח אשכולות (cluster) לנתונים על מנת להעמיק את ההבנה שלנו על הסביבה.

Reinforced learning למידת חיזוק - טכניקת למידה שבה נוקטים פעולות בתוך סביבה כדי למקסם את הרווח המצטבר כתוצאה מהפעולות הללו. למידה בצורה זו שונה מן הלמידה המפוקחת בכך שלא נדרש קלט ופלט מתויג ופעולות שאינן אופטימליות לא מתוקנות באופן מפורש. במקום זאת, ההתמקדות היא במציאת איזון בין חקירה (של שטח לא נודע) לבין ניצול (של הידע הנוכחי).

כפי שניתן לראות הגישה לפתרון הבעיה שלנו היא מסוג למידה מפוקחת וזאת משום שפתרון הבעיה הוא ע"י למידה מתוך אוסף דוגמאות קיים שכבר מתויג ועל סמך דוגמאות אלו נבנה מודל שבהינתן קלט מאותו צורה ידע להחזיר פלט.
כלומר יש לנו מידע נתון מראש שניתן להשתמש בו על המידע החדש ולכן זה supervised learning.
למידת מפוקחת היא הגישה המתאימה ביותר לבעיה שלנו מכיוון שבעזרת מאגרי מידע מתוייגים נוכל "לפקח" על אלגוריתמי הסיווג כדי לחזות תוצאות בצורה מדויקת, למשל ע"י השוואת התוצאות של המערכת שלנו מול התוצאות האמיתיות, כלומר השימוש במאגר מתויג נותן לנו דרך להעריך את המערכת שלנו ולגרום לה להגיע לדיוק טוב יותר .
בנוסף אלגוריתמי למידה לא מפוקחת לא מנסים לנבא שום דבר, אלא רק מנסים לזהות דפוסים בנתונים ולכן אלגוריתמים כאלו לא מתאים לפתרון הבעיה שלנו ואילו מטרת אלגוריתמי למידה מפוקחת היא לחזות תוצאות עבור נתונים חדשים כאשר אתה יודע מראש לאיזה סוג התוצאות לצפות.ולכן שימוש באלגוריתמים אלו נועדו בידיוק להשגת המטרה שלנו שהיא לחזות מחלה כאשר אנו יודעים שהתוצאה יכולה להיות כן או לא.

מה גם שלמידה מפוקחת היא שיטה פשוטה ללמידת מכונה המחושבת בד"כ באמצעות שימוש בתוכנות כמו Rאו python ובלמידה ללא פיקוח, נדרשים כלים רבי עוצמה לעבודה עם כמויות גדולות של נתונים לא מסווגים והמודלים נחשבים מורכבים מבחינה חישובית.

לאחר שתיארנו את גישת הלמידה בה בחרנו על מנת לפתור את הבעיה , נתאר את סוג הבעיה:

סוג הבעיה - הבעיה שבחרתי הינה בעיית binary classification , זאת אומרת שבהינתן דגימה, נרצה לסווג אותה ל label (תגית) אחד מתוך שני התגיות האפשריות.
הבעיה שלנו היא לקבוע אם מטופל הוא חולה במחלת לב או לא כלומר בהנתן דגימה של מאפיינים בריאותיים נרצה לסווג את הדגימה ולתת לה תגית/חותמת אחת מבין שתי החותמות האפשריות - האם הדגימה שייכת לאדם החולה במחלת לב או לא.

האלגוריתמים שנבחרו לפתרון הבעיה

את הבעיה ניתן לפתור על-ידי יישום של אלגוריתמי למידה מסוג learning from observation, כלומר למידה המתרחשת באמצעות התבוננות על דגימות.

מתחום ה classification בחרתי את האלגוריתמים הבאים:

1. Decision Trees

לאחר התבוננות במידע שעליו יתבסס הפרויקט וזיהוי הבעיה הסקתי כי המידע שלנו והמטרה מתאימים לבניית עצי החלטה מכמה סיבות:

- המידע שלנו ניתן לתיאור ע"י תכונות עם מספר מוגבל של אפשרויות כלומר כל המאפיינים הם בעלי כמות ערכים מוגבלת.
- הבעיה שלנו היא בעיית סיווג , ואלגוריתם לבניית עץ החלטה הינו אלגוריתם המיועד לבניית מודל להשגת מטרה זו.
- פונקציית המטרה הינה ערך דיסקרטי - 2 אפשרויות לקטגוריות ולכן אין צורך לחשב ערך מדויק אלא רק שיוך לכן ולא.
- יכול להיות שהמידע שלנו מכיל ערכים חריגים ערכים חסרים או מידע רועש ועצי החלטה עובדים טוב יחסית עם מקרים אלו.
- מכיוון שהמידע שלנו יחסית קטן וכבר הנראה לא מתפלג נורמלית יכול מאוד שבשלב האימון נתאים יותר מידי את המודל על הדוגמאות שלנו (overfitting) מה שיגרום לרמת דיוק מאוד גבוה על המידע המאומן ולרמת דיוק פחות טובה על המידע שנבחן ,ובעצי החלטה ישנם דרכים למניעת מצב זה למשל ע"י עצירת בניית העץ כאשר החלוקה כבר לא משמעותית או ע"י cross validation ו - boosting) .
- מסוגל להתמודד עם נתונים ממדיים גבוהים.
- הצגת ביצועים טובים במחקרים קשורים שקראתי.
- זהו הפרויקט הראשון שלי בתחום הבינה מלאכותית ולכן העדפתי כהתחלה לבחור באלגוריתם יחסית פשוט וקל להבנה ושימוש.
- בניית המערכת תתבצע דרך המחשב האישי שלי ולכן היה לי חשוב לבחור בצורת מימוש שתתאים ליכולות מחשב בסיסי ולכן מימוש ע"י עץ החלטה שהוא זול לאחסון הרצה (זמן ריצה לינארי) יסייע לי בכך.

2. Support Vector Machine

- ביצוע Non-linear classification בעזרת פונקציות ליבה שונות (Kernel).
- יעיל במרחבים בעלי ממדים גבוהים, עדיין יעיל במקרים בהם מספר הממדים גדול ממספר הדגימות.
- ורסטילי - ניתן ליישם פונקציות ליבה שונות (kernels) עבור פונקציית ההחלטה.
- SVM מתפקד היטב על מערכי נתונים שיש להם תכונות רבות
- SVM נחשב לטכניקה נפוצה ובשימוש רחב בתחום למידת המכונה ולכן היה חשוב לי להתנסות בה.

Decision Trees

עץ החלטה (Decision Tree) - הינו אלגוריתם השייך לתחום למידה מפוקחת (supervised learning) שמטרתו לבצע סיווג או חיזוי של event מסוים (רשימה של מאפיינים) לקטגוריה בלשהי, כאשר המאפיינים מסודרים לפי סדר החשיבות.
עצי החלטה מופעלים על תצפיות מהצורה $(\mathbf{x}, Y) = (x_1, x_2, x_3, \ldots, x_k, Y)$ כך ש $x_1, x_2, x_3, \ldots, x_k$ הם מאפייני הדוגמא/התצפית ו Y הוא הערך המתאים עבור תצפית זו (ערך המטרה שאותו האלגוריתם לומד).

עץ החלטה הוא עץ בינארי מלא המורכב מצמתי החלטה, כל צומת מייצגת מאפיין מסוים שבכל אחד מהם נבדק תנאי כלשהו עם הערך של התצפית עבור אותו מאפיין ,בנוסף ניתן להסתכל על הקשתות של העץ כערכים האפשריים למאפיין ולבסוף עלים המכילים את הערך החזוי עבור התצפית המתאימה למסלול שמוביל אליהם בעץ(ההחלטה/הסיווג).

סוגים של עצי החלטה הם עצי רגרסיה שבהם מותאם ערך רציף לכל תצפית (ערך המטרה לכל תצפית הוא מספר כלשהו בטווח מסויים) ועצי סיווג שבהם מותאם ערך בדיד(למשל ערך המטרה לתצפית הוא בוליאני) או סוג לכל תצפית(ערך המטרה לתצפית יכול להיות מחלקה כלשהי מקבוצה של מחלקות אפשריות). כמו כן קיימים עצי החלטה מסוג CART (Classification And Regression) Treeהמשלבים את שני סוגי החיזוי.

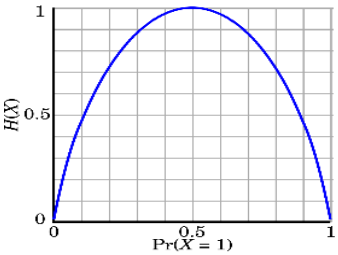
בנוסף ניתן לתאר עצי החלטה גם כסט של חוקים אלגבריים , דבר המסייע בתיאור, סיווג והכללה של קבוצה נתונה של נתונים ומצמצם את כמות הזיכרון שנצטרך על מנת לשמור את אותו העץ.

עץ החלטה הוא ייצוג פשוט לסיווג דוגמאות. למידה מבוססת עץ החלטה היא אחת הטכניקות המוצלחות ביותר ללמידה מפוקחת באמצעות סיווג. עץ יכול "ללמוד" על ידי פיצול קבוצת המקור לתתי קבוצות, המתבססות על מתן ערך לתכונה כאשר השורש יהיה המפריד הטוב ביותר (התכונה בעלת ההשפעה הגדולה ביותר על החלוקה) והתהליך יחזור על עצמו בכל תת-קבוצה באופן רקורסיבי כאשר בכל רמה בעץ בוחרים את המפריד הטוב ביותר .הרקורסיה מושלמת כאשר כל קבוצות המשנה בצומת בעלות אותו ערך מטרה, או כאשר הפיצול כבר אינו מוסיף ערך לתחזיות. תהליך זה של אינדוקציה מלמעלה למטה בעצי החלטה, הוא דוגמה לאלגוריתם נפוץ במיוחד לעצים אשר לומדים החלטה מהנתונים.

בכדי לבנות עץ החלטה, נצטרך לבחור בכל שלב את המאפיין שמפריד את הפריטים בקבוצה בצורה הטובה ביותר כלומר זה שהרווח שלו מהמידע הוא הגדול ביותר. לשם כך, קיימים שני מדדים לרווח ממידע -מדד Entropy ומדד Gini, לפי מדדים אלו נדע אם התכונה שנבחרה היא ה"הטובה" ביותר או לא:

1. מדד Gini המוגדר בתור $I_G(f) = \sum_{i=1}^{m} f_i(1-f_i) = \sum_{i=1}^{m}(f_i - {f_i}^2) = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} {f_i}^2 = 1 - \sum_{i=1}^{m} {f_i}^2$
2. מדד Information Gain המוגדר על פי נוסחת האנטרופיה $I_E(f) = -\sum_{i=1}^{m} f_i \log_2 f_i$

כאשר בשני המקרים לעיל $i = 1, 2, \ldots, m$ הוא כלל מחלקות הסוג האפשרויות לתצפיות ו $f_i \in [0,1]$ הוא השיעור של מחלקת הסוג i בקרב כלל התצפיות המשויכות לצומת שיש לפצלו בעץ. ככל שהמדד קרוב יותר ל-0, כך הסיווג הומוגני יותר.



- <u>אנטרופיה (Entropy)</u> – פונקציה / קונספט שבודק את מידת אי הוודאות/אי סדר שלי.

נניח לשם נוחות שיש לנו שני תיוגים: + , -.
האנטרופיה מודדת לנו את אי הודאות שיש לנו לגבי התיוג בקבוצה.
היא מוגדרת בצורה הבאה: $(S) = -p_+ \log(p_+) - p_- \log(p_-)$.
כאשר (p_+ , p_-) הם היחסים של התיוגים הרלוונטיים מתוך כלל הקבוצה ו- S זו הדגימה מאימון הדוגמאות.

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

<u>Information Gain</u>- אחד המדדים לרווח ממידע מבוסס על מדד אנטרופי כאשר המדד לרווח המידע שמתקבל הוא תוחלת הירידה באי הוודאות.
בעזרת האנטרופיה, משתמשים במדד זה לקביעת התכונה שתהיה המפריד שלנו בכל רמה :
כלומר, ההפרש בין האנטרופיה של הקבוצה לפני החלוקה, לבין הסכום הממושקל (לפי יחסי הגדלים של הקבוצות החדשות) של האנטרופיות בקבוצות החדשות .
בכל שנקטין את האנטרופיה בקבוצות, כך ה-Gain יגדל. לכן, מאפיין שייתן לנו Gain גדול יהיה עדיף.

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

- **ג'יני (Gini)** – בעץ CART (עץ סיווג ורגרסיה) משתמשים במדד הנ"ל וזאת על מנת למדוד כמה פעמים אלמנט שנבחר באופן אקראי מהסט יתויג בצורה שגויה.

אם הוא תויג באופן אקראי בהתאם לחלוקת התוויות בקבוצה המשנה, מדד Gini ניתן לחישוב על ידי סכום ההסתברות של כל פריט להיבחר כפול ההסתברות של סיווג שגוי של פריט זה.

ייתכן כי יגיע למינימום (אפס) כאשר כל המקרים שייכים לקטגוריית מטרה אחת.

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

- **גיזום (pruning)** – כדי להתמודד עם בעיית התאמת-יתר (overfitting), שבה העץ מספק תחזיות מדויקות עבור התצפיות שלפיהן נבנה, אך בעל שיעור שגיאה גבוה עבור תצפיות אחרות, מבוצע תהליך גיזום שבו מוחלפים חלק מהצמתים בעץ בעלים עם סיווג שנקבע על פי הסיווג של רוב הדוגמאות ששייכות לתת העץ שתחת אותו הצומת.

אלגוריתמים נפוצים לבניית עצי החלטה הם:

- ID3 – איטרטיבי (משתמש בinformation gain כדי למצוא את התכונה הטובה ביותר).
- C4.5 – יורשו של ID3 .
- Cart – עץ סיווג ורגרסיה.
- Chaid – מבצע פיצולים רבים כאשר מחשב עצי סיווג
- Mars – עץ החלטה מורחב אשר מתמודד עם נתונים מספריים טוב יותר.

יתרונות עץ ההחלטה

- פשוטים להבנה ופירוש - עצי החלטה תואמים לאופן שבו מרבית בני האדם חושבים על בעיה מסוימת והם פשוטים להסבר למי שאינם מומחים.
- דורש הכנה של מעט נתונים. טכניקות אחרות דורשות לעיתים קרובות נורמליזציה של הנתונים, יצירת משתני דמה והסרה של ערכים ריקים.
- מסוגל להתמודד עם נתונים מספריים ועם נתונים קטגוריאליים. טכניקות אחרות מתמחות בדרך כלל במערכי נתונים מסוג מסוים (ברשתות עצביות לדוגמה, ניתן להשתמש רק במשתנים מספריים).
- משתמש במודל קופסה לבנה. אם מצב מסוים נצפה במודל, ההסבר למצב מוסבר בקלות על ידי לוגיקה בוליאנית (ברשת עצבית לדוגמה, כיוון שהמודל הוא קופסה שחורה, ההסבר לתוצאה מסובך להבנה).
- חזק. פועל היטב, גם אם ההנחות שלו מופרות במקצת על ידי המודל האמיתי ממנו הופקו הנתונים.
- פועל היטב עם מערכי נתונים גדולים. כמויות גדולות של נתונים ניתנות לניתוח באמצעות משאבי מחשוב סטנדרטיים בזמן סביר.
- שימוש בעץ החלטה הוא בזמן לינארי
- נוח וזול לשימוש ואחסון (כאשר האחסון הוא ע"י סט של חוקים אלגבריים)
- עץ החלטה עובד טוב יחסית עם מידע רועש מידע חסר וחריגים

מגבלות

- לומדי עצי החלטה יכולים ליצור עצים מורכבים מדי שלא מכלילים בצורה טובה את נתוני האימון.
- ישנם מושגים קשים להבנה בגלל שעצי החלטה לא מבטאים אותם בקלות כמו XOR, בעיות שוויון ובעיות רבב. במקרים כאלה, עץ ההחלטה נהיה עצום בגודלו. גישות לפתרון הבעיה מערבות שינוי הצגת תחום הבעיה או שימוש באלגוריתמי למידה המבוססים על הצגה רחבה יותר של ביטויים (כמו למידת יחס סטטיסטי ותכנון תכנות הגיון מותווה).
- לנתונים הכוללים משתנים קטגוריאליים עם מספר רמות שונה, רווח מידע בעצי החלטה מוטה לטובת אלה עם תכונות עם יותר רמות.

Support Vector Machines

מכונת וקטורים תומכים (Support Vector Machine) או בקיצור SVM היא טכניקה של למידה מפוקחת המשמשת לניתוח נתונים לצורך סיווג וחיזוי כאחד.
כנהוג בלמידה מפוקחת, דוגמאות סט האימון מיוצגות כווקטורים במרחב ליניארי. עבור בעיות סיווג, בשלב האימון מתאימים מסווג שמפריד נכון ככל האפשר בין דוגמאות אימון חיוביות ושליליות. המסווג שנוצר ב SVMהוא המפריד הליניארי אשר יוצר מרווח גדול ככל האפשר בינו לבין הדוגמאות הקרובות לו ביותר בשתי הקטגוריות. כאשר נבחנת נקודה חדשה, האלגוריתם יזהה האם היא ממוקמת בתוך הקו המגדיר את הקבוצה, או מחוצה לו.

אלגוריתם זה פותר את בעיית הסיווג על ידי בניית משטח (או משטחים במקרה של multi-classes) מפריד בעל גודל ממדי קטן ב1 מגודל המרחב. משטח זה יבנה כמפריד הדגימות בצורה המיטבית ע"י יצירת מסווג שוליים מקסימלי (Maximal Margin classifier) בין 2 דוגמאות אימון בעלות המרחק הקטן ביותר, תוך אפשור של טעויות מדידה על-מנת לשמור על יחס הBias-Variance.
כל נקודה באוסף הנתונים מיוצגת על ידי וקטור בגודל קבוע. בגישה של SVM מחלקים את הנקודות במרחב הווקטורי, באמצעות על-מישור (Hyperplane) כך שהמרווח, המרחק, בין אותו על-מישור המחלק את הנקודות לבין הנקודות הממוקמות הכי קרוב אליו, יהיה המרחק המקסימלי האפשרי. דוגמאות האימון המתלכדות עם מישורי השוליים נקראות וקטורים תומכים ,ומכאן נגזר שם האלגוריתם.

את העל-מישור ניתן לייצג באמצעות הנקודות $\vec{x}$ המקיימות: $\vec{w}\cdot\vec{x}-b=0$ באשר $\vec{w}$ הוא וקטור נורמלי של העל-מישור (לאו דווקא מנורמל).

- <u>Decision Boundary</u> – גבול החלטה, מפריד נקודות לקבוצות הסיווג המתאימה להם.
- <u>Maximal Margin Classifier</u> – מסווג שוליים מקסימלי. נמקם את הגבול ההחלטה (ה-hyperplane) במיקום אשר ייתן לנו את הmargin המקסימלי המתאפשר.
- <u>הפרדה קשיחה (Hard Margin)</u> - במקרים של הפרדה קשיחה(hard margin) הפרדה מושלמת אכן אפשרית. האלגוריתם מוצא הנתיב הרחב ביותר האפשרי. כמובן שיש לבצע תחילה נרמול של הנתונים, הוקטורים התומכים הם למעשה התצפיות בקצה הנתיב.
- <u>הפרדה רכה (Soft Margin)</u> - נגדיר "הפרדה רכה (soft margin) כמצב שבו אין הפרדה ליניארית בין שתי הקבוצות. עבור בעיית "הפרדה רכה" אנו מחשבים את היקף ההפרה או מידת ההפרה עבור נתונים שסווגו לא נכון.
- <u>הפרדה א-לינארית (Non-linear Separation)</u>-המטרה היא ליצור מאפיינים חדשים כך שהגבול (boundary) יהפוך ללינארי.

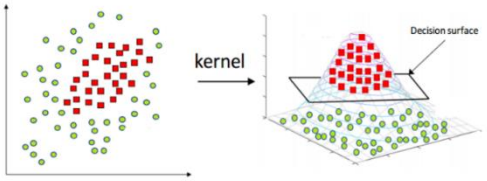
- <u>The Kernel Trick</u> –שיטה מתמטית, המקבלת כקלט 2 ווקטורי נתונים (Y,X) בעלי ממד נמוך יחסית שאינם ניתן לסיווג בצורה לינארית, ומחזירה את סט הנתונים בממד גבוה יותר, על-ידי מיפוי הנתונים לפי פונקציות התלויות באותם נתונים.

שיטה זו משמשת על-מנת למצוא מפריד לינארי כאשר אין כזה, על-ידי העלאת הממד ומציאת המפריד במימד הגבוה יותר ביחס לאותו ממד. הפונקציה יכולה להגיע עד לממד ה-N ולכן בסופו של דבר, ימצא ממד המאפשר הפרדה לינארית.

כאשר הנתונים מוחזרים בממד הגבוה ביותר, ביחס לאותו ממד – המפריד כבר אינו לינארי.

ישנם שני סוגי Kernel נפוצים, בהינתן ווקטורים אלו:

- <u>Linear Kernel</u> – לדוגמה, מהצורה $l(\vec{a},\vec{b}) = \gamma(\vec{a}\cdot\vec{b})$, כאשר $\vec{a},\vec{b}$ מהווים את ווקטורי נקודות התצפית ו- γ מגדיר את גודל השפעת הדגימה על שאר הנקודות. על מכפלת הdot product, נקבל מכפלה של 2 ווקטורים ממעלה גבוהה יותר, המייצגים את הנתונים בממד החדש.
- <u>Polynomial Kernel</u> – לדוגמה, מהצורה $p(\vec{a},\vec{b}) = \gamma(\vec{a}\cdot\vec{b}+r)^d$, כאשר $\vec{a},\vec{b}$ מהווים את ווקטורי נקודות התצפית, γ מגדיר את גודל השפעת הדגימה על שאר הנקודות, r הינו קבוע הפולינום ו-d יקבע את חזקת הפולינום. על מכפלת ה dot product, נקבל מכפלה של 2 ווקטורים ממעלה גבוהה יותר, המייצגים את הנתונים בממד החדש.
- <u>Radial basis Function (RBF) Kernel</u> – לדוגמה, מהצורה $rbf(\vec{a},\vec{b}) = e^{-\gamma(|\vec{a}-\vec{b}|)^2}$,
  כאשר $\vec{a},\vec{b}$ מהווים את ווקטורי נקודות התצפית ו- γ מגדיר את גודל השפעת הדגימה על שאר הנקודות. על מכפלת ה dot product, נקבל מכפלה של 2 ווקטורים ממעלה גבוהה יותר, המייצגים את הנתונים בממד החדש. ניתן ליישם את הrbf-kernel לאינסוף ממדים (באופן תיאורטי)
  יש לציין כי שיטה זו אינה מחשבת את הממד החדש, שכן הדבר יקר בזמן ומקום חישוביים, אלא רק מחשבת את היחסים עבור כל 2 נקודות במרחב באילו הן בממד גבוה יותר!.



סקירת התחומים לפתרון - המשך

חלוקת המידע

הליך פיצול המידע לtraining set ו-testing set משמש להערכת הביצועים של אלגוריתמי למידת מכונה, ניתן להשתמש בו לבעיות סיווג או רגרסיה וניתן להשתמש בו עבור כל אלגוריתם למידה בפיקוח. זהו הליך מהיר וקל לביצוע, שתוצאותיו מאפשרות לנו להשוות את הביצועים של המודל שבנינו אך ישנם מקרים שבהם אין להשתמש בהליך זה, למשל כאשר יש לנו מערך נתונים קטן או כאשר מערך הנתונים אינו מאוזן.
ההליך כולל לקיחת מערך נתונים וחלוקתו לשתי קבוצות משנה:

- תת-הקבוצה הראשונה משמשת כדי לאמן ולבנות את המודל והיא מכונה training set
- תת-הקבוצה השנייה המכונה test set משמשת לבחינת והערכת המודל שבנינו ע"י הכנסת הדוגמאות החדשות והשוואת התוצאות שקיבלנו מהמודל מול התוצאות האמיתיות.

הפיצול של מערך המודלים שלנו לדוגמאות לאימון ולדוגמאות למבחן הוא אחד משלבי העיבוד המוקדמים שעלינו לבצע כאשר חלוקת המידע צריכה להיות חלוקה רנדומלית בהתפלגות נורמלית כדי שנוכל להימנע מ overfitting .

אחוזי הפיצול הנפוצים הינם:

- אימון: 80%, מבחן: 20%
- אימון: 67%, מבחן: 33%
- אימון: 50%, מבחן: 50%

ישנם מספר טכניקות לחלוקת המידע ,אחת מהטכניקות הנפוצות היא הטכניקה של cross validation שמחלקת את סט הנתונים בין קבוצת אימון ומבחן יותר מאשר פעם אחת (בד"כ ל10 קבוצות). כאשר בכל פעם מאמנים את המודל על 9 מהקבוצות ובוחנים את התוצאות על הקבוצה שנותרה וכך שוב לכל קבוצה עד שלבסוף מעריכים את התוצאות ולוקחים את החלוקה הטובה ביותר.

בניית המערכת

לאחר שזיהינו והגדרנו היטב את הבעיה נצטרך למצוא מאגר מידע אמין שממנו ניצור את הדוגמאות שנוכל ללמוד מהם.
מאגר הנתונים שבו נשתמש נלקח ממאגר למידת מכונה של אוניברסיטת קליפורניה Irvine.

השלב הראשון בבניית המערכת: ניתוח המידע

לפני בניית המודל השלב הראשון והחשוב ביותר בלמידת מכונה זה הבנה וניתוח המידע שעליו נעבוד.
לכן נעבור על מאגר הנתונים שלנו וננתח אותו בעזרת הפונקציונליות של ספריית pandas:

1. נבין את המידע שעליו אנחנו עובדים - מה מייצגת כל עמודה, מה המשמעות מבחינה רפואית של כל תכונה (הבנה בסיסית), מהו הערך המינימלי והמקסימלי בכל עמודה ומהו הממוצע והחציון של הערכים.
2. נבדוק האם קיימים לנו תכונות לא רלוונטיות במאגר הנתונים.
3. נבדוק האם קיימים ערכים חסרים במאגר הנתונים במידה ויימצאו נמלא אותם בערכי הממוצע או החציון של אותה עמודה (ערכים חסרים יכולים לגרום לרעש לא רצוי שעלול להפריע למודל ללמוד)
4. נבדוק מהם הערכים המיוחדים בכל תכונה (עמודות בהם השונות גבוהה במיוחד לא תורמות, ואף עלולות להזיק כי הם מכניסות רעש שמפריע למודל ללמוד ולכן נשקול להסירן).
5. נבדוק האם כמות המופעים של אנשים החולים במחלת לב והכמות של אלו שלא מאוזנת.
6. נבדוק מהם סוגי התכונות במאגר הנתונים ומהם הטיפוסים של הערכים בכל תכונה.
7. נחפש ערכים Outliers ונטפל בהם.
8. ננתח את התכונות וננסה להבין אלו מבינהם בעלי ההשפעה הגדולה ביותר על משתנה המטרה.

השלב השני בבניית המערכת: עיבוד המידע

לאחר ניתוח והבנת המידע , נעבור לשלב השני שבו נעבד את המידע ונגדירו בצורה שנוכל לעבוד איתו:
1. נפצל את העמודות לתכונות קטגוריאליות ולתכונות נומריות.
2. נמיר תכונות קטגוריאליות למשתנה דמה וניצור עמודה חדשה לכל משנה כזה (בעזרת הפונקציה get_dummies).
(משתני דמה - כאשר למשתנה קטגורי יש יותר משתי קטגוריות אפשריות, הוא יכול להיות מיוצג על ידי קבוצה של משתני דמה, כל משתנה דמה ייצג את הערך האפשרי של אותה תכונה) לכל תכונה ניצור עמודה אחת לכל ערך בטווח שהיא יכולה לקבל ,כאשר מופע עם הערך של אותה תכונה יכיל את הערך 1 בעמודה המייצגת את התכונה ובשאר העמודות יכיל 0.
3. נבצע נירמול (Normalization) Feature scaling- שיטה המשמשת לנרמל את טווח המשתנים הבלתי תלויים או התכונות של הנתונים. בעיבוד נתונים, זה ידוע גם בשם נורמליזציה של נתונים. באופן כללי, יש צורך לנרמל תכונות כך שאף תכונה לא תכיל ערכים גדולים מידי באופן שרירותי (מרוכז) וכל התכונות יהיו באותו קנה מידה מכיוון שאלגוריתמים המנצלים מרחקים או קווי דמיון בין דגימות נתונים, כגון SVM, רגישים לתמורות תכונות (במסווגים המבוססים על מודל גרפי כמו עץ החלטה אין צורך לנרמל אך לפעמים זה מועיל גם בהם). הנירמול יתבצע ע"י הפונקציה MinMaxScaler

השלב השלישי בבניית המערכת: בניית המודל והערכתו

1. לכל דוגמא ניצור את וקטור המאפיינים X - שיכיל את כל התכונות מלבד ערך המטרה.
2. ניצור את הוקטור Y - שיכיל את ערך המטרה של של הדוגמא המיוצגת בוקטור X.
3.נפצל את וקטורי ה-X ווקטורי ה-Y ל **training set** שישמש להכנת ואימון המודל ול - **test set** שבו ננסה לחזות דוגמאות חדשות ומהתוצאות נוכל להעריך את ביצועי המודל שלנו. החלוקה תתבצע ע"י הפונקצייה train_test_split ,כך שחלוקת המידע תהיה חלוקה רנדומלית כאשר נשווה בין 2 חלוקות -חלוקה אחת בה היחס יהיה 85:15 וחלוקה שניה בה היחס יהיה 70:30
4. נבנה את המסווג SVM ע"י הפונקציה SVC כאשר נשווה בין שתי שיטות - Linear Kernel, ו- Radial basis Function (RBF) Kernel
5.נבנה את המסווג עץ החלטה ע"י הפונקציה DecisionTreeClassifier כאשר נשווה בין שתי אסטרטגיות לבחירת הפיצול - Entropy ו- Gini
5.נבחן את המודלים שקיבלנו מכל מסווג על הtest set ונעריכם ע"י מס' מדדי ביצוע שנפרט בהמשך.
הערכה תתבצע ע"י הפונקציות confusion_matrix ,classification_report

איך להריץ:
אין דרישה מיוחדת , המערכת נבנתה בpycharm ולא דורשת דברים חריגים.
יש להריץ בסביבת עבודה התומכת בשפת פייתון.
יש לוודא שכל הספריות שנדרשות מותקנות בסביבת העבודה (ככל הנראה יהיה צורך להתקין מספר מהספריות).
יש לוודא שמיקום מאגר הנתונים תואם למיקום בשורת הקוד ליבוא המאגר.

סקירת המערכת

על מנת לסקור את המערכת נתחיל בתיאור מאגר הנתונים שלנו :

Dataset

מאגר הנתונים שבו נשתמש :

- מורכב מ14 עמודות - כאשר 13 עמודות מתארות את התכונות ועמודה נוספת מכילה את התגית של משתנה המטרה שלנו .
- מכיל 303 שורות המייצגות נתוני נבדקים אמיתיים ואמינים כאשר כל שורה מייצגת מופע של נבדק אחד.

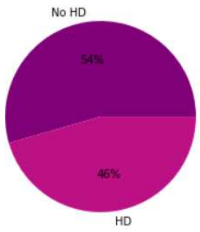
Features

מאגר הנתונים מכיל תכונות משלושה סוגים - תכונות בינארית {0,1} ,תכונות נומינליות בטווח מסוים ותכונות רציונליות בטווח מסוים.

להלן התכונות שבהן נשתמש כדי לחזות את משתנה המטרה שלנו (מחלת לב או ללא מחלת לב):

- age - The person's age in years
- sex - The person's sex (1 = male, 0 = female)
- cp - chest pain type :
  - 0 - Typical angina: chest pain related decrease blood supply to the heart
  - 1 - Atypical angina: chest pain not related to heart
  - 2 - Non-anginal pain: typically esophageal spasms (non heart related)
  - 3 - Asymptomatic: chest pain not showing signs of disease
- trestbps - resting blood pressure (in mm Hg), anything above 130-140 typically causes for concern
- chol - serum cholesterol in mg/dl (serum = LDL + HDL + .2 * triglycerides, above 200 is cause for concern)
- fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- restecg - resting electrocardiographic results
  - 0 - Nothing to note
  - 1 - ST-T Wave abnormality (signals non-normal heart beat ,can range from mild symptoms to severe problems)
  - 2 - Possible or definite left ventricular hypertrophy (enlarged heart's main pumping chamber)
- thalach - maximum heart rate achieved
- exang - exercise induced angina (1 = yes , 0 = no)
- oldpeak - ST depression induced by exercise relative to rest (looks at the stress of heart during exercise, unhealthy heart will stress more)
- slope - the slope of the peak exercise ST segment
  - 0 - Upsloping: better heart rate with exercise (uncommon)
  - 1 - Flat Sloping: minimal change (typical healthy heart)
  - 2 - Downsloping: signs of an unhealthy heart
- ca - number of major vessels (0-3) colored by fluoroscopy (colored vessel means the doctor can see the blood passing through, the more blood movement the better (no clots))
- thal - thallium stress result
  - 1,3 - normal
  - 6 - fixed defect: used to be a defect but ok now
  - 7 - reversible defect: no proper blood movement when exercising
-
- target - have disease or not (1=yes, 0=no)

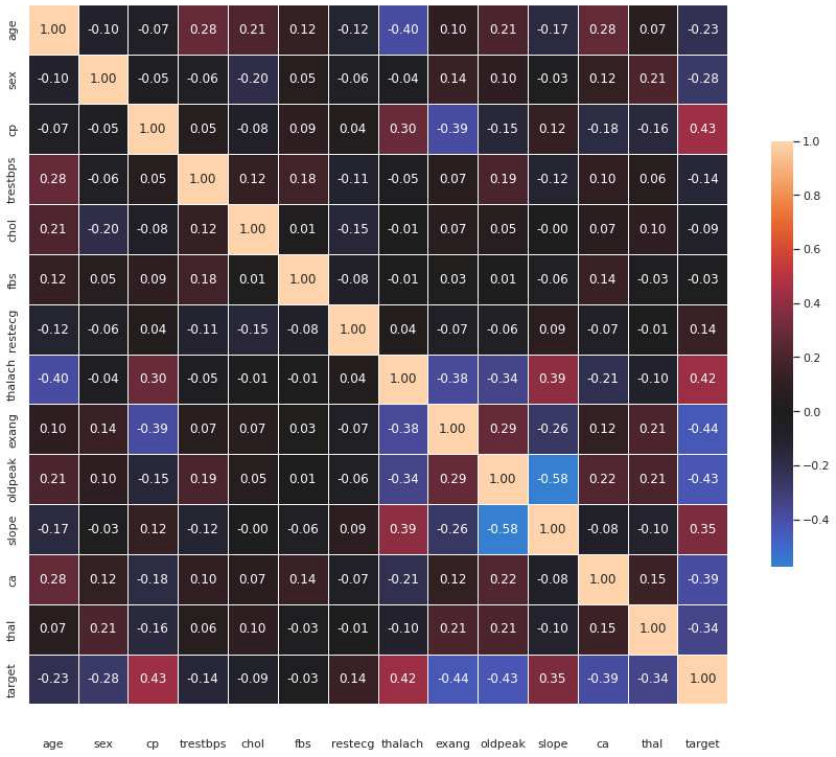
מסקנות מהשלב הראשון:



- מאגר הנתונים המקורי הכיל 76 תכונות , אך לאחר מחקר וניתוח של מומחים בתחום נמצא כי 14 התכונות שהצגנו הן הרלוונטיות ביותר לחיזוי מחלת לב ולכן לא נצטרך להתעסק בסינון תכונות לא רלוונטיות אלא רק במציאת התכונות המשפיעות ביותר על משתנה המטרה שלנו.
- טיפוסי הערכים במאגר המידע שלנו הם מהסוגים int ו-float
- בנוסף ראינו כי לא קיימים ערכים חסרים במאגר הנתונים.
- מצאנו כי המידע שלנו מחולק ל138 אנשים בריאים ול165 אנשים חולים ולכן המידע שלנו מאוזן.

מסקנות נוספות שעלו מניתוח המידע :

- גברים אובחנו עם מחלת לב יותר מנשים.
- אפילו חולים אסימפטומטיים מאובחנים לעתים קרובות עם מחלת לב.
- מחלת לב מאובחנת לעתים קרובות כאשר קיימת אנגינה הנגרמת על ידי פעילות גופנית.
- אם ה ST_Slope שטוח/למטה, הוא נחשב לעתים קרובות כמחלת לב.
- fbs ו-chol הם התכונות בעלי ההשפעה הנמוכה ביותר עם משתנה המטרה לשאר התכונות יש השפעה יחסית גבוה על המטרה
- כדי להבין את השפעת כל תכונה על משתנה המטרה נשתמש בפונקציה שתחזיר לנו את מטריצת התאימות, להלן המטריצה שקיבלנו:



סקירת המערכת - המשך

סקירת הכלים בהם אנחנו משתמשים:

ספריות שנייבא :

* pandas - מיועדת לניתוח ועיבוד נתונים , נשתמש בה כדי לקרוא ולהקצות את מערך הנתונים לתוך DataFrame.
* numpy - מאפשרת לעבוד עם מערכים רב-מימדיים, ומספקת פונקציות מתמטיות לעבודה עם המערכים נשתמש בה כדי לייצג את הדוגמאות שלנו כוקטורים ולא כרשימות.
* sklearn - היא ספריית המיועדת ללמידת מכונה ,מספקת כלים שונים להתאמת מודלים, עיבוד מקדים של נתונים, בחירת מודלים והערכתם, וכלים רבים אחרים.

פונקציות מרכזיות שנשתמש בהן :

פונקציות להערכת מדדי הביצוע של המסווגים:

confusion_matrix - תצוגה מפורטת של כל השגיאות של המסווג , בציר

accuracy_score - פונקציה לחישוב רמת הדיוק של המסווג

classification_report- מחזיר דוח המציג את מדדי הסיווג העיקריים.

פונקציות למסווגים:

SVC -פונקציה למסווג SVM , מקבל את הפרמטרים הבאים :

0.1 = gamma,C=1.0, kernel={‘linear’, , ‘rbf’, }, probability=True, max_iter=-1

DecisionTreeClassifier - פונקציית למסווג עץ החלטה, מקבל את הפרמטרים הבאים:

criterion{“gini”, “entropy”} - למדידת איכות הפיצול. הקריטריונים הנתמכים הם "Gini " עבור Gini impurity ו"אנטרופיה" לרווח המידע.

max_depthint -העומק המרבי של העץ.

min_impurity_decreasefloat - צומת יפוצל אם פיצול זה גורם לירידה של ה impurity גדולה או שווה לערך זה.

פונקציה לחלוקת המידע לאימון ומבחן :

()train_test_split -פונקציה לפיצול הנתונים ל training data ולtest data מקבלת את הערכים הבאים:

test_size- מייצג את היחס של חלוקת המידע כאשר הערך במשתנה מייצג את האחוז של המידע למבחן מכלל הנתונים

random_state-שולט בערבוב שהוחל על הנתונים לפני החלת הפיצול .

הערכת ביצועי מסווג ביארי

בתום אימון המערכת נהוג להעריך את ביצועיה כדי לבדוק את טיב ההיפותזה שנלמדה. בדרך כלל משתמשים בהערכות סטטיסטיות מגוונות, אך תמיד נהוג להציג את אחוזי הדיוק של המודל על סט הבדיקה ואת ערך פונקציית ההפסד שלו.
לשם כך שמרנו את המידע המתויג שלא השתתף בתהליך האימון (testing data) ובעזרתו נוכל להעריך את ביצועיי המסווג ע"י כך שנפעיל את המודל שבנינו על המידע החדש ונשווה את תוצאות החיזוי שהתקבלו עם תגיות האמת של הדוגמאות כאשר נרצה להבחין בין ארבעה מצבים :

2 מקרים בהם המודל צודק:

True positive - זיהוי נכון של דוגמא חיובית

True negative -זיהוי נכון של דוגמא שלילית

2 מקרים בהם המודל טועה:

False positive- זיהוי של דוגמא שלילית כחיובית

False negative - זיהוי של דוגמא חיובית כשלילית

כדי לזהות את סוגי השגיאות מתארים את התוצאות באמצעות **Confusion matrix** שמתוכה מחשבים מדדים להערכת המודל (model evaluation metrics).

ישנם מספר מדדי הערכה למסווג , נציג את המרכזיים שמבניהם :

- **Accuracy -** שיעור הזיהוי הנכון מתוך כל הדוגמאות (חיוביות או שליליות). המדד accuracy יודע להעריך באופן כללי עד כמה המודל הוא מדויק אבל הוא לא יודע להגיד לנו אילו שגיאות הוא עשה ולפעמים לסוג טעות אחת תהיה יותר משקל מסוג הטעות שניה ולכן מדד זה לא תמיד מייצג את ביצועי המודל האמיתיים.
המדד מחושב כך:

$$Accuracy = \frac{True\ positive + True\ negative}{True\ positive + True\ negative + False\ positive + False\ negative}$$

- **sensitivity/Recall -** היא הפרופורציה של דוגמאות חיוביות שזוהו נכונה (True positive) מכל הדוגמאות החיוביות שהמודל זיהה.
המדד מחושב כך:

$$Recall = \frac{True\ positive}{True\ positive + False\ negative}$$

- **Precision** - שיעור הזיהוי הנכון מתוך הדוגמאות החיוביות כלומר היחס של תצפיות חיוביות שהמודל זיהה נכונה מכל התצפיות שהמודל זיהה שהם חיוביות (בצדק או שלא בצדק) . המדד מחושב כך:

$$Precision = \frac{True\ positive}{True\ positive + False\ positive}$$

- **F1** - מדד F1 עושה ממוצע הרמוני של ה-Precision וה- Recall ובכך לוקח בחשבון את השגיאות משני הסוגים.
כדי לקבל ערכים גבוהים במדד זה, נצטרך שגם ה-precision יהיה טוב וגם ה-recall.כך נראית הנוסחה:

$$F1\ score = \frac{2 * Recall * Precision}{Recall + Precision}$$

כמובן שנרצה שהמודל שלנו יהיה המדויק ביותר וגם הרגיש ביותר אבל יש בעיה בגלל שה-**precision** וה-**recall** נותנים בדרך כלל תוצאות הפוכות.
לכן כדי להעריך את המודל שלנו נשתמש במדד שהכי מתאים לבעיה ולכל סוג טעות ניתן מחיר אחר (Biased error pricing).
לדוגמא אם אנחנו מעוניינים להעריך את יעילותו של מודל לזיהוי מחלה מסוימת החשש הוא יותר מחולים שלא נצליח לזהות (**false negative**) וכבל הנראה לא יזכו לטיפול מאשר מזיהוי שגוי של אדם בריא כחולה במחלה (**false positive**). במקרה זה נרצה לעשות אופטימיזציה של המודל ל-**recall**. לדוגמה, על ידי הורדת הסף שמעליו נגדיר אדם כחולה.
לעומת זאת במקרה של מודל לזיהוי דואר זבל, רוב האנשים יעדיפו שמיילים מסוג ספאם יגיעו אליהם לתיבת הדואר הנכנס מאשר שדואר לגטימי ילך לתיבת הספאם. בגלל שההעדפה היא ל-**false positive** כדאי לנו לעשות אופטימיזציה ל-**precision**.
לא תמיד המקרים הם כל כך ברורים ועל כן חשוב להשתמש במדד **F1** בהערכת המודל מפני שהוא לוקח בחשבון את שני המדדים, **recall** ו- **precision**.

השוואת ביצועי המודל

תחילה נשווה בין 2 המודלים שהתקבלו מכל מסווג

SVM:

1. מסווג SVM עם Kernel = Radial basis Function (RBF):

```
Train split: 85%
Train Result:
================================================
Accuracy Score: 92.61%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.931624    0.921429   0.92607    0.926526      0.926189
recall       0.908333    0.941606   0.92607    0.924970      0.926070
f1-score     0.919831    0.931408   0.92607    0.925620      0.926002
support    120.000000  137.000000   0.92607  257.000000    257.000000
_______________________________________________
Confusion Matrix:
 [[109  11]
 [  8 129]]
```

```
Test split: 15%
Test Result:
================================================
Accuracy Score: 89.13%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy  macro avg  weighted avg
precision    0.842105    0.925926  0.891304   0.884016      0.893127
recall       0.888889    0.892857  0.891304   0.890873      0.891304
f1-score     0.864865    0.909091  0.891304   0.886978      0.891785
support     18.000000   28.000000  0.891304  46.000000     46.000000
_______________________________________________
Confusion Matrix:
 [[16  2]
 [ 3 25]]
```

```
Train split: 70%
Train Result:
================================================
Accuracy Score: 91.98%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.920000    0.919643  0.919811    0.919821      0.919813
recall       0.910891    0.927928  0.919811    0.919410      0.919811
f1-score     0.915423    0.923767  0.919811    0.919595      0.919792
support    101.000000  111.000000  0.919811  212.000000    212.000000
_______________________________________________
Confusion Matrix:
 [[ 92   9]
 [  8 103]]
```

```
Test split: 30%
Test Result:
================================================
Accuracy Score: 92.31%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy  macro avg  weighted avg
precision    0.894737    0.943396  0.923077   0.919067      0.923612
recall       0.918919    0.925926  0.923077   0.922422      0.923077
f1-score     0.906667    0.934579  0.923077   0.920623      0.923230
support     37.000000   54.000000  0.923077  91.000000     91.000000
_______________________________________________
Confusion Matrix:
 [[34  3]
 [ 4 50]]
```

2. מסווג SVM עם Linear Kernel

```
Train split: 85%
Train Result:
================================================
Accuracy Score: 87.16%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.914286    0.842105  0.871595    0.878195      0.875808
recall       0.800000    0.934307  0.871595    0.867153      0.871595
f1-score     0.853333    0.885813  0.871595    0.869573      0.870647
support    120.000000  137.000000  0.871595  257.000000    257.000000
_______________________________________________
Confusion Matrix:
 [[ 96  24]
 [  9 128]]
```

```
Test split: 15%
Test Result:
================================================
Accuracy Score: 89.13%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy  macro avg  weighted avg
precision    1.000000    0.848485  0.891304   0.924242      0.907773
recall       0.722222    1.000000  0.891304   0.861111      0.891304
f1-score     0.838710    0.918033  0.891304   0.878371      0.886993
support     18.000000   28.000000  0.891304  46.000000     46.000000
_______________________________________________
Confusion Matrix:
 [[13  5]
 [ 0 28]]
```

```
Train split: 70%
Train Result:
================================================
Accuracy Score: 84.91%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.870968    0.831933  0.849057    0.851450      0.850530
recall       0.801980    0.891892  0.849057    0.846936      0.849057
f1-score     0.835052    0.860870  0.849057    0.847961      0.848569
support    101.000000  111.000000  0.849057  212.000000    212.000000
_______________________________________________
Confusion Matrix:
 [[81 20]
 [12 99]]
```

```
Test split: 30%
Test Result:
================================================
Accuracy Score: 90.11%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy  macro avg  weighted avg
precision    0.911765    0.894737  0.901099   0.903251      0.901660
recall       0.837838    0.944444  0.901099   0.891141      0.901099
f1-score     0.873239    0.918919  0.901099   0.896079      0.900346
support     37.000000   54.000000  0.901099  91.000000     91.000000
_______________________________________________
Confusion Matrix:
 [[31  6]
 [ 3 51]]
```

- מבחינת חלוקת המידע, החלוקה הטובה ביותר בשתי השיטות הייתה החלוקה ליחס של 70:30
- מבחינת כל מדדי הדיוק , השיטה הטובה ביותר למסווג SVM כדי לחזות מחלת לב היא בעזרת (RBF) Kernel

Decision Tree Classifier

1. מסווג עץ החלטה עם criterion='entropy'

```
Train split: 85%
Train Result:
================================================
Accuracy Score: 94.16%

------------------------------------------------
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.948718    0.935714  0.941634    0.942216      0.941786
recall       0.925000    0.956204  0.941634    0.940602      0.941634
f1-score     0.936709    0.945848  0.941634    0.941279      0.941581
support    120.000000  137.000000  0.941634  257.000000    257.000000

------------------------------------------------
Confusion Matrix:
 [[111   9]
 [  6 131]]
```

```
Test split: 15%
Test Result:
================================================
Accuracy Score: 80.43%

------------------------------------------------
CLASSIFICATION REPORT:
                    0           1  accuracy  macro avg  weighted avg
precision   0.695652    0.913043  0.804348   0.804348      0.827977
recall      0.888889    0.750000  0.804348   0.819444      0.804348
f1-score    0.780488    0.823529  0.804348   0.802009      0.806687
support    18.000000   28.000000  0.804348  46.000000     46.000000

------------------------------------------------
Confusion Matrix:
 [[16  2]
 [ 7 21]]
```

```
Train split: 70%
Train Result:
================================================
Accuracy Score: 99.53%

------------------------------------------------
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.990196    1.000000  0.995283    0.995098      0.995329
recall       1.000000    0.990991  0.995283    0.995495      0.995283
f1-score     0.995074    0.995475  0.995283    0.995275      0.995284
support    101.000000  111.000000  0.995283  212.000000    212.000000

------------------------------------------------
Confusion Matrix:
 [[101   0]
 [  1 110]]
```

```
Test split: 30%
Test Result:
================================================
Accuracy Score: 79.12%

------------------------------------------------
CLASSIFICATION REPORT:
                    0           1  accuracy  macro avg  weighted avg
precision   0.704545    0.872340  0.791209   0.788443      0.804116
recall      0.837838    0.759259  0.791209   0.798549      0.791209
f1-score    0.765432    0.811881  0.791209   0.788657      0.792995
support    37.000000   54.000000  0.791209  91.000000     91.000000

------------------------------------------------
Confusion Matrix:
 [[31  6]
 [13 41]]
```

2. מסווג עץ החלטה עם criterion = 'gini'

```
Train split: 85%
Train Result:
================================================
Accuracy Score: 85.21%

------------------------------------------------
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.836066    0.866667   0.85214    0.851366      0.852378
recall       0.850000    0.854015   0.85214    0.852007      0.852140
f1-score     0.842975    0.860294   0.85214    0.851635      0.852207
support    120.000000  137.000000   0.85214  257.000000    257.000000

------------------------------------------------
Confusion Matrix:
 [[102  18]
 [ 20 117]]
```

```
Test split: 15%
Test Result:
================================================
Accuracy Score: 84.78%

------------------------------------------------
CLASSIFICATION REPORT:
                    0           1  accuracy  macro avg  weighted avg
precision   0.761905    0.920000  0.847826   0.840952      0.858137
recall      0.888889    0.821429  0.847826   0.855159      0.847826
f1-score    0.820513    0.867925  0.847826   0.844219      0.849372
support    18.000000   28.000000  0.847826  46.000000     46.000000

------------------------------------------------
Confusion Matrix:
 [[16  2]
 [ 5 23]]
```

```
Train split: 70%
Train Result:
================================================
Accuracy Score: 86.32%

------------------------------------------------
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.891304    0.841667  0.863208    0.866486      0.865315
recall       0.811881    0.909910  0.863208    0.860896      0.863208
f1-score     0.849741    0.874459  0.863208    0.862100      0.862683
support    101.000000  111.000000  0.863208  212.000000    212.000000

------------------------------------------------
Confusion Matrix:
 [[ 82  19]
 [ 10 101]]
```

```
Test split: 30%
Test Result:
================================================
Accuracy Score: 89.01%

------------------------------------------------
CLASSIFICATION REPORT:
                    0           1  accuracy  macro avg  weighted avg
precision   0.864865    0.907407   0.89011   0.886136       0.89011
recall      0.864865    0.907407   0.89011   0.886136       0.89011
f1-score    0.864865    0.907407   0.89011   0.886136       0.89011
support    37.000000   54.000000   0.89011  91.000000      91.00000

------------------------------------------------
Confusion Matrix:
 [[32  5]
 [ 5 49]]
```

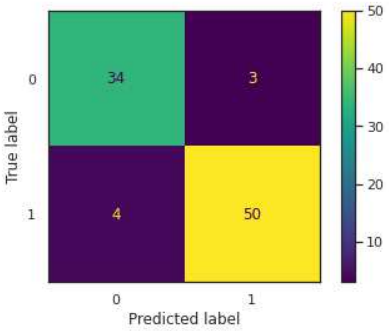
- מבחינת חלוקת המידע, החלוקה הטובה ביותר בשתי השיטות הייתה החלוקה ליחס של 70:30
- מבחינת רמת דיוק האסטרטגיה לבחירת המפריד הטוב ביותר היתה ע"י 'gini'
- בנוסף ניתן לראות שכאשר משווים את רמת הדיוק של המודל המאומן לבין רמת הדיוק של המודל שבנינו ישנו פער קטן יותר כאשר משתמשים באסטרטגיית 'gini', כלומר הוא מונע overfitting יותר מאשר באסטרטגיה השניה

כעת נשווה בין שתי המודלים הטובים ביותר מכל מסווג:

כדי להשוות בין מדדי הביצוע נציג את מטריצת הטעויות שקיבלנו מכל אחד מהמודלים

נזכיר כי במערכת שלנו אנחנו יותר רגישים לטעויות מסוג false negative כי אבחון אדם חולה במחלת לב כבריא יגרום לנזק רב יותר (לא יזכה לטיפול ומצבו יחמיר למשל) מאשר אבחון אדם בריא כחולה (שכן אולי בעתיד יבצע בדיקות נוספות וכך תשלל האבחנה השגויה)

:SVM

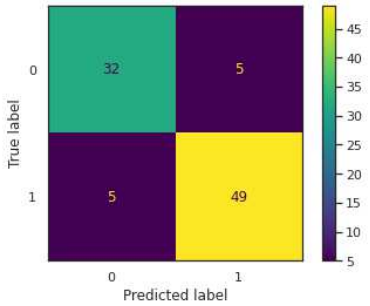


ניתן לראות כי המודל **חזה נכון** 34 דוגמאות חיוביות ו50 דוגמאות שליליות.

בנוסף המודל **חזה לא נכון** שלוש דוגמאות שליליות כחיוביות ו4 דוגמאות חיוביות כשליליות.

לכן מדד הרגישות שלנו במודל זה הינו 92.31%

Decision Tree Classifier



ניתן לראות כי המודל **חזה נכון** 32 דוגמאות חיוביות ו49 דוגמאות שליליות.

בנוסף המודל **חזה לא נכון** 5 דוגמאות שליליות כחיוביות ו5 דוגמאות חיוביות כשליליות.

לכן מדד הרגישות שלנו במודל זה הינו 89.01%

מסקנות:

כלומר מסווג SVM הגיע לתוצאות מדויקות יותר לעומת DT
בנוסף ניתן לראות כי מסווג SVM מונע התאמת יתר טוב יותר מאשר עץ החלטה
מבחינת זמני ריצה ,סיבוכיות מקום וסיבוכיות שימוש במודל שתי המודלים בעלי אותו סדר גודל.
מבחינת סיבוכיות בניית המודל , לי באופן אישי היה יותר מסובך לבנות את המודל המשתמש במסווג SVM מכיוון שהיה לי קשה להבין אילו מהפרמטרים משפיעים על התוצאות ולמה.
לכן ניתן להסיק ששימוש בSVM טוב יותר לחיזוי מחלת לב מאשר DT אך דורש ידע רב רחב יותר.

מסקנות אישיות

בפרוייקט זה התנסתי לראשונה בתחום למידת המכונה , לאחר קריאת מספר מאמרים ומדריכים ובמהלך יצירת הפרויקט הבנתי שלמידת מכונה ובפרט בעיות סיווג זהו תחום רחב מאוד שדורש המון ידע וניסיון ועיקר העבודה בלמידת מכונה מתבססת על הבנה וניתוח מעמיק של הנתונים שדורש ידע רחב בניתוח נתונים.

נבחתי לראות כי גם שינוי של פרמטר קטן שנראה לנו לא עיקרי , יכול להשפיע מאוד על המערכת ותוצאות המודל ולכן כאשר רוצים לבנות מודל יש לבחון את התנהגותו בשילובי פרמטרים רבים (מה שלצערי לא יכלתי לבצע בפרויקט זה מכיוון שלכל מסווג היו המון פרמטרים שיכלנו להכניס וכל פרמטר דורש הבנה מעמיקה על אופן פעילותו והשפעתו ולכן העדפתי להתמקד בפרמטרים שנלמדו בכיתה).

בנימה אישית חייבת לומר שמאוד נהנתי מהתהליך ולמדתי המון (גם בפן הרפואי שמאוד מושך אותי), אך בעיקר בתחום למידת המכונה ולמרות שהמערכת שבניתי מאוד חובבנית ובטח שלא אופטימלית למדתי דרכה המון ואף היתי שמחה מאוד להמשיך לבחון טכניקות ומסווגים שונים על הבעיה שלנו , היתי רוצה לבחון את עץ ההחלטה עם שיטת החלוקה cross validation מכיוון שבמערכת שבניתי ראיתי כי המודל שלנו הותאם יתר על המידה ולכן לא חזה טוב דוגמאות חדשות ו היתי שמחה גם לבחון את המערכת על מאגר נתונים גדול יותר , ובנוסף היתי רוצה לבחון את המערכת עם מסווג של יער אקראי מכיוון שבמאמרים שקראתי דובר רבות על יעילותו בבעיות סיווג כאלה.

לסיום, תחום למידת המכונה מסקרן אותי מאוד וללא ספק אמשיך לבחון גישות נוספות בתחום.

בביליוגרפיה

- הסבר על חישוב מדדי ביצוע של מסווג עם ספריית sklearn
- https://www.youtubeScores with sklearn.com
- עצי החלטה - רקע תיאורתי https://www.wikiwand.com
- מסווג SVC

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

- https://downloads.hindawi.com/journals/mis
- https://downloads.hindawi.com/journals/misy/2018/3860146.pdf
- https://github.com/ShubhankarRawat/Heart-Disease-Prediction