פרויקט ML חלק ב



מגישים:

אור עזר 205781396

שוהם שבת 308160456

**תוכן ענייניים**

[הכנת הנתונים לאימון ובחינת מערכת לומדת 3](#_Toc43717218)

[Decision Trees 3](#_Toc43717219)

[1. עץ החלטה מלא 3](#_Toc43717220)

[2. כיוונון פרמטרים 3](#_Toc43717221)

[3. אימון עץ החלטה מיטבי 4](#_Toc43717222)

[Neural Networks 5](#_Toc43717223)

[1. רשת ברירת מחדל 5](#_Toc43717224)

[2. כיוונון פרמטרים 5](#_Toc43717225)

[3. אימון רשת מיטבית 7](#_Toc43717226)

[K-Means 7](#_Toc43717227)

[1. מודל ברירת מחדל 7](#_Toc43717228)

[2. דיון בתוצאות האשכול 7](#_Toc43717229)

[3. אימון מודלים שונים עם ערכי K שונים 8](#_Toc43717230)

[4. ביצוע שיטת אשכול נוספת 9](#_Toc43717231)

[השוואה בין מודלים 9](#_Toc43717232)

[המודל הנבחר 10](#_Toc43717233)

[הגשת חיזויים סופיים 11](#_Toc43717234)

[סעיף בונוס 11](#_Toc43717235)

[נספחים 12](#_Toc43717236)

[נספח 1- עץ החלטה עם הקונפיגורציה הטובה ביותר שהתקבלה 12](#_Toc43717237)

[נספח 2- סיווג תצפיות סט האימון למחלקות- K Clusters 13](#_Toc43717238)

[נספח 3- טבלת פרמטרים ANN 14](#_Toc43717239)

[נספח 4- חישוב ציון סופי לכל מודל 15](#_Toc43717240)

[נספח 5- תיעוד קוד 16](#_Toc43717241)

# **הכנת הנתונים לאימון ובחינת מערכת לומדת**

חילקנו את סט הנתונים המלא לשני חלקים- 80% מהנתונים (169 רשומות) בסט האימון, ו-20% מהנתונים (43 רשומות) בסט הבחינה שישמש להשוואה בין המודלים. בחרנו בחלוקה זו מכיוון שרצינו לאזן בין הרצון שסט האימון יהיה גדול ככל האפשר ויכיל מספר רב של רשומות, לבין הרצון שסט הבחינה יכיל הרבה רשומות כדי שנקטין את הסיכויים לקבלת תוצאות קיצוניות בסט הבחינה. בנוסף, במהלך בניית המודלים וכוונון הפרמטרים נבצע תהליך של קרוס- ולידציה בעזרת k-fold כאשר k=9 על סט האימון.

# **Decision Trees**

## עץ החלטה מלא

נציג את אחוזי הדיוק הממוצעים על סט האימון וסט הבחינה:

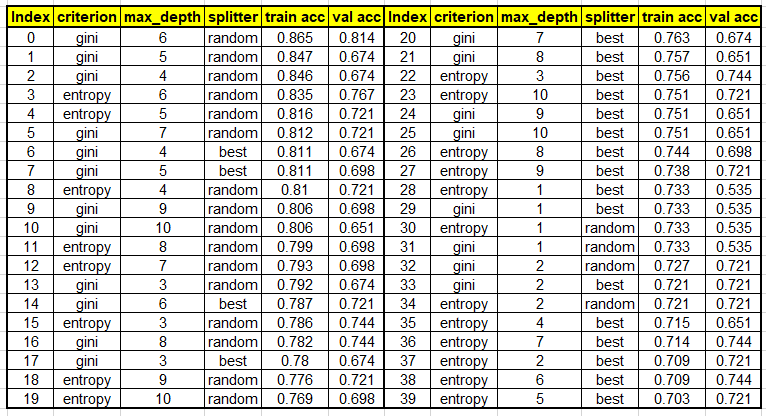
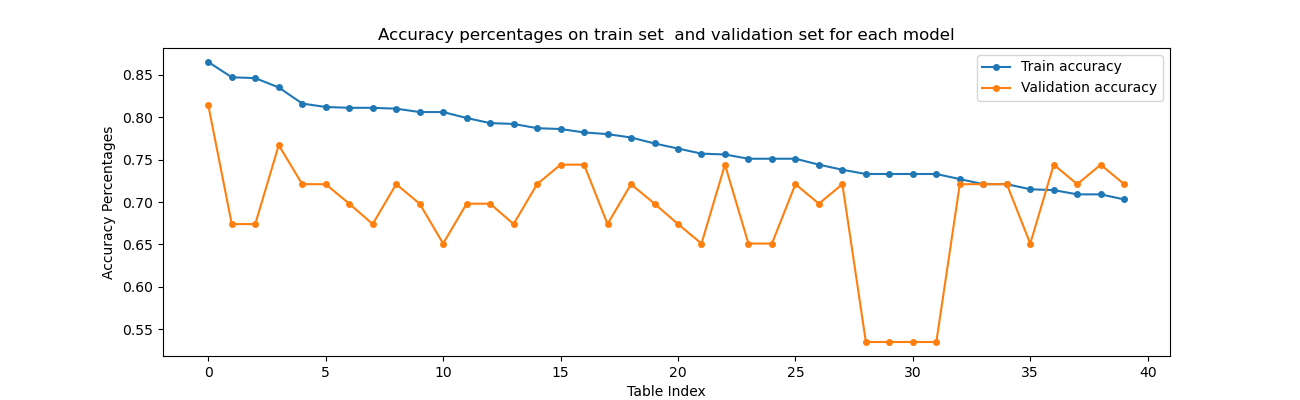
|  |  |
| --- | --- |
| אחוזי דיוק על סט האימון | 100% |
| אחוזי דיוק על סט קרוס ולידציה (k-fold) | 71.1% |
| אחוזי דיוק על סט הבחינה | 62.8% |

מנתונים אלו ניתן לראות כי יש over-fitting של המודל לסט האימון. ניתן לראות כי יש דיוק מוחלט של המודל על סט האימון שנובע מכך שאנו בונים עץ החלטה מלא- שממשיך ליצור ענפים ופיצולים עד שהוא יפריד כל רשומה בסט האימון בצורה מלאה. פועל יוצא של מצב זה הוא אחוזי דיוק נמוכים יחסית בסט הבחינה.

## כיוונון פרמטרים

הפרמטרים שבחרנו לבדוק ולכוונן הם עומק העץ, קריטריון בחירת פיצ'ר בפיצולים והשיטה לבחירת פיצול. את הכיוונון ביצענו באמצעות grid search- כלומר בדיקה של כלל המודלים האפשריים- סה"כ 40 מודלים שונים.

|  |  |  |  |
| --- | --- | --- | --- |
| פרמטר | ערכים שנבדקו | ערך נבחר | השפעת הפרמטר על עץ ההחלטה |
| עומק העץ (max\_depth) | 1-10 | 6 | עומק העץ הוא פרמטר בעל חשיבות רבה במודל. הגדלה של ערך זה מייצרת עץ בעל מספר פיצולים הולך וגדל. בדומה לסעיף הקודם, עץ גדול מידי יכול להביא למצב של over-fitting על סט האימון. מנגד, הקטנה של הערכים תביא לעצים קטנים שלא מבצעים מספיק השוואות וכך יכולות להתקבל שגיאות רבות בחיזויים. |
| קריטריון בחירת פיצ'ר בפיצולים (criterion) | "gini", "entropy" | "gini" | בחרנו לכוונן היפר-פרמטר זה מכיוון שקריטריון בחירת פיצול הוא מרכיב משמעותי במודל. שינוי של פרמטר זה ייצור סדר פיצולים שונה בעץ, ולכן יתקבלו עצים שונים בתכלית- שיכולים להביא לתוצאות שונות של המודל. |
| שיטה לבחירת פיצול (splitter) | "best", "random" | "random" | פרמטר זה קובע את מספר הפיצ'רים שייבדקו בכל פיצול. בשיטת "best"- המודל יבדוק את כל הפיצ'רים ויבחר בפיצ'ר הטוב ביותר (לפי הקריטריון שנבדק). שיטת "random" המודל בודק מספר פיצ'רים שנבחרים רנדומלית ובוחר מותכם את הפיצ'ר הטוב ביותר. בחרנו לכוונן פרמטר זה כי להחלטה זו השפעה רבה על אופן התרחבות העץ ולכן גם השפעה רבה על הביצועים שלו. |

מכיוון שאנו מכוונים 3 פרמטרים בו- זמנית נרצה להציג גרף שמציג את הקשר בין כלל האפשרויות הנבדקות לבין אחוזי הדיוק על סט האימון וסט הבחינה. לשם כך, יצרנו טבלה המציגה את אחוזי הדיוק על סט האימון וסט הבחינה עבור כל אפשרות. בנוסף, נציג גרף המציג את הנתונים כאשר ציר X- הוא אינדקס השורה בטבלה (ממוינת לפי אחוזי הדיוק על סט האימון) וציר Y הוא אחוזי הדיוק.

מהסתכלות על הגרף והטבלה יחד ניתן ללמוד מספר דברים:

1. נסתכל על אינדקסים 0,1- שני מודלים אלו בעלי אחוזי הדיוק הגבוהים ביותר על סט האימון- אולם למודל 0 יש אחוזי דיוק גבוהים משמעותית על סט הבחינה ביחס למודל 1. מהסתכלות בטבלה, ניתן לראות כי כל ההבדל בין המודלים הוא בעומק העץ- עומק 6 במודל 0, ואילו עומק 5 במודל 1. כלומר, שינוי קטן של יחידה אחת בלבד בעומק העץ- הביאה לשינוי כל כך גדול בביצועי המודל.
2. בחרנו במודל 0 למודל הסופי מכיוון שהביא לאחוזי הדיוק הגבוהים ביותר על סט האימון. ניתן לראות מהגרף כי מודל זה גם הביא לאחוזי הדיוק הגבוהים ביותר על סט הבחינה- מה שמחזק את בחירתנו במודל.

## אימון עץ החלטה מיטבי

|  |  |
| --- | --- |
| אחוזי דיוק על סט האימון | 86.5% |
| אחוזי דיוק על סט הבחינה | 81.4% |

נאמן עץ החלטה לפי הפרמטרים שהתקבלו מסעיף 2- עומק עץ 6, קריטריון ג'יני, ושיטת פיצול רנדומלית. אחוזי הדיוק שהתקבלו הם:

מתוצאות אלו ניתן לראות כי תופעת ה-over fitting שראינו בעץ המלא נחלשה או נעלמה לחלוטין. מצד אחד הדיוק על סט האימון ירד- אך הדיוק על סט הבחינה עלה משמעותית- עלייה של כמעט 20%. כלומר, כיוונון הפרמטרים הביא להקטנה של מידת ההתאמה של המודל לסט האימון- אך גרם להגדלה ביכולת ההכללה של המודל (generalization capability) ולכן המודל שהתקבל לאחר הכיוונון הוא טוב יותר.

תצוגה של העץ המלא שהתקבל ניתן לראות [בנספח 1](#_נספח_1).

הדגמת עץ ההחלטה על רשומה מסט הבחינה:

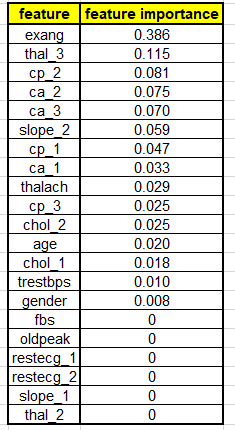
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | cp 3 | cp\_2 | cp\_1 | oldpeak | exang | thalach | fbs | tresbps | gender | age | id |
| 0 | 1 | 0 | 1.2 | 0 | 97 | 0 | 130 | 0 | 62 | 295 |
| y | chol 2 | chol 1 | thal 3 | thal 2 | ca 3 | ca 2 | ca 1 | slope 2 | slope 1 | restecg 2 | restecg 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |

1. התנאי מתקיים ולכן נמשיך לענף השמאלי
2. התנאי לא מתקיים ולכן נמשיך לענף הימני
3. התנאי מתקיים ולכן נמשיך לענף השמאלי
4. התנאי לא מתקיים ולכן נמשיך לענף הימני
5. התנאי מתקיים ולכן נמשיך לענף השמאלי
6. התנאי לא מתקיים ולכן נמשיך לענף הימני- נסווג את הרשומה לקלאס כלומר- כפי שהרשומה מתויגת במציאות.

מבנה העץ:

ממבנה העץ ניתן לראות כי המשתנים exang ו-cp נמצאים בפיצולים גבוהים של העץ (exang בשורש העץ). משתנים אלו קשורים לכאבים בחזה שחווה המטופל. אנו יודעים כי סוג הכאבים והופעה/אי הופעה של כאבים בזמן מאמץ חשובים לאבחנה של התקף לב- ומעניין לראות שגם האלגוריתם זיהה את משתנים אלו כבעלי חשיבות גבוהה.

בנוסף, 9 מתוך 22 העלים בעץ (40%) מסווגים סאמפל למחלקה 1 (heart attack), והשאר (60%) למחלקה 0 (no heart attack). זוהי סטייה ביחס לחלוקת המחלקות בסט האימון (שם 54% מסווגים למחלקה 1). הסבר אפשרי לכך הוא שהעץ לא גדל עד לעומק המלא ולכן יש שגיאות חיזוי.

חשיבות המשתנים:

נציג לכל פיצ'ר את חשיבותו בעץ ההחלטה שהתקבל (בעזרת הפונקציה feature\_importances\_). החשיבות של כל פיצ'ר מחושבת לפי ההפחתה הכוללת בערך הקריטריון (אצלנו מדד ג'יני) שנוצרה בעקבות שימוש בפיצ'ר. החשיבות מנורמלת לאחוזים כך שכל פיצ'ר מקבל ערך בין 0 ל-1, וככל שהערך גדול יותר כך חשיבות הפיצ'ר גדולה יותר. מכיוון שבזמן בניית עץ ההחלטה- בכל פיצול נבחר הפיצ'ר שמבצע את ההפחתה הגדולה ביותר- נצפה לראות כי לפיצ'רים בענפים גבוהים תהיה חשיבות גדולה יותר ביחס לפיצ'רים בתחתית העץ, והחשיבות הגבוהה ביותר תופיע אצל הפיצ'ר שבראש העץ. משתנים שלא מופעים בעץ ההחלטה יקבלו ערך 0.

ניתן לראות כי הפיצ'ר שקיבל את החשיבות הגבוהה ביותר הוא exang המעיד על קיום של כאבים בחזה בזמן מאמץ. כבר בחלק הקודם של העבודה ראינו קשר חזק בין משתנה המטרה לפיצ'ר זה- ולכן ציפינו לראות גם חשיבות גבוהה לפיצ'ר.

מנגד, פיצ'ר fbs המציג את רמת הסוכר בדם של המטופל, הראה קשר חלש למשתנה המטרה ולכן צפינו שיקבל חשיבות נמוכה. ואכן, ניתן לראות כי חשיבות הפיצ'ר היא 0- כלומר לא משתתף בעץ ההחלטה כלל.

# **Neural Networks**

## רשת ברירת מחדל

ראשית, ביצענו נרמול של הערכים לפני הרצת המודל ע"י Standard Scaler. נציג מספר פרמטרים בקונפיגורציה הדיפולטיבית:

* מספר נוירונים בשכבת הכניסה- זהו מספר הפיצ'רים בנתונים פלוס 1. סה"כ ברשת יש 14 נוירונים
* מספר שכבות חבויות ומספר נוירונים חבויים- בקונפיגורציה הדיפולטיבית ישנה שכבה אחת המכילה 100 נוירונים

|  |  |
| --- | --- |
| אחוזי דיוק על סט האימון | 99% |
| אחוזי דיוק על סט קרוס ולידציה (k-fold) | 84.1% |
| אחוזי דיוק על סט הבחינה | 67.4% |

ניתן לראות שישנה ירדה של כ- 15% בין אחוזי הדיוק של סט האימון לאחוזי הדיוק על הסט ולידציה, וירידה של כ- 32% כאשר בדקנו את המודל על סט הבחינה. כלומר ניתן לראות ירידה משמעותית, שנובעת מכך שהמודל הדיפולטיבי הינו מודל פשוט ולכן יש לכוונן את הפרמטרים כדי להתאימם לנתוני הבעיה הנחקרת, ובכדי להגיע לאחוזי דיוק גבוהים יותר בין סט האימון לסט הבחינה.

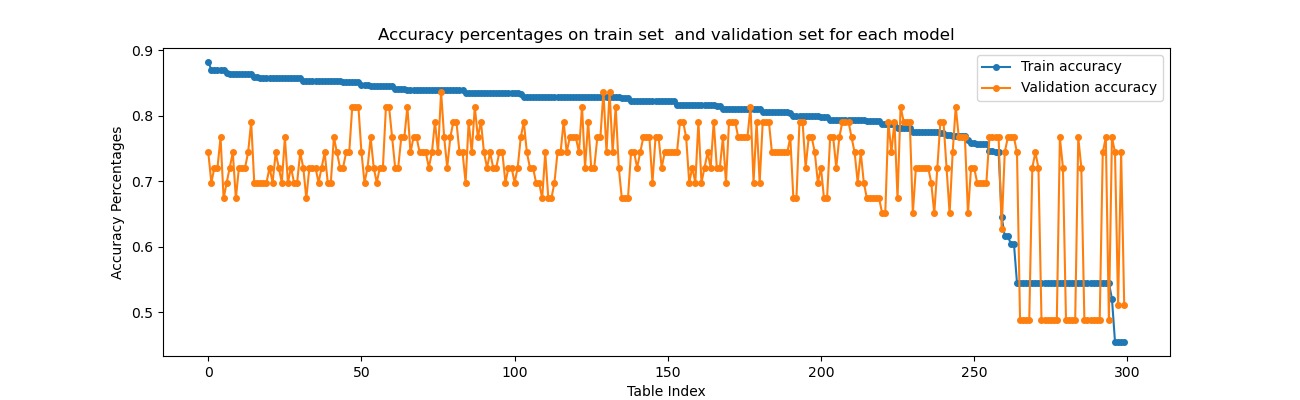
## כיוונון פרמטרים

תחילה החלטנו להשתמש בחיפוש אקראי על הפרמטרים מכוון שאנו לא ידענו אילו פרמטרים יניבו את המודל הטוב ביותר, וכדי לבדוק את שלל האפשריות נצטרך המון בדיקות שלוקחות לא מעט זמן. לכן החלטנו להשתמש בחיפוש אקראי ולקבוע מספר איטרציות- וכך נוכל לחפש בשלל האופציות בצורה אקראית בזמן סביר.

החלטנו לבחור 300 איטרציות, הפרמטרים שבדקנו :

|  |  |  |  |
| --- | --- | --- | --- |
| פרמטר | ערכים שנבדקו | השפעת הפרמטר על עץ ההחלטה | מדוע לכוון את הפרמטר |
| פונקציית אקטיבציה (activation) | logistic, tanh, relu | פונקציית הפעלה לשכבה המוסתרת כל אחת מהשיטות משפיעה על היחסים בין השכבות החבויות בצורה שונה . | הפרמטר הדיפולטיבי relu משתמש בפונקציית הפעלה לינארית מתוקנת דבר שלא בהכרח מתאים למודל שלנו, ולכן נרצה לבדוק את השיטות ההפעלה השונות. |
| מספר איטרציות מקסימלי ((max\_iter | 100-1500 בקפיצות של 25 | המודל רץ על מספר האיטרציות שנקבע או עד שיגיע להתכנסות. מספר האיטרציות יכול להשפיע במקרים בהם הוא לא מגיע להתכנסות כי המודל יעצור לאחר מספר שנקבע מראש ולא יכול לשפר את עצמו. | מספר האיטרציות הדיפולטיבי הינו 200 דבר שיכול להגביל את פיתוחו של במודל ולכן נרצה להרחיב זאת. |
| קצב למידה התחלתי (learning rate\_init) | 0.001-0.05 בקפיצות של 0.005 | קצב הלמידה הראשוני בו נעשה שימוש. זה שולט בגודל הצעד ובעדכון המשקולות. | שינוי המשקולות וקצב הלמידה יכול להשפיע על התוצאות המודל ולכן ננסה מגוון רחב של קצבי למידה ועדכון המשקולות. |
| גישה לפתרון (solver) | "lbfgs", "sgd" | פותר בשיטות שונות המבוססות על ירידת שיפוע סטוכסטית/מיטוב מבוסס-שיפוע/ /family of quasi-Newton methods | הפרמטר הדיפולטיבי adam יותר מתאים לקבוצות גדולות של נתונים 1000 דגימות ומעלה ו lbfgs- , sgd מתאימים יותר לנתונים עם מספר דגימות נמוך יותר כמו שלנו. |
| מספר שכבות חבויות ומספר נוירונים חבויים (hidden layer\_size) | 1-4 שכבות,  בכל שכבה ישנם 3-9 נוירונים | מספר הנוירונים בכל שכבה ומספר השכבות משפיעות על מספר הקשרים בין השכבות. | הפרמטר הדיפולטיבי הוא שכבה אחת המכילה 100 נוירונים , רצינו לבדוק אם שכבות נוספות בגדלים שונים יוכלו להעמיק את הקשרים במודל ולשפרו. |

מכיוון שאנו מכוונים 5 פרמטרים בו- זמנית נרצה להציג גרף שמציג את הקשר בין כלל המודלים שנבדקו לבין אחוזי הדיוק על סט האימון וסט הבחינה. לשם כך, יצרנו טבלה המציגה את אחוזי הדיוק על סט האימון וסט הבחינה עבור כל אפשרות. בנוסף, נציג גרף המציג את הנתונים כאשר ציר X- הוא אינדקס השורה בטבלה (ממוינת לפי אחוזי הדיוק על סט אימון מהגבוהה לנמוך) וציר Y הוא אחוזי הדיוק.



ניתן לראות שישנם תוצאות בטווח גדול בין 0.882 – 0.455 אחוזי דיוק על סט האימון, ותוצאות בטווח גדול בין 0.837 – 0.488 אחוזי דיוק על סט הוולידציה ([לנספח 3- טבלת פרמטרים](#_נספח_3-_טבלת)).

## אימון רשת מיטבית

הפרמטרים של המודל הנבחר:

גישה לפתרון (solver) - sgd

מספר איטרציות מקסימלי (max\_iter) – 1350

פונקציית אקטיבציה (activation) - relu

קצב למידה התחלתי (learning\_rate\_init) – 0.016

מספר שכבות חבויות ומספר נוירונים חבויים (hidden\_layer\_size)- (6,6) – 2 שכבות של 6 נוירונים בכל אחת .

מספר נוירונים בשכבת הכניסה- זהו מספר הפיצ'רים בנתונים פלוס 1. סה"כ ברשת יש 14 נוירונים.

ניתן לראות שרוב הפרמטרים השתנו חוץ מפונקציית האקטיבציה.

|  |  |
| --- | --- |
| אחוזי דיוק על סט קרוס ולידציה (k-fold) | 88.2% |
| אחוזי דיוק על סט הבחינה | 62.8% |

השינוי באחוזי הדיוק על סט הקרוס ולידציה הם : שיפור ב- 4.1%.

השינוי באחוזי הדיוק על סט הבחינה הם : שיפור ב- 7%.

ניתן להסיק ששינוי הפרמטרים שיפר את האחוזים על סט האימון וסט הבחינה מכוון שבחנו 300 אופציות אפשריות למודלים שונים עם פרמטרים שונים ובכך ניסנו להגיע לרשת מתאימה יותר לנתונים.

אך בשל העבודה שסט הבחינה המורכב מ- 20% מכלל הדגימות, יכול להיווצר מצב שכל שינוי קטן בפרמטרים יכול לשנות את אחוזי הדיוק בו. לכן כאשר נבדוק את מודל זה על כמות גדולה של נתונים או על נתונים אחרים ישנה אפשרות שנראה שיפור גדול מהמודל הדיפולטיבי.

# **K-Means**

## מודל ברירת מחדל

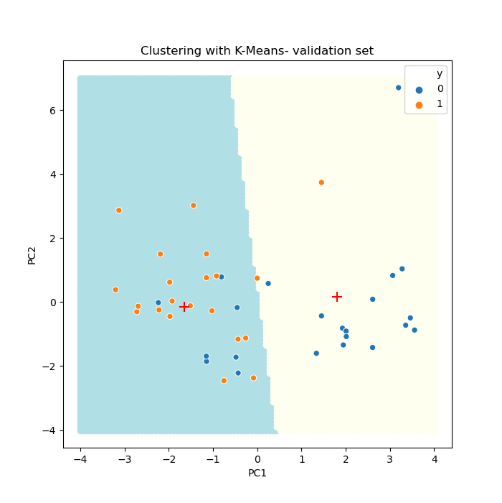
לפני הרצת המודל, ביצענו PCA על סט האימון וסט הבחינה כדי להוריד את הממדים ל-2 ממדים בלבד. מהסתכלות בתרשימי הפיזור שהתקבלו ניתן לראות כי פיזור התצפיות לפי כל מחלקה שונה מאוד בין הסטים: בסט האימון מרבית התצפיות שמסווגות כ- "0" נמצאות במד השמאלי של הגרף ואילו תצפיות שמסווגות כ- "1" נמצאות בצד הימני. בסט הבחינה המצב הוא הפוך. אנו צופים כי הבדל זה יביא לביצועים יחסית נמוכים של המודל.

## דיון בתוצאות האשכול

|  |  |
| --- | --- |
| 8 | 69 |
| 73 | 19 |

בגרף הבא ניתן לראות את תוצאות האשכול של האלגוריתם לצד הסיווג למחלקות האמיתיות של התצפיות. כל התצפיות שנמצאות בשטח הכחול- סווגו ע"י האלגוריתם למחלקה 0, והתצפיות בשטח הלבן למחלקה 1. מרכז מח' 0 הוא (-1.655, -0.155) ומרכז מח' 1 הוא (1.798, 0.168 ניתן לראות כי מרבית החיזויים של האלגוריתם בוצעו בצורה נכונה. אחוזי הדיוק של האלגוריתם על סט האימון הם 84%.

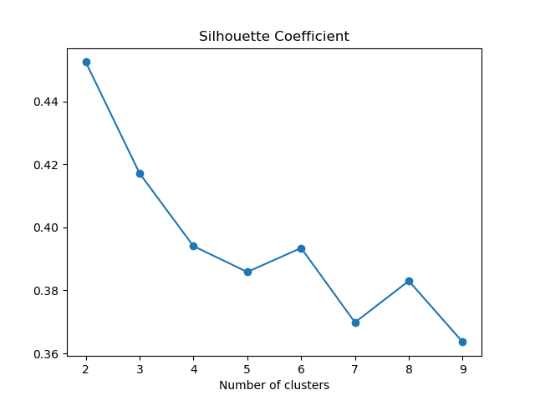
מטריצת המבוכה על סט האימון (169 רשומות):

אולם, ניתן לראות כי בסט הבחינה תוצאות האשכול אינן תואמות את הסיווג למחלקות המקוריות. אנו חושבים כי תוצאות אלו נובעות מההבדל בפיזור התצפיות לפי מחלקות בין סט האימון וסט הבחינה- כפי שהוסבר בסעיף הקודם. אחוזי הדיוק על סט האימון הם 18.6% בלבד.

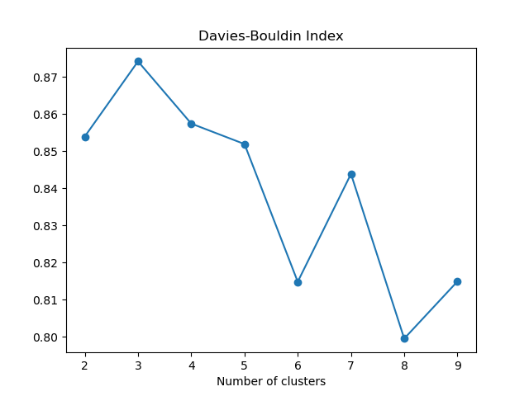
|  |  |
| --- | --- |
| 15 | 7 |
| 1 | 20 |

מטריצת המבוכה על סט הבחינה (43 רשומות):

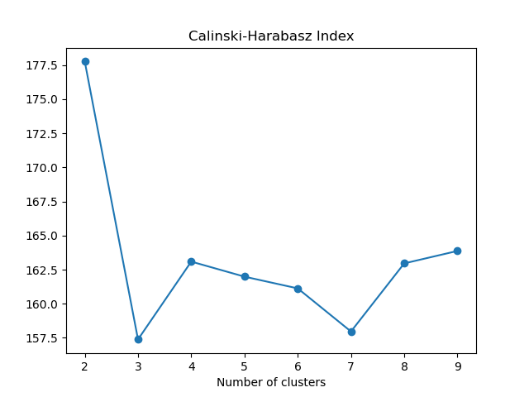
## אימון מודלים שונים עם ערכי K שונים

כעת, במנותק מסיפור המקרה, נאמן 8 מודלים כאשר בכל אחד מהם מספר מחלקות שונה- בין 2 ל-9 מחלקות. גרפים המציגים את סיווג התצפיות למחלקות בכל מודל מצורף [בנספח 2](#_נספח_2-_סיווג).

בכדי לבחור את מספר המחלקות הטוב ביותר (ערכו של K) נסתכל על 3 מדדים שאינם תלויים בידיעה מוקדמת של מספר המחלקות:

1. **Silhouette Coefficient-** מדד זה מתאים לכל רשומה ערך המורכב מהמרחק הממוצע של הרשומה מכל שאר הרשומות שסווגו לאותה המחלקה של הרשומה (a), ובנוסף מהמרחק הממוצע של הרשומה מכל שאר הרשומות שסווגו למחלקה הקרובה ביותר לרשומה (b). ערך המדד לכל תצפית הוא- . ערך המדד הסופי הוא ממוצע של כל הערכים שהתקבלו עבור כל התצפיות. זהו מדד שנרצה למקסם- כלומר נרצה לבחור ב-K שיביא את הערך הגבוה ביותר.

**לפי מדד זה נבחר בשתי מחלקות.**

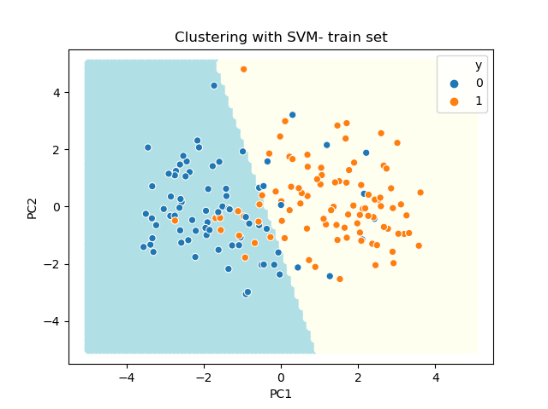
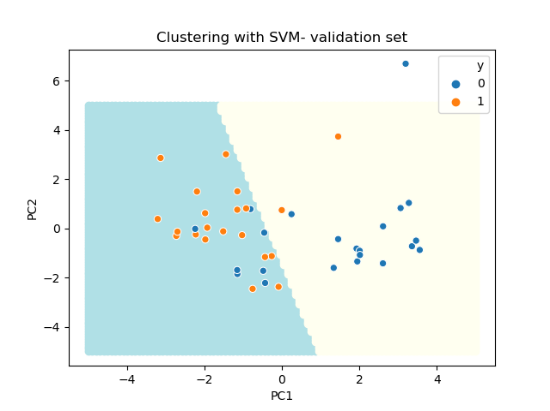
1. **Davies-Bouldin Index-** מדד זה מתאים לכל מודל ערך המעיד על רמת הדמיון בין המחלקות (לפי שקלול של מרחק בין המחלקות וגודלן של המחלקות). זהו מדד שנרצה למזער- כלומר נרצה לבחור ב-K שיביא את הערך הנמוך ביותר. **לפי מדד זה נבחר בשמונה מחלקות.**
2. **Calinski-Harabasz Index-** מדד זה מחשב לכל מודל את היחס שבין הפיזור בין המחלקות לבין היחס של התצפיות בתוך כל מחלקה. (הפיזור מחושב על ידי סכום ריבועי המרחקים). נקרא גם Variance Ratio Criterion. זהו מדד שנרצה למקסם- כלומר נרצה לבחור ב-K שיביא את הערך הגבוה ביותר. **לפי מדד זה נבחר בשתי מחלקות.**

מניתוח התוצאות עולה כי לפי שניים מהמדדים נרצה לבחור בשתי מחלקות, בעוד לפי מדד אחד נרצה לבחור בשמונה מחלקות. בנוסף, ניתן לראות כי ההפרש בערכים עבור 2 ו-8 מחלקות עבור המדדים התומכים ב-2 מחלקות הוא משמעותי יותר (הפרש של כ-15% במדד Silhouette והפרש של כ-8.5% במדד Calinski-Harabasz) לעומת ההפרש בערכים במדד התומך ב-8 מחלקות (הפרש של כ- 5.8% במדד Davies-Bouldin). לכן נבחר בשתי מחלקות- כלומר K=2- כפי שמתקיים בסט הנתונים האמיתי.

## ביצוע שיטת אשכול נוספת

|  |  |  |  |
| --- | --- | --- | --- |
| פרמטר | ערכים שנבדקו | ערך נבחר | השפעת הפרמטר אופן הסיווג |
| פונקציית קרנל "kernel" | "linear", "poly", "rbf", "sigmoid" | "linear" | לפונקציית קרנל חשיבות רבה באופן יצירת ההפרדה בין 2 המחלקות. שינוי של הפונקציה יביא ליצירת מישור הפרדה שונה ולכן לביצועים שונים של המודל- ולכן נרצה לבחון באיזו פונקציה הכי מתאים להשתמש בבעיה שלנו. |
| פרמטר ויסות (Regularization Parameter)  "C" | 0.05-3.5 בקפיצות של 0.5 | 0.55 | פרמטר זה נותן "קנס" לקלסיפיקציה שגויה של המודל. כך, ככל שערך הפרמטר גדל- קו ההפרדה ישתנה כך שמרבית התצפיות יסווגו כראוי. |
| מקדם קרנל "gamma"  (רלוונטי עבור פונקציות קרנל "poly", "rbf", "sigmoid" בלבד) | "scale", "auto" | "scale" | פרמטר זה מגדיר את מידת ההשפעה של כל סאמפל בודד. "scale" נותן לכל סאמפל משקל השווה ל- . "auto" נותן לכל סאמפל משקל השווה ל- . מכיוון שלבסוף נבחרה פונקציית קרנל לינארית- בחירת פרמטר זה חסרת חשיבות (הפרמטר לא חלק מנוסחת הפונקציה) |

החלטנו לבצע אשכול בעזרת SVM. ראשית נכוונן 3 היפר- פרמטרים של המודל באמצעות grid search:

ניתן לראות בגרף את סכמת האשכול שהתקבלה עבור המודל הטוב ביותר יחד עם המחלקות המקוריות של הנתונים בסט האימון. ניתן לראות כי ישנה הפרדה יחסית טובה בין המחלקות. עם זאת, גם במודל זה אחוזי הדיוק על סט הבחינה נמוכים מאוד. גם במקרה זה, אנו מעריכים כי אחוזי הדיוק הנמוכים נובעים בעיקר מכך שסט הבחינה שונה מאוד מסט האימון.

|  |  |
| --- | --- |
| אחוזי דיוק על סט האימון | 86.3% |
| אחוזי דיוק על סט הבחינה | 20.9% |

# **השוואה בין מודלים**

למשימת הסיווג נעדיף DT או ANN לעומת K-Means מכיוון שאלו אלגוריתמים ללמידה מונחית- בה אנו יודעים מה כמות מחלקות משתנה המטרה ומה הסיווג המחלקתי של כל רשומה בסט האימון. זהו מידע חשוב שלא קיים בלמידה הלא מונחית שמאפשר לאמן מודל טוב יותר. יתרון נוסף של הלמידה המונחית היא שישנה משמעות ידועה מותאמת לבעיה הנחקרת לכל מחלקה, בעוד אלגוריתמי הלמידה הלא מונחית כדוגמת K-Means, לא מספקים את המשמעות לקלאסטר. כך למשל בבעיה שלנו- אנו נוכל לחלק את הנתונים ל-2 קבוצות שונות- אך לא נדע האם קלאסטר מסוים זה אדם עם התקף לא או ללא התקף לב.

בכדי להשוות בין כל המודלים בחרנו לבחון שני ערכים: אחוז הדיוק על סט הבחינה ושקלול החיזויים השגויים לפי מטריצות המבוכה של כל מודל. מכיוון שאנו עוסקים בעולם תוכן רפואי- יש משמעות רבה יותר לפספוס אבחנה בו מטופל מסווג כאדם שלא חווה התקף לב- למרות שאכן חווה התקף לב (under treatment, תא שמאלי תחתון במטריצת מבוכה- מסומן באדום), מאשר למצב ההפוך- בו מטופל מסווג כאדם שחווה התקף לב אך בפועל לא חווה התקף לב (over treatment, תא ימני עליון במטריצת מבוכה). לכן, נרצה לתת משקלים שונים לכל סוג טעות- 125% לטעות מהסוג הראשון החמורה יותר, ו-75% לטעות השנייה שהיא פחות חמורה.

נוסחת החישוב:

הציון הסופי של כל מודל יהיה ממוצע בין אחוז הדיוק על סט הבחינה ואחוז הדיוק לאחר שקלול הטעויות:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מודל | אחוזי דיוק על סט האימון | אחוזי דיוק על סט הבחינה | אחוזי דיוק טעויות משוקללות | ציון סופי ([נספח 4](#_נספח_4-_חישוב)) |
| DT | 86.5% | 81.4% | 83.7% | 82.5% |
| ANN | 88.2% | 74.4% | 76.1% | 75.2% |
| K-Means | 84% | 18.6% | 15.7% | 17.1% |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ANN** | | **DT** | | **k-means** | |
| 7 | 15 | 6 | 16 | 15 | 7 |
| 17 | 4 | 19 | 2 | 1 | 20 |

מטריצות מבוכה של כל מודל:

ניתן לראות ממטריצות המבוכה כי למודל עץ ההחלטה בעל מספר כולל של טעויות הקטן ביותר ובנוסף המספר הקטן ביותר של טעויות מסוג under treatment- מה שמסביר את הציון הסופי הגבוה שקיבל המודל. לעומתו, למודל ה- K-Means ציון סופי נמוך במיוחד. ציון זה נובע מכך שלמודל מספר גדול משמעותית של טעויות חיזוי ביחס לשני המודלים האחרים ובעיקר מספר רב של טעויות ה- under treatment (כמעט 50%).

לסיכום, מכיוון שמודל ה-DT קיבלת את הציון הגבוה ביותר נמליץ עליו כמודל הנבחר לביצוע החיזויים.

# **המודל הנבחר**

המודל הסופי שבחרנו להשתמש בו לביצוע החיזויים הסופיים הוא עץ ההחלטה כפי שהתקבל בסעיפים הקודמים. הפרמטרים של המודל הם:

עומק עץ- 6, קריטריון- מדד ג'יני, שיטת פיצול- רנדומלית

|  |  |
| --- | --- |
| 6 | 16 |
| 19 | 2 |

כזכור, ערכי הפרמטרים נקבעו בשיטה של grid search בשלב כיוונון הפרמטרים של עץ ההחלטה. ערכים אלו הביאו לאחוזי הדיוק הגבוהים ביותר על סט האימון. (הסבר מפורט של הפרמטרים נמצא בסעיף "[כוונון פרמטרים](#_כיוונון_פרמטרים)" בפרק עץ ההחלטה).

מטריצת המבוכה של המודל:

לפי מטריצת המבוכה ניתן לראות כי למודל 8 טעויות על סט הבחינה מתוך 43. מתוכן, רק 2 טעויות מסוג ה-under treatment בו אדם מאובחן ללא התקף לב- למרות שהוא חווה בפועל התקף לב. כפי שראינו בסעיף הקודם, למודל שבחרנו המספר הקטן ביותר של טעויות מסוג זה מבין כל המודלים שנבחנו. מסקנה זו מחזקת את בחירתנו במודל זה למודל הסופי על פני כל שאר המודלים שנבדקו.

יתרון נוסף של עץ ההחלטה הוא, שנוכל בקלות יחסית להסביר את ההחלטות שמבצע המודל לאנשים שאינם מומחים בעולם הלימוד מכונה- כדוגמת הצוות הרפואי המטפל.

# **הגשת חיזויים סופיים**

החיזויים הסופיים לסט x\_test בוצעו ע"י עץ ההחלטה (המודל הנבחר). תוצאות החיזויים מפורטים בקובץ y\_test.

# **סעיף בונוס**

החלטנו לאמן מודל Random Forest על הבעיה שלנו ולבחון אם מודל זה יהיה טוב יותר מהמודל הסופי שבחרנו (עץ ההחלטה). לטובת האימון, החלטנו לבצע כיוונן של מספר היפר-פרמטרים בעזרת 100 איטרציות של random search. פירוט הערכים שנבדקו מוצגים בטבלה הבאה:

|  |  |  |  |
| --- | --- | --- | --- |
| פרמטר | ערכים שנבדקו | ערך נבחר | השפעת הפרמטר על עץ ההחלטה |
| מספר עצים  n\_estimators)) | 10-100 בקפיצות של 10 | 20 | למספר עצי ההחלטה ביער השפעה רבה על ביצועי המודל ולכן בחרנו לכוונן פרמטר זה. ככל שמספר העצים גדל- לכל סיווג של עץ בודד השפעה נמוכה יותר על ההחלטה הסופית של המודל. |
| עומק העצים (max\_depth) | 1-10 | 10 | עומק העצים הוא פרמטר בעל חשיבות רבה במודל. הגדלה של ערך זה מייצרת עצים בעלי מספר פיצולים הולך וגדל. עצים גדולים מידי יכולים להביא למצב של over-fitting על סט האימון. מנגד, הקטנה של הערכים תביא לעצים קטנים בעלי טעויות חיזוי רבות. על סמך התוצאות של כיוונון עץ ההחלטה בסעיפים הקודמים- נבדוק תחום ערכים דומה. |
| קריטריון בחירת פיצ'ר בפיצולים (criterion) | "gini", "entropy" | "gini" | בחירת פיצול הוא מרכיב משמעותי במודל ולכן נבצע כיוונון שלו. שינוי של פרמטר זה ייצור סדר פיצולים שונה בעצים, ולכן יתקבלו עצים שונים- שיביאו להחלטות שונות של המודל. |
| מספר פיצ'רים לבדיקה בכל פיצול ((max\_features | "sqrt", "log2" | "log2" | פרמטר זה קובע את מספר הפיצ'רים שייבדקו בכל פיצול. בשיטת "sqrt"- המודל יבדוק מספר פיצ'רים השווה לשורש מספר הפיצ'רים בנתונים. בשיטת "log2"- יבדוק מספר פיצ'רים השווה ללוג2 על מספר הפיצ'רים בנתונים. בחרנו לכוונן פרמטר זה כי להחלטה זו השפעה רבה על אופן התרחבות העצים ולכן גם השפעה רבה על ההחלטה הסופית של היער. |

ביצועי המודל:

|  |  |
| --- | --- |
| אחוזי דיוק על סט האימון | 84.6% |
| אחוזי דיוק על סט הבחינה | 72.1% |

|  |  |
| --- | --- |
| 8 | 14 |
| 17 | 4 |

מטריצת מבוכה:

ציון סופי של המודל: 79.5%

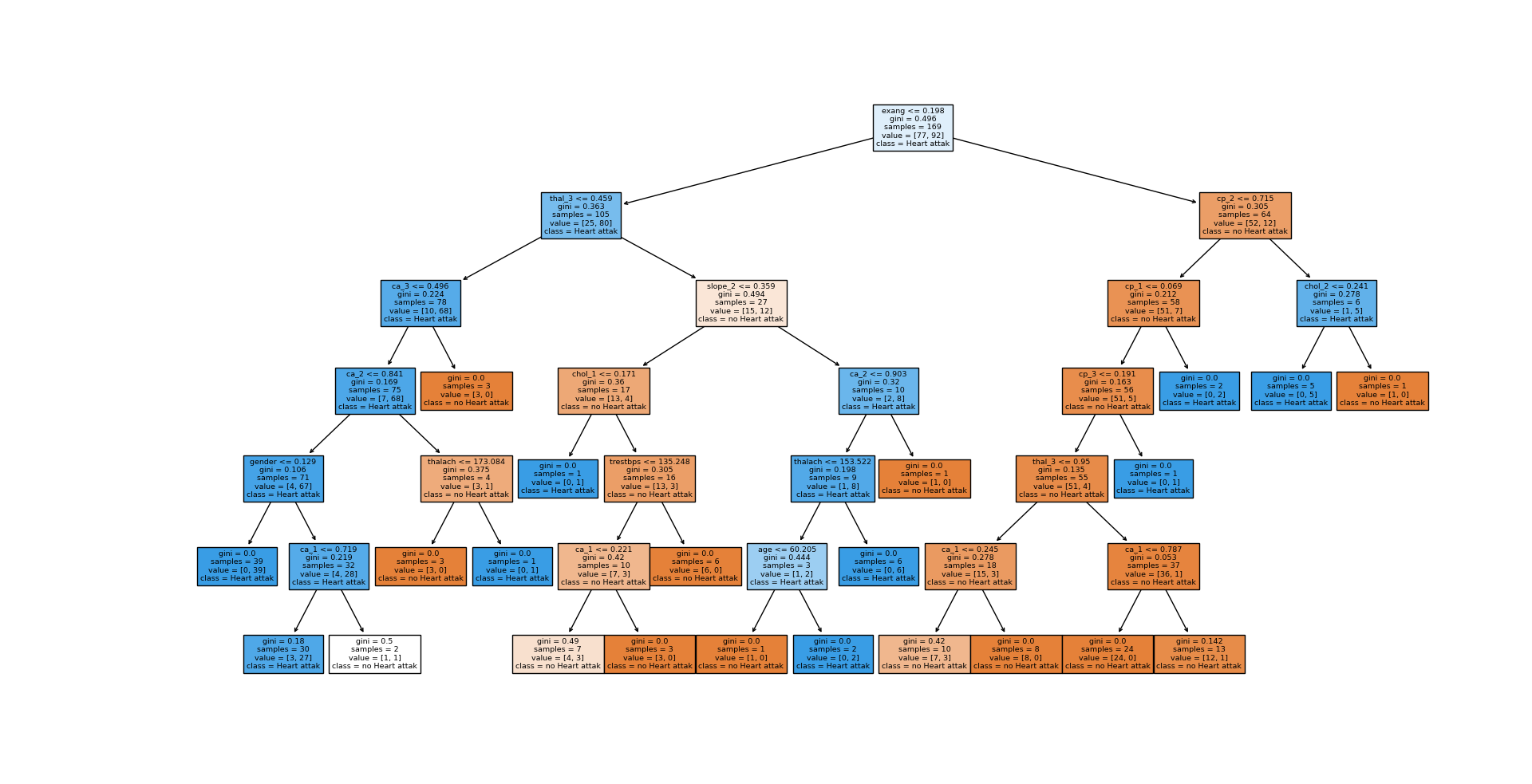
ניתן לראות כי למודל ה-random forest מספר גדול יותר של טעויות under treatment מאשר מודל עץ ההחלטה שאימנו. בנוסף, למרות שלמודל ה- random forest אחוזי דיוק גבוהים יותר על סט האימון- הציון הסופי שקיבל המודל על פי החישוב מהסעיף "השוואה בין מודלים" נמוך יותר.

לדעתנו, ניתן לשפר את ביצועי ה- random forest ולעקוף את ביצועי מודל עץ ההחלטה ע"י כיוונון של פרמטרים נוספים והגדלה של מספר האיטרציות ב- random search או מעבר לשימוש ב- grid search. לטובת זאת, נדרשים זמן עבודה נוסף ובעיקר כוח חישוב גדול יותר.

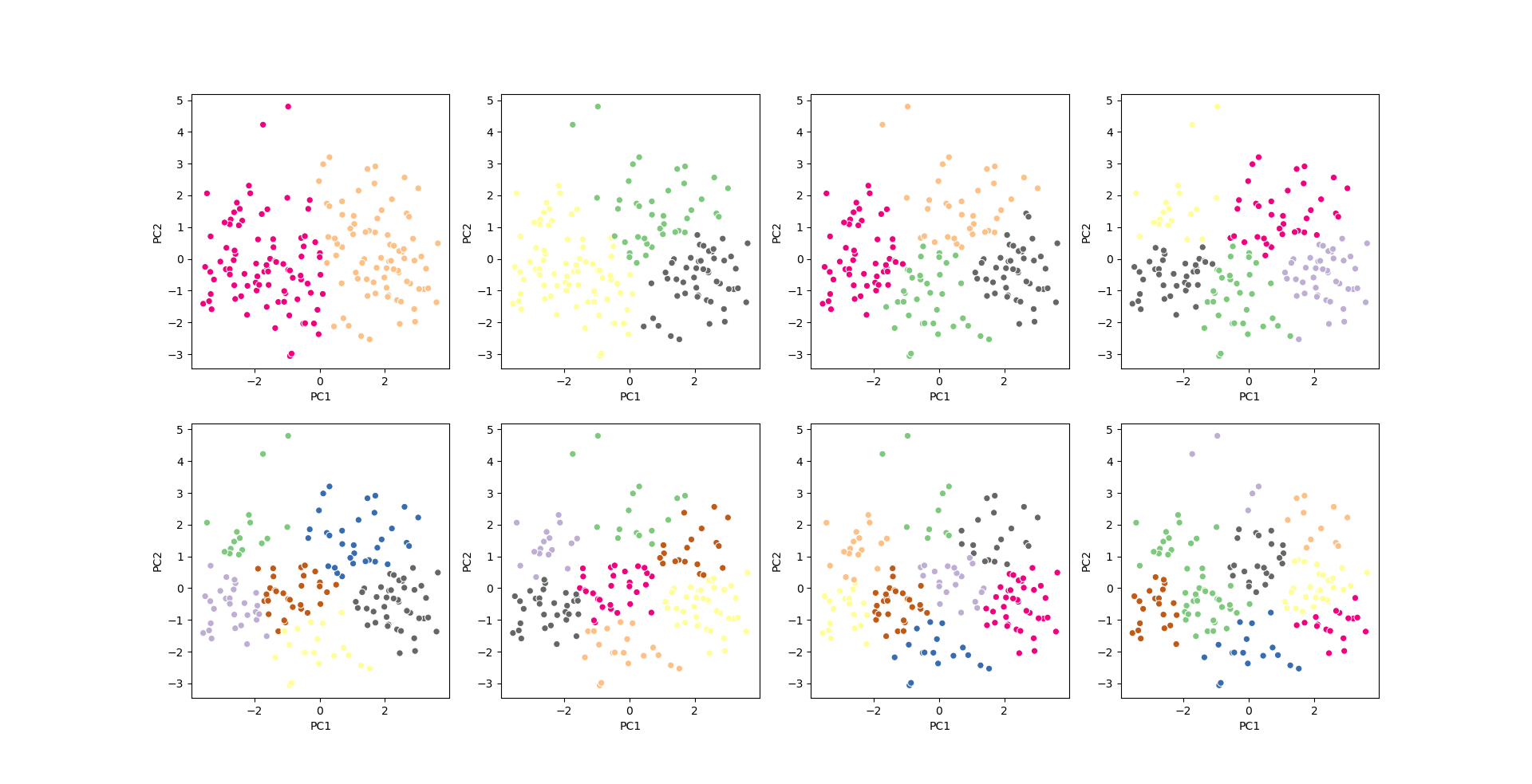
# **נספחים**

## נספח 1- עץ החלטה עם הקונפיגורציה הטובה ביותר שהתקבלה

([לחזרה לעמוד](#_אימון_עץ_החלטה))



## נספח 2- סיווג תצפיות סט האימון למחלקות- K Clusters

([לחזרה לעמוד](#_אימון_מודלים_שונים))

## נספח 3- טבלת פרמטרים ANN

([חזרה לעמוד](#_כיוונון_פרמטרים_1))

ישנם 300 רשומות בטבלה זו בחרנו להציג את ה 25 הכי טובות מבחינת אחוזי דיוק על האימון:



## נספח 4- חישוב ציון סופי לכל מודל

([חזרה לעמוד](#_השוואה_בין_מודלים))

DT:

:ANN

:K-Means

## נספח 5- תיעוד קוד

import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
import seaborn as sns  
from matplotlib import ticker  
from scipy.stats import chi2\_contingency  
from statsmodels.graphics.mosaicplot import mosaic  
from scipy.stats import multivariate\_normal  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.tree import plot\_tree  
from sklearn.model\_selection import GridSearchCV  
from sklearn.model\_selection import KFold  
from sklearn.model\_selection import RandomizedSearchCV  
from sklearn.metrics import accuracy\_score  
from sklearn.metrics import confusion\_matrix  
from sklearn.neural\_network import MLPClassifier  
from sklearn.preprocessing import StandardScaler, MinMaxScaler  
from sklearn.cluster import KMeans  
from sklearn.decomposition import PCA  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import silhouette\_score, davies\_bouldin\_score, calinski\_harabasz\_score  
from tqdm import tqdm  
from sklearn.svm import SVC  
from sklearn.ensemble import RandomForestClassifier  
  
###-------------------Create the data frame-------------------------------###  
  
data=pd.read\_csv('C:/Users/shoha/Desktop/ML\_part2/Xy\_train.csv') #read the excel file from location on pc  
results=pd.read\_csv('C:/Users/shoha/Desktop/ML\_part2/results.csv') #read the excel file from location on pc  
results\_ann=pd.read\_csv('C:/Users/shoha/Desktop/ML\_part2/results\_ann.csv') #read the excel file from location on pc  
  
###-------------------Change invalid values -------------------------------###  
  
np.random.seed(356) #seed from normal distribution  
age120=data[data['age']<120] #subset age under 120  
agemean =np.round\_(np.mean(age120['age'])) #mean to all age under 120  
agestd = np.std(age120['age']) #std to all age under 120  
for i in data.index: #changing ages over 120 to random number fro normal distribution with agemean, agestd  
 if (data['age'].values[i] > 120):  
 data['age'].values[i] = np.round\_(np.random.normal(agemean, agestd, 1))  
  
###-------------------Changing missing values -------------------------------###  
  
#CA  
data.loc[data['ca'] == 4,'ca'] = 0  
  
#Thal  
data.loc[data['thal'] == 0,'thal'] = 2  
  
###-------------------Changing to dummies -------------------------------###  
  
#Chol changing to categorical variable  
data.loc[data['chol'] <= 200, 'chol'] = 0 #change chol to 0\1 , 0= normal 1=high  
data.loc[data['chol'] > 300, 'chol'] = 2 #change chol to 0\1 , 0= normal 1=high  
data.loc[data['chol'] > 200,'chol'] = 1 #change chol to 0\1 , 0= normal 1=high  
  
dataDummies = pd.get\_dummies(data, columns=['cp', 'restecg', 'slope', 'ca', 'thal' , 'chol'], drop\_first=True)  
dataDummies=dataDummies.drop(['id'], axis=1)  
  
for col in dataDummies.columns:  
 print(col)  
  
###---------------------------- Validation set & Train set -----------------------------###  
  
X\_train = dataDummies.drop(['y'], axis=1).values  
y\_train = dataDummies['y'].values  
X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.2, random\_state=123)  
  
print("val")  
count=0  
for i in np.arange(0,len(y\_val)):  
 if (y\_val[i]==1):  
 count=count+1  
print(count)  
print(len(y\_val))  
  
print("train")  
count2=0  
for i in np.arange(0,len(y\_train)):  
 if (y\_train[i]==1):  
 count2=count2+1  
print(count2)  
print(len(y\_train))  
  
###---------------------------- Kfold 9 - Full Max Depth Tree -----------------------------###  
  
kfold = KFold(n\_splits=9, shuffle=True, random\_state=123)  
  
DT\_res = pd.DataFrame()  
for train\_idx, val\_idx in kfold.split(X\_train):  
 modelDT = DecisionTreeClassifier(criterion='entropy', random\_state=123)  
 modelDT.fit(X\_train[train\_idx], y\_train[train\_idx])  
 accTrain=accuracy\_score(y\_true=y\_train[train\_idx], y\_pred=modelDT.predict(X\_train[train\_idx]))  
 accVal = accuracy\_score(y\_train[val\_idx], modelDT.predict(X\_train[val\_idx]))  
 DT\_res = DT\_res.append({'accVal': accVal , 'accTrain' : accTrain}, ignore\_index=True)  
  
print("Max Depth Tree Performances:")  
print(round(DT\_res,3))  
print(round(DT\_res.mean(),3))  
  
preds\_DT = modelDT.predict(X\_val)  
print("Max Depth Tree- Validation accuracy: ", round(accuracy\_score(y\_val, preds\_DT), 3))  
print()  
  
###--------------------------------------Hyperparameter tuning - Tree-------------------------------###  
  
DTGrid= {'max\_depth' : np.arange(1, 11, 1),  
 'criterion' : ['gini', 'entropy'],  
 'splitter' : ['best', 'random']}  
  
grid\_search = GridSearchCV(estimator=DecisionTreeClassifier(random\_state=42), param\_grid=DTGrid, refit=True, cv=9)  
grid\_search.fit(X\_train, y\_train)  
  
#Show Results  
#results\_dt=pd.DataFrame(grid\_search.cv\_results\_)  
#results\_dt=results\_dt.drop(['mean\_fit\_time','std\_fit\_time','mean\_score\_time','std\_score\_time','rank\_test\_score'],1)  
#results\_dt.to\_csv("C:/Users/shoha/Desktop/results.csv")  
  
print("Best DT Hyper Parameters: ", grid\_search.best\_params\_)  
print("Best DT Score: ", round(grid\_search.best\_score\_,3))  
  
best\_model = grid\_search.best\_estimator\_  
preds\_DT = best\_model.predict(X\_val)  
print("DT- Validation accuracy: ", round(accuracy\_score(y\_val, preds\_DT), 3))  
  
DTGrid\_df = pd.DataFrame(columns=['index','max\_depth', 'criterion', 'splitter', 'train ac', 'val ac'])  
criterion\_list = {'gini', 'entropy'}  
splitter\_list={'best', 'random'}  
  
print("confusion matrix DT:")  
print(confusion\_matrix(y\_true=y\_val, y\_pred=preds\_DT))  
  
  
for j in np.arange(0,40,1):  
 model\_csi = DecisionTreeClassifier(criterion=results['param\_criterion'][j], splitter=results['param\_splitter'][j], max\_depth=results['param\_max\_depth'][j], random\_state=42)  
 model\_csi.fit(X\_train, y\_train)  
 preds\_csi = model\_csi.predict(X\_val)  
 DTGrid\_df.at[j, 'index'] = j  
 DTGrid\_df.at[j, 'max\_depth'] = results['param\_max\_depth'][j]  
 DTGrid\_df.at[j, 'criterion'] = results['param\_criterion'][j]  
 DTGrid\_df.at[j, 'splitter'] = results['param\_splitter'][j]  
 DTGrid\_df.at[j, 'train ac'] = round(results['mean\_test\_score'][j], 3)  
 DTGrid\_df.at[j, 'val ac'] = round(results['val ac'][j], 3)  
 #DTGrid\_df.at[j, 'val ac'] = round(accuracy\_score(y\_val, preds\_csi), 3)  
 #results.at['val ac'][j]= round(accuracy\_score(y\_val, preds\_csi), 3)  
  
#results.to\_csv("C:/Users/shoha/Desktop/results.csv")  
print(DTGrid\_df)  
plt.figure(figsize=(13, 4))  
plt.plot(DTGrid\_df['index'], DTGrid\_df['train ac'], marker='o', markersize=4)  
plt.plot(DTGrid\_df['index'], DTGrid\_df['val ac'], marker='o', markersize=4)  
plt.legend(['Train accuracy', 'Validation accuracy'])  
plt.title('Accuracy percentages on train set and validation set for each model')  
plt.xlabel('Table Index')  
plt.ylabel('Accuracy Percentages')  
plt.show()  
  
###---------------------------------------------------print trees--------------------------------------------------------------##  
  
plt.figure(figsize=(20, 10))  
plot\_tree(best\_model, filled=True, max\_depth=3,class\_names=['no Heart attak', 'Heart attak'], feature\_names=['age', 'gender', 'trestbps','fbs','thalach','exang' ,  
 'oldpeak','cp\_1','cp\_2','cp\_3','restecg\_1','restecg\_2',  
 'slope\_1','slope\_2','ca\_1','ca\_2','ca\_3', 'thal\_2','thal\_3',  
 'chol\_1','chol\_2'])  
plt.show()  
  
plt.figure(figsize=(20, 10))  
plot\_tree(best\_model, filled=True, class\_names=['no Heart attak', 'Heart attak'], feature\_names=['age', 'gender', 'trestbps','fbs','thalach','exang' ,  
 'oldpeak','cp\_1','cp\_2','cp\_3','restecg\_1','restecg\_2',  
 'slope\_1','slope\_2','ca\_1','ca\_2','ca\_3', 'thal\_2','thal\_3',  
 'chol\_1','chol\_2'])  
plt.show()  
  
###---------------------------------------------------features importances---------------------------------------------##  
  
print("features:")  
print(best\_model.n\_features\_)  
  
print('features importances:')  
print(best\_model.feature\_importances\_)  
  
###---------------------------------------------------ANN-scale values--------------------------------------------------------------##  
  
scaler = StandardScaler()  
X\_train\_scaled = scaler.fit\_transform(X\_train)  
X\_val\_scaled = scaler.fit\_transform(X\_val)  
  
###---------------------------------------------------ANN-default network--------------------------------------------------------------##  
  
ANNres = pd.DataFrame()  
for train\_idx, val\_idx in kfold.split(X\_train\_scaled):  
 modelANN= MLPClassifier(activation='relu', alpha=0.0001, batch\_size='auto', beta\_1=0.9,  
 beta\_2=0.999, early\_stopping=False, epsilon=1e-08,  
 hidden\_layer\_sizes=(100,) , learning\_rate='constant',  
 learning\_rate\_init=0.001, max\_fun=15000, max\_iter=200,  
 momentum=0.9, n\_iter\_no\_change=10, nesterovs\_momentum=True,  
 power\_t=0.5, random\_state=None, shuffle=True, solver='adam',  
 tol=0.0001, validation\_fraction=0.1, verbose=False,  
 warm\_start=False)  
 modelANN.fit(X\_train\_scaled[train\_idx], y\_train[train\_idx])  
 accTrain2=accuracy\_score(y\_true=y\_train[train\_idx], y\_pred=modelANN.predict(X\_train\_scaled[train\_idx]))  
 accVal2 = accuracy\_score(y\_train[val\_idx], modelANN.predict(X\_train\_scaled[val\_idx]))  
 ANNres = ANNres.append({'accVal': accVal2 , 'accTrain' : accTrain2}, ignore\_index=True)  
  
print("Default ANN Performances:")  
print(round(ANNres,3))  
print(round(ANNres.mean(),3))  
  
preds\_ANN = modelANN.predict(X\_val\_scaled)  
print("Default ANN- Validation accuracy: ", round(accuracy\_score(y\_val, preds\_ANN), 3))  
print()  
  
###---------------------------------------------------ANN-Best network--------------------------------------------------------------##  
  
AnnGrid= {'activation' : ['logistic', 'tanh', 'relu'],  
 'max\_iter' : np.arange(100,1500,25),  
 'learning\_rate\_init' : np.arange(0.001 ,0.050, 0.005),  
 'solver' : ['lbfgs' , 'sgd' ],  
 'hidden\_layer\_sizes' : [(3,),(3,3,),(3,3,3,),(3,3,3,3,),(4,),(4,4,),(4,4,4,),(4,4,4,4,),(5,),(5,5,),(5,5,5,),(5,5,5,5,),  
 (6,6,6,6,),(6,6,6,),(6,6,),(6,),(7,),(7,7,),(7,7,7,),(7,7,7,7,),(8,),(8,8,),(8,8,8,),(8,8,8,8,),  
 (9,),(9,9,),(9,9,9,),(9,9,9,9,)]}  
  
  
random\_search = RandomizedSearchCV(MLPClassifier(random\_state=42), param\_distributions=AnnGrid, cv=9, random\_state=123, n\_iter=300, refit=True)  
random\_search.fit(X\_train\_scaled, y\_train)  
print("Best ANN Hyper Parameters: ", random\_search.best\_params\_)  
print("Best ANN Score: ", round(random\_search.best\_score\_,3))  
  
best\_model\_ANN = random\_search.best\_estimator\_  
preds\_ANN = best\_model\_ANN.predict(X\_val\_scaled)  
print("ANN- Validation accuracy: ", round(accuracy\_score(y\_val, preds\_ANN), 3))  
print("confusion matrix ANN:")  
print(confusion\_matrix(y\_true=y\_val, y\_pred=preds\_ANN))  
  
#Show.Results  
#results\_ann=pd.DataFrame(random\_search.cv\_results\_)  
#results\_ann=results\_ann.drop(['mean\_fit\_time','std\_fit\_time','mean\_score\_time','std\_score\_time','rank\_test\_score'],1)  
#results\_ann.to\_csv('C:/Users/shoha/Desktop/ML\_part2/results\_ann.csv')  
  
  
ANNGrid\_df = pd.DataFrame(columns=['index','activation', 'max\_iter', 'learning\_rate\_init', 'solver', 'hidden\_layer\_sizes','train ac', 'val ac'])  
activation\_list = {'idntity', 'logistic', 'tanh', 'relu'}  
max\_iter\_list= np.arange(100,1500,25),  
learning\_rate\_init\_list=np.arange(0.001 ,0.050, 0.005)  
solver\_list = {'lbfgs' , 'sgd' , 'adam'}  
hidden\_layer\_sizes\_list= {(3,),(3,3,),(3,3,3,),(3,3,3,3,),(4,),(4,4,),(4,4,4,),(4,4,4,4,),(5,),(5,5,),(5,5,5,),(5,5,5,5,),  
 (6,6,6,6,),(6,6,6,),(6,6,),(6,),(7,),(7,7,),(7,7,7,),(7,7,7,7,),(8,),(8,8,),(8,8,8,),(8,8,8,8,),  
 (9,),(9,9,),(9,9,9,),(9,9,9,9,)}  
  
for j in np.arange(0,300,1):  
 str = results\_ann['param\_hidden\_layer\_sizes'][j]  
 s2 = ","  
 number = str[1]  
 # + str[2]  
 numberof = str.count(s2)  
 if (numberof == 1):  
 x = (int(number),)  
 if (numberof == 2):  
 x = (int(number), int(number),)  
 if (numberof == 3):  
 x = (int(number), int(number), int(number),)  
 if (numberof == 4):  
 x = (int(number), int(number), int(number), int(number),)  
  
 model\_i = MLPClassifier(activation=results\_ann['param\_activation'][j], max\_iter=results\_ann['param\_max\_iter'][j],  
 learning\_rate\_init=results\_ann['param\_learning\_rate\_init'][j], solver=results\_ann['param\_solver'][j],  
 hidden\_layer\_sizes=x, random\_state=42)  
 model\_i.fit(X\_train\_scaled,y\_train)  
 preds\_i = model\_i.predict(X\_val\_scaled)  
 ANNGrid\_df.at[j, 'index'] = j  
 ANNGrid\_df.at[j, 'activation'] = results\_ann['param\_activation'][j]  
 ANNGrid\_df.at[j, 'max\_iter'] = results\_ann['param\_max\_iter'][j]  
 ANNGrid\_df.at[j, 'learning\_rate\_init'] = results\_ann['param\_learning\_rate\_init'][j]  
 ANNGrid\_df.at[j, 'solver'] = results\_ann['param\_solver'][j]  
 ANNGrid\_df.at[j, 'hidden\_layer\_sizes'] = results\_ann['param\_hidden\_layer\_sizes'][j]  
 ANNGrid\_df.at[j, 'train ac'] = round(results\_ann['mean\_test\_score'][j], 3)  
 #ANNGrid\_df.at[j, 'val ac'] = round(results\_ann['val ac'][j], 3)  
 ANNGrid\_df.at[j, 'val ac'] = round(accuracy\_score(y\_val, preds\_i), 3)  
 results\_ann.at[j,'val ac']= round(accuracy\_score(y\_val, preds\_i), 3)  
  
#results\_ann.to\_csv('C:/Users/shoha/Desktop/ML\_part2/results\_ann.csv')  
  
print(ANNGrid\_df)  
plt.figure(figsize=(13, 4))  
plt.plot(ANNGrid\_df['index'], ANNGrid\_df['train ac'], marker='o', markersize=4)  
plt.plot(ANNGrid\_df['index'], ANNGrid\_df['val ac'], marker='o', markersize=4)  
plt.legend(['Train accuracy', 'Validation accuracy'])  
plt.title('Accuracy percentages on train set and validation set for each model')  
plt.xlabel('Table Index')  
plt.ylabel('Accuracy Percentages')  
plt.show()  
  
###--------------------------------------------------- PCA --------------------------------------------------------------##  
  
pca = PCA(n\_components=2)  
pca.fit(X\_train\_scaled)  
X\_train\_pca = pca.transform(X\_train\_scaled)  
X\_train\_pca = pd.DataFrame(X\_train\_pca, columns=['PC1', 'PC2'])  
  
pca.fit(X\_val\_scaled)  
X\_val\_pca = pca.transform(X\_val\_scaled)  
X\_val\_pca = pd.DataFrame(X\_val\_pca, columns=['PC1', 'PC2'])  
  
X\_train\_pca['y'] = y\_train  
sns.scatterplot(x='PC1', y='PC2', hue='y', data=X\_train\_pca)  
plt.title('Train set after PCA')  
plt.show()  
  
X\_val\_pca['y'] = y\_val  
sns.scatterplot(x='PC1', y='PC2', hue='y', data=X\_val\_pca)  
plt.title('Validation set after PCA')  
plt.show()  
  
###--------------------------------------------------- KMeans- default clustring ------------------------------------##  
  
X\_train\_pca = X\_train\_pca.drop(['y'], axis=1)  
X\_val\_pca = X\_val\_pca.drop(['y'], axis=1)  
  
kmeans = KMeans(n\_clusters=2, init='k-means++', n\_init=10, max\_iter=300, tol=1e-4,  
 precompute\_distances='deprecated', verbose=0, random\_state=None,  
 copy\_x=True, n\_jobs='deprecated', algorithm='auto')  
modelK = kmeans.fit(X\_train\_pca)  
  
pred\_train=modelK.predict(X\_train\_pca.values)  
for i in np.arange(0,len(pred\_train),1):  
 if(pred\_train[i]==1):  
 pred\_train[i]=0  
 else: pred\_train[i]=1  
print("Default KMeans- train accuracy: ",round(accuracy\_score(y\_true=y\_train,y\_pred=pred\_train),3))  
print("confusion matrix- train set:")  
print(confusion\_matrix(y\_true=y\_train, y\_pred=pred\_train))  
  
pred\_val=modelK.predict(X\_val\_pca.values)  
for i in np.arange(0,len(pred\_val),1):  
 if(pred\_val[i]==1):  
 pred\_val[i]=0  
 else: pred\_val[i]=1  
print("Default KMeans- Validation accuracy: ",round(accuracy\_score(y\_true=y\_val,y\_pred=pred\_val),3))  
print("confusion matrix- validation set:")  
print(confusion\_matrix(y\_true=y\_val, y\_pred=pred\_val))  
  
centers=modelK.cluster\_centers\_  
print(centers)  
print(round(centers[0][0],3))  
print(round(centers[0][1],3))  
print(round(centers[1][0],3))  
print(round(centers[1][1],3))  
  
x\_test = np.linspace(-4, 4, 100)  
y\_test = np.linspace(-4, 7, 100)  
predictions = pd.DataFrame()  
for x in tqdm(x\_test):  
 for y in y\_test:  
 pred = modelK.predict(np.array([x, y]).reshape(-1, 2))[0]  
 predictions = predictions.append(dict(X1=x, X2=y, y=pred), ignore\_index=True)  
  
plt.figure(figsize=(7,7))  
plt.scatter(x=predictions[predictions.y == 1]['X1'], y = predictions[predictions.y == 1]['X2'], c='powderblue')  
plt.scatter(x=predictions[predictions.y == 0]['X1'], y = predictions[predictions.y == 0]['X2'], c='ivory')  
X\_train\_pca['y'] = y\_train  
sns.scatterplot(x='PC1', y='PC2', hue='y', data=X\_train\_pca)  
plt.scatter(centers[:, 0], centers[:, 1], marker='+', s=100 ,color='red')  
plt.title("Clustering with K-Means- train set")  
plt.show()  
  
plt.figure(figsize=(7,7))  
plt.scatter(x=predictions[predictions.y == 1]['X1'], y = predictions[predictions.y == 1]['X2'], c='powderblue')  
plt.scatter(x=predictions[predictions.y == 0]['X1'], y = predictions[predictions.y == 0]['X2'], c='ivory')  
X\_val\_pca['y'] = y\_val  
sns.scatterplot(x='PC1', y='PC2', hue='y', data=X\_val\_pca)  
plt.scatter(centers[:, 0], centers[:, 1], marker='+', s=100 ,color='red')  
plt.title("Clustering with K-Means- validation set")  
plt.show()  
  
###--------------------------------------------------- KMeans- K clustres ------------------------------------##  
  
X\_train\_pca = X\_train\_pca.drop(['y'], axis=1)  
X\_val\_pca = X\_val\_pca.drop(['y'], axis=1)  
  
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(20, 10))  
for n\_clusters in range(2, 10, 1):  
 kmeans = KMeans(n\_clusters=n\_clusters, max\_iter=300, n\_init=10, random\_state=42)  
 kmeans.fit(X\_train\_pca)  
 assignments = kmeans.predict(X\_train\_pca)  
 scheme = X\_train\_pca.copy()  
 scheme = pd.DataFrame(scheme, columns=['PC1', 'PC2'])  
 scheme['cluster'] = assignments  
 i = 0 if n\_clusters in [2, 3, 4, 5] else 1  
 j = 0  
 j = 1 if n\_clusters in [3, 7] else j  
 j = 2 if n\_clusters in [4, 8] else j  
 j = 3 if n\_clusters in [5, 9] else j  
 sns.scatterplot(x='PC1', y='PC2', data=scheme, hue='cluster', ax=axes[i, j], palette='Accent\_r', legend=False)  
plt.show()  
  
iner\_list = []  
dbi\_list = []  
sil\_list = []  
ch\_list = []  
  
for n\_clusters in tqdm(range(2, 10, 1)):  
 kmeans = KMeans(n\_clusters=n\_clusters, max\_iter=300, n\_init=10, random\_state=42)  
 kmeans.fit(X\_train\_pca)  
 assignment = kmeans.predict(X\_train\_pca)  
 iner = kmeans.inertia\_  
 sil = silhouette\_score(X\_train\_pca, assignment)  
 dbi = davies\_bouldin\_score(X\_train\_pca, assignment)  
 ch = calinski\_harabasz\_score(X\_train\_pca, assignment)  
 dbi\_list.append(dbi)  
 sil\_list.append(sil)  
 ch\_list.append(ch)  
 iner\_list.append(iner)  
  
plt.plot(range(2, 10, 1), iner\_list, marker='o')  
plt.title("Inertia")  
plt.xlabel("Number of clusters")  
plt.show()  
  
plt.plot(range(2, 10, 1), sil\_list, marker='o')  
plt.title("Silhouette Coefficient")  
plt.xlabel("Number of clusters")  
plt.show()  
  
plt.plot(range(2, 10, 1), dbi\_list, marker='o')  
plt.title("Davies-Bouldin Index")  
plt.xlabel("Number of clusters")  
plt.show()  
  
plt.plot(range(2, 10, 1), ch\_list, marker='o')  
plt.title("Calinski-Harabasz Index")  
plt.xlabel("Number of clusters")  
plt.show()  
  
###--------------------------------------------------- SVM ------------------------------------##  
  
print("svm:")  
SVMGrid= {'kernel' : ['linear', 'poly', 'rbf', 'sigmoid'], 'C' : np.arange(0.05, 3.5, 0.5), 'gamma' : ['scale', 'auto']}  
  
grid\_searchSVM = GridSearchCV(estimator=SVC(random\_state=42), param\_grid=SVMGrid, refit=True, cv=9)  
grid\_searchSVM.fit(X\_train\_pca, y\_train)  
print("Best SVM Hyper Parameters: ", grid\_searchSVM.best\_params\_)  
print("Best SVM Score: ", round(grid\_searchSVM.best\_score\_,3))  
  
best\_modelSVM = grid\_searchSVM.best\_estimator\_  
preds\_SVM = best\_modelSVM.predict(X\_val\_pca)  
print("before")  
print("SVM- Validation accuracy: ", round(accuracy\_score(y\_val, preds\_SVM), 3))  
print(confusion\_matrix(y\_true=y\_val, y\_pred=preds\_SVM))  
  
for i in np.arange(0,len(preds\_SVM),1):  
 if(preds\_SVM[i]==1):  
 preds\_SVM[i]=0  
 else: preds\_SVM[i]=1  
  
print("SVM- Validation accuracy: ", round(accuracy\_score(y\_val, preds\_SVM), 3))  
print("after")  
print(confusion\_matrix(y\_true=y\_val, y\_pred=preds\_SVM))  
  
x\_test = np.linspace(-5, 5, 100)  
y\_test = np.linspace(-5, 5, 100)  
predictions = pd.DataFrame()  
for x in x\_test:  
 for y in y\_test:  
 pred = best\_modelSVM.predict(np.array([x, y]).reshape(-1, 2))[0]  
 predictions = predictions.append(dict(X1=x, X2=y, y=pred), ignore\_index=True)  
  
plt.scatter(x=predictions[predictions.y == 0]['X1'], y = predictions[predictions.y == 0]['X2'], c='powderblue')  
plt.scatter(x=predictions[predictions.y == 1]['X1'], y = predictions[predictions.y == 1]['X2'], c='ivory')  
X\_train\_pca['y']=y\_train  
sns.scatterplot(x='PC1', y='PC2', data=X\_train\_pca, hue='y')  
plt.title("Clustering with SVM- train set")  
plt.show()  
  
  
plt.scatter(x=predictions[predictions.y == 0]['X1'], y = predictions[predictions.y == 0]['X2'], c='powderblue')  
plt.scatter(x=predictions[predictions.y == 1]['X1'], y = predictions[predictions.y == 1]['X2'], c='ivory')  
X\_val\_pca['y']=y\_val  
sns.scatterplot(x='PC1', y='PC2', data=X\_val\_pca, hue='y')  
plt.title("Clustering with SVM- validation set")  
plt.show()  
  
X\_train\_pca=X\_train\_pca.drop(['y'],1)  
X\_train\_pca['cluster'] = best\_modelSVM.predict(X\_train\_pca.values)  
sns.scatterplot(x='PC1', y='PC2', hue='cluster', data=X\_train\_pca, palette='Accent')  
plt.title('SVM clustering on train set')  
plt.show()  
  
###--------------------------------------------------- Final Model Predictions ------------------------------------##  
  
X\_test=pd.read\_csv('C:/Users/shoha/Desktop/ML\_part2/X\_test.csv') #read the excel file from location on pc  
y\_pred = pd.DataFrame(columns=['y'])  
  
#Change invalid values  
np.random.seed(356) #seed from normal distribution  
age120=data[data['age']<120] #subset age under 120  
agemean =np.round\_(np.mean(age120['age'])) #mean to all age under 120  
agestd = np.std(age120['age']) #std to all age under 120  
for i in X\_test.index: #changing ages over 120 to random number fro normal distribution with agemean, agestd  
 if (X\_test['age'].values[i] > 120):  
 X\_test['age'].values[i] = np.round\_(np.random.normal(agemean, agestd, 1))  
  
#Changing missing values  
#CA  
X\_test.loc[X\_test['ca'] == 4,'ca'] = 0  
#Thal  
X\_test.loc[X\_test['thal'] == 0,'thal'] = 2  
  
#Changing to dummies  
#Chol changing to categorical variable  
X\_test.loc[X\_test['chol'] <= 200, 'chol'] = 0 #change chol to 0\1 , 0= normal 1=high  
X\_test.loc[X\_test['chol'] > 300, 'chol'] = 2 #change chol to 0\1 , 0= normal 1=high  
X\_test.loc[X\_test['chol'] > 200,'chol'] = 1 #change chol to 0\1 , 0= normal 1=high  
  
X\_test\_dum = pd.get\_dummies(X\_test, columns=['cp', 'restecg', 'slope', 'ca', 'thal' , 'chol'], drop\_first=True)  
X\_test\_dum=X\_test\_dum.drop(['id'], axis=1)  
  
y\_pred['y']= best\_model.predict(X\_test\_dum.values)  
y\_pred.to\_csv('C:/Users/shoha/Desktop/ML\_part2/y\_test.csv')  
  
###--------------------------------------------------- Random Forest - Bonus Question ------------------------------------##  
  
ramdom\_forest\_grid= { 'n\_estimators': np.arange(10,110,10), 'criterion' : ['gini', 'entropy'], 'max\_depth': np.arange(1,11,1), 'max\_features' :['sqrt', 'log2']}  
grid\_search\_rf = RandomizedSearchCV(estimator=RandomForestClassifier(random\_state=42), param\_distributions=ramdom\_forest\_grid, cv=9, random\_state=123, n\_iter=100, refit=True)  
grid\_search\_rf.fit(X\_train,y\_train)  
  
print("Best RF Hyper Parameters: ", grid\_search\_rf.best\_params\_)  
print("Best RF Score: ", round(grid\_search\_rf.best\_score\_,3))  
  
best\_model\_rf = grid\_search\_rf.best\_estimator\_  
preds\_rf = best\_model\_rf.predict(X\_val)  
print("RF- Validation accuracy: ", round(accuracy\_score(y\_val, preds\_rf), 3))  
  
preds\_rf = best\_model\_rf.predict(X\_val)  
print("confusion matrix RF:")  
print(confusion\_matrix(y\_true=y\_val, y\_pred=preds\_rf))