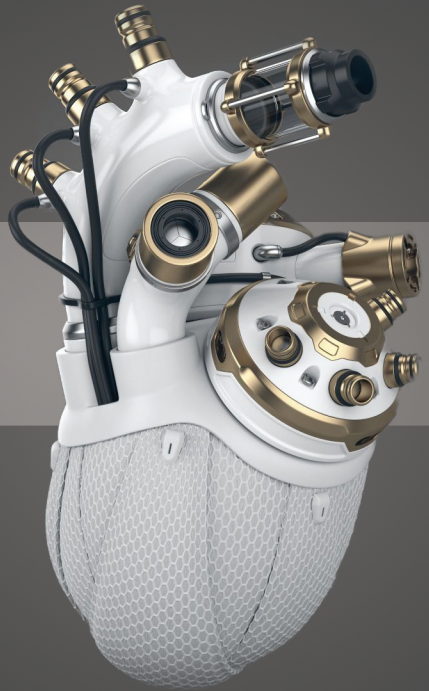
# ממ"ן 21



מתן כהן

**1 – הגדרת הבעיה והכנת הנתונים**

א. נגדיר את מטרות כריית המידע, נציין את ההנחות וההפשטות בהן השתמשנו -
מטרת כריית המידע היא חיזוי הישרדות חולים בעל כשל לבבי מתוך סט תכונות הנתונות לנו בקובץ csv בעל 299 רשומות, כאשר עמודת DEATH_EVENT היא העמודה עליה נבצע חיזוי.

ב. נגדיר את הנתונים בהם נשתמש בפרויקט מבחינת תכונות, סוג נתונים, נתונים חסרים, תחומי ערכים וכו' -
על מנת לבצע זאת, יש **להבין** את ה data המוצג בפנינו, לשם כך נעזרתי בגוגל על מנת להרחיב את הידע שלי על הבעיה ולהבין את עמודות הטבלה בצורה טובה יותר, לשם כך יצרתי טבלה חדשה עם תכונות ומידע נוסף שאני מסיק מהן, הטבלה כוללת 12 שורות כמס' התכונות.
אציין שבדיקת הערכים החסרים התבצעה באמצעות פייתון ושימוש ב – df.isnull().any() עבור df המייצג את הנתונים.

| תכונה | תיאור | סוג נתונים | תחומי ערכים | ערך שכיח | ערכים חסרים |
|---|---|---|---|---|---|
| age | גיל | numeric | 40-95 | 60 | אין |
| anemia | אנמיה | category | 0 או 1 | 0 | אין |
| creatinine_phosphokinase | רמות CPK בדם | numeric | 23-7,861 | 582 | אין |
| diabetes | סוכרת | category | 0 או 1 | 0 | אין |
| ejection_fraction | מקטע פליטה | numeric | 14-80 | 35 | אין |
| high_blood_pressure | לחץ דם גבוה | category | 0 או 1 | 0 | אין |
| platelets | כמות טסיות בדם | numeric | 25,100-850,000 | 26,3358 | אין |
| serum_creatinine | קריאטינין | numeric | 0.5-9.4 | 1 | אין |
| serum_sodium | נתרן | numeric | 113-148 | 136 | אין |
| sex | מגדר | category | 0 או 1 | 1 | אין |
| smoking | מעשן/ת | category | 0 או 1 | 0 | אין |
| time | ימים בהם נבדק | numeric | 4-285 | 187, ממוצע 130 | אין |

נעזרתי בפונקציה בסיסית שנתנה לי את המידע בצורה מהימנה:

```
[4]: def get_info(column):
         column_obj = eval(f'df.{column}')
         print(f'mode of {column} = {column_obj.mode()[0]}')
         print(f'mean of {column} = {column_obj.mean()}')
         print(f'min of {column} = {column_obj.min()}')
         print(f'max of {column} = {column_obj.max()}')

     for column in df.columns:
         get_info(column)
         print()
```

ג. בהמשך לסעיפים א' ו-ב' , נגדיר ונתאר את שלבי ה KDD עבור הבעיה הנתונה.

1. מטרת כריית המידע: כנאמר בסעיפים קודמים, היא חיזוי הישרדות חולים בעלי כשל לבבי.
2. איסוף, איגוד ואחסון נתונים: הקובץ שניתן בממ"ן והקישור הנוסף שעזר לי להבין חלק מהנתונים (כמו time ומה מסמל כל ערך במין הנבדק). את קובץ ה csv פתחתי בעזרת pandas בפייתון וכך גם ידעתי לבדוק ערכים חסרים וערכים שכיחים או ממוצע.
3. ניקוי הנתונים ועיבוד מוקדם: בעזרת שימוש נוסף בפייתון, עברתי על הנתונים ובדקתי ערכים חריגים, נתונים חסרים (כפי שכבר נאמר), רשומות כפולות ועוד וכמובן ניסיתי לסנן נתונים ע"פ מידע שמצאתי באינטרנט לגבי טווחי ערכים תקינים ובעזרת היסטוגרמות ומפות חום של קורלציות בין פיצ'רים.
4. טרנספורמציות על נתונים ורדוקציה: אין סיבה לעשות רדוקציה שכן 299 רשומות זהו מספר מרשים אך לא גדול במידה כזו שמצריכה רדוקציה, כמו-כן הטרנספורמציה היחידה שהפעלתי היא box-cox לנרמול מידע שגורם לצידוד (skewness) ודיסקרטיזציה על ידי חלוקה ל bins במידה והנרמול שביצעתי לא מתקבל על ידי צוות הקורס.
5. בחירת האמצעים לתהליך כריית המידע: השתמשתי ב – Python, pandas, scipy, sklearn על מנת לנתח את הנתונים בצורה מועילה והשתמשתי ב cross-validation על תהליכי החיזוי אשר אותם עשיתי בעזרת ספריית פייתון חיצונית בשם decision-tree-id3 וספריית sklearn.
בחנתי מודלים שונים של אלגוריתמים מוכרים של עצי החלטה.
6. סקירת תוצאות: לאחר בניית מודלים כאלו ואחרים והרצתם קיבלתי תוצאות שונות.
בדיקות בפייתון עזרו לי להעריך את המודלים לפי דיוק על סט האימון שלהם, רלוונטיות, מדדי confusion matrices שלהם וכמובן זאת בזמן שאני בודק ערכים חריגים ואפשרויות לאמן את המודלים שלי בצורה קצת שונה בעזרת מניפולציות כאלו ואחרות על סט המידע שלי.
7. הסקת מסקנות: תחילה, לאחר ניקוי קצר של נתונים שהיו outliers הגעתי לאחוזי דיוק יחסית מספקים ובחרתי להמשיך לבדוק את סט המידע שלי, לנסות לנרמל ערכים כאלו ואחרים או לחלק אותם ל bins מתאימים על מנת להעלות את אחוזי הדיוק של המודל.
ניתן להסיק בבירור שמשך זמן המעקב אחר מטופלים חולי לב קשור באופן מובהק לסיכויי הישרדותם לפי העובדה שזמן המעקב היה תמיד בשורש עץ ההחלטה בכמעט כל עץ שיצרתי.

ד. בהמשך לסעיפים א ו-ב נערוך סקירה השוואתית לחלופות האפשריות לביצוע כריית מידע. נתייחס ליתרונות ולחסרונות של כל אחת מהחלופות בהקשר לבעיה הנתונה.

1. <u>עץ החלטה מבוסס אלגוריתם cart עם מדד gini index:</u>

**סקרתי אופציה זו כיוון שגם ממומשת בקוד של scikit-learn.**

עץ החלטה בינארי בלבד אפשר מפותח במלואו ללא שיקולי גיזום מוקדם ורק לאחר פיתוח מפעילים גיזום מאוחר על העץ.

קריטריון הפיצול של העץ הוא מדד gini index ברוב המקרים.

שימוש בקריטריון gini index מניח שכל התכונות רציפות וכי יש כמה נקודות פיצול אפשריות לכל תכונה.

אלגוריתם זה מעדיף פיצול לפני שתי קבוצות שיהיו <u>יחסית</u> שוות בגודלן, לאחר בניית העץ המלא, האלגוריתם יוצר קבוצה של תתי-עצים גזומים ובוחר בעץ עם פונקציית העלות-סיבוכיות המינימלית על מנת לצמצם overfit.

החיסרון העיקרי – זמן ריצת האלגוריתם ארוך יותר משאר האלגוריתם שנלמדו.

2. <u>עץ החלטה מבוסס אלגוריתם ID3 עם מדד IG:</u>

זהו אלגוריתם חמדן ורקורסיבי אשר מפצל רשומות המשויכות לצומת בהתאם לקריטריון הפיצול שנבחר – הפיצול של צומת יכול גם להיות לא בינארי כך שיוצרים בן לכל צומת עבור כל תוצאת פיצול, על כך חוזרים בצורה רקורסיבית לכל תת-קבוצה של רשומות עד תנאי העצירה שכל הרשומות בעלות סיווג זהה **או** שאין יותר תכונות לפצל על פיהן.

אלגוריתם זה מצריך חלוקת משתנים רציפים למרווחים בדידים – דיסקרטיזציה של משתנים רציפים, ועליהם נוכל לבצע את חישוב ה IG ולפיכך לבחור את הפיצול האידיאלי.

היתרון – בניית העץ מהירה וקל להבין את מבנה העץ, מאידך החיסרון הגדול ביותר שלו הוא שקל מאוד להגיע איתו למצב של overfit במידה ויש לנו מעט מידי רשומות או עמודה עם המון אפשרויות שונות (דיסקרטיזציה לא תועיל לנו ממש), החיסרון הוא שהמדד IG בעל נטייה לבחור תכונות בעלות ערכים רבים.

3. <u>רגרסיה לינארית:</u>

חלופה בסיסית זו מבוססת על חיזוי ערכים רציפים כלשהם (כמו הדוגמאות הקלאסיות של חיזוי מחיר בית, מנייה, כמות מבריאים מקורונה ביום וכו' וכמו-כן ידועה מאוד בקרב תחומי מדעי החברה) משתמשים בה אם מצליחים למצוא התאמה לינארית כזו שעמודות במאגר נתונים יכולות "לחזות" את עמודת המטרה שלנו (בצורה אינטואיטיבית יותר, האם יש קורלציה חזקה בין עמודת פיצ'ר לעמודת מטרה כמו לדוגמה גודל בית ומיקומו שיכולים להשפיע על מחיר הבית).

היתרון – יחסית פשוטה לביצוע ולמימוש וגם נורא קל להבין אותה בצורה אינטואיטיבית ואפילו ניתן ליצור גרף המתאר אותה!

החיסרון – היא לא מתאימה לחיזוי פרמטרים מורכבים אשר עבורם נצטרך מודלים חזקים הרבה יותר (כמו לדוגמה ניסיון ללמידת מסלולי גרפים בעזרת למידת חיקוי עם RL ושימוש ב MDP).

4. <u>עץ החלטה המבוסס על אלגוריתם 4.5c עם המדד gain-ration:</u>

עץ החלטה אשר מנסה להתגבר על הבעיות של מדד הרווח האינפורמטיבי שהיא הטיה לבחירת תכונות בעלות ערכים רבים (חיסרון שדיברנו עליו כבר קודם) ולכן אלגוריתם זה מבצע שינוי ומגדיר קריטריון חדש לפיצול והוא יחס הרווח האינפורמטיבי – gain ratio.

נשיג קריטריון זה באמצעות נרמול ליחס האינפורמטיבי על ידי שימוש באינפורמציית הפיצול, התכונה בעלת היחס רווח אינפורמטיבי **המקסימלי** תבחר כתכונה המפצלת.

היתרון – נקבל דיוק טוב יותר אם ב data שלנו יש הרבה תכונות וכל תכונה מחולקת להרבה קטעים.

החיסרון – הפיצול מתעדף חלוקה פחות מאוזנת ויכול ליצור עלים לא מאוזנים באופן ניכר.

ה. נתאר את שלבי הכנת הנתונים. נתייחס לבעיות באיכות הנתונים כמו טיפול בערכים חסרים, תצוגה גרפית של הנתונים, ניקוי הנתונים, שילוב והמרה של נתונים ועוד.
תחילה נבחין כי אין כלל נתונים חסרים שכן df.info() החזיר לנו שכלל הרשומות הן Non-Null.
כמו כן, אציין את הנתונים בשפה שקל להבינה (שאבתי את הנתונים מכאן):

1. age – גיל האדם בעל כשל לבבי
2. anemia – חוסר בתאי דם אדומים או המוגלובין נמוך (בוליאני)
3. creatinine_phosphokinase – רמות אנזים CPK בדם (mcg/L).
4. diabetes – סוכרת (בוליאני)
5. ejection_fraction – אחוז הדם שעוזב את הלב בזמן כיווץ (אחוזים)
6. high_blood_pressure – לחץ דם גבוה (בוליאני)
7. platelets – טסיות בדם (kiloplatelets/mL)
8. serum_creatinine – רמות קריאטין בדם (mg/dL)
9. serum_sodium – רמות נתרן בדם (mEq/L)
10. sex – מגדר (בוליאני – 0=אישה, 1=גבר)
11. smoking – מעשן/ת (בוליאני)
12. time – זמן שבו הנבדק היה במעקב (ימים)
13. DEATH_EVENT – האם המטופל נפטר בזמן המעקב (בוליאני)

ניווכח אם כך, שהמידע שקיבלנו כולל ערכים שעל פי pandas מסווגים כערכים מספריים וערכים בינאריים (בוליאניים).
נחקור תחילה את הערכים הנומריים:

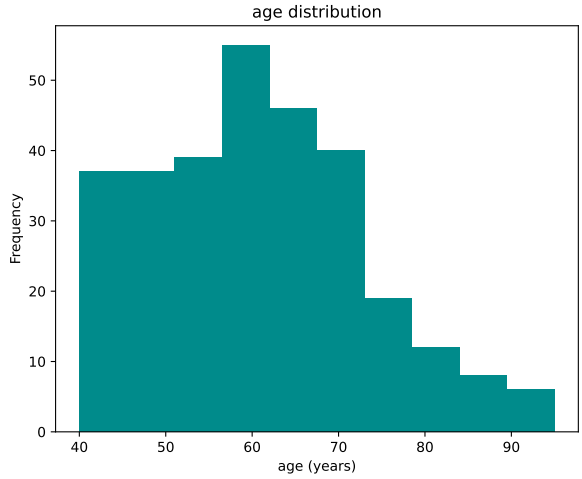
<u>ערכים נומריים</u>
ניצור היסטוגרמות פשוטות שיעזרו לנו להבין את המידע בצורה מועילה.
לשם כך, כתבתי בפייתון פונקציה פשוטה שתקבל שם משתנה ותייצר פלוט של ההיסטוגרמה המתאימה לו (ערך נומרי):

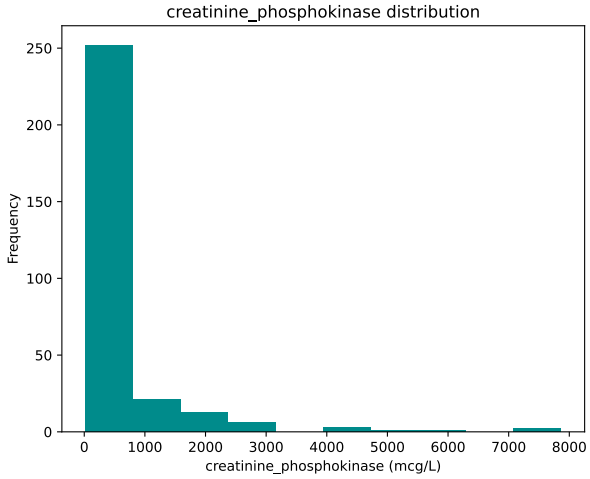
```
units = {
    'age': 'years',
    'creatinine_phosphokinase': 'mcg/L',
    'ejection_fraction': '%',
    'platelets': 'kiloplatelets/mL',
    'serum_creatinine': 'mg/dL',
    'serum_sodium': 'mEq/L',
    'time': 'days'
}
def draw_numeric_histograms():
    fig, axs = plt.subplots(4, 2, figsize=(15, 25))
    axes = axs.ravel()
    for variable, ax in zip(units.keys(), axes):
        ax.hist(X[variable], color="darkcyan")
        ax.set_xlabel(f'{variable} ({units[variable]})')
        ax.set_ylabel("Frequency")
        ax.set_title(f'{variable} distribution')
    fig.delaxes(ax=axs[3, 1])
    plt.savefig('numeric.svg')
draw_numeric_histograms()
```

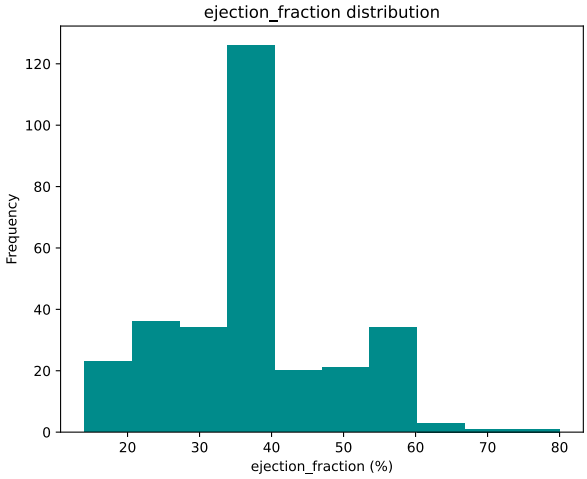
<u>ערכים בוליאניים</u>
עבור כל ערך קטגוריאלי גם יצרתי היסטוגרמה מתאימה:

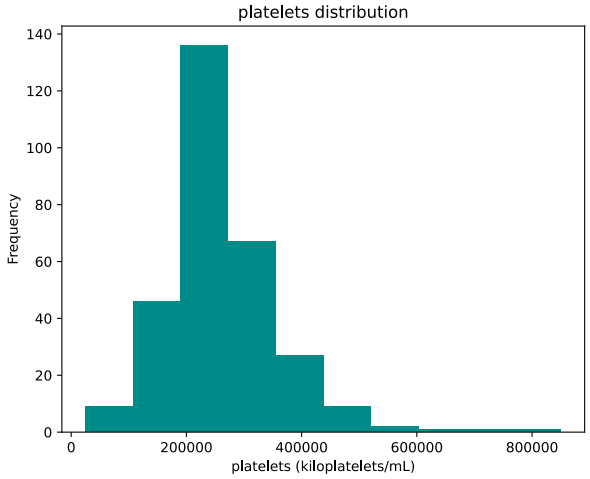
```
categorical_columns = [var for var in X.columns if len(X[var].value_counts()) == 2]
def bar_plot(variable):
    fig, axs = plt.subplots(3, 2, figsize=(15, 25))
    axes = axs.ravel()
    for variable, ax in zip(categorical_columns, axes):
        values = X[variable]
        counts = values.value_counts()
        ax.bar(counts.index, counts, color='skyblue')
        ax.set_xticks(counts.index)
        ax.set_ylabel('Frequency')
        ax.set_title(f'{variable} distribution')
    fig.delaxes(ax=axs[2, 1])
    plt.savefig('categorical.svg')
bar_plot(categorical_columns)
```

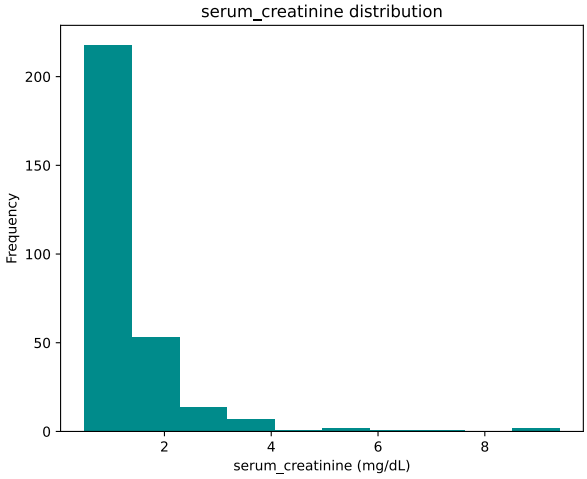
ההיסטוגרמות מוצגות בעמודים הבאים

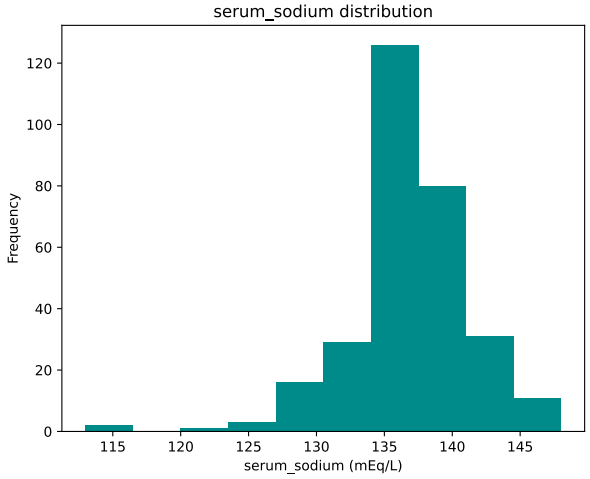

age distribution
Frequency
age (years)

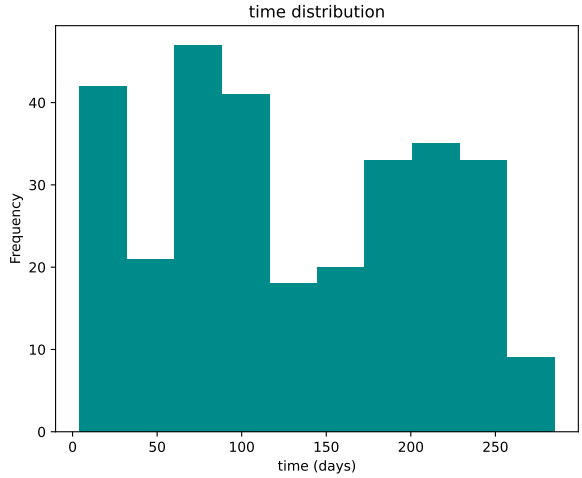

creatinine_phosphokinase distribution
Frequency
creatinine_phosphokinase (mcg/L)

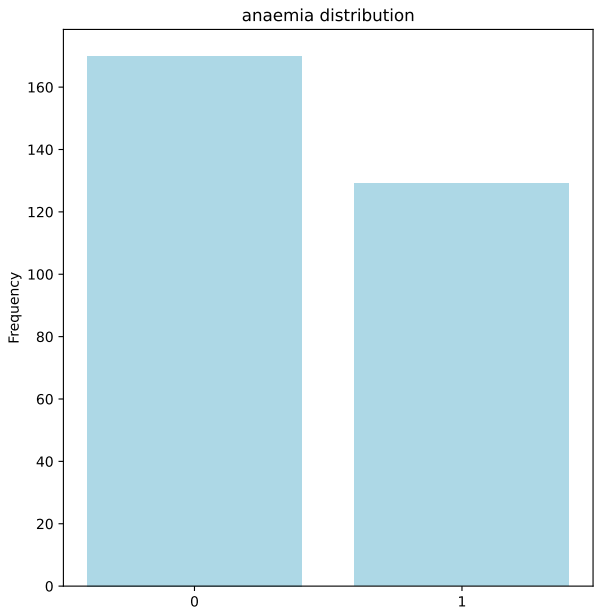

ejection_fraction distribution
Frequency
ejection_fraction (%)

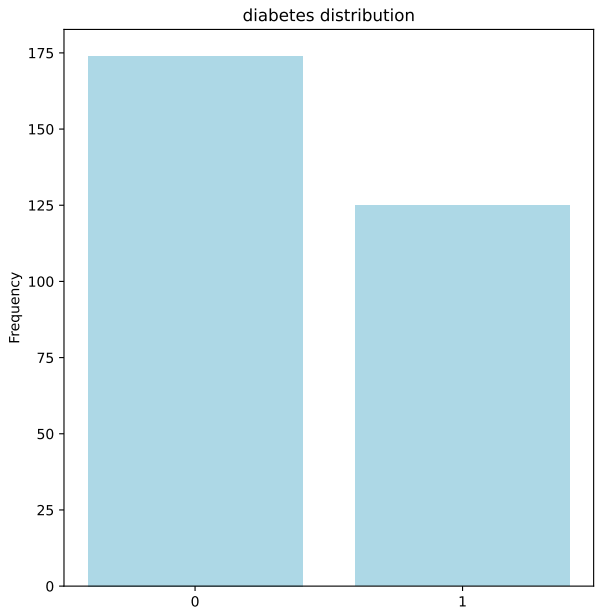

platelets distribution
Frequency
platelets (kiloplatelets/mL)

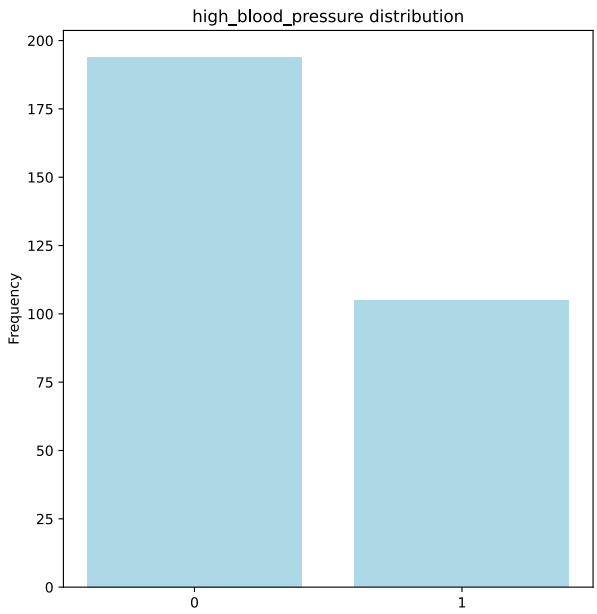

serum_creatinine distribution
Frequency
serum_creatinine (mg/dL)

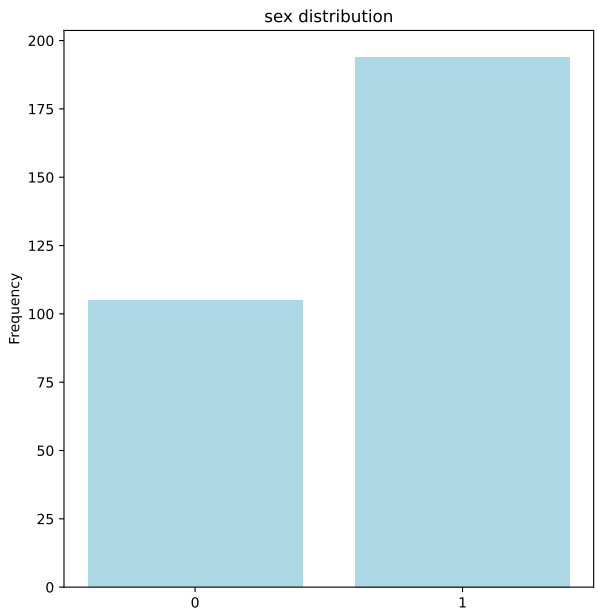

serum_sodium distribution
Frequency
serum_sodium (mEq/L)

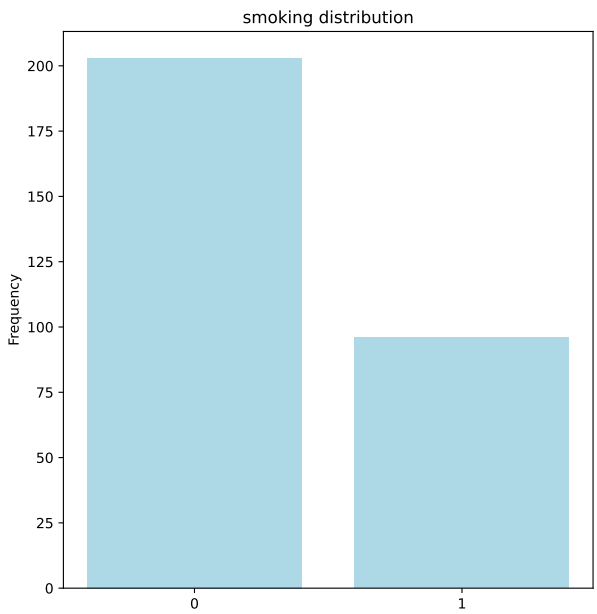

time distribution
Frequency
time (days)


anaemia distribution
Frequency
0
20
40
60
80
100
120
140
160
0
1


diabetes distribution
Frequency
0
25
50
75
100
125
150
175
0
1

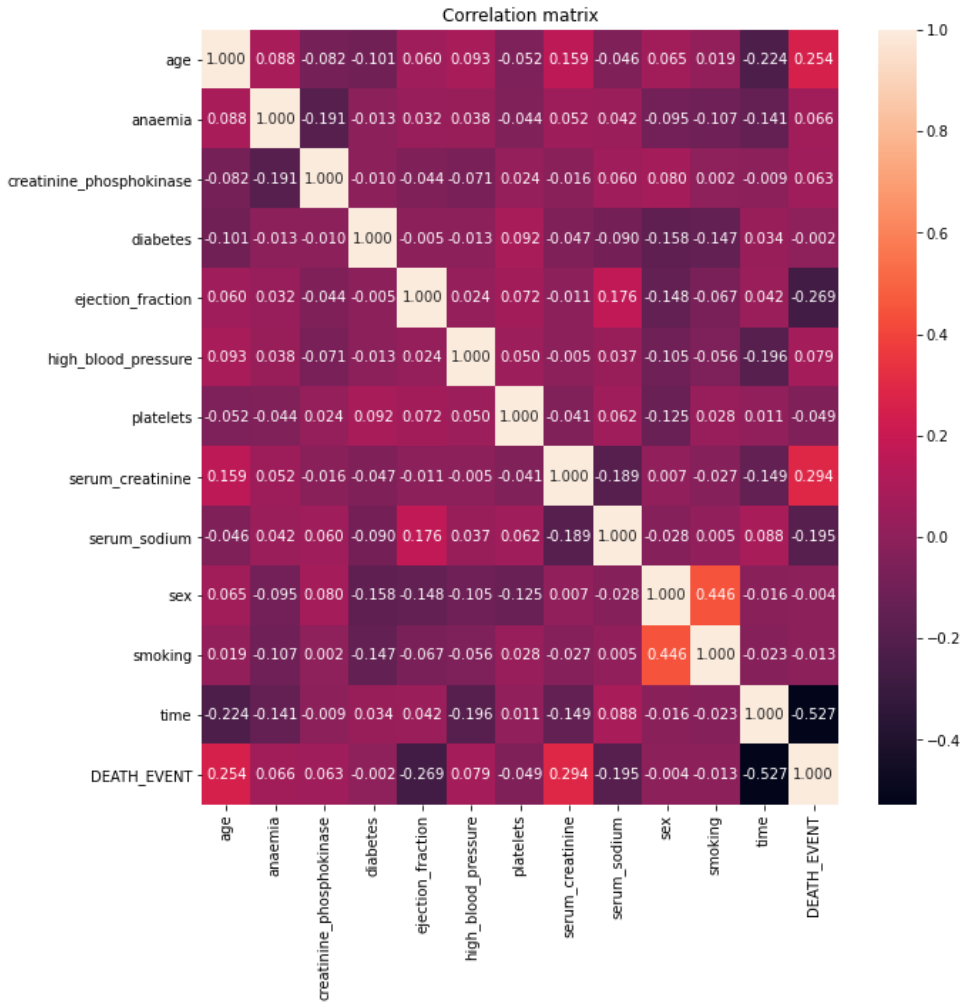

high_blood_pressure distribution
Frequency
0
25
50
75
100
125
150
175
200
0
1


sex distribution
Frequency
0
25
50
75
100
125
150
175
200
0
1


smoking distribution
Frequency
0
25
50
75
100
125
150
175
200
0
1

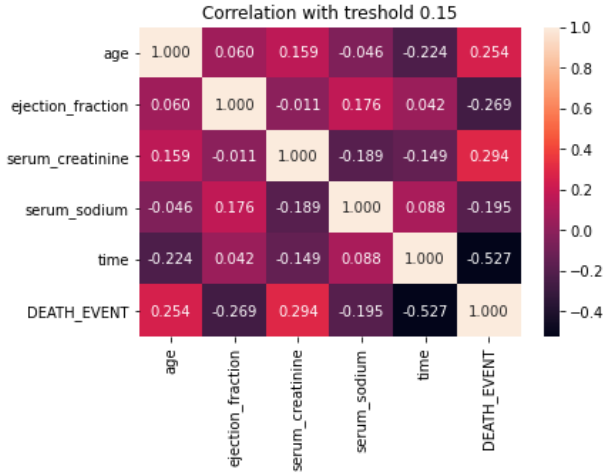
כעת, לאחר שראינו את הנתונים שלנו בצורה ויזואלית, נוכל להתחיל גם לבדוק קורלציה כזו או אחרת בין הפיצ'רים ולאחר מכן לחקור ולנסות לגלות נתונים חריגים!
תחילה נבחן קורלציות בין פיצ'רים ונציג את המידע שלנו בצורה שתהיה לנו נוחה יותר אח"כ.
לשם כך, נבדוק בעזרת seaborn ו pandas מפת קורלציה של המשתנים (הפעם כולל עמודת המטרה).

```
correlation_df = df.corr()
plt.figure(figsize=(10, 10))
sns.heatmap(correlation_df, annot=True, fmt='.3f')
plt.title('Correlation matrix')
plt.show()
```



כעת, נבחן את כלל היחסים בין הפיצ'רים.
לשם כך, נגדיר תנאי סף עבורו נרצה לבדוק יחס בין עמודת המטרה לבין פיצ'ר מסוים.
לאחר שבדקתי את ערכי הקורלציה ביחס לעמודת המטרה, הבחנתי שתנאי סף מתאים יהיה 0.15.
סיננתי ערכים עבורם מקדם המתאם במפת הקורלציה הוא נמוך מערך מוחלט של תנאי הסף שלנו (שכן מקדם מתאם יכול להיות שלילי):

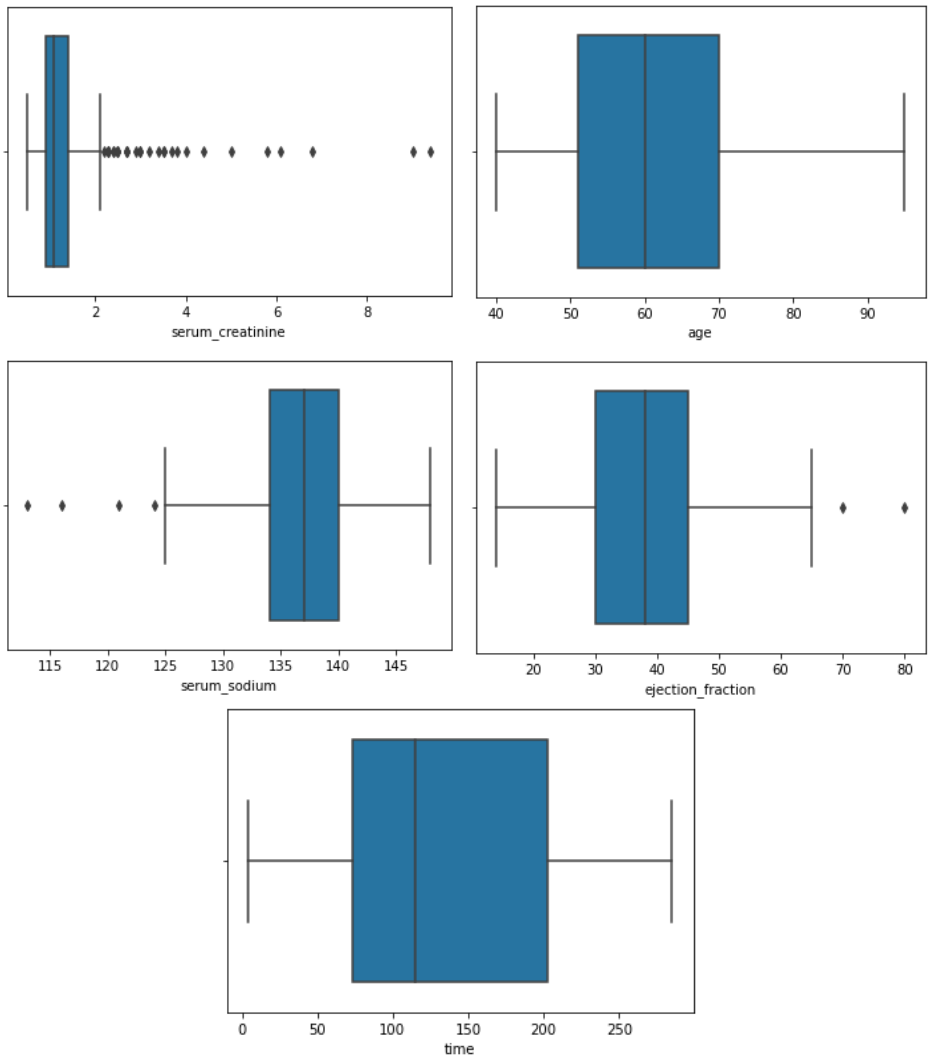
```
treshold = 0.15
filtered = np.abs(correlation_df.DEATH_EVENT) > treshold
wanted_features = correlation_df.columns[filtered].tolist()
sns.heatmap(df[wanted_features].corr(), annot=True, fmt='.3f')
plt.title(f'Correlation with treshold {treshold}')
plt.show()
```




כבר עכשיו ניתן לראות קורלציה יחסית טובה בין סיכוי לפטירה של חולה לבין הזמן מעקב, מה שמרמז מעתה על המודל שעתיד להגיע בהמשך.
בחרתי לסנן מסט הנתונים שלי את העמודות שלא נכללות במפת החום מעלה.
נשארתי עם הפיצ'רים: age, ejection_fraction, serum_creatinine, serum_sodium, time

## ערכים חריגים (Outlier)

נתחיל מלבדוק box plot על מנת למצוא באופן ויזואלי ערכים חריגים בפיצ'רים שנותרו:




נבחין כי עבור ערכי ejection_fraction יש שני ערכים שחשודים כחריגים, עבור serum_sodium ישנם ארבעה.
עבור serum_creatinine ניתן להבחין בצורה מאוד ברורה בחוסר סימטריות בהתפלגות ויש סיכוי שנראה צידוד של הערכים, אני אשאיר את הפיצ'ר הנ"ל כשחשוד בלבד בינתיים אך נראה שאין סיבה למחוק ממנו רשומות.

נתחיל בבדיקה יותר מדודה, נרצה למצוא ערכים חריגים (חריג חשוד טעות) על מנת לנקות את המידע שקיבלנו מערכים שיכולים לפגוע לנו בתהליך החיזוי.
בעזרת המידע שצברתי בדף הוויקיפדיה שקישרתי אליו, בחרתי להשתמש ב Tukey's fences (נוסחה די פשוטה שתוכל להניב לנו קטע של ערכים האמורים להיות בו.

$$[Q_1 - k(Q_3 - Q_1), Q_3 + k(Q_3 - Q_1)]$$

הרעיון הוא חלוקה לרביעונים, לשם כך אסביר מה הכוונה ברביעון (מה שמודגם גם ב *boxplots* שהצגתי).
כידוע, בסט הנתונים שלנו, לכל אחד מהפיצ'רים יש טווח והתפלגות על גבי הישר הממשי (ואף במקרים מסוימים ההתפלגות היא דיסקרטית), רביעון הוא מדד לפיזורם של ערכים במדגם.
אנו מכירים את מושג החציון מהמון קורסים קודמים, שם אחר לחציון הוא הרביעון השני.
נבחין כי בעצם על מנת למצוא רבעונים יש לחלק את טווח הערכים ל 4 רבעים שווי שכיחות (בכל אחד מהרבעים יש 25% מהשכיחויות).
לדוגמה, הרביעון התחתון (הרביעון הראשון) הוא ערך שעד אליו 25% ומעליו 75% מהערכים.
הרביעון השני הוא החציון והרביעון העליון (השלישי) הוא ערך שמתחתיו 75% מההתפלגות ומעליו 24% ממנה.

כעת, לאחר שהבנו את מושג הרביעון, נוכל לממש פונקציה בסיסית שתוכל לחשב את הטווח ערים למציאת חריגים, מהנוסחה מעלה נסיק: $Q_3$ הוא הרביעון העליון ו $Q_1$ הוא הרביעון התחתון ו $k$ הוא ערך לא שלילי שעל פי גישתו של ג'ון טוקי (עליו נקראת השיטה) כל ערך שהוא חריג חשוד טעות יהיה מחוץ לגבולות הקטע כאשר $k = 1.5$, לפיכך נגדיר כך את ה $k$ – שלנו. (אציין שבהתחלה חשבתי ש $k$ יהיה היפר-פרמטר וניסיתי לבדוק איזה $k$ יניב לי תוצאה טובה ביותר אך הדבר תרם ל *overfitting* רציני ולכן בחרתי ללכת ע"פ הנאמר)
כל ערך מחוץ לקטע הסגור שהגדרנו יהיה *outlier*.
כתבתי פונקציה פשוטה שתחשב לי את הקטע לכל אחד מהפיצ'רים שקיבלנו ושילבתי אותה בתוך פונקציה אחרת שתחזיר לנו זיהוי של ערכים חריגים:

```
from collections import Counter
def detect_outliers(df, features, k=1.5):
    outliers = []
    for feature in features:
        Q1 = np.percentile(df[feature], 25)
        Q3 = np.percentile(df[feature], 75)
        delta = Q3 - Q1
        lower_bound, upper_bound = Q1 - k * delta, Q3 + k * delta
        print(f'{feature}: [{lower_bound}, {upper_bound}]')
        outliers_idxs = df[(df[feature] < lower_bound) | (df[feature] > upper_bound)].index
        outliers.extend(outliers_idxs)
    outliers_counts = Counter(outliers)
    outliers = [idx for idx, cnt in (filter(lambda x: x[1] > 1, outliers_counts.items()))]
    return outliers
```

ערכים חריגים חיפשתי סביב הפי'צרים שלנו ומצאתי 2 ערכים חריגים:

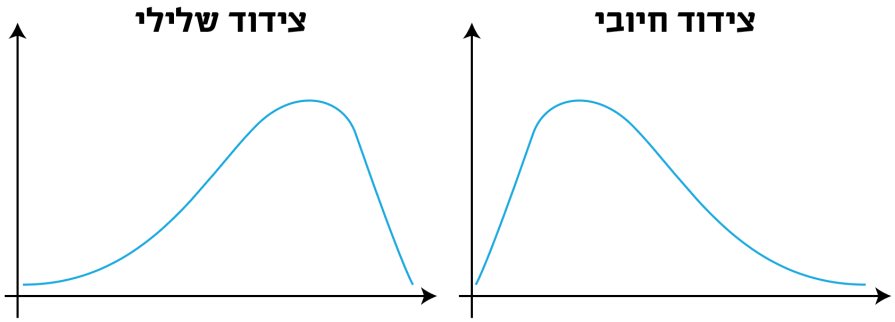
```
age: [22.5, 98.5]
ejection_fraction: [7.5, 67.5]
serum_creatinine: [0.15000000000000024, 2.1499999999999995]
serum_sodium: [125.0, 149.0]
time: [-122.0, 398.0]
```

| | age | ejection_fraction | serum_creatinine | serum_sodium | time | DEATH_EVENT |
|---|---|---|---|---|---|---|
| **217** | 54.0 | 70 | 9.0 | 137 | 196 | 1 |
| **4** | 65.0 | 20 | 2.7 | 116 | 8 | 1 |

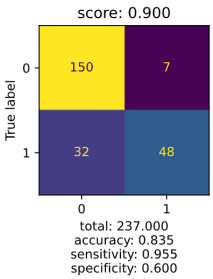
נעבור לבדיקת צידוד של ערכים.

<u>בדיקת צידוד וניסיון לתיקון</u>

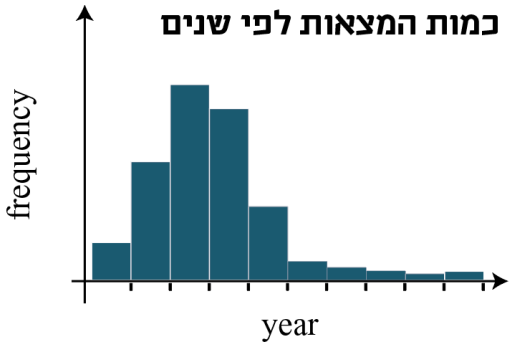
אחד מן הדברים אשר יכולים להשפיע על מודל טוב ואחוזי דיוקו הוא צידוד.
צידוד הוא מונח סטטיסטי אשר מקביל לחוסר סימטריה של פונקציית ההתפלגות של משתנה מקרי רציף כלשהו.
נבחין כי לכלל המשתנים המקריים הרציפים שלנו (הפיצ'רים הלא בוליאניים) יש אופציה להיות בעלי צידוד.
מצידוד של נתון מסוים נוכל לאיזה כיוון יהיו מרבית הסטיות שגדולות מן הממוצע.
בתמונה הבאה נוכל בקלות להבחין איך נראה צידוד חיובי ושלילי (ביחס, כמובן להתפלגות נורמלית בה הפעמון נמצא בדיוק באמצע וכך קל למצוא תוחלת ממוצע ועוד):




**על הצידוד חשבתי** לאחר שאימנתי מודל מבוסס עץ החלטה עם קריטריון אנטרופיה מבוסס *CART* בניסיון להקדים את המאוחר ולהבין האם יש צורך לבצע עוד מניפולציה על הנתונים וקיבלתי תוצאות טובות, אך חשבתי כיצד אפשרי לשפר את המודל (מיותר לציין שכאן לא השתמשתי ב *cross-validation* כיוון שזהו שלב ביניים):




לפיכך, מכמה מאמרים שקראתי ב *arxiv* מצאתי שלמודלים יש קושי בחיזוי עבור נתונים שיש להם צידוד.
אסביר זאת עם הקבלה פשוטה כדי שנוכל להמשיך בתהליך הבדיקה הנ"ל.
נניח ויש בידינו התפלגות של המצאות מדעיות לפי שנים (התפלגות מומצאת לחלוטין):



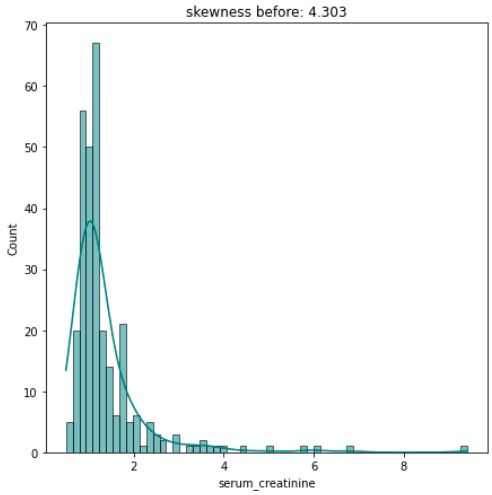

באופן וודאי ניתן לראות שיש צידוד חיובי, אם נרצה להשתמש בפיצ'ר זה בעתיד, נניח עבור מודל שיחזה שנים מוצלחות במדע, כיוון שיש צידוד חיובי, נוכל לראות שערכים נמוכים (שנים) נמצאים בתדירות גבוהה בסט האימון שלנו, דבר שיכול להוביל לחיזוי טוב יותר של שנים מוצלחות למדע עבור שנים נמוכות (שנים מהעבר) לעומת העתיד.
בנוסף, בצירוף ניקוי הנתונים הקודם שכלל ניקוי חריגים חשודי טעות, בעזרת צידוד נוכל להבחין בכיוון בוא החריגים "נעים" (אם כי צידוד לא תורם במספר החריגים אלא רק בכיוון הצידוד).
בעמוד הבא אתאר כיצד טיפלתי בצידודים בסט הנתונים.

<u>בדיקת צידוד בעזרת פייתון</u>

בדקתי את ערכי ה *skewness* לכל עמוד, כאשר ערכים מחוץ לטווח $[-1,1]$ מראים על צידוד.

| | skew_value |
|---|---|
| age | 0.418714 |
| ejection_fraction | 0.534021 |
| serum_creatinine | 4.303300 |
| serum_sodium | -0.830014 |
| time | 0.132660 |

תחילה, אראה כיצד נראה צידוד באחד מהסטים של האימון על מנת לקבל תמונה כללית על איך נראה המידע שלי תחת הצידודים (נראה שהצידוד של טסיות הדם לא גדול בכלל, ניקח זאת בחשבון):




נראה כי פעמון גאוס אצלה לא שלם, דבר שיכול להעיד על מצב שבו טרנספורמציה כזו או אחרת יכולה להוביל לפגיעה בסימטריות ובטווח הערכים, ניקח זאת בחשבון בזמן בדיקת הטרנספורמציות שאממש.

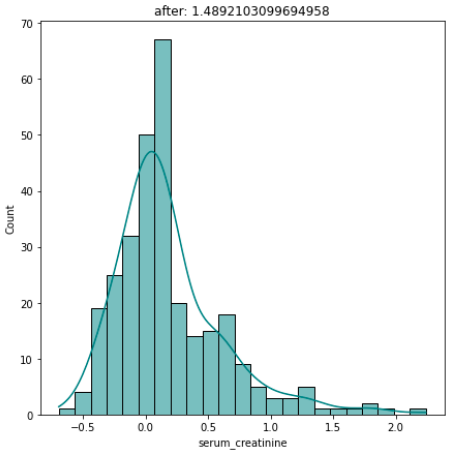
בחיפוש מהיר בגוגל ניתן להבחין שטרנספורמציות מקובלות הן פונקציות מהמשפחות:

- *Power transformation*
- *Log transformation*
- *Exponential transformation* (טרנספורמציה שכנראה לא תועיל לנו מתוקף טווחי הערכים שלנו)

בחרתי לבחון 2 פונקציות שונות על מנת להסיק איזה מהן תוכל להניב לי שתי מטרות שאני מכוון אליהן:

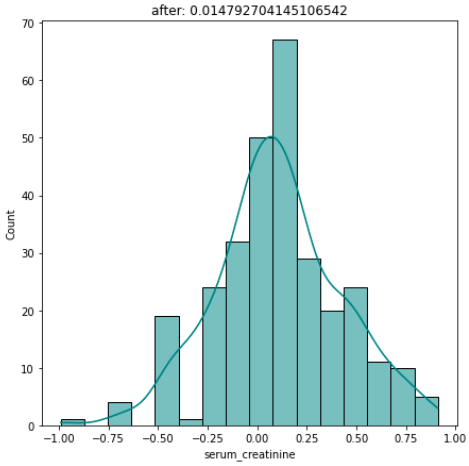
1. פונקציית התפלגות שתתקרב בזהותה לפונקציית ההתפלגות הנורמלית (פעמון גאוס)
2. הפיכת ה *skewness* לערכים בטווח הנורמה $[-1,1]$.

א. <u>שימוש בפונקציית *Log*:</u>
טרנספורמציית לוגריתם היא פשוטה ומבצעת לכל ערך $x$ את הפעולה $\log(x)$.
כך בעצם כאשר ההתפלגות לא מצטיירת כפעמון גאוס, נוכל "לנרמל" את הערכים למצב שמקנה לנו מעין פעמון גאוס.
טרנספורמציה זו עוזרת בעיקר עבור מידע שמתפלג בצורה שדומה להתפלגות לוג-נורמלית, לפיכך נוכל להשתמש בפונקציה זו על מנת לנרמל את העמודה הנ"ל ולקבל התפלגות שיותר דומה להתפלגות נורמלית אך מצאתי שאמנם יש שיפור, אבל הוא לא ניכר מספיק, שכן הצידוד עדיין נותר גדול מ 1:




**בחרתי לוותר על הטרנספורמציה הנ"ל.**

ב. <u>שימוש בפונקציית *Power transform*</u>
פונקציות אלו אמורות גם הן לגרום להתפלגות מסוימת של משתנה מקרי להיות יותר גאוסייני.
ישנן שתי פונקציות פופולריות: *box-cox* ו *Yeo-Johnson* כאשר הראשונה מתאימה לנו יותר כיוון שהיא מתאימה לערכים חיוביים וכלל הערכים שלנו הם ערכים חיוביים.
השתמשתי בטרנספורמציה הנ"ל ומצאתי שגם איתה יש שיפור אך התוצאה בעייתית כיוון שכידוע, הפיצ'ר הנ"ל שמודד קריאטינין אמור להיות חיובי בבדיקות דם – והטרנספורמציה יכולה לגרום ליצירה של ערכים שליליים.
בחרתי להשתמש בכל זאת בטרנספורמציה הנ"ל כיוון שהתאימה לי לשאר הערכים בצורה טובה והערכים התקרבו ל 0. להלן השינוי שיצרה בהתפלגויות:




אכן, כעת הצידוד הוא מינימלי בערכים אלו (מתואר מעל כל אחת מההיסטוגרמות), נבדוק שוב את רמת הצידוד בסט המידע:

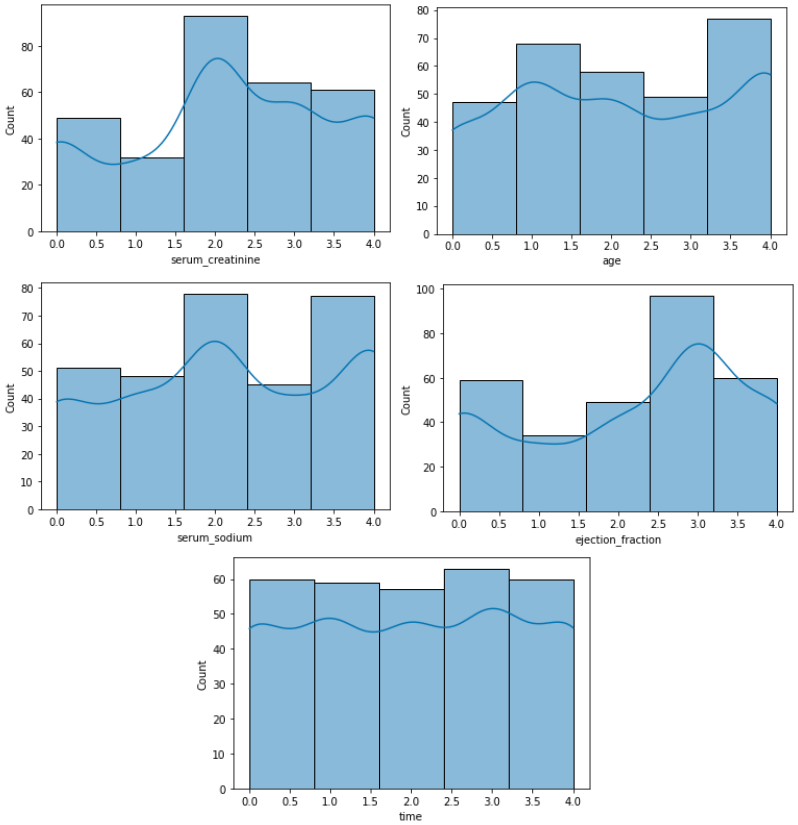
| | skew_value |
|---|---|
| age | 0.418714 |
| ejection_fraction | 0.534021 |
| serum_creatinine | 0.014793 |
| serum_sodium | -0.830014 |
| time | 0.132660 |

<u>שימוש ב *Binning* לפתרון בעיית ההתפלגות</u>

ביחד עם הטרנספורמציות שהצגתי מעלה, בניסיון נוסף (**במידה ואסור להשתמש בטרנספורמציות הללו**) השתמשתי בדיסקרטיזציה על ידי *Binning* כאשר בכל *bin* ניסיתי להכניס מספר ערכים די זהה. לאחר הסתכלות והתרשמות בחרתי לחלק את הנתונים בצורה הבאה:

| שם פיצ'ר | מספר בינים |
|---|---|
| *Age* | 5 |
| *Ejection_fraction* | 5 |
| *Serum_creatinine* | 5 |
| *Serum_sodium* | 5 |
| *Time* | 5 |

ושמרתי את הנ"ל בסט נתונים חדש על מנת לא לפגוע בטרנספורמציות שכבר עשיתי.
אימנתי את המודלים בשתי צורות, פעם אחת על הבינים שיצרתי ופעם אחרת על הנתונים שעברו טרנספורמציית *boxcox* על מנת לבדוק את כלל האופציות שחשבתי עליהן.
הנתונים לאחר *Binning*:




**לא היו בסט המידע ערכים פגומים או למחיקה, לכן לא פירטתי על כאלו.**

**2 – סיווג וחיזוי**

א. נבחר שתי שיטות לחיזוי הנתונים.
מתוך השיטות שהגדרתי בהתחלה, בחרתי להשתמש בשיטות הבאות:
1. עץ *ID3* אשר לא משתמש בגיזום (זהו העץ המוכר לנו מקורס מבוא לבינה מלאכותית ושעובד לפי קריטריון אנטרופיה), משתמש במדד *IG* וקל להגיע איתו ל *overfit* אך יש ספרייה ממומשת שהקלה עלי עם העבודה איתו.
2. עץ החלטה מבוסס *Cart* אשר משתמש בגיזום – קיצור העץ בשביל למנוע *overfit* ובכך להגדיל את דיוק העץ על נתוני האימון.

ב. נתאר את שלבי השיטות שבחרתי בסעיף א'
תחילה פיצלתי את סט הנתונים לאימון ולטסטים כאשר גודל סט הטסט הוא 20% מכלל המידע.
1. עץ החלטה מבוסס *ID3* שמומש בעזרת פייתון וספרייה מ *PyPi*.
אלגוריתם זה משתמש בקריטריון אנטרופיה לפיצול העץ לפי מדד *IG*, השתמשתי בעץ זה לאחר ניקוי הנתונים וכמובן השתמשתי ב *k-fold* בשביל *cross-validation* עם *10 folds*, **נבחין כי** במקרה של עץ *ID3* לא הייתה לי אפשרות להשתמש ב *GridSearchCV* של *sklearn* ולכן פיצלתי בעצמי את הסטים ואימנתי את המודל, ככה יכלתי להראות גם 5 מודלים טובים ביותר שיצאו לי מהאימון.
2. עץ החלטה מבוסס *Cart* זהו העץ שמומש בספריית *sklearn*, השימוש בעץ זה התבצע באותה צורה כמו השימוש בעץ מבוסס *ID3* שכן גם פה לאחר ניקוי הנתונים השתמשתי ב *k-fold* על מנת לבצע *cross-validation* אך הפעם ביחד עם *GridSearchCV* ואת פרמטר העלים שיניתי כדי לבדוק שינויים באחוזי הדיוק.

יצירת מקבץ המידע, עמודת המטרה, מקבצי *k-fold*:

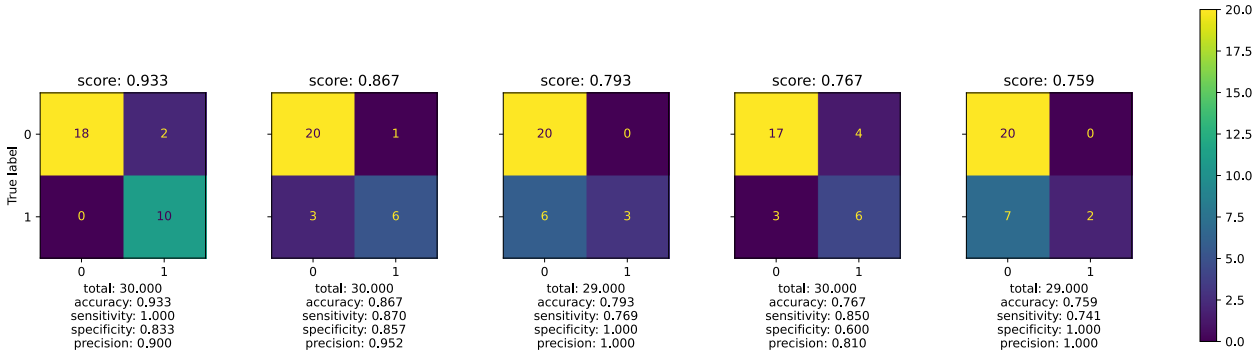
```
# K-fold with k=10
skf = StratifiedKFold(n_splits=10)
skfs = []
for train_index, test_index in skf.split(X, y):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    skfs.append([X_train, X_test, y_train, y_test])
```

הצגת *confusion matrix* לכל *decision tree* ביחס לנתוני הבדיקה שלו (5 מטריצות):

```
def get_cm_data(cm):
        data = {}
        data['total'] = sum(sum(cm))
        data['accuracy'] = (cm[0, 0] + cm[1, 1]) / data['total']
        data['sensitivity'] = cm[0, 0] / (cm[0, 0] + cm[0, 1])
        data['specificity'] = cm[1, 1] / (cm[1, 0] + cm[1, 1])
        return data
def plot_cms(best):
    f, axes = plt.subplots(1, 5, figsize=(20, 5), sharey='row')
    for idx, (dt, score, X_test, y_test) in enumerate(best):
        y_pred = dt.predict(X_test)
        cm = confusion_matrix(y_test, y_pred)
        cm_data = '\n'.join([f'{key}: {value:.3f}' for key, value in get_cm_data(cm).items()])
        cmd = ConfusionMatrixDisplay(cm)
        cmd.plot(ax=axes[idx])
        cmd.ax_.set_title(f'score: {score:.3f}')
        cmd.im_.colorbar.remove()
        cmd.ax_.set_xlabel(cm_data)
        if idx != 0:
            cmd.ax_.set_ylabel('')
    plt.subplots_adjust(wspace=0.40, hspace=0.1)
    f.colorbar(cmd.im_, ax=axes)
    plt.savefig(f'{type(best[0][0])}_cm.svg')
    plt.show()
```

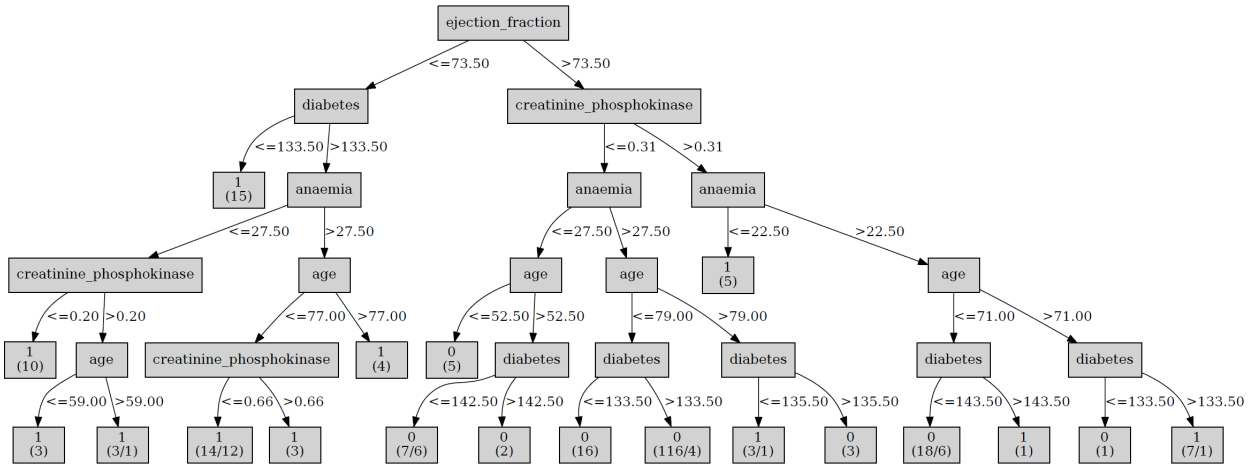
ג. <u>תוצאות הניתוחים של המודלים הדיוק של כל אחת מהשיטות ומידות הדיוק</u>

1. **עבור הנתונים שעברו טרנספורמציית *boxcox*** - אלגוריתם *ID3* אומן עם כלל הסטים של *k-fold* ומצאתי שחמשת התוצאות הטובות ביותר ו *confusion-matrix* מתאימים ביחד עם מדדי *total, accuracy, sensitivity, specificity, precision*:

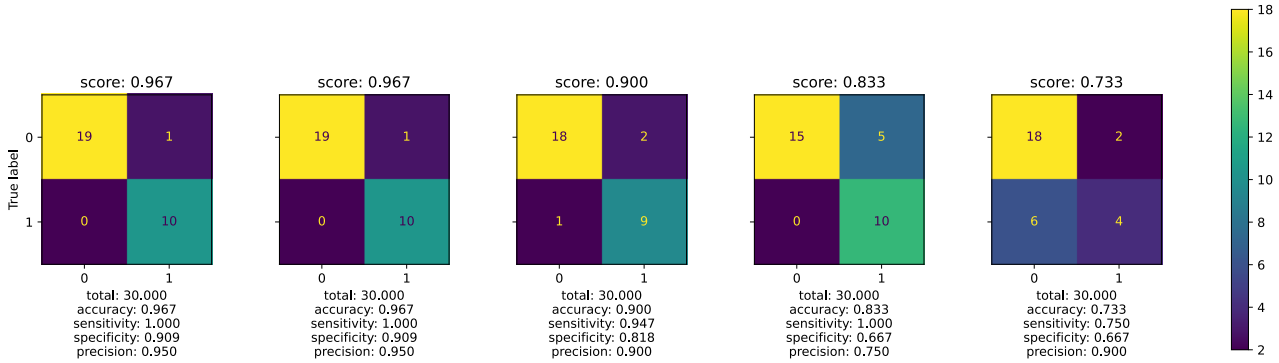


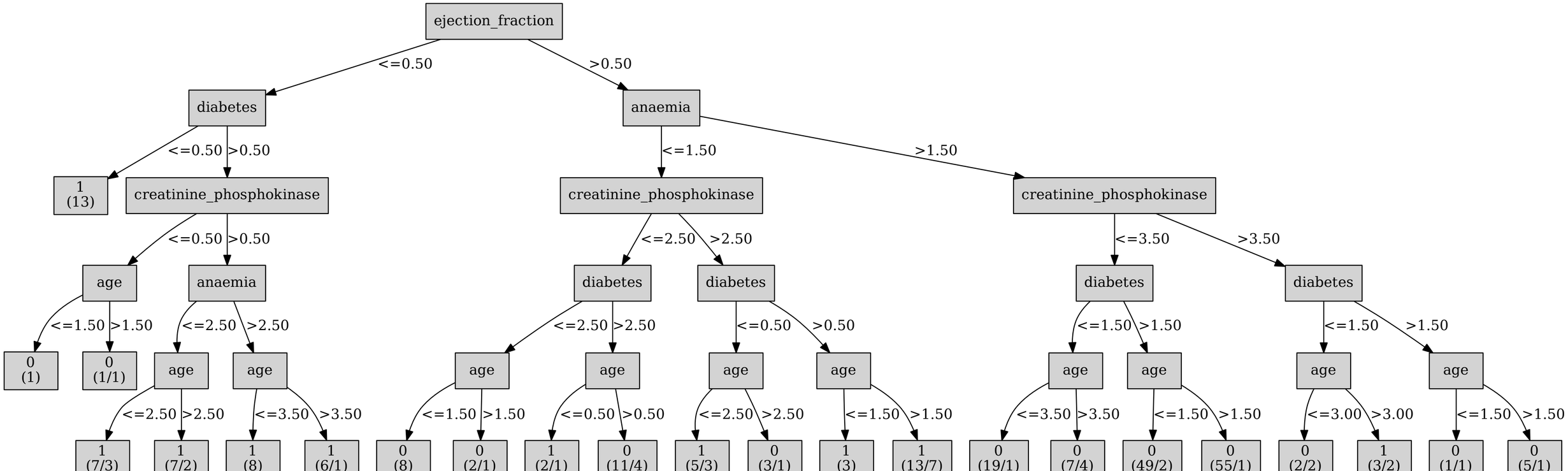

כאשר הדיוק המקסימלי הוא 93.3% וממוצע המודלים הוא 72.42% דיוק (יש יותר מ 5 מודלים שבתמונה).
קל להבין שהמודל הטוב ביותר ב *ID3* ככל הנראה נמצא במצב של *overfitting*, על פי הנתונים והדבר ברור מאליו כיוון שהמודל לא גוזם.

העץ שמומש:

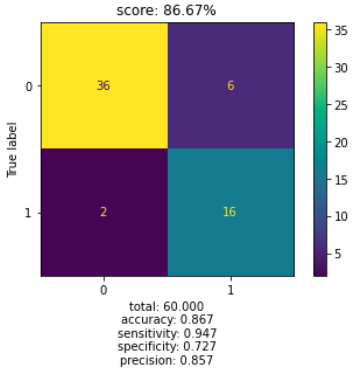


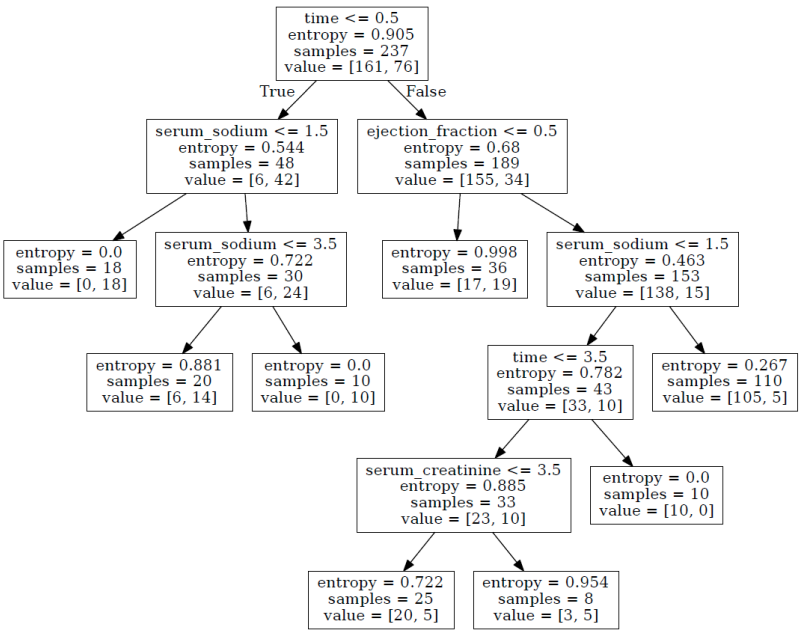

**עבור הנתונים שעברו *Binning*** - ביצעתי פעולות זהות וקיבלתי תוצאות טובות יותר (!) דבר שיכול ללמד על מצב של *overfit* בהסתכלות על העץ שנוצר, ניתן גם להבחין בזה לפי כמות העלים הגדולה:



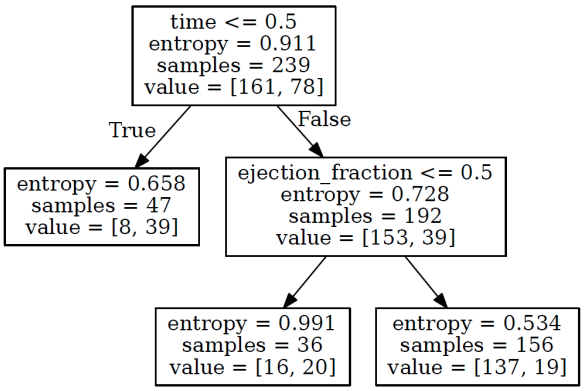

ejection_fraction
<=0.50
>0.50
diabetes
anaemia
<=0.50
>0.50
<=1.50
>1.50
1
(13)
creatinine_phosphokinase
creatinine_phosphokinase
creatinine_phosphokinase
<=0.50
>0.50
<=2.50
>2.50
<=3.50
>3.50
age
anaemia
diabetes
diabetes
diabetes
diabetes
<=1.50
>1.50
<=2.50
>2.50
<=0.50
>0.50
<=1.50
>1.50
<=1.50
>1.50
0
(1)
0
(1/1)
age
age
age
age
age
age
age
age
<=2.50
>2.50
<=3.50
>3.50
<=1.50
>1.50
<=0.50
>0.50
<=2.50
>2.50
<=1.50
>1.50
<=3.50
>3.50
<=1.50
>1.50
<=3.00
>3.00
<=1.50
>1.50
1
(7/3)
1
(7/2)
1
(8)
1
(6/1)
0
(8)
0
(2/1)
1
(2/1)
0
(11/4)
1
(5/3)
0
(3/1)
1
(3)
1
(13/7)
0
(19/1)
0
(7/4)
0
(49/2)
0
(55/1)
0
(2/2)
1
(3/2)
0
(1/1)
0
(5/1)

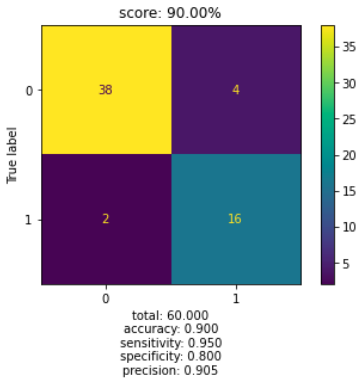
2. **עבור הנתונים שעברו טרנספורמציית *boxcox*** - אלגוריתם *CART* אומן עם *grid search cv* שמבצע *cross-validation* על סט הנתונים ומחזיר את המודל הטוב ביותר שמצא, ניסיתי להריץ עם שינוי מספר העלים בנוסף כדי לנסות לקבל תוצאות טובות, אחוז הדיוק גבוה, אך לא מספיק טוב בהשוואה לאחרים, לצידו העץ:




**עבור הנתונים שלא עברו את הטרנספורמציה אבל נכנסו ל*Binning*** – האלגוריתם אומן באותה צורה, עם *k-fold* ואכן התוצאות מרשימות גם כן! נראה שהדיוק של המודל הוא 90% והעץ מצורף:



ה. <u>ננתח השוואתית את התוצאות ונסיק מסקנות כולל הצעות לשיפור.</u>

שני האלגוריתמים רצו אחרי ביצוע *stratifiedt k-fold* עם *k=10*.

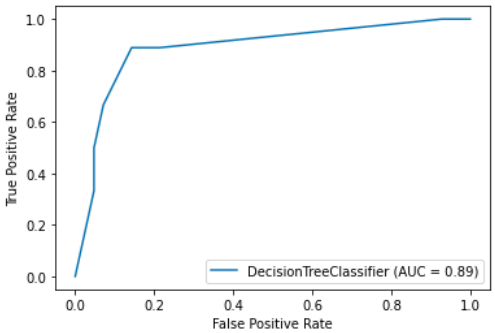
נבחין כי מכלל המודלים שאלגוריתם *ID3* ייצא, הטוב מביניהם הוא בעל אחוזי דיוק מאוד גבוהים כאשר הדיוק המקסימלי הוא 96.667% וממוצע המודלים הוא 78.4% דיוק.
קל להבין שהמודל הטוב ביותר ב *ID3* ככל הנראה נמצא במצב של *overfitting*, על פי הנתונים והדבר ברור מאליו כיוון שהמודל לא גוזם ויש המון עלים.

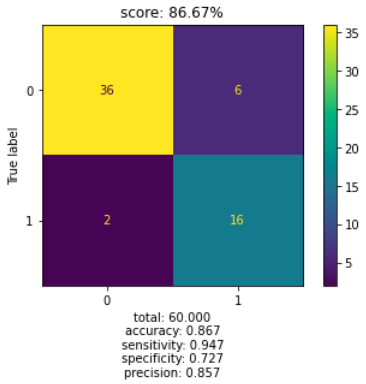
כיאה למודל שמבוסס על האלגוריתם, יש סיכוי לא קטן שנתקלנו ב *overfit* **בשתי הגישות**, שכן האלגוריתם, כפי שנאמר, לא גוזם ענפים והדבר יכול לגרום ל *overfit* (הוא נוטה לכך, נאמר כבר בסקירת האפשרויות בתחילת הפרויקט).

מכלל המודלים שאלגוריתם *CART* ייצא, הטוב מביניהם היה בעל אחוזי דיוק טובים גם כן – 90% בשימוש ב - *Binning* ופחות מכך, 86.67% בעזרת טרנספורמציות שביצעתי.
לפי אלגוריתם זה, פחות סביר שנתקלנו בבעיה של *overfitting* שכן הוא נוטה לגזום ענפים שיכולים לגרום לכך.

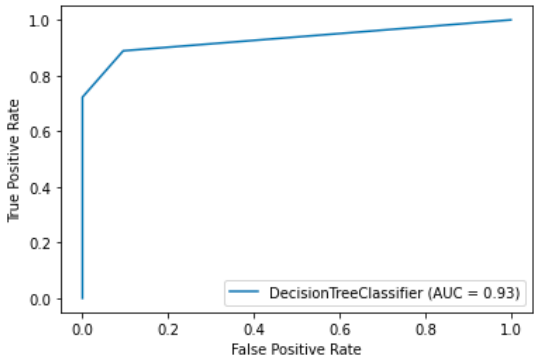
אלגוריתם ה *ID3* הטוב ביותר בנה עץ עם כ 22 עלים ואלגוריתם *CART* בנה עץ עם כ 3 עלים בלבד (כתוצאה מההגדרות שנתתי אך בלי ההגדרות הגיע גם ליותר), כמות העלים הגדולה ב *ID3* יכולה להצביע בהחלט על *overfit* מסוים.
מבחינת הנתונים היבשים, אציג *confusion matrix* ביחד עם מדדי הרגישות, ייחודיות, מדויקות ודיוק:

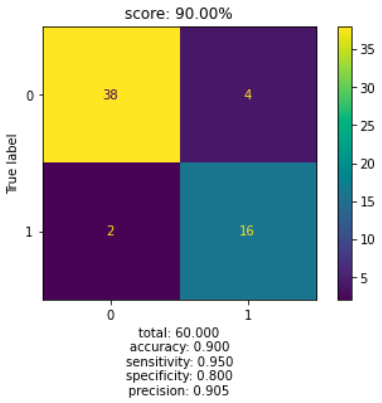
<u>אלגוריתם *CART* עם טרנס' *boxcox*, לצידו *ROC* עם שטח 0.89:</u>



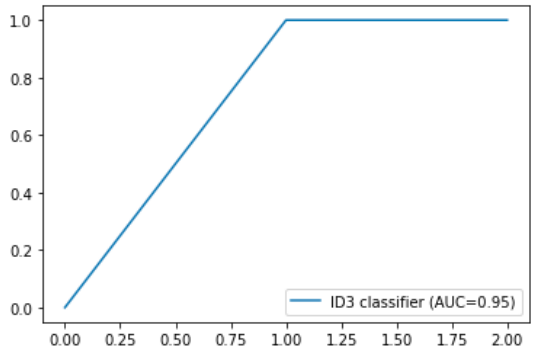



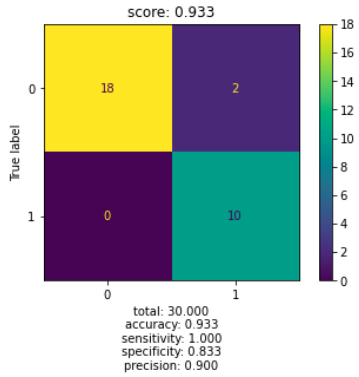

<u>אלגוריתם *CART* עם *Binning* לצידו *ROC* עם שטח 0.93:</u>



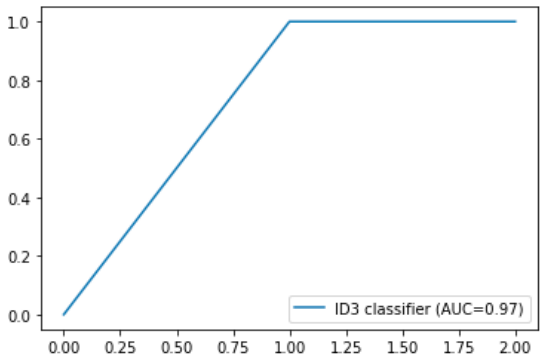



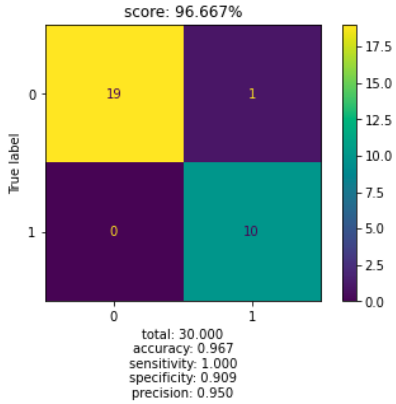
אלגוריתם *ID3* עם *boxcox*, הטוב ביותר לצידו *ROC* עם שטח 0.95:




אלגוריתם *ID3* עם *Binning*, הטוב ביותר, לצידו *ROC* עם שטח 0.97:




לסיכום הנתונים, הצלחתי להגיע לשטחים קרובים מאוד ל 1 מתחת ל *ROC curve* בעזרת רוב המודלים שבניתי.
מודל *CART* מבוסס *Binning* יצא המודל **המדויק ביותר הכולל גיזום** עם נתוני דיוק של 90%, רגישות 95%, ספציפיות 80% ומדויקות 90.5%.
מודל *ID3* מבוסס *Binning* יצא המודל **המדויק ביותר שלא כולל גיזום** עם נתוני דיוק של 96.667%, של רגישות 100%, של ספציפיות 90.9% ושל מדוייקות 95%.

שני האלגוריתמים רצו בזמן קצר, כיוון שסט האימון לא היה גדול במידה שיכולה להאט את האימון בצורה שנבחין בה.
ניתן להבחין כי **התכונה המשפיעה ביותר** ברוב העצים היא כמובן ה *time* שנמצא בשורשי העצים.
עם זאת, מקטע הפליטה היה שורש העצים של אלגוריתם *ID3* (בצורה די תמוהה לטעמי)
מצד אחד, נראה שתכונת הזמן יכולה להוביל לבעיות שכן היא יכולה להתפרש כהטיית הישרדות אך נבחין כי הניתוח הסטטיסטי שהתבצע לא התחשב רק באוכלוסיית המדגם שסיימה את הניסוי, אלא גם בחולים שמתו במהלך הניסוי, לכן הפרמטר הנ"ל הכרחי וחשוב!

לטעמי, ניתן לשפר את המודלים שיצרתי בעזרת שימוש בשינוי מספר העלים המקסימלי בכל ענף בעץ ההחלטה של אלגוריתם *CART*.
יתרה מזאת, ניסיתי לבצע זאת ואכן התקבל דיוק של 90-100% בחלק מהמודלים אך בכלל המידע המצומצם שיש בידינו שכולל כ 299 רשומות בלבד (ולאחר ניקוי עוד פחות) צריך לחשוב על ייצור מידע סינטטי על מנת לבדוק אופציות כאלו כיוון שנראה שאנחנו נמצאים ב *overfitting* רציני בהתאם לשינוי העלים.

הצעה נוספת לשיפור היא שימוש במודלים נוספים שלא למדנו בקורס, לדוגמה *RandomForest* או *SVM*, *ANN* וגם רגרסיה לוגיסטית.

**המסקנה האישית שלי**

ממ"ן 11 של הקורס היווה תירגול מעניין לנושא פשטני, אך הממ"ן הנ"ל היה פשוט כיף גדול לכתיבה וביצוע.
מצאתי עצמי משקיע שעות רבות בהבנה של **המון (!)** חומר חדש וישן שנשכח ולמדתי המון ממנו!
נושאים כמו צידוד (שאני מקווה שזה תקין שהשתמשתי בזה, אודה מאוד לדעת אם לא, בכל מקרה הוספתי את החלוקה ל *bins* כדי למנוע בעיות) או מציאת *outliers* בצורה מדודה אלו דברים שאנחנו לא מבצעים ביום-יום, אך הפרויקט הנ"ל עזר לי ללמוד הרבה דברים חדשים שהרחיבו את הידע שלי המון וגרמו לי לקפוץ ראש למים עמוקים – דבר מבורך, למדתי המון דברים, אך החשובים ביותר הם שאין דרך אחת לבצע משהו, אין דרך נכונה לבצע משהו, אבל מצאתי דרכים מעניינות להסתכל על המידע שלי והבנתי אותו בצורה די טובה לטעמי, צורה כזו שגרמה לי לנסות דבר אחד והיפוכו ואף לחזור אחורה על מנת להתקדם קדימה בצורה טובה יותר!

מתן