

עבודת מחשב 2 - השוואה בין רשתות עצביות, עצי החלטה ורגסיה לוגיסטיות באמצעות חיזוי הירידות לחולות סרטן השד

בית ספר להנדסת חשמל

קורס עיבוד אותות אקראיים 20337

מג'יסטים: ניצן איסמאלווב 961254082

גל נוימן 020261316

שם המרצה: פרופסור יצחק לפידות

A Comparison of Artificial Neural Network and שם המאמר:  
Decision Trees with Logistic Regression as Classification Models  
for Breast Cancer Survival

קישור: [A Comparison of Artificial Neu.pdf](#)

## תוכן עניינים

3 .....	מבוא ומודולציה לפרויקט
4 .....	רקע תיאורטי
4.....	למידת מכונה
5.....	אופן הפעולה של למידה מכונה
5.....	מסווגים
6.....	נורמליזציה
6.....	IQR
7.....	– רגרסיה לוגיסטיבית
8.....	– עצי החלטה
9.....	CRT
11.....	ANN – רשתות עצביות מלאכותיות
14.....	EDA
15.....	קורלציה ומטריצת קורלציה
16.....	הערכת ביצועי מודל וקריטריון טיב
18.....	K-Folds Cross Validation
21.....	עיבוד מקדים
28.....	Logistic Regression Model
30.....	Decision Tree Model
35.....	ANN Model
38.....	סיכום ומסקנות

## מבוא:

سرطان השד הוא סוג הסרטן השכיח ביותר בקרב נשים ברחבי העולם, והוא מהוות בעיה בריאותית חמורה המשפיעה על מיליון נשים מדי שנה. על פי נתוני ארגון הבריאות העולמי, (WHO) מדי שנה מأובחנות כ-3.2 מיליון נשים ברחבי העולם בחולות סרטן השד, והוא גורם לכ-685,000 מקרי מוות. למורת זאת, בעשרות השנים האחרונות התקדמות משמעותית בטיפולים, ובזכות גילוי מוקדם, שיעורי ההישרדות עלילו ניכר.

בישראל, כמו ברוב המדינות המפותחות, סרטן השד מהוות כ-30% מכלל מקרי הסרטן המאובחנים בקרב נשים, עם כ-5,500 מקרים חדשים בשנה. כ-90% מהנשים המאובחנות הסרטן השד מוקדם שורdot את המחלת לטווח של חמש שנים, נתן המעיד על החשיבות הקритית של בדיקות סקר ובחון מוקדם. הסיכון לפתח סרטן השד עולה עם הגיל, באשר רוב המקרים מאובחנים בקרב נשים מעל גיל 50, אך קיימים גם מקרים של נשים צעירות.

לסיכום, גילוי מוקדם של סרטן השד מגדיל משמעותית את סיכוי ההחלמה, עם שיעורי ההישרדות של מעל 90% באשר המחלת מתגלה בשלבים הראשונים. ככלים כמו מוגרפיה ובדיקה עצמית מאפשרים זיהוי גידולים קטנים לפני שהם מתפשטים, מה שמחזק את הצורך בטיפולים מוקדמים ומסייע במניעת תמותה. מוגרפיה תקופתית לנשים מעל גיל 50 הוכחה כמצמצמת את התמותה מהמחלה בכ-30%-40%. הعلاאת מודעות וערנות לשינויים בשדי הן קריטיות לשיפור סיכוי ההישרדות.

## מוסמיצה:

המוסמיצה נובעת מ.zaeuR לבחן ולזהות בצוואר מדעית ויעילה את הסיכויים להישרדות של חולות, להתבסס על מאפיינים קליניים וביוולוגיים שונים. על ידי השוואת יבולות הניבוי של כל אחת מהשיטות, ניתן יהיה להבין איזה מהן מספקת את התוצאות המדויקות והאמינות ביותר לצורך תכנון טיפול מותאם אישית בהמשך.

## פרק תיאורטי:

### למידת מוכנה:

למידת מוכנה (Machine Learning) הוא תחום מחקר ויישום בתחום הרוחב יותר של בינה מלאכותית (AI) המתמקד בפיתוח אלגוריתמים ומודלים המאפשרים למחשבים ללמידה ולקבל תחזיות או החלטות מביי להיות מתוכנתים במפורש. למידת מוכנה עוסקת בפיתוח מערכות שיכולות ללמידה והסתגל לנתחים, ולשפר את הביצועים שלהם לאורך זמן. אלגוריתמי למידת מוכנה נועדו למתוך אוטומטית במגוון גדולים של נתונים, לזהות דפוסים ולבצע תחזיות או החלטות על סמך הדפוסים שהתגלו. אלגוריתמים אלו לומדים מדוגמאות או תוצאות מסוימות, המשמשות כנתוני אימון, ומשתמשים בטכניקות סטטיסטיות כדי להכפיל מתרן נתונים אלו ולהימנע את מה שלמדו על נתונים חדשים, בלתי נראים. הרעיון המרכזי מlógה למידת מוכנה הוא לאפשר למחשבים ללמידה מנתחים ולקבל תחזיות או החלטות מדויקות, מביי להידרש לתוכנות מפורש עבור כל שימושה או בעיה ספציפית. היכולת הזו ללמידה ולהסתגל אוטומטית הופכת את למידת המוכנה לשימושית במיוחד עבור בעיות מורכבות שבין גישות תכנות מסורתיות מבוססות כללים עשויה להיות בלתי מעשית או בלתי ישירות. למידת מוכנה (ML) משמשת במגוון תחומיים, כולל עיבוד שפה טבעית (NLP), ראייה ממוחשבת (Computer Vision), זיהוי דיבור, חיזוי ורוביוטיקה. למידת מוכנה משמשת גם בתחוםים כמו פיננסים, בריאות ושיווק, שבהם מנהלים במגוון גדולים של נתונים כדי לבצע תחזיות ולקבל החלטות.

למידת מוכנה פועלת בכך שהיא מאפשרת למחשבים ללמידה מנתחים ולשפר את הביצועים שלהם במשימה או בעיה ספציפית מביי להיות מתוכנתים במפורש.

## אופן הפעולה של למידת מכונה

שלבים:

1. איסוף נתונים (Data Collection) זהו השלב הראשון שבו אוספים נתונים רלוונטיים לבעה שברצוננו לפתחה. הנתונים יכולים להציג מקורות שונים כמו מסדי נתונים, חישנים, אתרי אינטרנט, או קבצים. ככל שיש יותר נתונים, כך המודלים יהיו מדויקים יותר.
2. הכנת הנתונים (Data Preparation) הנתונים שנאספו צריכים לניוקו והבנה. שלב זה כולל טיפול בערכים חסרים, הסרת נתונים כפולים, התאמת פורמטים, ושינוי משתנים לבניה החדש. לעיתים נדרש גם שלבים של נרמול (Normalization) או קידוד (Encoding) של נתונים קטגוריאליים.
3. חלוקת הנתונים (Data Splitting) הנתונים מחולקים לשתי קבוצות עיקריות:
  - אימון (Training Set): הקבוצה שבה המודל למד את הקשרים בין התכונות השונות.
  - בדיקה (Test Set): קבוצה נשמרת כדי לבדוק את ביצוע המודל לאחר האימון.
  - לעיתים מחלקים גם קבוצה שלישית לשנקרת וlidציה (Validation Set) לצורך כוונון היפר-פרמטרים של המודל.
4. בחירת מודל (Model Selection) בשלב זה בוחרים את סוג המודל ללמידה. ישנו מודלים שונים שמתאימים לסוגי בעיות שונות:
  - גראסיה לינארית, עצי החלטה, יער אקראי (Random Forests), רשותות נירוניים, מכונת קטרורים תומכים (SVM), ועוד.
  - הבחירה תלויה בבעיה הספציפית (גראסיה, סיווג, וכו') ובMORECOMBOwnות הנתונים.
5. אימון המודל (Model Training) לאחר בחירת המודל, מבצעים את תהליכי האימון. הנתונים מועברים דרך המודל, והוא לומד לזהות את הדפוסים והקשרים בין נתונים. בתהליך זה המודל מתאים את הפרמטרים שלו כדי לצמצם את השגיאה בין התוצאות שלו לבין התוצאותจริง.
6. הערכת המודל (Model Evaluation) המודל נבחן באמצעות קבוצת הבדיקה שנשמרה מראש. מדדים כמו דיוק (Accuracy), דיוק מיקורי (Precision), ורישות (Recall), עיקומת ROC, ואחרים משמשים להערכת הביצועים של המודל.
7. כוונון היפר-פרמטרים (Hyperparameter Tuning) לאחר הערכת המודל, ניתן לשפר את ביצועיו באמצעות כוונון היפר-פרמטרים, כמו עומק עץ ההחלטות או שיעור הלמידה ברשותות נירוניים. תהליכי זה נעשו בדרך כלל על קבוצת הווילדייז.
8. פריסת המודל (Model Deployment) לאחר שהמודל עבר אימון והערכה, והוא מספק תוצאות טובות, הוא מוכן לפריסת (deployment). השלב זה כולל שילוב המודל במערכות קיימות, כגון אפליקציות או שירותים חיצוניים, לצורך שימוש בתרחישים אמיתיים.

### מסווגים

בסיוג אנו רוצים לקבל החלטה לפי מספר ה-class – אם שישנם אלגוריתם שלנו כך שבעור קלט אנו רוצים לקבל החלטה לאיזה class הוא שייך. לדוגמה, אם הקלט שלנו הוא תמונה של גבר ויש לנו שני classes גבר ('0') ואישה ('1'), אנו נרצה לשירות את הקלט שלנו ל-class '0'. שיטות שונות של מסווגים נפוצות בתחום למידת המכונה בעזרת למידה מפוקחת שמטרתה לאמן מודל שיבצע סיוג כך שהמודל מואמן בעזרת נתונים שאנו יודעים את התיאוג שלהם מראש.

### נורמליזציה

נורמליזציה היא טכניקה שבה משתמשים בחלוקת מהכנת הנתונים למידת מכונה. מטרת הנורמליזציה היא לשנות את ערכי העמודות המספריות במרקם הנתונים לסקללה משותפת מוביל לעותם הבדלים בטוחה' הערכים. בລמידת מכונה, לא כל מערך נתונים דרוש נורמליזציה. תחילה זה מדרש רק באשר לתכונות יש טווחים שונים. בפרויקט שלנו נNORMALIZATE את הנתונים בעזרת שיטתה - Z- SCORE .

שיטת זו מייצגת כמה סטיות תקן מתחת או מעל הממוצע הערך עבור כל עמודת הערכים

$$Z_{score} = \frac{data - \mu}{\sigma}$$

יתרנו שיטה זו היא שיטה זו מטפלת בערכים חריגים ביחס לממוצע אבל אם נקבע מאגר נתונים אשר יוכל לעמודות עם סדרי גודל שונים אחד מהשני שיטת נרמול זו לא תעוזר כיון שהוא עובדת על כל עמודה בנפרד. במאגר הנתונים שלנו סדרי הגודל בכל עמודה מספיק קרובים כדי לנרמל בעזרת שיטת z-score .

### Interquartile Range (IQR)

בעברית "טוח בין-רביעוני", הוא ממד סטטיסטי שמציג את הפיזור של ערכים בסט נתונים. הוא נחשב ברוח בין הרביעון הראשון (Q1) לרביעון השלישי (Q3) של הנתונים, ובכך מספק מידע על הטוח שבו נמצא 50% מהערכים המרכזיים בסט הנתונים.

чисוב ה-IQR:

1. רביעון ראשון (Q1): זה הערך שמתחתיו נמצא 25% מהנתונים. כלומר, זהה הערך שבспособו של דבר יפריד את הרביעון הראשון של הנתונים מהשאר.

2. רביעון שלישי (Q3): זה הערך שמתחתיו נמצא 75% מהנתונים, כלומר הוא מפריד בין הרביעון העליון של הנתונים לשאר הנתונים.

3. חישוב ה-IQR:

$$IQR = Q_3 - Q_1$$

שימושים של IQR:

- זיהוי חריגים: IQR משמש לעיתים קרובות לזיהוי ערכים חריגים (outliers).

ערכים שנמצאים מעל  $Q_3 + IQR * 1.5$  או מתחת ל-  $Q_1 - IQR * 1.5$  נחישבים לחריגים.

- תיאור פיזור: IQR מספק הבנה טוביה יותר של הפיזור של הנתונים, במיוחד כאשר ישניםערכים קיצוניים או הפצה לא סימטרית.

- השוואות בין קבוצות: IQR יכול לשמש להשוואת הפיזור של קבוצות שונות בסט הנתונים.

יתרונות של IQR:

- IQR הוא ממד עמיד לרעש ולערכים חריגים, שכן הוא מתמקד במרכז הנתונים ולא מושפע מערכים קיצוניים כמו ממוצע או טווח.

באופן כללי, IQR הוא כל שימושי להבנת הפיזור של נתונים סטטיסטיים ולזיהוי חריגים בסטי נתונים.

מתוך המאמר שלנו, אנו נמשש שלושה אלגוריתמים:

Logistic Regression	.1
Artificial neural networks (ANN)	.2
Decision Tree	.3

## גרסיה לוגיסטיבית - Logistic Regression

גרסיה לוגיסטיבית היא מודל סטטיסטי המתאר קשר בין משתנה שמי, כמו רמות המשנה המוסבר, לבין משתנים אחרים המכונים "משתנים מסבירים". שימוש נפוץ במודל הוא כאשר המשתנים המסבירים הם בעלי ערך רציף כלשהו. במקרים כאלו, הגרף שנוצר מתאר את ההסתברות להתרחשות מאורע כלשהו כתלות במסתנים הרציפים. המודל לבדו אינו מספיק כדי לקבוע קשר סיבתי בין המשתנים המסבירים והמשנה המוסבר.

ייחי  $Y$  משתנה איבוטי המתקבל שני ערכים: 0 ויחי  $X$  משתנה מסביר כלשהו.

כדי לבנות מודלקשר אפשרי בין  $X$  ל $Y$  ישנו צורך במספר הנחות:

1. קיים משתנה רציף  $Y^*$  כך שערכו של  $Y$  שווה 0 אם ערכו של  $Y^*$  קטן מסף כלשהו ושווה ל 1 אם ערכו של  $Y^*$  גדול מסף זה. יש לשים לב  $Y^*$  חלק מהנתונים אלא חלק מהמודל.
2. קיים קשר ביןאי בין  $Y^*$  ובין  $X$  המבוטא ע"י המודל:

$$Y^* = \beta_0 + \beta_1 * x + \epsilon$$

3. התפלגותו של  $\epsilon$  היא התפלגות לוגיסטיבית:

$$F(t) = \frac{e^t}{1 + e^t}$$

כל  $t$  ממשי.

למען הפשטות, נניח כי  $X$  הוא משתנה בדיד.

$$P(Y = 0|X = x) = 1 - p = P(Y = 1|X = x)$$

לכן, ע"פ ההנחה השנייה:

$$p = P(Y^* > 0|X = x) = P(\beta_0 + \beta_1 * x + \epsilon > 0) = P(\epsilon < \beta_0 + \beta_1 * x)$$

ע"פ ההנחה השלישית אודות התפלגות  $\epsilon$ :

$$p = P(\epsilon < \beta_0 + \beta_1 * x) = \frac{e^{\beta_0 + \beta_1 * x}}{1 + \beta_0 + \beta_1 * x}$$

לכן,

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 * x$$

ייחי  $Y$  משתנה מקרי ביןאי המתקבל את הערכים 0 ו- 1 ויחי  $X$  משתנה מקרי רב ממדדי

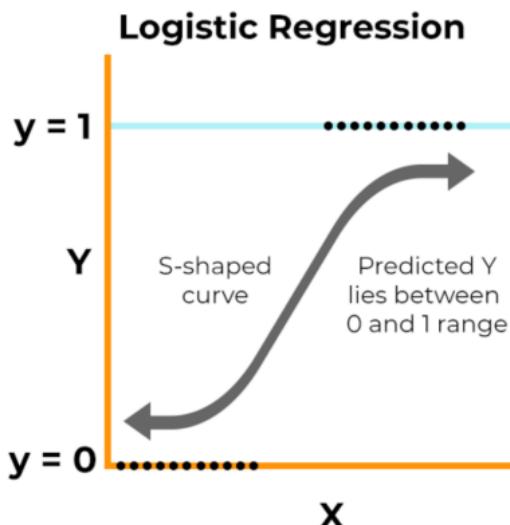
$$X = (X_1, X_2, \dots, X_I)$$

נסמן:

$$x = (x_1, x_2, \dots, x_I) \text{ ו } \pi(x) = P(Y = 1|X_1 = x_1, X_2 = x_2, \dots, X_I = x_I)$$

מודל הגרסיה הלוגיסטיבית הוא:

$$\log\left(\frac{\pi(x)}{1 - \pi(x)}\right) = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \dots + \beta_I * x_I$$



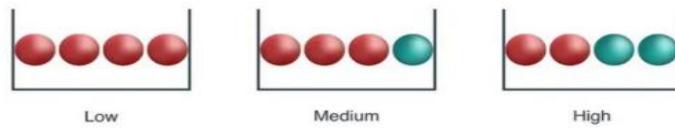
איור 1 פונקציית sigmoid עבור רוגסיה לוגיסטיית

## עץ החלטה- Tree Decision

עץ החלטה הוא מודל חיזוי תחומי הסטטיסטייה והלמידה החישובית (כגון למידת מכונה). עץ זה משתמש במשתנים מנכבים על מנת לסוג ערכים של משתנה מנכבה לתצפיות. תהליך היסויו הוא רב שלבי, כאשר בכל שלב נעשה שימוש בשאלת על אחד מהמשתנים המנכבים שתשובהו היא כן או לא. כאשר המשתנה המנכਬ הוא רציף, נהוג לבנות את עץ ההחלטה עץ גרסיה ובאשר המשתנה המנכਬ הוא קטגוריאי מקובל לבנות את עץ ההחלטה עץ סיווג. אלגוריתם עץ ההחלטה הוא אחד מאלגוריתמי למידת המכונה הפופולריים ביותר. זהו אלגוריתם למידת מכונה מפוקחת. ברוב המקרים העץ נבנה על סמך המידע מלמעלה למטה, קרי עליים מפצלים צמצמי החלטה בהתאם למידע שנוסף לעץ. ככל שלב נבחר מאפיין שעל פי יתבצע פיצול ב策ומת ההחלטה וערך סף מתאים קר שהतוצאות שיישוכו לכל תת-עץ תחת הצומת יהיו אחידות בערך ההחלטה שבוון בכל שנייתן. על מנת לבנות עץ ההחלטה, יש צורך להחליט על מدد אשר מבטא אותו העץ מנסה למחזר או למקסם. כך, שבמקרה אחר חלוקות העץ, התוצאות המשתייכות לחולקות אלו יופיעו על ידי ערך מינימלי או מקסימלי במדד זה. מגדדים מסווג נפוץ הם ממדדי הומוגניות של משתנה היעד בתוך תת-קובוצות, כאשר העץ מנסה למаксם את ההומוגניות.

סביר על ידי דוגמה:

נניח ישנן 3 קערות עם כדורים בשני צבעים:



איור 2 דוגמה למקורה עבור סוג הקבוצה

בוחרים כדור באופן רנדומלי, מכל הקערה. כמה מידע נדרש כדי להגיד במדויק מהו צבע הכדור?

ניתן להבחן שעבור הקערה השמאלית נctrוך הכדור כמעט מידע כיון שככל ה כדורים באוטו צבע, הצבע האדום ומאותה סיבה הקערה הימנית זקופה להבי הרבה מידע כיון שיש לה שני כדורים בצבע אדום ושני כדורים בצבע ירוק. מידע יכול להיות מدد לטהור ולכן ניתן לומר שמדובר שמאלית טהורה למרי ואילו שתי הקערות האחריות פחות טהורות (לא באותה מידת). חוסר הטהור יהווה ממד לפיצול עבור צומת מסוים.

### CRT (Classification and Regression Trees)

CRT היא שיטה נפוצה נוספת לחילוץ, המשמשת באלגוריתם "פיזול בינהř" (Binary Splits), ככלומר בכל צומת העץ, הפיזול מתבצע תמיד לשתי קבוצות בלבד (כן/לא). CRT מבוסס על הפחלה חוסר טויה (impurity), ככלומר, מנסה למקסם את ההבדלים בין הקבוצות שנוצרו.

#### :תכונות עיקריות של CRT

- פיזול בינהř: בכל צומת מתבצע פיזול לשתי קבוצות בלבד, וזה ממשיך עד שהעץ מגע לגודל מקסימלי או עד לעמידה בקריטריון עצירה.
- מדידת טויה: CRT משתמש במידדים של חוסר טויה (בגון מzd ג'י' או אנטרופיה) כדי לבחור את המסתנה המפצל בצורה האופטימלית ביותר.
- גיזום CRT: מבצע פיזול יתר (Pruning): ולאחר מכן גיזום של ענפים מיוחדים כדי להגיע לעץ אופטימלי.

#### :תרכונות CRT:

- יותר גמיש ופשוט להבנה עם פיצולים בינהרים ברורים.
- טוב להתמודד עם משתנים רציפים וגם עם קטגורים.
- מאפשר גיזום של העץ לצורך מניעת פיזול יתר (overfitting), מה שבוטב למודל עם יותר יכולת הכללה.

#### :חסרונות CRT:

- יכול להוביל לעצים גדולים ומורכבים יותר בגין פיזול יתר, אם לא מביצעים גיזום נכון.
- עלול להיות מושפע מחריגים ורעש בתנתונים אם לא מטפלים בכך נכון.

### Information Gain- Entropy

מדד information gain המוגדר על פי נוסחת האנטרופיה:

$$H(x) = - \sum_{i=1}^n p_i * \log_2 p_i$$

כאשר  $X$  הוא מרחיב הסתבותות סופי עם הסתבותויות  $p_1, p_2, \dots, p_n$  המיצגות את המאורעות השונים במרחב.

אנטרופיה היא במתמטית המידע הדורשה כדי לתאר במדויק את המדגם.

ערך האנטרופיה יכול לעמוד בין 0 ל-1. אם המדגם הוא הומוגני, פירוש הדבר שכל האלמנטים דומים ולכן האנטרופיה תהיה 0 (אין אי-ודאות) ואילו אם המדגם מחולק שווה בשווה אז האנטרופיה תהיה מקסימלית -1.

ובדוגמה שלנו - לערך השמאלית יש את האנטרופיה הנמוכה ביותר ואילו לערך הימנית יש את האנטרופיה הגבוהה ביותר.

הנוסחה Information Gain (חוות המידע) יתואר ע"י הנוסחה הבאה:

$$IG(X, a) = H(X) - H(X|a) = Entropy(X) - \sum_0^m \frac{X_m}{X} Entropy(X_m)$$

באשר:

- $a$  מייצג תכונה או קטgorיה מסוימת.

- אנטרופיה של  $X$  זה האנטרופיה של מאגר מידע  $X$ .
- $\frac{X_u}{X}$  מייצג את היחס בין הערך  $X_u$  למספר הערכים באותו מאגר המידע  $X$
- אנטרופיה של  $X_u$  זה האנטרופיה של מאגר מידע  $X_u$

לסיום,  $X$  זו האנטרופיה של כל סט המידע ו-  $X_u$  זו האנטרופיה של כל מאפיין בנפרד. מכך זה, לפי ההפרש, מראה כיצד לאחר הנחתת  $a$  או הוזאות שלנו מופחתת.

ז"א עברו חלוקה מסוימת בצומת מסוים, נמדד כיצד או הוזאות לגבי תיאור של המדגם, השתנה בעקבות אותה חלוקה.

### Gini Impurity

מדד ג'יני הוא מודד לאו – שווין במדגם. יש לו ערך בין 0 ל-1. ערכו של מודד ג'יני הינו 0 פירושו שהمدגם הומוגני לחילוטן וכל האלמנטים דומים ואילו אם מודד ג'יני בעל ערך 1 פירושו או שווין מקסימלי בין האלמנטים. זהו סכום ריבוע ההסתברויות של כל מחלקה, לפי הנוסחה:

$$Gini \text{ impurity} = 1 - \sum_{i=0}^n p_i^2$$

באמור, מודד ג'יני מודד את ההומוגניות במדגם הנתונים. אם המדגם הומוגני אז המדגם הוא מאותה מחלוקת. CRT היא שיטה המשמשת לייצרת עצי החלטה (Decision Trees) לצורכי סיוג וחיזוי נתונים. היא משתמש לביצוע פיצולים של נתונים כדי להזות דפוסים קשורים בין משתנים בלתי תלויים (המאפיינים) לבין משתנה תלוי (הפלט או המטרה).

## רשתות עצביות מלאכותיות (ANN)

רשתות עצביות מלאכותיות (Artificial Neural Networks, ANN) הן מערכות חישוביות שבנו בהשראת אופן פועלות המוח האנושי. הן מהוות את אחת הטכנולוגיות המרכזיות והמשמעותיות בתחום למידת מובנה ולמידה عمוקה (Deep Learning). רשתות עצביות משמשות בעיקר לפתרון בעיות מורכבות כגון גזע זיהוי, תמונה, עיבוד שפה טבעית (NLP), זיהוי דבר, משחקים אוטומטיים, ועוד.

### מבנה של רשת עצפית מלאכותית (ANN):

רשת עצפית מלאכותית מורכבת מספר שכבות של "נוירונים" מלאכותיים (או צמתים), בדומה לנוירונים במוח. כלנוירון מקבל קלט (Input), מבצע חישוב, ועביר את התוצאה קדימה לנוירונים אחרים בשכבות הבאות.

הרשת בנויה מ-3 סוגים עיקריים של שכבות:

#### 1. שכבת קלט (Input Layer):

שכבה זו מקבלת את המידע הראשוני (הקלט) שאוטו אנו רצים לעבד. מספר הנוירונים בשכבה זו תואם למספר התבוננות או המשתנים של הנתונים.

#### 2. שכבות נסתרות (Hidden Layers):

שכבות אלה נמצאות בין שכבת הקלט לשכבת הפלט, ומבצעות את רוב העיבוד והחישובים. כל שכבה כזו מורכבתenoiron, וכלנוירון בשכבה מחוברenoirons בשכבה הקודמת והבאה. רשתות יכולות לכלול שכבה נסתרת אחת או יותר, ורשתות בעלות מספר רב של שכבות נקראות רשתות למידה عمוקה (Deep Learning Networks).

#### 3. שכבת פלט (Output Layer):

השכבה האחורונה ברשת שמייקה את תוצאות הרשת – לחוב סיווג או חיזוי. מספר הנוירונים בשכבה זו תלוי במספר הפלטים הנדרשים. לדוגמה, אם יש צורך בסיווג ביןאר, תהיהנוירון אחד בפלט (עם ערך 0 או 1).

## איך ANN פועל?

הפעולה של רשת עצפית מבוססת על תהליך של למידה, שבו הרשת לומדת לזהות דפוסים מתוך נתונים:

#### 1. קלט והפצת מידע קדימה (Forward Propagation):

הקלט (בגון תמונה, טקסט, או כל סוג אחר של נתונים) מועבר דרך שכבת הקלט לשכבות הנסתרות. כלנוירון בשכבה נסתרת מקבל את התוצאה מהנוירונים בשכבה הקודמת, משקל את התוצאות באמצעות פונקציית הפעלה (Activation Function), ועביר את התוצאה קדימה לשכבה הבאה. בסופה של דבר, המידע מגיע לשכבת הפלט.

#### 2. פונקציות הפעלה (Activation Functions):

- כלנוירון מבצע חישוב על הקלט שלו ועביר את התוצאה דרך פונקציית הפעלה. הפונקציות הנפוצות כוללות:

- סיגמויד (Sigmoid): משמשת להחזרת ערכים בין 0 ל-1.

$$F(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}$$

- ReLU: מחדירה 0 אם הקלט קטן מ-0, ואת הקלט עצמו אם הוא חיובי.

$$f(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

- Tanh: פונקציה שמתאימה את הערכים להיות בין -1 ל-1.

#### 3. שגיאה והפצת שגיאה לאחר (Backward Propagation):

- לאחר שהרשת מספקת תוצאות, משווים את התוצאות לתוצאות האמיתית באמצעות פונקציית שגיאה (בגון ריבועי ההפרש או פונקציית חצייה). ההבדל בין התוצאות לתוצאות האמיתית מוכנה שגיאה.

- בשלב זה מתבצע תהליך של "הפצת שגיאה לאחר" (backpropagation), שבו הרשת מעדכנת את המשקלים של הקשרים בין הנוירונים כדי למצוות את השגיאה בתוצאה הסופית. זה נעשה על ידי חישוב הנגדות של פונקציית השגיאה לפי המשקלים השונים, ועדכון המשקלים בהתאם.

4. אימון ואופטימיזציה:  
תהליכי הלמידה מתבצעו לאורך מספר מחזוריים (Epochs), שבמהלכם הרשת ממשיכה לעדכן את המשקלים לשגיאה שנמצברת. ככל שהרשת מתאימה יותר, כך היא לומדת להזות את הדפוסים ולהפחית את השגיאה בתוצאות שלה. אלגוריתמים כמו ירידת מפל (Gradient Descent) משמשים לאופטימיזציה של המשקלים.

#### דוגמה לשימוש ב-ANN:

נניח שיש לנו מערכת ליהוו תמונות של חתולים וכלאים. מערכת ANN לומדת את הדפוסים בתמונות (צבעים, קצאות, צורות) באמצעות שכבות של נוירונים. ככל שהרשת מעמיקה יותר שכבות נסתרות, היא לומדת להזות דפוסים מורכבים יותר עד שלבסוף היא יכולה להזות אם בתמונה יש חתול או כלב ברמת דיוק גבוהה.

#### פונקציית log-likelihood:

פונקציית הלוג-ליקelihood (Log-Likelihood) היא אחת מפונקציות העלות הנפוצות ביותר בלמידה מכונה, במיוחד במקרים סטטיסטיים כמו **גרסיה לוגיסטי** או **רשתות נוירונים** עבור בעיות סיווג. פונקציה זו מודדת עד כמה המודל מתאים לנתחים בהתבסס על הסתברויות שהמודל מנבב עבור כל דוגמה בסט הנתונים.

במילים אחרות, הלוג-ליקelihood מנסה להעריך את הסיכוי שהמודל יניב את התוצאות שנצפו בסט הנתונים, ולכן המטריה שלנו היא למסם את פונקציית הלוג-ליקelihood, כלומר להגיע למודל שמספק את הסיכויים הגבוהים ביותר עבור התוצאות הנכונות.

הקשר לפונקציית העלות - בבעיות למידה מוכנה, אנחנו בעצם מודדים את פונקציית העלות, ולא ממסים את הלוג-ליקelihood. לכן, פונקציית העלות של גרסיה לוגיסטי היא בעצם הנטיב של הלוג-ליקelihood. לעומת זאת, במקרה למסם את ההסתברות, אנחנו מודדים את הנגטיב שלו:

פונקציית העלות ( $J(\theta)$ ) מוגדרת כך:

$$J(\theta) = -\log L(\theta) = - \sum_{i=1}^n (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i))$$

- $y_i$  זה 0 או 1.
- $\hat{y}_i$  זה ההסתברות לחזות ש  $y_i = 1$ .
- $y_i - \hat{y}_i$  זה ההסתברות לחזות ש  $y_i = 0$ .

#### רגוליזציה מסוג 2:

רגוליזציה מסוג 2 היא טכניקה המשמשת למניעת אימון יתר (overfitting) במקרים של למידה מוכנה. במצבים שבהם המודל מתאים יתר על המידה לנתחי האימון, הוא עלול ללמוד את הרעש והפרטים הללוonlyניים שבנתונים, מה שמקשה עליו להכיל על נתונים חדשים. רגוליזציה עוזרת להקטין את הסיכוי שזה יקרה על ידי הוספה נוספת על עריכים גדולים מדי של הפרמטרים במודל.

#### איך עובדת רגוליזציה מסוג 2?

רגוליזציה מסוג 2, שנקראת גם רגוליזציה Ridge, מוסיפה למודל ענישה שمبرובוסת על סכום הריבועים של הפרמטרים (או המשקלות) של המודל. הענישה הזאת מתווספת לפונקציית העלות, כך שבנוסף להסתאמת המודל לנתחים, המודל גם נדרש לשמור על הפרמטרים שלו קטנים ככל האפשר.

הצורה המתוקנת של פונקציית העלות עם רגולרייזציה מסוג 2 נראית כה:

$$J(\theta) = -\sum_{i=1}^n (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)) + \frac{\lambda}{2} \sum_{j=1}^p \theta_j^2$$

באשר:

- $\lambda$  הוא הפרמטר של הרגולרייזציה, השולט בעוצמת העונישה.
- $\sum_{j=1}^p \theta_j^2$  הוא סכום הריבועים של הפרמטרים  $\theta$  (המשקלות של המודל).

#### האינטרואיציה מאחרוי 2:

1. מנייעת ערכים גדולים של פרמטרים: העונישה הנוספת מבוססת על גודל המשקלות. אם המשקלות של המודל נעות גדולות מדי, זה עשוי להשפיע שהמודל מתאים את עצמו יתר על המידה לנתחי האימון. רגולרייזציה מסוג 2 מנסה לצמצם את ערכי המשקלות כך שלא יהיו גבוהים מדי.
2. הקטנת מרכיבות המודל: על ידי הקטנת ערכי המשקלות, הרגולרייזציה גורמת לכך שהמודל יתמקדש בתכונות החשובות ביותר ולא נסה להתאים את עצמו באופן מדויק מדי לכל דוגמה בסט האימון, מה שמקטין את מרכיבות המודל ועזר לו להכפיל טוב יותר.
3. שיליטה על עצמת הרגולרייזציה: הפרמטר  $\lambda$  קובע במה חשבה העונישה על ערכים גבוהים של פרמטרים. אם  $\lambda$  קטן מאוד, הרגולרייזציה תחיה חלה ומודול עשוי להתאים יתר על המידה. אם  $\lambda$  גדול מדי, המודל עשוי להיות פשוט מדי ולסבול מחתה-התאמה (underfitting).

#### למה זה עוזר למניע אימון יתר?

- אימון יתר קורה כאשר המודל מתאים את עצמו בצורה מדויקת מדי לנתחי האימון, כולל הרעש והחריגות. בתוכה מכבר, המודל לא מצליח להכפיל ברاءו על נתונים חדשים. רגולרייזציה מסוג 2 מוסיפה עונישה שמנועת מהמודל להסתמך יתר על המידה על הרכות מסוימות או להתאים יתר על המידה לנתחים הספציפיים של האימון. בכך שהיא "معدיפה" מודלים עם משקלות קטנות יותר, היא גורמת למודל להיות פשוט יותר ולשמור על יכולת הכללה טובה יותר.
- ביסיכום רגולרייזציה מסוג 2 היא טבניקה שמוסיפה עונישה לפונקציית העלות כדי להקטין את ערכי המשקלות של המודל. היא מסייעת למניע אימון יתר על ידי כך שהיא "מכריכה" את המודל להימנע מההתאמה מופרתת של הפרמטרים לנתחים, ובכך משפרת את היכולת של המודל להכפיל על נתונים חדשים.

#### בעיית התאמת יתר (Overfitting) ותת התאמה (Underfitting)

התאמת יתר (Overfitting) היא בעיה יסודית בסטטיסטיקה ובלימידת מכונה שבה המודל מותאם יתר על המידה לאוסף הנתונים ועל כן מצליח פחות ביצוע תחזיות. התאמת יתר מתרחשת כאשר המודל נקבע ע"י יותר פרמטרים מאשר הנתונים מצדדים.

כמו כן, ישנה בעיה נוספת והיא תת התאמה (Underfitting).

תופעה זו מתרחשת כאשר המודל הסטטיסטי פשוט מדי להציג כראוי את המבנה הבסיסי של הנתונים, למשל בעקבות מיעוט בפרמטרים המגדירים את המודל. דוגמה לכך היא למשל ניסויו להשתמש במודל לינארי לתיאור התנagoות לא לינארית.

## EDA

(Exploratory Data Analysis) EDA, או בעברית ניתוח נתונים חקרני, הוא שלב חשוב בתהיליך למידת מכונה וניתוח נתונים. מטרתו היא לחקור ולסכם את התכונות העיקריות של הנתונים לפני בניית מודלים או ביצוע ניתוחים סטטיסטיים מורכבים יותר. EDA מספק תובנות ראשוניות על מבנה הנתונים, פיזורם, מגמות, קורלציות וחרגים.

תהליכיים עיקריים ב-EDA:

1. בדיקת הנתונים:
  - בחינת המבנה הכללי של הנתונים: כמות השורות והעמודות, סוג הנתונים (מספריים, קטגוריאליים, תאריכים, טקסטים).
  - איתור נתונים חסרים (missing data) והחלטה כיצד לטפל בהם: להחליפם, להסיר או לאמוד.
2. סטטיסטיות תיאוריות:
  - חישוב מדדים כמו ממוצע, חציון, סטיית תקן, מינימום, מקסימום וכדומה, כדי להבין את ההתפלגות הבסיסית של כל משתנה.
3. הדמויות גרפיות:
  - היסטוגרמות: להראות את פיזור הנתונים ולהבין את הצורה שלהם (נורמלית, פעומנית, לא סימטרית).
  - דיאגרמת קופסה (Boxplot): ליזהו ערכים חריגים והבנה של הפיזור ברבעונים.
  - גרפי פיזור (Scatter plot): לבחון את הקשרים והמתאמים בין משתנים שונים.
  - מטריצות קורלציה: לבדוק את רמת הקשר בין משתנים מספריים.
4. זיהוי חריגים (Outliers)
  - באמצעות כל ניתוח כמו דיאגרמת קופסה (Boxplot) או IQR, ניתן לזהות ערכים חריגים שעולים מהשיפு על הנתונים או המודל.
5. קשרים בין משתנים:
  - זיהוי מתאימים וקשרים בין משתנים: איר משתנים מסוימים משפיעים אחד על השני, האם יש קשר ליניארי או לא.

מטרות EDA:

- הבנת הנתונים: EDA עוזר לקבל הבנה עמוקה של הנתונים לפני שנכנסים לבניית מודלים.
- הבנת הנתונים: במהלך EDA, ניתן לזהות בעיות בניתוחים (חסרים, חריגים, נתונים לא מדויקים) ולהחליט על הטיפול בהן.
- בדיקת הנחות: EDA מאפשר לבדוק האם הנתונים עומים על הנחות מסוימות כמו נורמליות, לניניאריות או איזומטריות, הנחוצות לנתחים סטטיסטיים או למודלים מסוימים.

לסיכום EDA הוא שלב חיוני להבנת הנתונים באופן חזותי וסטטיסטי לפני ניתוחים מתקדמים יותר או בניית מודלים. הוא עוזר לגלוות את התכונות הראשונות, להזות בעיות ולהבין את הנתונים בצורה מיטבית לשלבים הבאים של הניתוח.

**Log transformation** (המרה לוגריתם) הוא תהליך מתמטי שבו לוקחים את הלוגריתם של הנתונים כדי לשנות את ההתפלגות שלהם. מטרת הממרה היא לעזור להקטין את ההתווות בנתונים (skewness) ולהפוך אותם להתפלגות יותר נורמלית (דמי פעומן), מה שיעזר בשיפור הביצועים של מודלים סטטיסטיים או של למידת מכונה. ההמרה תבצע רק על ערכים חיוביים.

**Square Root Transformation** (המרה שורש ריבועי) הוא תהליך מתמטי שמטרתו לשנות את ההתפלגות של הנתונים על ידי לקיחת השורש הריבועי של כל ערך בנתונים. בדומה ל, **log-log transformation** (skewness) הוא לשפר את ההתפלגות של הנתונים, להפחית הטיה, ולצמצם את ההשפעה של ערכים קיצוניים (outliers). נציין כי עבור ערכים שליליים נבצע שורש מהערך המוחלט.

## קורלציה ומטריצת קורלציה

**מטריצת קורלציה** היא כלי סטטיסטי המשמש לניתוח הקשרים בין משתנים שונים במבנה נתונים. מטריצה זו מציגה את מקדמי הקורלציה בין כל זוג של משתנים, ובכך מספקת תובנות על עצמת וכוון הקשר בין המשתנים השונים.

### **מהי קורלציה?**

קורלציה היא ממד שמצביע על הקשר בין שני משתנים. היא יכולה להיות חיובית, שלילית, או אפסית:

- **קורלציה חיובית**: כאשר ערך של משתנה אחד עולה, גם הערך של המשתנה השני עולה (לדוגמה, ככל שהగובה עולה, גם הממשק עשו לעלות).
- **קורלציה שלילית**: כאשר ערך של משתנה אחד עולה, הערך של המשתנה השני יורדת (לדוגמה, ככל שהמחיר של מוצר עולה, הביקוש אליו עשוי לרדת).
- **קורלציה אפסית**: אין קשר בין המשתנים (השינוי בערך של משתנה אחד אינו משפיע על השני).

מקדמי הקורלציה נעים בטווח שבין -1 ל-1:

- 1: קורלציה חיובית מלאה (קשר חזק חיובי).
- -1: קורלציה שלילית מלאה (קשר חזק שלילי).
- 0: אין קשר בין המשתנים.

## One-Hot Encoding

קידוד (One-Hot) הוא טכנית המשמשת להמרת משתנים קטגוריים (משתנים בעלי ערכים מוגבלים וספצייפים, כמו צבע, מין, עיר וכו') לייצוג מספרי, על מנת שנitin יהיה להשתמש בהם במודלים של למידה מוכונה.

במודלים של למידה מוכונה, בעיקר אלה המבוססים על למידה ממוחשבת, (Machine Learning) משתנים קטגוריים לא יכולים להיות מזונים ישירות למודלים, כיוון שהם לא יודעים להתמודד עם מחוזות או שמות של קטגוריות. כדי לפתור זאת, אנחנו משתמשים ב One-Hot Encoding.

## הערכת ביצועי מודל וקריטריון טיב

### Confusion Matrix

Confusion Matrix פולטת 4 ערכים:

- החוליה סוג ב – '0' אך בפועל הוא '1' – False Positive.
- החוליה סוג ב – '0' והואאמת '0' – True Negative.
- החוליה סוג ב – '1' אך הוא בפועל '0' – False Negative.
- החוליה סוג ב – '1' והואאמת '1' – True Positive.

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

confusion matrix 3 אויר 3

ניתן לחישב ממדדים שונים לביצועי המודל:

1. Recall /Sensitivity – קובעת את שיעור ה – True Positive.

$$\text{Sensitivity/Recall} = \frac{TP}{TP + FN}$$

שיעור ה – sensitivity זהה להסתברות שלבדיקה חוליה יש תוצאות בדיקה חיוביות.

TP מתייחס למקרים שבהם חולים (אמיתיים) מסוווגים עם מחלה ע"י המודל.

FN מתייחס למקרים שבהם מטופלים בריאים מסוווגים בחולים.

2. Precision – זה מודד את המספר הכלול של ההצלחות של המודל באשר הוא סיווג חיובי ואכן זה חיובי חלקי המספר הכלול של חיוביות חזיות ע"י המודל. מתתקבל ע"י הנוסחה הבאה:

$$\text{Precision} = \frac{TP}{TP + FP}$$

3. Specificity - הסתברות שלבדיקה בריאה תהיה חיילתית, בעצם אנו מודדים את ההצלחות של המודל של התוצאות השליליות חלקו סך הפעם שישנה תוצאה שלילית. נקבל זאת באמצעות הנוסחה הבאה:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

כאשר,

NT מתייחס למקרים שבהם מטופלים בריאים אכן מסוווגים כבריאים.

FP מתייחס למקרים שבהם מטופלים חולים מסוווגים כבריאים.

4. ממד נוסף שnitן לחשב הוא ממד שמשמעותו את שתי הסתברויות האחרונות שחוושבו ומודד את רמת הדיק ה collateral (הצלה "לזהות" חולמים וכן הצלחה "לזהות בראים"). בעצם אנו מודדים את המספר הכללי של הסיווגים הנכונים חלקו המספר הכללי של המקרים. מתקובל ע"י הנוסחה הבאה:

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

F1 score .5

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

מדד שמאז בין הדיק (Precision) לבין הרגישות (Recall) באמצעות הממוצע ההרמוני שלהם. זהו מודד מועיל במיוחד בשיש חוסר איזון בין הנתונים החיוביים לשיליים.

### עקומת ROC

עקומת ROC (Receiver Operating Characteristic Curve) היא גרפ שמשמש להערכת הביצועים של מודל סיווג ביןארי (או רב-קטגוריאלי) על פני מגוון של ספי חיתוך. העקומה מציגה את הקשר בין רגישות (Recall) לבין שיעור התראות שווים (False Positive Rate, FPR) של המודל, אשר סף הסיווג משתנה.

מנוחים חשובים לפניהם שבין את העקומה:

- רגישות (Sensitivity / Recall): אחוז המקרים החיוביים שהמודל זיהה נכון מתוך כל המקרים החיוביים בפועל.

$$Sensitivity/Recall = \frac{TP}{TP + FN}$$

- שיעור התראות שווים (False Positive Rate, FPR): אחוז המקרים השיליים שהמודל סיווג בטעות כחיוביים מתוך כל המקרים השיליים בפועל.

$$FPR = \frac{FP}{FP + TN}$$

### :בנייה עקומת ROC

עקומת ROC נבנית על ידי שינוי הסף (threshold) שמעליו המודל מסוו את הדוגמאות חיוביות. ככל שהסף משתנה, בר גם ייחס רגישות ושיעור התראות השווים משתנה, מה שמאפשר להפיק את העקומה.

בציר ה-Y של הגרף נציג את הרגישות (Sensitivity). בציר ה-X נציג את שיעור התראות השווים (FPR). העקומה מראה כיצד המודל מאנז בין זיהוי חיובי נכון (TP) לבין הפקת התראות שווים (FP) ב嚷ון ספי חיתוך.

### :איך לקרוא את עקומת ROC

נקודה טובה על העקומה: עקומה שנמצאת קרוב לציר ה-Y ובזווית רחוכה מוקו האלבסוני (המסמן ביצועים אקראים) נחשבת טובה. העקומה האידיאלית היא פינהعلינה שמאלית של הגרף (כלומר, רגישות גבוהה ושיעור FP נמוך).

קו אלבסוני (ביצוע אקראי): אם עקומת ROC של המודל קרובהקו האלבסוני, המשמעות היא שהמודל כמעט לא מבצע הבחנה בין הקטגוריות והביצועים שלו הם כמו ניחוש אקראי.

עקומה גבוהה: ככל שהעקומה קרובה יותר לפינה השמאלית העליונה, בר המודל טוב יותר. המשמעות היא שהוא מושג רגישות גבוהה תוך שמירה על שיעור נמוך של FP.

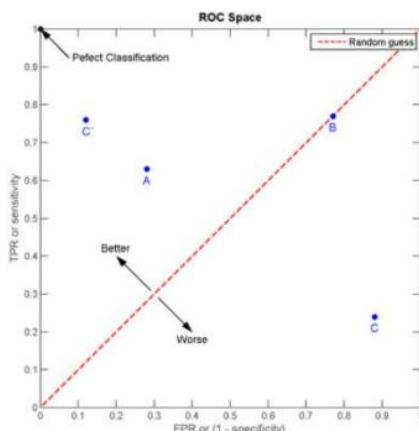
### מדד ה-(AUC (Area Under Curve)

המדד המركבי שמשופק מעוקמת ROC הוא AUC – שטח תחת העוקמה (Area Under the Curve). מzd זה מייצג את הביצועים הכלליים של המודל בכל ספֵי החיתוך האפשרי:

- $AUC = 1$ : מודל מושלם שמסוג נכון כל הדוגמאות.
- $AUC = 0.5$ : מודל שמתפרק כמו ניחוש אקראי.
- $AUC$  קטן מ-0.5: מודל שמסוג בטעות גורעה יותר מניחוש אקראי (ביצועים גרועים).

יתרונות עוקמת ROC:

הערכת ביצועים מרובים: ROC מאפשרת להעריך את ביצועי המודל על פני מגוון ספִים, ולא רק בנקודה מסוימת אחת כמו Confusion Matrix, מה שמספק הבנה רחבת יותר של ביצועי המודל.  
השוואה בין מודלים: ניתן להשוות בין מספר מודלים באמצעות עוקמות ROC שלהם, כאשר המודל עם ה-AUC הגדול ביותר הוא המודל הטוב יותר.  
ביצועים בלתי תלויים בספֵי: ניתן לראות את כל טווח הביצועים של המודל על פני כל הספִים, ולא להיות תלויים בבחירה ספֵי ספציפי.



איור 4 עוקמת ROC

### Cross Validation

קורס-olidzia (Cross-Validation) היא שיטה להערכת ביצועים של מודל למידת מכונה על ידי חלוקת הנתונים לשיטים שונים של אימון ובדיקה בצורה שיטית. המטרה העיקרית של קروس-olidzia היא להימנע מההתאמה יתרה (overfitting) לנתחי האימון ולהבטיח שהמודול יוכל להצליח גם על נתונים שלא ראה לפני כן.

במהלך קروس-olidzia, הנתונים מחולקים לכמה חלקים, ובכל איטרציה המודול מאומן על חלק אחד של הנתונים ומוסרך על חלק אחר. התהליך חוזר על עצמו מספר פעמים, ולאחר מכן מבוצעת ממוצע הביצועים כדי לקבל מדד כולל של ביצועי המודל.

### k-fold cross-validation

בשיטת ה-k-fold cross-validation, מחלקים את הנתונים ל-k חלקים שווים בגודלם, הנקראים "קיפולים" (folds). בכל איטרציה מאמנים את המודל על 1-k קיפולים, ובזקרים את ביצועי המודל על הקיפול הנotor. ההתהליך זה חוזר k פעמים, כאשר בכל פעם קיפול אחד משמש לבדיקה, והקיפולים האחרים משמשים לאימון.

איך השיטה עובדת:

1. חלוקת הנתונים:

נניח שיש לנו סט נתונים, למשל 100 דוגמאות, נבחר ב- $k$  מסויים, למשל 5, ועכשיו נחלק את הנתונים ל-5 חלקים שווים בגודלם, שככל אחד מהם מכיל 20 דוגמאות.

2. אימון ובדיקה:

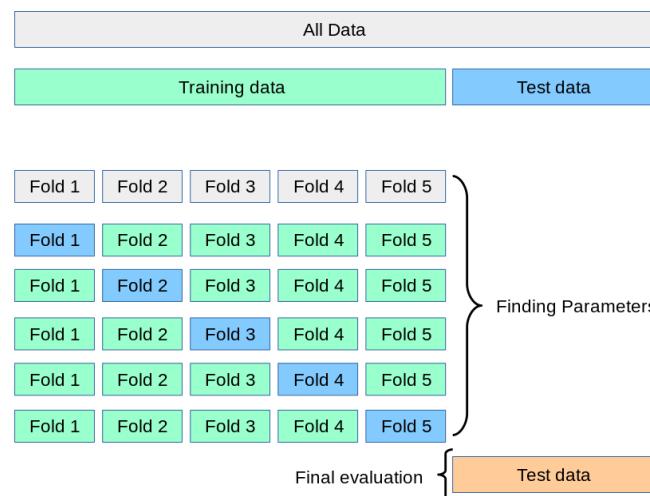
באייטרציה הראשונה, המודל יאמון על 80 דוגמאות (4 קיפולים) ויבדק על 20 הדוגמאות הננותראות מהקיטול החמשי. באיטרציה השנייה, נשתמש ב-4 קיפולים שונים לאימון ונבדוק על קיטול אחר, וכך הלאה עד שככל אחד מהקיטולים ישמש פעם אחת כקובוצת הבדיקה.

3. חישוב ממוצע הביצועים:

לאחר שכל k האיטרציות הסתיימו, מחשבים את ממוצע הביצועים מכל האיטרציות כדי לקבל הערכה כוללת של המודל. בדרך זו מקבלים מدد מדויק יותר של ביצועי המודל, שכן הוא נבדק על חלקים שונים של הנתונים.

**תרונות של k-fold cross-validation:**

- שימושiesel בעיל בנתונים: כל דוגמה בסט הנתונים משמשת פעם אחת לבחינת המודל ופעם אחת לאימון, מה שמבטיח ניצול מרבי של כל הדוגמאות.
- הפחיתה הטיות: מכיוון שהערכת הביצועים נעשית מספר פעמים על קבוצות שונות של נתונים, השיטה מקטינה את הסיכוי שההתוצאות תושפנה ממבנה ספציפי של נתונים.
- הקטנת overfitting: השיטה מאפשרת למודל ללמוד מכל הדוגמאות בצורה שיטית, ומונעת מצב שבו המודל מתאים עצמו יתר על המידה לסט נתונים מסוים.



איור 5 CV בשיטת K-Folds

### פונקציית loss:

פונקציית Loss (או פונקציית הפסד) היא מושג מרכזי בלמידת מכונה ובאופטימיזציה, והוא משמשת להערכת כמה טוב המודל מבנה את התוצאות הנכונות ביחס לנתחי האימון. המטרה של כל מודל בלמידת מכונה היא לזרע את פונקציית הפסד, כלומר לצמצם את ההבדל בין התוצאות החזויות לבין התוצאות האמיתיות ככל האפשר.

ביצד עובדת פונקציית Loss?

במהלך אימון המודל, עברו כל תצפיות במערכת הנתונים, המודל מייצר תחזית. פונקציית loss מחשבת את השגיאה בין תחזית זו לבין הערך האמתי. אם פונקציית הפסד מחזירה ערך גבוה, זה אומר שהמודל עשה תחזית גרועה, ואם הערך נמוך, המודל עשה תחזית טובה.

### **פונקציית Cross-Entropy Loss**

אחד מהדוגמאות לפונקציית הפסד היא Cross-Entropy Loss :

$$L = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

כasher:

- $y_i$  הוא הערך האמתי (0 או 1)
- $\hat{y}_i$  הוא ההסתברות החזויה עבור הקטגוריה החיובית
- $n$  הוא מספר הדוגמאות

פונקציה זו מודדת את ההפרש בין ההסתברויות החזויות לבין הערכים האמתיים. המטרה במהלך האימון היא לזרע את פונקציית הפסד זו, מה שמצויב על תוצאות טובות יותר של המודל.

## חיזוי היישרות של חולות סרטן השד בהתקפס על מידע קודם של החולה:

### עיבוד מקדים:

העבודה מתחילה מכך שאנחנו בוחרים את בסיס הנתונים שלנו, המידע המשמש לניטוח במחקר זה מגיע ממאגר ה-"SEER" שכולל נתונים מפורטים על מקרי סרטן השד בארה"ב. הדאטה כולל 47,167 רשומות של חולות סרטן השד עם פרטים כמו גיל המטופלת, גודל הסרטן, שלב הסרטן, סוג הטיפול שהוענק ומשך זמן מהאבחנה ועד למועד הזמן שבו נלקחים הנתונים ועוד. לצורך העבודה שלנו נלקח מהמאגר 569 רשומות, ונציין כי מספר הקטגוריות במאגר שלנו הוא 37. נציין כי מערכת הנתונים שלנו בניו בקרה שכל שורה מתאימה לאדם פרטי וכל עמודה מצינית תכונה ספציפית.

הרעין העיקרי של המערכת הוא מיצי תכונה- היבט מרכזי בלמידה מוכנה, שבו נתונים הופכים לייצוגים עם משמעות הניתנים לניטוח.

נציין כי בדרך כלל בעיבוד מקדים של הנתונים יש צורך להחליט מה לעשות עם מטופלים שתכונה מסוימת אצל אינה קיימת או אינה תקינה מה שמבטיח ניקיון אויכות הנתונים, שהם תנאי מקדים לאימון והערכה של מודלים אמיתיים, (مونע הטיות), אך המאגר שאנו השתמשנו בו בבר מטפל בבעיה זו ולכן הנתונים שלנו מלאים.

כמו כן נזכיר כי בסוף כל התהליך אנחנו צריכים תוצאה ביןרית האם החולה תשרוד או לא וכן הפעולה הראשונה שנעשהזה לשנות את התכונה של הסתברות היישרות (Survival probability) לתכונה שנקראת target וממספר בין 0 ל-1 לתשובה ביןרית של 0 (לא שרדה) ו-1 (שרדה), נעשה זאת ע"י בדיקה פשוטה אם ההסתברות גדולה 0.5 אזי התוצאה תהופיע ל"1" ואם קטן מ-0.5 אזי התוצאה תהופיע ל"0". וכך גם בבדיקה אם את התכונה censor הורדנו למלי כי אין לה חשיבות בחישובים שלנו

survival_probability	target
0.829205	1
0.780743	1
0.507499	1
0.480011	0
0.485004	0

איור 6 נתונים survival prob לפני ואחרי השינוי

בשלב הבא מערך הנתונים מפוצל לחת-קבוצות – אימון ובדיקה באשר היחסים הם: 80% אימון, 20% בדיקה. הקפדנו לשמור את התפלגות משתנה היעד באמצעות פרמטר ריבוד. ככלומר נראה שההתפלגות דומה גם בסט האימון וגם בסט הבדיקה.

Stratification of train set: 0.6681318681318681

Stratification of test set: 0.6666666666666666

איור 7 התפלגות הריבוד בסט האימון והבדיקה

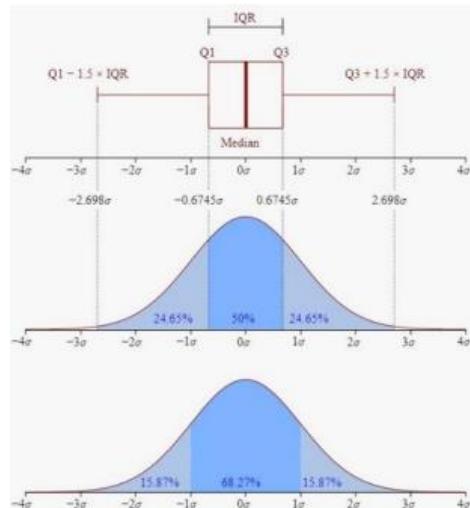
בשלב הבא נתאים את הנתונים לסוגי נתונים קטגוריים ומספריים:

את תכונת diagnosis מאייה סוג הפכנו לtrue/false, את עמידות stage of cancer הוחבנו ל4 עמידות של שלב 1,2,3,4 שהן גם מחזירות true/false, כמו כן גם סוג הטיפול הרחבנו לארבעת הסוגים שיבולים להחזר true/false

זהו חריגם- כדי להמשיך לנתח את הנתונים וליצור מערך נתונים כזה שלא ישפייע באופן חריג על תוצאות חיזוי המודל שלנו, ז"א מערך נתונים אמיתי, נרצה להציג אותו בעוד דרך ויזואלית כדי שנוכל לראות הימצאות

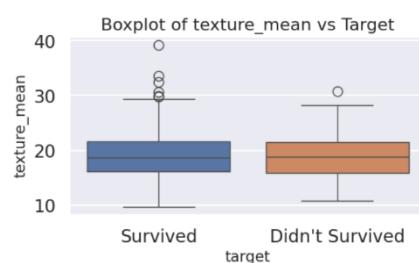
ערכים חריגים עבור כל תכונה אצל המטופלים. אך, חישבנו את ה – IQR\* 1.5 – ונקבעו את תכונת "Boxplot" נצפו מטופלים בעלי ערכים חריגים.

כאשר IQR מסתכם על החיצון ונוטן תחום סביבו לפי האIOR מס' 8 ייצורו Boxplot (לפי מה שקבענו לראות באדום)

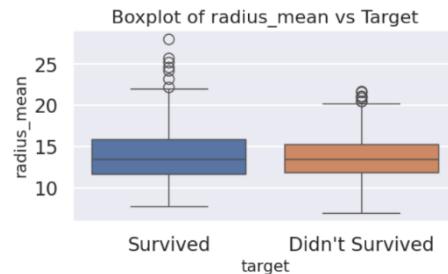


איור 8 בניית IQR

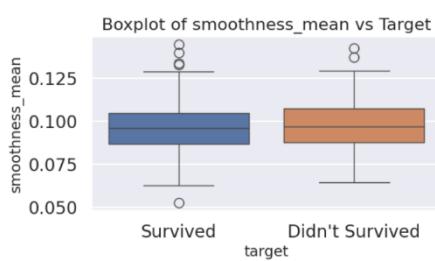
להלן מספר דוגמאות לתוצאות התכונות:



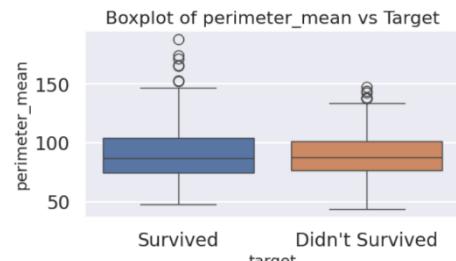
איור 10 דוחי חריגים לפי טקסטורה



איור 9 דוחי חריגים לפי רדיוס



איור 12 דוחי חריגים לפי חלקות



איור 11 דוחי חריגים לפי היקף

באירועים הב"ל ניתן לראות חריגות כאשר כל שגול מייצג חוליה שערבה עבור תכונה זו חרגת.

נציין כי כל חוליה אשר יש לה יותר מחרג אחד הורידנו אותה מערך הנתונים. סה"כ הורידנו 176 חוליות.

בשלב הבא תחלמו את תהליךEDA (নিয়ন্ত্রণ নথুন চৰকৰণ), אנו רוצים לבדוק האם עבור משתנים נומריים (מספריים) ישן הטוות של המשתנה (איסימטריות של התפלגות) ונרצה לתקן בהתאם באמצעות טרנספורמציה.

ישנו צורך בתיקון התפלגיות בעלי הטיה מכמה סיבות:

- התפלגיות מוטות יכולות להכניס הטיה לתחזיות המודל. אם הנתונים מוטים מאוד לכך אחד של התתפלגות, המודל עשוי להיות מאמין להעדיין ביצוע תחזיות המתאימות למעמד הרוב או למגמה השולטת בנתונים, מה שיבילל להוצאות מוטות.
  - יעילות משפיע על הממוצע והחיצון של הנתונים בצורה שונה. בתתפלגיות מוטות חיוביות (זנב ארוך ימיןה), הממוצע בדרכו כל גדול מהחיצון, בעוד בתתפלגיות מוטות שליליות (זנב ארוך לשמאלו), הממוצע בדרך כלל קטן מהחיצון. זה יכול להשפיע על אלגוריתמים המסתמכים על חישובים מוצעים או חיצוניים, כגון אלגוריתמי עצי החלטה.
  - נתונים מוטים יכולים להפוך מודלים בעלי רגשות גבואה. נתונים החירגים בעלי הטיה, עלולים להיות השפעה לא פרופורציונלית על אימון המודל, כגון באלגוריתמים כמו ורגסיה לינארית, שביהם המודל מנסה לפחות את סכום השגיאות בריבוע.

#### **התפלגות הנתוניים:**



איור 13 התפלגות הנתונים

כדי לדעת עד כמה הנתונים מוטים נשתמש בפונקציית `skew()`

```
skewed_features = train_data[numerical_features].skew().sort_values(ascending=False)

skewness_df = pd.DataFrame({'Skew': skewed_features})

print(skewness_df)
```

איור 14 פונקציית `skew`

והתוצאה

	Skew
area_se	1.659755
id	1.484636
concavity_mean	1.369991
area_worst	1.280565
perimeter_se	1.176789
radius_se	1.166624
area_mean	1.141010
fractal_dimension_se	1.115306
concave_points_mean	1.087700
concavity_se	1.071583
compactness_worst	0.992317
compactness_se	0.987362
concavity_worst	0.961041
symmetry_se	0.824792
perimeter_worst	0.791964
radius_worst	0.775978
fractal_dimension_worst	0.773376
compactness_mean	0.712874
smoothness_se	0.695264
texture_se	0.659345
perimeter_mean	0.627970
radius_mean	0.593535
concave_points_worst	0.569812
fractal_dimension_mean	0.559172
texture_mean	0.451940
symmetry_worst	0.420912
smoothness_worst	0.310679
texture_worst	0.308110
smoothness_mean	0.288099
concave_points_se	0.269053
age	0.094380
symmetry_mean	0.092489
duration	-0.095762
target	-0.643194

איור 15 תוצאות פונקציית `skew` לתבונן

עבשים נבצע Log Transformation עברו הטיה ימנית וthreshold Square Root Transformation עברו הטיה שמאלית, נציין כי ה threshold שלנו הוא 0.75 עברו ימני ומינוס 0.75 עברו שמאל.

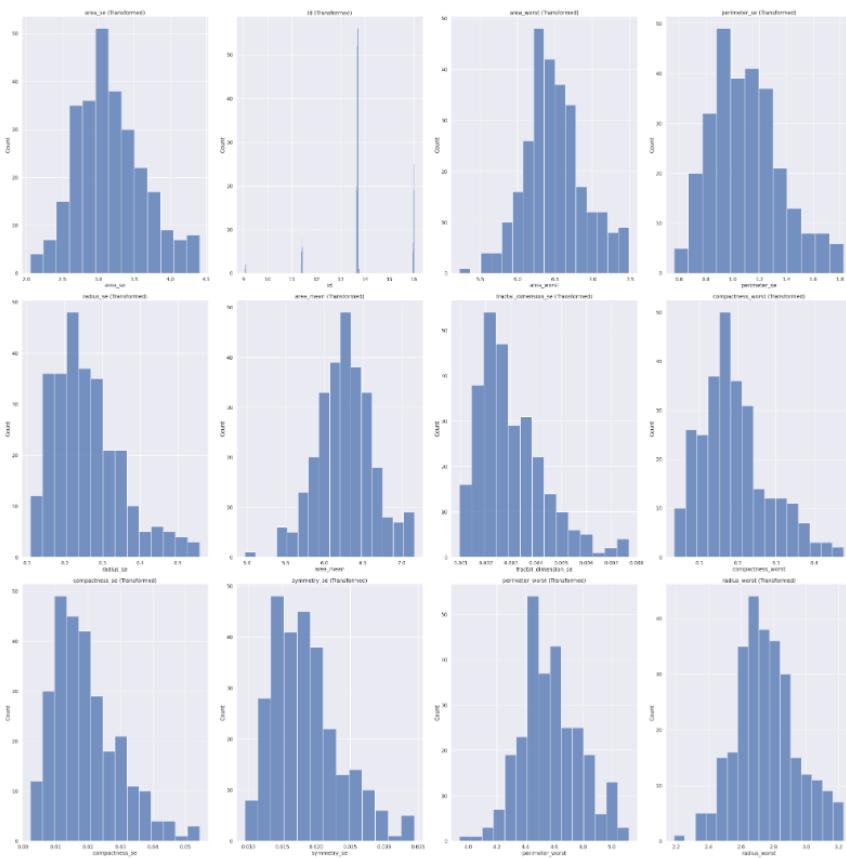
\* נציין כי יש 4 תכונות שדים גם עליהם והן: concavity mean, concave points mean, concavity se, concavity worst. הסיבה שדים גם עליהם היא שיש להם ערכים שליליים או אפס בDATA.

מבין כל הרשימה למעלה באירור 15 נראה רשימה חדשה של התכונות שביצעו בהן את הטרנספורמציות.

Features with Skewness Changes: Feature: area_se Original Skewness: 1.659755220881787 New Skewness: 0.42398965934030164	Feature: compactness_worst Original Skewness: 0.9923165777030903 New Skewness: 0.7611325185574692
Feature: id Original Skewness: 1.4846359778232827 New Skewness: -0.45710444975178427	Feature: compactness_se Original Skewness: 0.9873615564845798 New Skewness: 0.9610520072425602
Feature: area_worst Original Skewness: 1.2805654737601706 New Skewness: 0.1948748362214927	Feature: symmetry_se Original Skewness: 0.824791940484055 New Skewness: 0.8117734004862318
Feature: perimeter_se Original Skewness: 1.1767893013645345 New Skewness: 0.507598934388331	Feature: perimeter_worst Original Skewness: 0.7919635852266184 New Skewness: 0.22795226120103185
Feature: radius_se Original Skewness: 1.166623893774386 New Skewness: 0.9095294477822646	Feature: radius_worst Original Skewness: 0.7759777264470128 New Skewness: 0.24573454652637125
Feature: area_mean Original Skewness: 1.1410095006193648 New Skewness: -0.02312677896589771	Feature: fractal_dimension_worst Original Skewness: 0.773376441251809 New Skewness: 0.7416643622410701
Feature: fractal_dimension_se Original Skewness: 1.1153064068330594 New Skewness: 1.1112102628401086	

איור 16 קודם וחדר עבור התכונות שעבורו טרנספורמציה

וההטפלות לאחר הטרנספורמציות:



[איר 17 התפלגות חדשה לאחר הטרנספורמציה](#)

לאחר הטרנספורמציות, נסיר את התבוננות שעדיין יש להן הינה גבואה

Features to be removed due to high skewness (>0.75):

- concavity\_mean
- fractal\_dimension\_se
- concave points\_mean
- concavity\_se
- compactness\_se
- concavity\_worst
- radius\_se
- symmetry\_se
- compactness\_worst

Number of features removed: 9

[איר 18 התבוננות שהוסה בגל ההטיה](#)

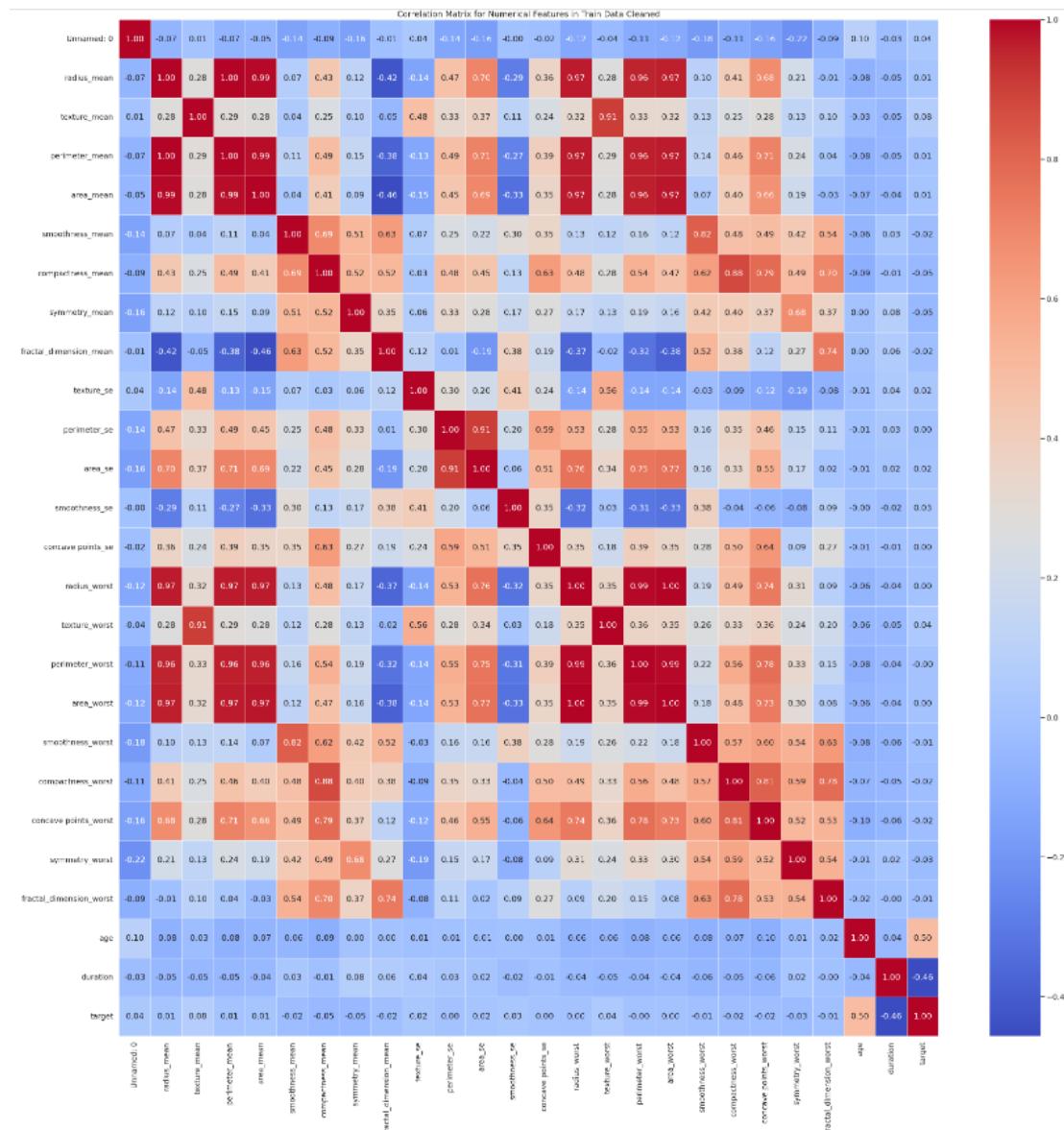
סה"כ הורידנו 9 תבוננות בגל ההטיה הגבואה.

בשלב הבא נעשה בדיקה על סט הנתונים שלנו בעזרת מטריצת קורלציה:

זהו תהליך איטרטיבי שבו אנו מסירים תכונה אחת בכל פעם, מחשבים מחדש את מטריצת הקורלציה, ולאחר מכן מעריכים מחדש אילו תכונות להסיר לאחר מכן. גישה זו יכולה לעזור לטפל בכל שינוי בדינמיקת הקורלציה לאחר הסרת כל תכונה.

המטרה המרכזית בשימוש במטריצת קורלציה בקוד זה היא לבצע בחירת תכונות (Feature Selection) על בסיס קורלציה בין התכונות השונות. בשיש קורלציה גבוהה בין תכונות (כלומר, כאשר שתי תכונות מקיימות קשר חזק אחד עם השנייה), אחת מהן עשויה להיות מיותרת עבור המודל. אם שני משתנים מאוד דומים זה לזה, הם עלולים להשיבא של בעיה של Multicollinearity (רב-קורלינאריות), מה שעלול לגרום ביעילות המודל, להגדיל את המורכבות, ולגרום לאימון יerde.

להלן התוצאה לפני הסרת התכונות:



איור 19 מטריצת קורלציה המקורית

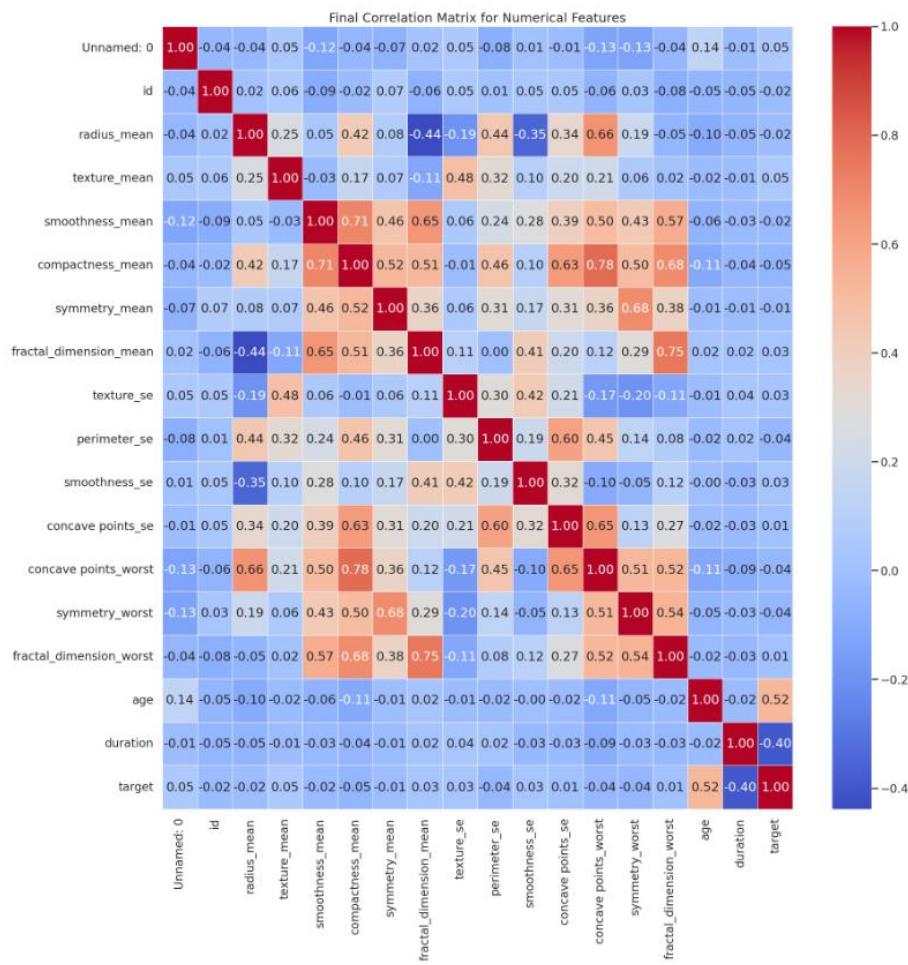
נקבע את סף הקורלציה להיות 0.8.

הפרמטרים שהוסרו בעקבות קורלציה גבוהה:

```
Removed perimeter_mean due to high correlation with radius_mean
Removed area_mean due to high correlation with radius_mean
Removed area_se due to high correlation with perimeter_se
Removed radius_worst due to high correlation with radius_mean
Removed texture_worst due to high correlation with texture_mean
Removed perimeter_worst due to high correlation with radius_mean
Removed area_worst due to high correlation with radius_mean
Removed smoothness_worst due to high correlation with smoothness_mean
```

#### איור 20 הפרמטרים שהוסרו בגל קורלציה גבוהה

וכך נראה מטריצה הקורלציה לאחר ההסרה:



#### איור 21 מטריצת הקורלציה לאחר ההסרה

בשלב הבא ננормל ונתקן את הנתונים בעבר רשת ANN, כי חשוב שהתכונות יהיו בסקללה דומה.

אנו מבצעים טרנספורמציה של משתנים (של מי שיש צורך) וממפיקים קטגוריות מסוימות לתוכאות בינהיות דבר זה נקרא one hot coding והוא מסמן מספר תכונות שאוthon נרצה לשנות לצורה בינהית על מנת שהמודלים יהיו היכי אפקטיבים ולא תתבצע העדפה מסוימת לפי הערכים השונים.

```
def one_hot_encode(X):
    categorical_columns = ['diagnosis', 'stage_of_cancer', 'treatment_administered']
    X = pd.get_dummies(X, columns=categorical_columns, drop_first=True)
    return X
```

#### איור 22 קטע הקוד אשר מבצע one hot coding

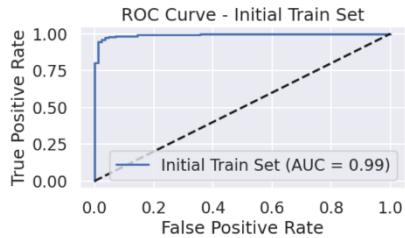
## Logistic Regression Model Building

תחילה, ביצענו סיווג בעזרת מסובג וגרסיה לוגיסטיית מתוך ספריית sklearn (ללא כוונון היפר פרמטרים) ובדקנו את ביצועי המודול על סט האימון.

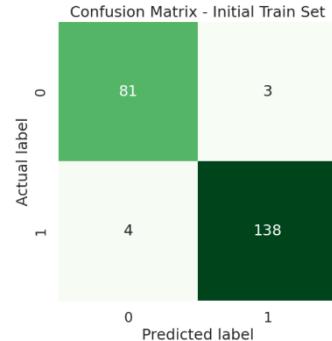
כמו כן ביצענו וגרסיה לוגיסטיית יחד עם Cross-Validation כדי לבדוק שהמודל אמין ומותאם בראוי. המטרה כאן היא להשתמש בגרסיה לוגיסטיית כדי ליצור מודל סיווגBINARI, אבל לבצעAIMOT צולב כדי לבחור את הפרמטרים האופטימליים של המודל ולראות שמודל זה מושך מהתוצאות כמו אימון יתר או תחת-אימון. נעשה זאת בשיטת `k-folds` עם 5 קיפוליים.

ולבסוף ביצענו וגרסיה לוגיסטיית על סטן הבדיקה כדי לבדוק את התוצאות.

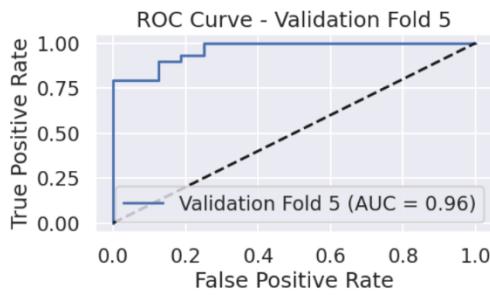
בנוסף, כדי להבין את טיב המודל, כפי שהוא בתאוריה, אנו נפלוט את עקומת ROC ובין את התנהגות המודול עבור סף שונה בין קצב השגיאות להישרדות של חוליה לבין קצב ההצלחות להישרדות של חולב עבור המצביע הרגיל, עבור שיטת `k` folds ועבור סט הבדיקה. כמו כן נפלוט את confusion matrix עבור כל מודל כדי לראות בקרה ויזואלית את מספר ההצלחות אל מול מספר הבשלאות.



איור 23 עקומה ROC במקורה הרגיל



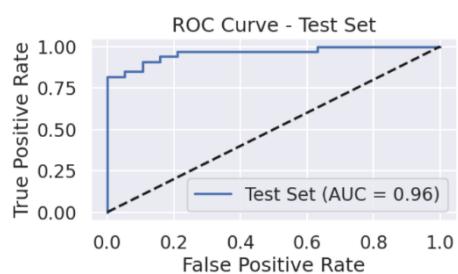
איור 24 confusion matrix במקורה הרגיל



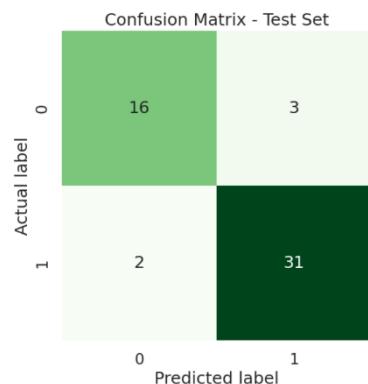
איור 25 עקומה ROC עבור k-folds



איור 26 confusion matrix עבור k-folds



איור 27 עקומה ROC עבור סט המבחן



איור 28 מטריצה confusion עבור סט המבחן

## Predicted label

	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.969027	0.993125	0.971831	0.964286
<b>Logistic Regression (CV)</b>	0.924734	0.981829	0.944721	0.892895
<b>Logistic Regression (Test)</b>	0.903846	0.961722	0.939394	0.842105

איור 29 השוואת ביצועי מודל וගרסיה לוגיסטיות

ניתן לראות ביצועים טובים: המודל מפגין ביצועים חזקים עם דיק ו-AUC ROC גבוהים בכל הסטים.  
אין סימני Overfitting: למחרת ירידת ביצועים קלה ב-CV וב-Test, התוצאות נשארות קרובות וגובהות, מה שמצויב על כך שהמודל לא סובל מאימון יתר חמור.

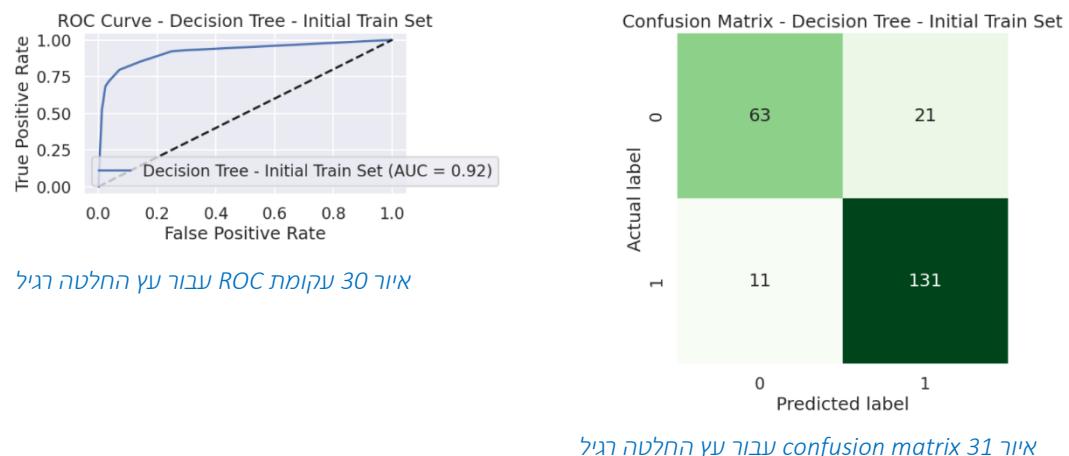
## Decision Tree Model Building

ראשית, נוצר אובייקט מסווג של עץ החלטות ('dt') באמצעות המחלקה 'DecisionTreeClassifier' (dt) באמצעות המחלקה 'DecisionTreeClassifier'. לאחר מכן, המשוג עבר אימון באמצעות נתונים נתוני האימון ('x\_train' ו-'y\_train') באמצעות שיטת 'fit()' שלב זה כולל בניית מודל עץ ההחלטה בהתקساس על התכונות ('x\_train') ותווית היעד המתאימות להן ('y\_train').

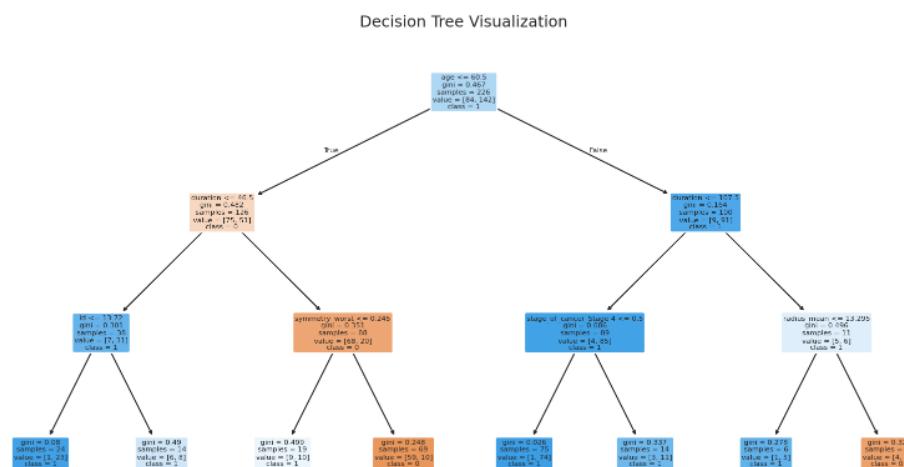
לאחר מכן, נעשה שימוש במסווג כדי לחזות את התגובה עבור מערכת הנתונים של הבדיקה ('x\_test') באמצעות שיטת 'predict()'. שלב זה יוצר תוצאות המבוססות על הדפסים הנלמדים מתוינו האימון. לאחר השגת חיזויים עבור מערכת הנתונים של הבדיקה, מודיע ביצועים כגון accuracy, sensitivity, specificity ROC AUC, sensitivity, specificity. מודיעים אלו מספקים תובנות לגבי ביצועי הסיווג מבוחנת דיקס סיווג ואיזון בין דיקס לדיוק.

העץ מכיל צמתים כאשר לכל צומת יש אופציית פיצול ולבסוף נקבע האם אותה חוליה תשודד או לא. בהתאם לביצוע האימון של עץ החלטה ייחד עם קרוס ולידיציה בשיטת k-folds. כמו שעשינו ברגסיה לוגיסטייה ולאחר האימון ביצע את המודל על סט הבדיקה שלנו לצורך בדיקה עד כמה המודל טוב.

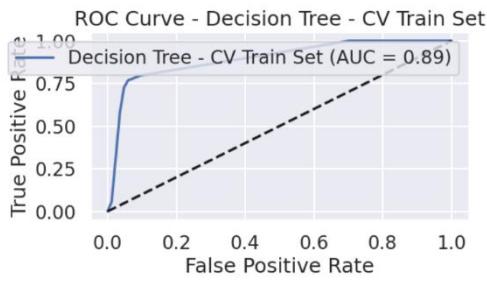
להלן תוצאות גרפי ROC ו-ROC/confusion matrix.



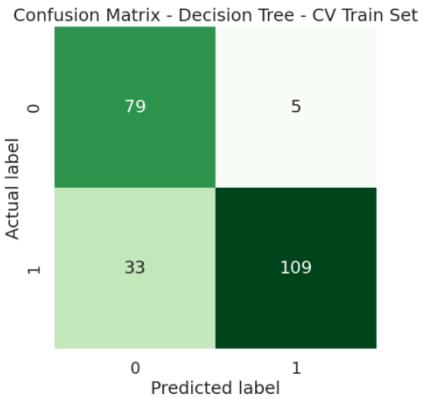
איור 30 עוקמת ROC עבור עץ החלטה רגיל



איור 32 עץ ההחלטה הרגיל

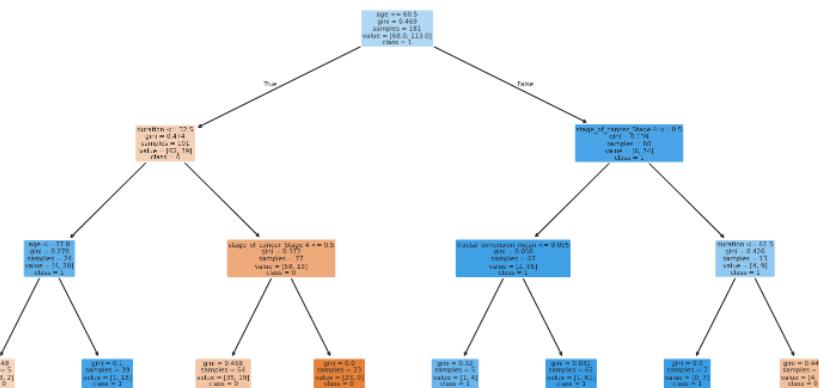


איור 33 עיקומת ROC עבור שיטת  $k$ -folds

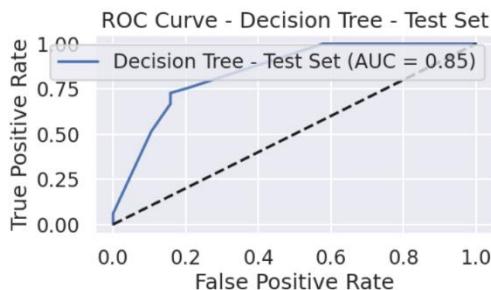


איור 34 עברור שיטת confusion matrix

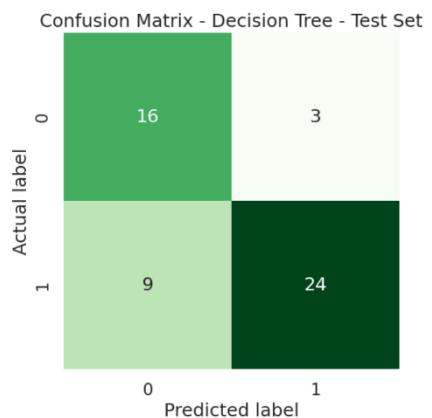
Decision Tree Visualization



איור 35 תוצאות העץ עבור שיטת  $k$ -folds



איור 36 עיקומת ROC עבור סט בדיקה



איור 37 עברור שיטת confusion matrix סט בדיקה

טבלה המסכםת את הנתונים כולל תוצאות הרגסיה הלוגיסטיות באIOR 38.

	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.969027	0.993125	0.971831	0.964286
<b>Logistic Regression (CV)</b>	0.924734	0.981829	0.944721	0.892895
<b>Logistic Regression (Test)</b>	0.903846	0.961722	0.939394	0.842105
<b>Decision Tree</b>	0.858407	0.921320	0.922535	0.750000
<b>Decision Tree (CV)</b>	0.831858	0.894869	0.767606	0.940476
<b>Decision Tree (Test)</b>	0.769231	0.845295	0.727273	0.842105

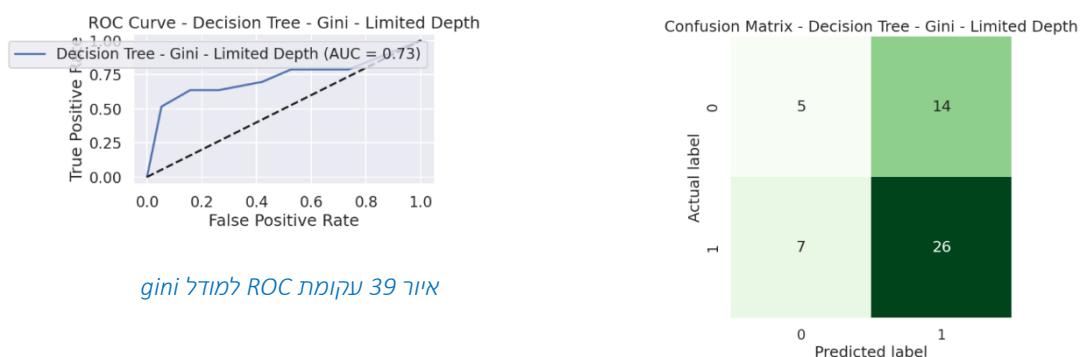
AIOR 38 השוואת עצי החלטה יחד עם דגימה לוגיסטיבית

כמו כן העץ הרגיל הוא ללא הגבלות לצורך דיק אך ניתן לצורך ביוון של היפר פרמטרים לבצע כמה החלטות בחירה:

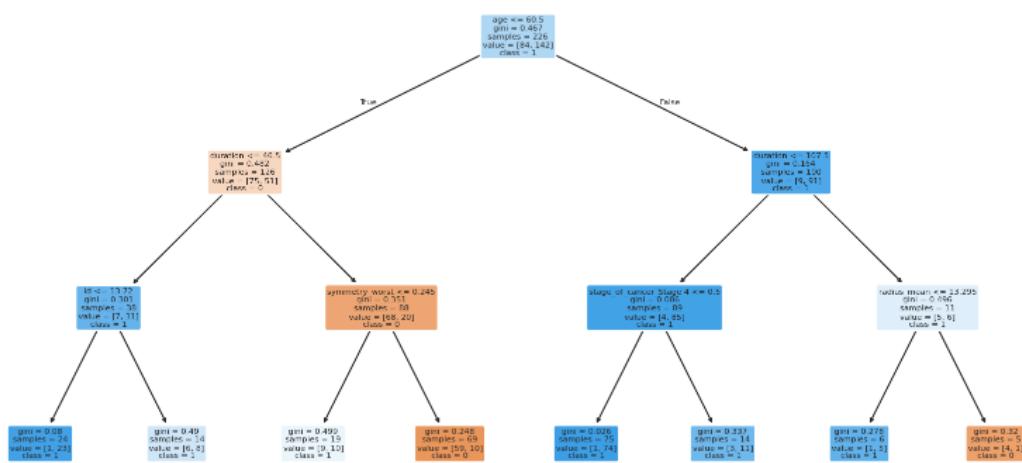
1. Limited depth: היפרפרמטר זה שולט בעומק המקסימלי של עץ ההחלטה. הגדתו כ "None" מאפשרת לעץ להרחיב עצמו. ערכים גבוהים של עומק העץ עשויים לגרום להתקנת יתר, בעוד ערכים נמוכים יכולים להביא לתת-התאמת. נבחר אצלו להיות 3.
2. leaf Min samples: היפרפרמטר זה מקבע את מספר הדוגמאות המינימלי הנדרשות כדי להיות בעלה. הוא מסייע במניעת יצירת צמתים שקולטים רעש בנתחי האימון ובכך שולט על רמת המורכבות של העץ.
3. Criterion: הפרמטר קритריון קובע את הפונקציה המשמשת למידת איכות ההחלטה. ערכים נפוצים לפרמטר זה כוללים `gini` ו- `entropy`. המיצגים את הקритריונים. עבור חישוב ג'יני ואנטרופיה בהתאם. RandomizedSearchCV מעריכה את שני הקритריונים על מנת לקבוע איזה מהם מביא לביצועים טובים יותר עבור המודל של עץ ההחלטה.

על ידי חקירת מרכיבי היפרפרמטרים בקרה יعلاה ובחירה השילוב האופטימלי של היפרפרמטרים, RandomizedSearchCV מסייעת בשיפור ביצועי המודל של עץ ההחלטה על ידי מציאת היפרפרמטרים שמתאימים היבט לנתחים לא מקוריים ומונעת מעבר.

תוצאות של `gini`:



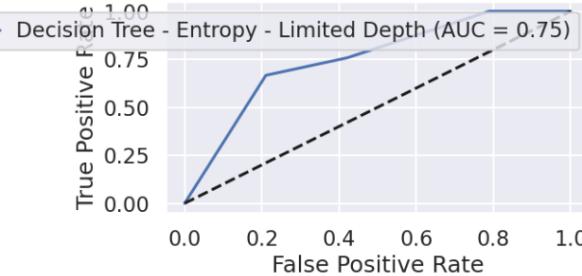
### Decision Tree Visualization



איור 41 תוצאות העז gini

תוצאות של entropy

ROC Curve - Decision Tree - Entropy - Limited Depth



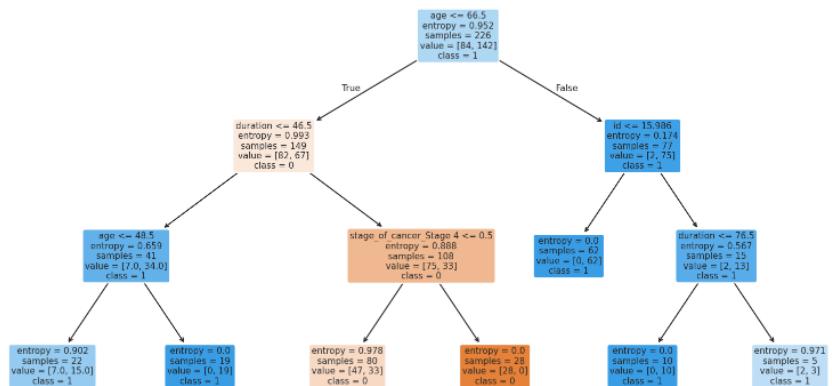
איור 42 עקומה ROC entropy

Confusion Matrix - Decision Tree - Entropy - Limited Depth

		Actual label	
		0	1
Predicted label	0	11	8
	1	8	25

confusion matrix entropy 43 אייר

### Decision Tree Visualization



איור 44 עץ החלטה entropy



טבלה המסכםת את כל התוצאות באIOR: 45

	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.969027	0.993125	0.971831	0.964286
<b>Logistic Regression (CV)</b>	0.924734	0.981829	0.944721	0.892895
<b>Logistic Regression (Test)</b>	0.903846	0.961722	0.939394	0.842105
<b>Decision Tree</b>	0.858407	0.921320	0.922535	0.750000
<b>Decision Tree (CV)</b>	0.831858	0.894869	0.767606	0.940476
<b>Decision Tree (Test)</b>	0.769231	0.845295	0.727273	0.842105
<b>Decision Tree - Gini - Limited Depth</b>	0.596154	0.725678	0.787879	0.263158
<b>Decision Tree - Entropy - Limited Depth</b>	0.692308	0.754386	0.757576	0.578947

IOR 45 השוואת כל עצי החלטה כולל גרסה לוגיסטיות

עץ החלטה בעומק מלא מראה ביצועים טובים יותר באופן כללי, במיוחד בסט האימון ובקרוס-ולידציה, עם איזון טוב בין רגישות לספציפיות.

קרוס-ולידציה (CV) מורידה מעט את הדיקוק אך משפרת את הסpecificity, מה שמראה שהמודל עשוי להיות כללי יותר ולהתמקד במקרים שליליים.

הביצועים על סט המבחן חושפים ירידת בתוצאות, מה שמצוין על כך שהמודל לא מצליח להוביל טוב על נתונים שלא נראו בעבר.

עץ החלטה בעומק מוגבל (ג'יני ונטרופיה) מראים ביצועים פחות טובים, במיוחד בסpecificity. יתרון שמודלים אלו פשוטים מדי כדי ללבוד את המורכבות של הנתונים.

## ANN (רשתות עצביות מלאכותיות)

אנו משתמשים ב (Multilayer Perceptron - MLPClassifier) של ספריית scikit-learn כדי ליצור ולאמן רשת עצבית מלאכותית. תהילך הפעולה של הרשת ב ANN- כולל מספר שלבים:

### 1. ארכיטקטורת הרשת: הרשת שלנו מוגדרת עם פרמטרים מסוימים כמו:

- hidden\_layer\_sizes=100 - המשמעות הוא שיש לנו שכבה נוספת המכילה 100 נוירונים.
- activation='relu' - אנחנו משתמשים בפונקציית אקטיבציה ReLU בשכבות המוסתרות, שעזרה לרשת למודד באופן יעיל יותר על ידי שימירה על שיפוע הגרדיינט.
- max\_iter=1000 - מספר האיטרציות המקסימלי במהלך האימון, כלומר, מספר הפעמים שהרשת תעבור על הנתונים ותעדכן את המשקלים שלה.

### 2. תהילך האימון: (Training Process)

- הרשת מתחילה עם משקלים אקראיים בין הנוירונים, ולאחר מכן, הנתונים מועברים בשכבות הרשת. על סמך הקלטים, הרשת מבצעת תחזית ראשונית (תוצאה) ומשווה אותה לתוצאות האמיתיות.
- פונקציית הפסד (Loss Function) : אנחנו משתמשים ברשת שלנו ב-cross-log-loss (entropy loss) כדי לבדוק את השגיאה בין התוצאות של הרשת לבין התוצאות האמיתיות.
- אופטימיזציה : האלגוריתם משתמש ב-backpropagation-בשילוב עם גרדיינט יורד (Gradient Descent) כדי לעדכן את המשקלים בין הנוירונים במטרה למצער את הפסד לאורך זמן.

### 3. תהילך התוצאות: (Prediction) :

- לאחר סיום תהילך האימון, המודל שלנו מסוגל לייצר תוצאות על סמך נתונים חדשים. בשכבת הפלט, המודל נותן הסתברויות עבור כל קטgorיה תוך שימוש ב-ROC AUC-Sensitivity, Specificity.

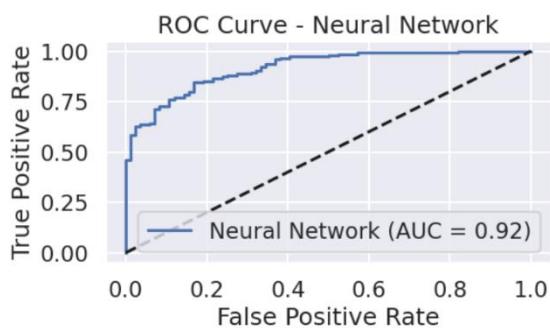
\*הפקודה אשר מאננת את הרשת (איור 46)

```
def train_neural_network(X_train, y_train, hidden_layer_sizes=(100,), activation='relu', max_iter=1000):  
    ann_model = MLPClassifier(hidden_layer_sizes=hidden_layer_sizes, activation=activation, max_iter=max_iter, random_state=42, verbose=True)  
    ann_model.fit(X_train, y_train)  
    return ann_model
```

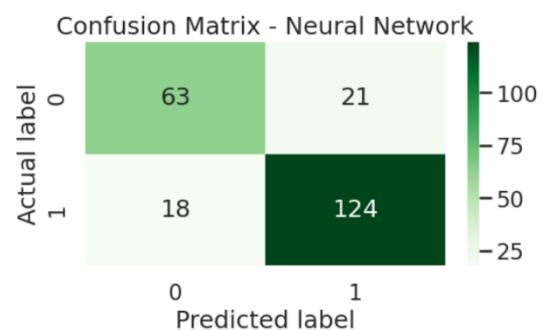
איור 46 הפקודה אשר מאננת את הרשת

נזכיר כי גם במודל זהה השתמשנו אימנו את המערכת, עשינו אימון קROSS- ולייזיציה בשיטת ה- k-folds כמו קודם, ובסיוף בדקנו את המודל על סט הבדיקה. להלן התוצאות:

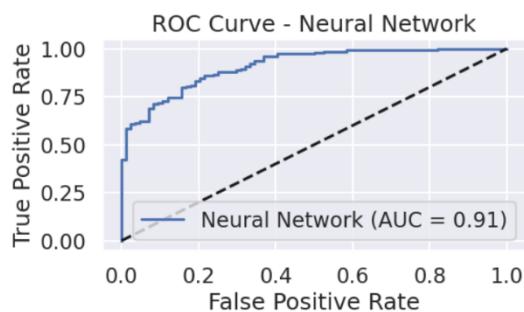
להלן התוצאות לכל ניסוי - עקומות ROC וconfusion matrix



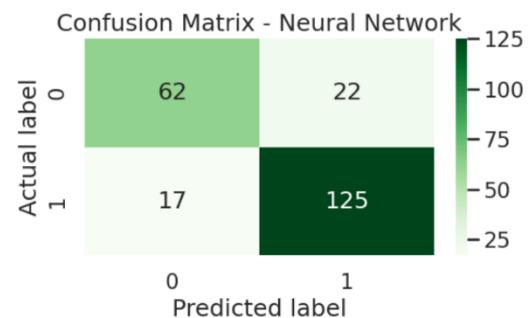
איור 47 עקומת ROC עבור ANN במקורה המקורי



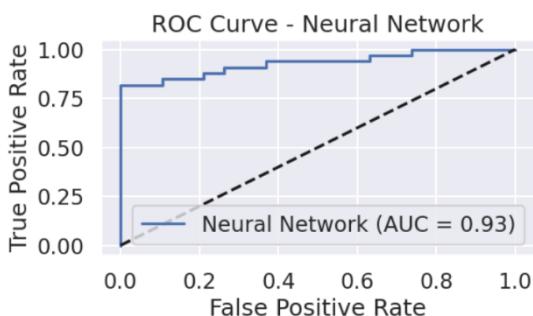
איור 48 confusion matrix עבור ANN במקורה המקורי



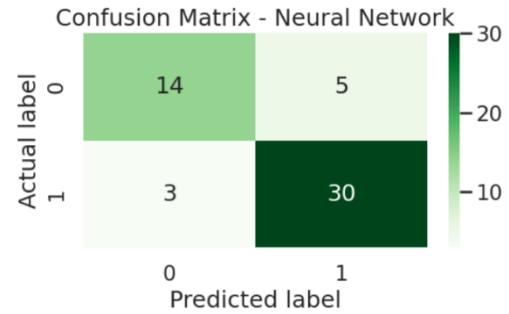
איור 49 עקומת ROC עבור ANN עם k-folds CV בשיטת הבדיקה



איור 50 confusion matrix עבור ANN עם k-folds CV בשיטת הבדיקה



איור 51 עקומת ROC עבור ANN עם סט הבדיקה



איור 52 confusion matrix עבור ANN עם סט הבדיקה

	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.969027	0.993125	0.971831	0.964286
<b>Logistic Regression (CV)</b>	0.924734	0.981829	0.944721	0.892895
<b>Logistic Regression (Test)</b>	0.903846	0.961722	0.939394	0.842105
<b>Decision Tree</b>	0.858407	0.921320	0.922535	0.750000
<b>Decision Tree (CV)</b>	0.831858	0.894869	0.767606	0.940476
<b>Decision Tree (Test)</b>	0.769231	0.845295	0.727273	0.842105
<b>Decision Tree - Gini - Limited Depth</b>	0.596154	0.725678	0.787879	0.263158
<b>Decision Tree - Entropy - Limited Depth</b>	0.692308	0.754386	0.757576	0.578947
<b>ANN</b>	0.827434	0.917170	0.873239	0.750000
<b>ANN (CV)</b>	0.827434	0.912140	0.880282	0.738095
<b>ANN (Test)</b>	0.846154	0.929825	0.909091	0.736842

איור 53 השוואת ANN עם עצי החלטה ורגסיה לוגיסטי

ביצועים עקביים: הביצועים בין ה-ANN הרגיל, קורס-וילדייה וסת המבחן הם יחסית עקביים, מה שמעיד על בר שאון בעיה משמעותית של אימון יתר (overfitting). המודלים מצליחים להתאים את עצם היפט לסת המבחן ולא רק לסת האימון.

גישות גבואה: בכל המודלים, הריגשות (יכולת זהה מקרים חיוביים) היא גבואה וחסית, בעיקר במודל שנבדק על סט המבחן (90.91%). זהו מدد חשוב מאוד כאשר עוסקים בעיות כמו גילוי מחלות.

ROC AUC טוב מאוד: כל המודלים מראים ערכיהם גבוהים של ROC AUC, כאשר המודל על סט המבחן מושג את התוצאה הטובה ביותר (0.929), מה שמעיד על יכולת המודל להבחין בין מקרים חיוביים לשיליים.

## סיכום ומסקנות:

**סיכום:**

בפרויקט זה נבדקה יכולת של שלושה מסוגים שונים עבור הסיכוי שחולת סרטן השד תשרוד את המחלת:

- Logistic Regression
- Decision Tree
- ANN

כאשר בtarget, 1 - תשרוד את המחלת, 0 - לא תשרוד את המחלת.

בשלב הראשון של הפרויקט ביצענו עיבוד וטיפול בננתונים כך שנוכל להשתמש בהם למטרות אנליהזה. שבוד הננתונים כולל בדיקה ונקיוי לפי שלבים שונים כך שכל הערכים לבסוף הם מספריים ואין מקום שאין בו ערך כלשהו. בנוסף ביצענו דיווק outliers וביצוע נרמול ופיזול מערך הננתונים לסט אימון וסט בדיקה. לאחר מכן, בשלב בניסוי, ביצענו ניתוחים על כל אחד מהמסוגים בצורה וgilah, עם קרוס ולידציה בשיטת k-folds וניסוי על סט בדיקה.

בחנו את העבודה של המודלים וביצענו השוואה בין ביצועים לפי מדדים שונים:

- Accuracy
- ROC AUC
- Sensitivity
- Specificity

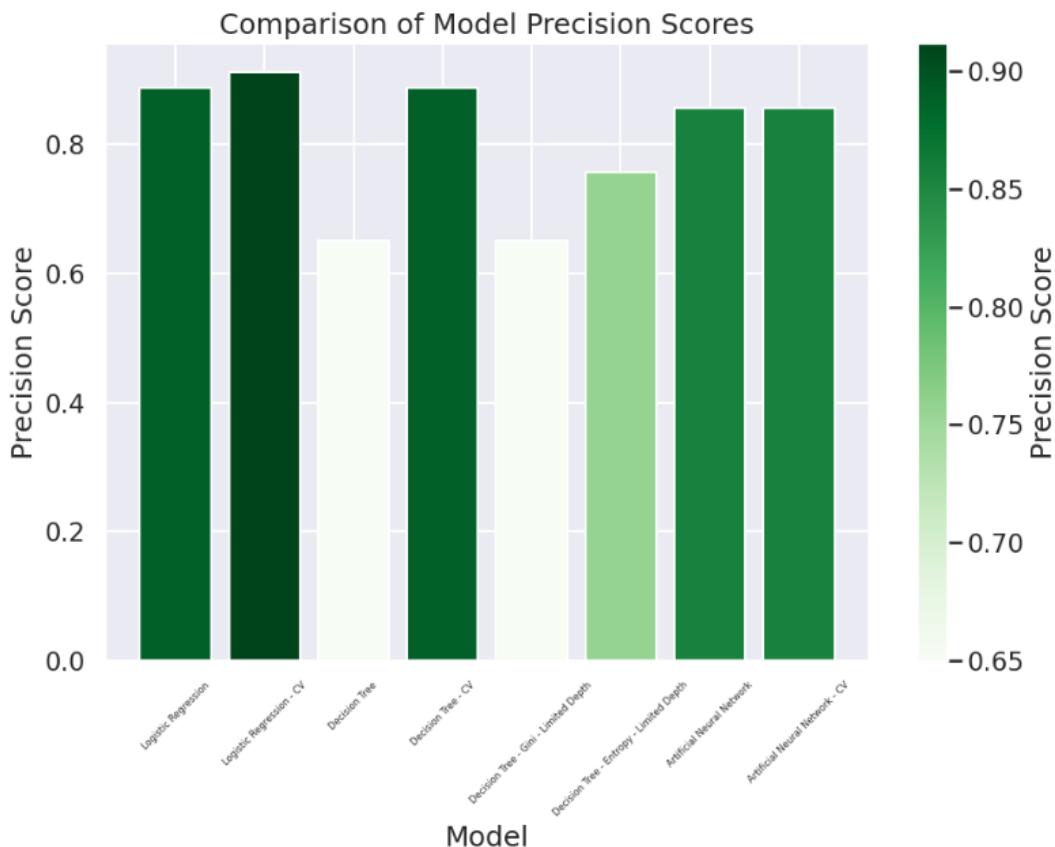
את התוצאות הטובות ביותר קיבלנו עבור מסוג של **Logistic Regression**, המודל מספק את הביצועים הטובים ביותר בכל המדדים, עם דיווק ROC AUC גבוה מאוד, גם באימון וגם בקרוס-ולידציה. על סט המבחן הביצועים קצת ירודים, אך עדין הם טובים מאוד.

תוצאות מסכימות מבחינת המדדים הנבדקים:

	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.969027	0.993125	0.971831	0.964286
<b>Logistic Regression (CV)</b>	0.924734	0.981829	0.944721	0.892895
<b>Logistic Regression (Test)</b>	0.903846	0.961722	0.939394	0.842105
<b>Decision Tree</b>	0.858407	0.921320	0.922535	0.750000
<b>Decision Tree (CV)</b>	0.831858	0.894869	0.767606	0.940476
<b>Decision Tree (Test)</b>	0.769231	0.845295	0.727273	0.842105
<b>Decision Tree - Gini - Limited Depth</b>	0.596154	0.725678	0.787879	0.263158
<b>Decision Tree - Entropy - Limited Depth</b>	0.692308	0.754386	0.757576	0.578947
<b>ANN</b>	0.827434	0.917170	0.873239	0.750000
<b>ANN (CV)</b>	0.827434	0.912140	0.880282	0.738095
<b>ANN (Test)</b>	0.846154	0.929825	0.909091	0.736842

### אior 54 תוצאות כלל המודלים

נסיף ונאמר כי מההבנה שלנו על כלל המודלים כנראה שעבור גרסיה לוגיסטיבית היה לנו קצת אימון יתר כי אנחנו ציפינו שמודל ANN ישפוך את התוצאות הביולוגיות במבנה שלו, בغال סט הננתונים המורכב שלנו, ובגלו האופן שהוא פועל.



איור 55 precision עבור כל המודלים בהם מתרבעת השוואה

Precision מודד את המספר הכללי של ההצלחות של המודל באשר הוא סיווג חובי אכן זה חובי חלק המשOPERNUMר הכללי של חוביות חזיות ע"י המודל. לכן, להגיד מהו המודל הטוב ביותר זו שאלת שאין עליה תשובה חד משמעית, תלוי מה מחפשים. הדיאגרמה באירור מס' 55 יכולה לתת אינדיקציה לטיב המודל עבור הישרדות של חולה.

#### מסקנות:

1. עיבוד מקדים של הנתונים הוא שלב חוני שמשמעותו לשפר את הביצועים והאמינות של המודלים החזויים, ובכך תורם להצלחת הפרויקט כולו.
2. חשיבות בחירת תכונות: השימוש במטריצת הקורלציה ובחירה תכונות מהוות גישה טובה להפחחת מושתנים שאינם רלוונטיים ולמניעת בעיות של אימון יתר. תהליך זה עוזר לשפר את הביצועים הכלליים של המודלים.
3. באופן כללי, השימוש ב-k-Folds-Kruschke הוא כלי עצמאי שיכול לשפר את איכות המודלים והבנתם, ולסייע במניעת בעיות פוטנציאליות בעת ביצוע חזיות על נתונים חדשים ואימון יתר.
4. הערכת נתונים: התוצאות מצביעות על כך שבחירה נכונה של מדדי ביצוע (בגון ROC, AUC, דיקן, רגישות וספציפיות) חיונית להבנת תפקוד המודלים, במיוחד כאשר הנתונים אינם מאודנים.

~ 40 ~

# Assignment No. 2

Lecturer: Professor Yitzhak Lapidot

Gal Neumann - 316120260 & Nizan Ismailov - 208412569

## A Comparison of Artificial Neural Network and Decision Trees with Logistic Regression as Classification Models for Breast Cancer Survival

The data set was import from: NIH - National Cancer Institute (SEER)

### Imports

```
In [3]: from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import seaborn as sns
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_predict, StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, auc, f1_score, precision_score, recall_score, accuracy_score
from sklearn.tree import export_text
import pandas as pd
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
from sklearn.tree import plot_tree
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold, train_test_split, cross_validate, cross_val_score
from scipy.stats import randint
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import f1_score, precision_score, recall_score
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import numpy as np
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score, precision_score, recall_score, accuracy_score, recall_weighted
from IPython.display import display
```

### Load the data

```
In [4]: data = pd.read_csv('data.csv')
print(data.head())
```

```
diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0          M           17.99        10.38         122.80      1001.0
1          M           20.57        17.77         132.90      1326.0
2          M           19.69        21.25         130.00      1203.0
3          M           11.42        20.38         77.58       386.1
4          M           20.29        14.34         135.10      1297.0

smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0            0.11840          0.27760          0.3001          0.14710
1            0.08474          0.07864          0.0869          0.07017
2            0.10960          0.15990          0.1974          0.12790
3            0.14250          0.28390          0.2414          0.10520
4            0.10030          0.13280          0.1980          0.10430

symmetry_mean  ...  concavity_worst  concave points_worst  symmetry_worst  \
0            0.2419    ...          0.7119          0.2654          0.4601
1            0.1812    ...          0.2416          0.1860          0.2750
2            0.2069    ...          0.4504          0.2430          0.3613
3            0.2597    ...          0.6869          0.2575          0.6638
4            0.1809    ...          0.4000          0.1625          0.2364

fractal_dimension_worst  age  stage_of_cancer  treatment_administered  \
0            0.11890    74      Stage 1             Radiation
1            0.08902    77      Stage 1             Surgery
2            0.08758    83      Stage 4             Hormonal
3            0.17300    30      Stage 3             Radiation
4            0.07678    33      Stage 2             Hormonal

duration  censor  survival_probability
0          20      1            0.829205
1          67      0            0.780743
2          93      0            0.507499
3          38      1            0.480011
4          62      1            0.485004
```

[5 rows x 37 columns]

## Data PreProcessing

### Taking care of missing data

```
In [5]: data.replace('?', pd.NA, inplace=True)
data.dropna(inplace=True)
data.shape
data.describe()
```

Out[5]: (569, 37)

Out[5]:

	<b>radius_mean</b>	<b>texture_mean</b>	<b>perimeter_mean</b>	<b>area_mean</b>	<b>smoothness_mean</b>	<b>compactness_mean</b>
<b>count</b>	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
<b>mean</b>	14.127292	19.289649	91.969033	654.889104	0.096360	0.104324
<b>std</b>	3.524049	4.301036	24.298981	351.914129	0.014064	0.052871
<b>min</b>	6.981000	9.710000	43.790000	143.500000	0.052630	0.019384
<b>25%</b>	11.700000	16.170000	75.170000	420.300000	0.086370	0.064923
<b>50%</b>	13.370000	18.840000	86.240000	551.100000	0.095870	0.092638
<b>75%</b>	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400
<b>max</b>	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400

8 rows × 34 columns

As we can see there are no missing values, "SEER" datasets should be clean and without missing values.

Change the target (Survival\_probability) to binary terms and remove "survival probability" and "censor" from data

According to the article, the goal is to predict whether a patient survived or not.

"1" ≡ Survived , "0" ≡ Didn't Survived

```
In [6]: data['target'] = (data['survival_probability'] > 0.5).astype(int)
Binary_data = data.copy()

data = data.drop('survival_probability', axis=1)
data = data.drop('censor', axis=1)
data.to_csv('data_01.csv', index=False)

data.head()
```

Out[6]:

<b>diagnosis</b>	<b>radius_mean</b>	<b>texture_mean</b>	<b>perimeter_mean</b>	<b>area_mean</b>	<b>smoothness_mean</b>	<b>compactness_mean</b>
<b>0</b>	M	17.99	10.38	122.80	1001.0	0.11840
<b>1</b>	M	20.57	17.77	132.90	1326.0	0.08474
<b>2</b>	M	19.69	21.25	130.00	1203.0	0.10960
<b>3</b>	M	11.42	20.38	77.58	386.1	0.14250
<b>4</b>	M	20.29	14.34	135.10	1297.0	0.10030

5 rows  $\times$  36 columns

## Now we will split the data into 80% train and 20% test.

We made sure to preserve target variable distribution using Stratification Parameter.

```
In [7]: X = data.drop(columns=['target'])
y = data['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y,
train_data = pd.concat([X_train, y_train], axis=1)
test_data = pd.concat([X_test, y_test], axis=1)

print(f"Stratification of train set: {y_train.mean()}")
print(f"Stratification of test set: {y_test.mean()}")
```

Stratification of train set: 0.6659340659340659

Stratification of test set: 0.6666666666666666

Exports the sets

```
In [8]: train_data.to_csv('train_data.csv')
test_data.to_csv('test_data.csv')
```

## Adjust the data into Categorical and Numerical datatypes

```
In [9]: objectAttributesKey = ["diagnosis", "stage_of_cancer", "treatment_administered"]
integerAttributesKey = ["age"]

data[objectAttributesKey] = data[objectAttributesKey].astype('category')
data[integerAttributesKey] = data[integerAttributesKey].astype(int)

data = pd.get_dummies(data, columns=objectAttributesKey, drop_first=True)

data.to_csv('train_data_01.csv', index=False)
```

```
In [10]: data.head()
```

```
Out[10]:   radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  compactness_mean  concavity_mean  concave_points_mean  symmetry_mean  fractal_dimension_mean  radius_se  texture_se  perimeter_se  area_se  smoothness_se  compactness_se  concavity_se  concave_points_se  symmetry_se  fractal_dimension_se  radius_worst  texture_worst  perimeter_worst  area_worst  smoothness_worst  compactness_worst  concavity_worst  concave_points_worst  symmetry_worst  fractal_dimension_worst
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_points_mean	symmetry_mean	fractal_dimension_mean	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave_points_se	symmetry_se	fractal_dimension_se	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave_points_worst	symmetry_worst	fractal_dimension_worst
<b>0</b>	17.99	10.38	122.80	1001.0	0.11840	0.27760																								
<b>1</b>	20.57	17.77	132.90	1326.0	0.08474	0.07864																								
<b>2</b>	19.69	21.25	130.00	1203.0	0.10960	0.15990																								
<b>3</b>	11.42	20.38	77.58	386.1	0.14250	0.28390																								
<b>4</b>	20.29	14.34	135.10	1297.0	0.10030	0.13280																								

5 rows × 41 columns

We will check for the distribution of 'target'

```
In [11]: train_data['target'].value_counts()
```

Out[11]: count

target	count
1	303
0	152

dtype: int64

## Identify outliers

we'll calculate the 1.5 times the Interquartile Range (IQR) and visualize them using boxplots.

```
In [12]: train_data['target'] = train_data['target'].map({1: 'Survived', 0: "Didn't Survived"})

numerical_columns = train_data.select_dtypes(include=['int64', 'float64']).columns
numerical_columns = [col for col in numerical_columns if col != 'target']

sns.set(rc={'figure.figsize': (6, 3)})
sns.set_context("talk")

for column in numerical_columns:
    plt.figure(figsize=(6, 3))
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")

    plt.title(f'Boxplot of {column} vs Target', fontsize=16)
    plt.xlabel('target', fontsize=14)
    plt.ylabel(column, fontsize=14)

    plt.show()
```

Out[12]: <Figure size 600x300 with 0 Axes>

<ipython-input-12-c8da27f086f5>:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

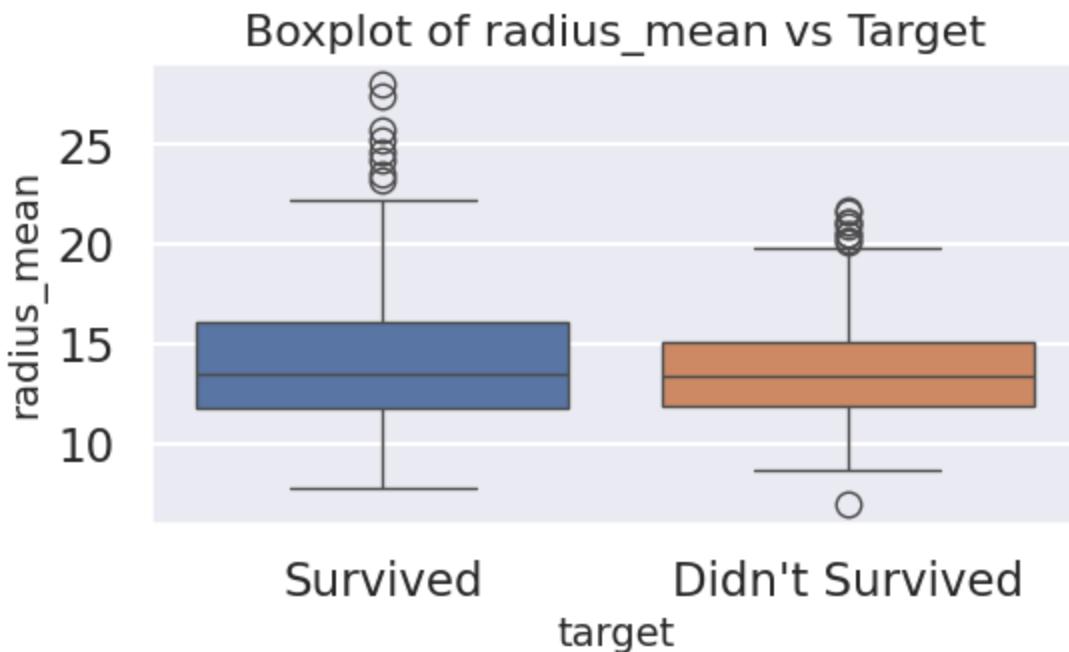
sns.boxplot(x='target', y=column, data=train\_data, orient="v", palette="deep")

Out[12]: <Axes: xlabel='target', ylabel='radius\_mean'>

Out[12]: Text(0.5, 1.0, 'Boxplot of radius\_mean vs Target')

Out[12]: Text(0.5, 0, 'target')

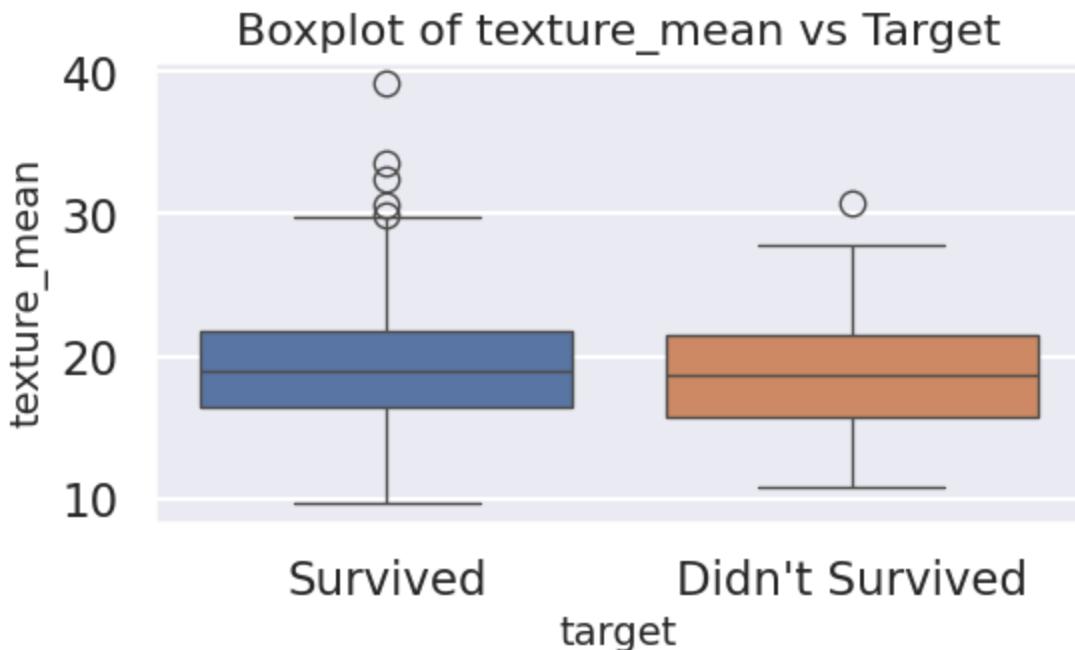
Out[12]: Text(0, 0.5, 'radius\_mean')



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

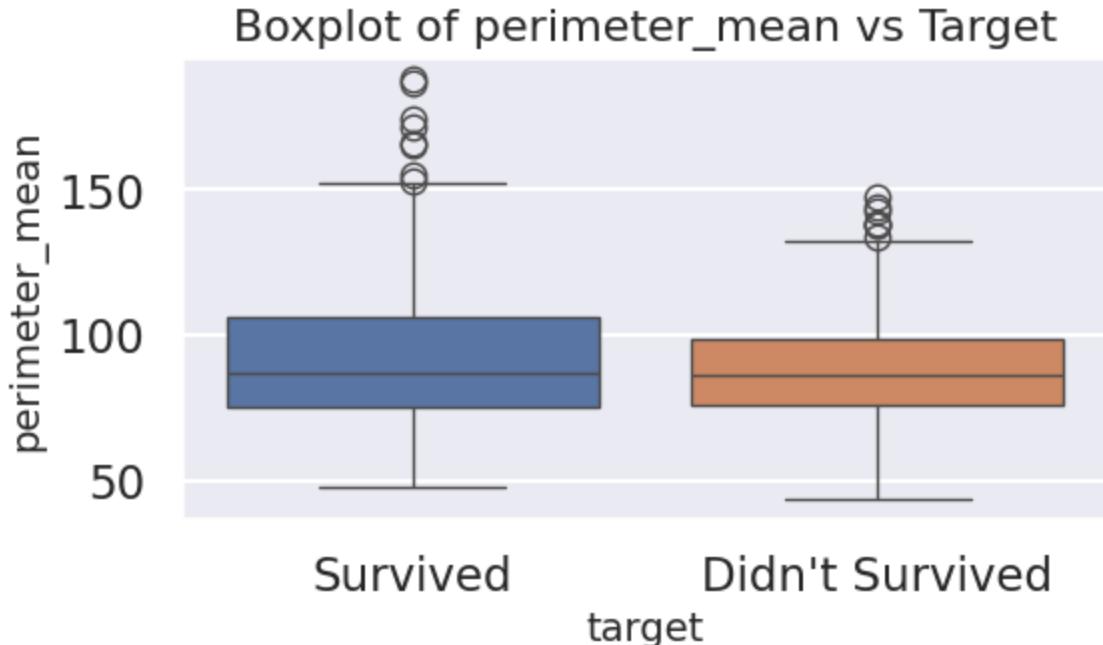
```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
<Axes: xlabel='target', ylabel='texture_mean'>  
Text(0.5, 1.0, 'Boxplot of texture_mean vs Target')  
Text(0.5, 0, 'target')  
Text(0, 0.5, 'texture_mean')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

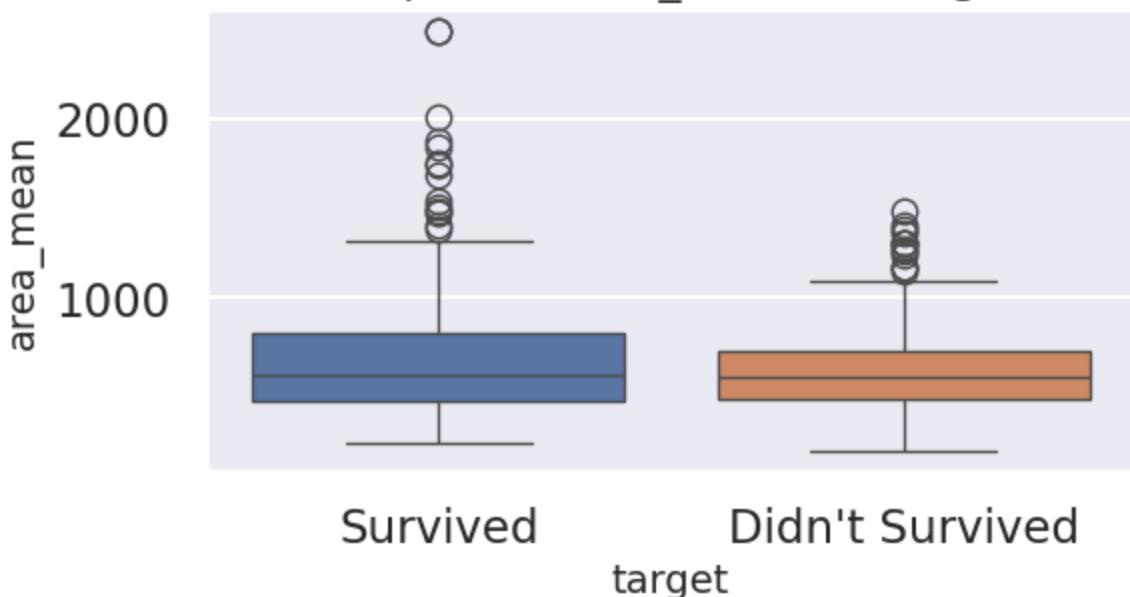
```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='perimeter_mean'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of perimeter_mean vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'perimeter_mean')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='area_mean'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of area_mean vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'area_mean')
```

### Boxplot of area\_mean vs Target

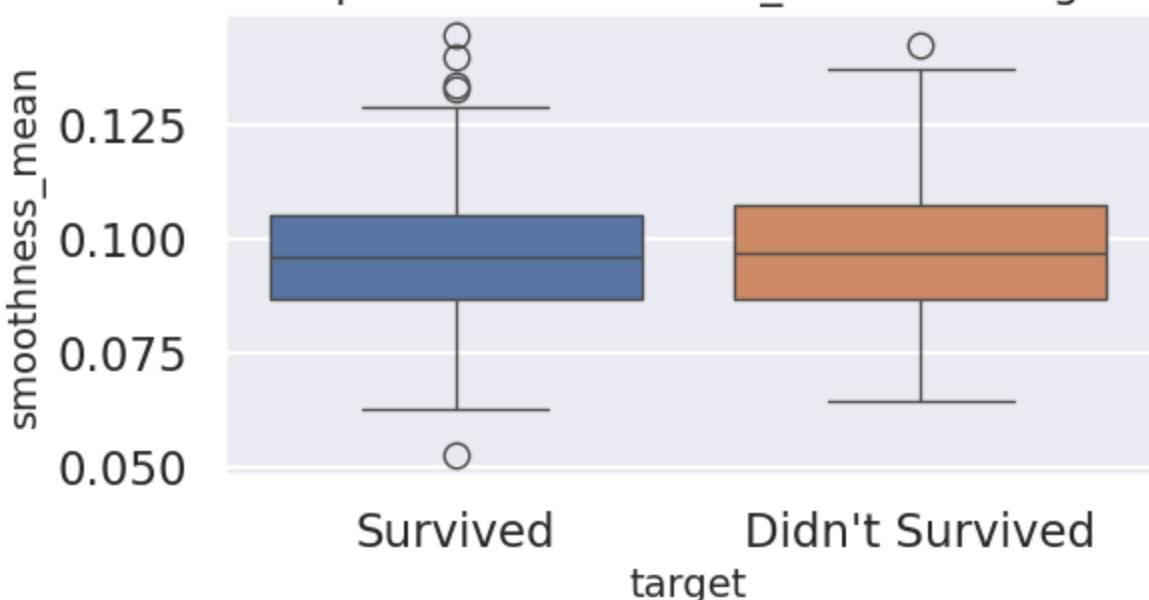


```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

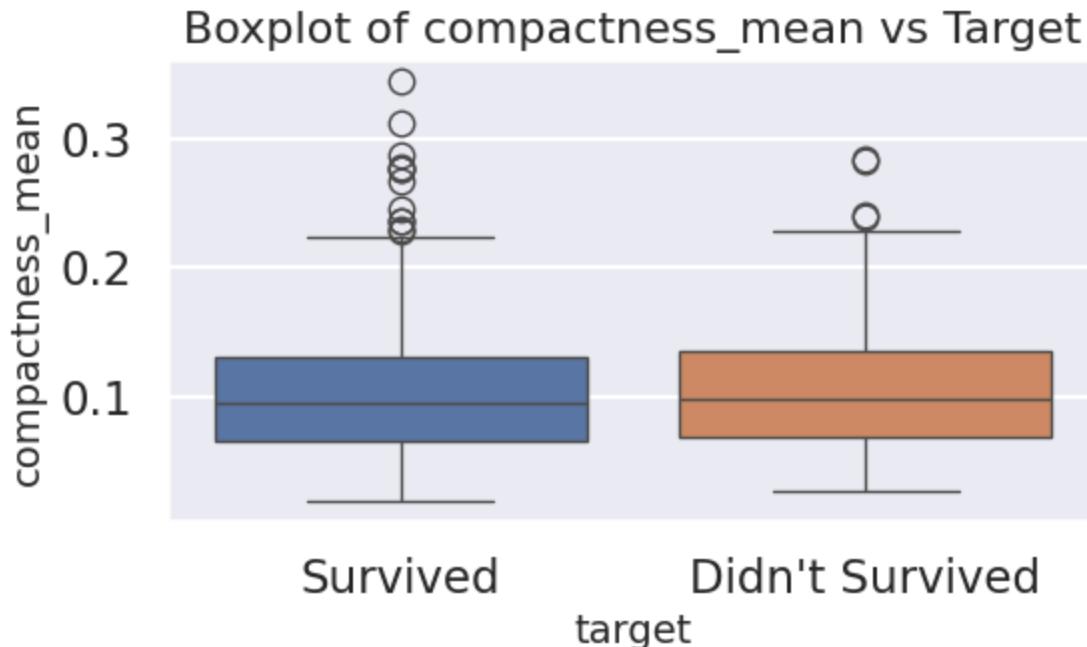
```
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
<Axes: xlabel='target', ylabel='smoothness_mean'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of smoothness_mean vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'smoothness_mean')
```

### Boxplot of smoothness\_mean vs Target

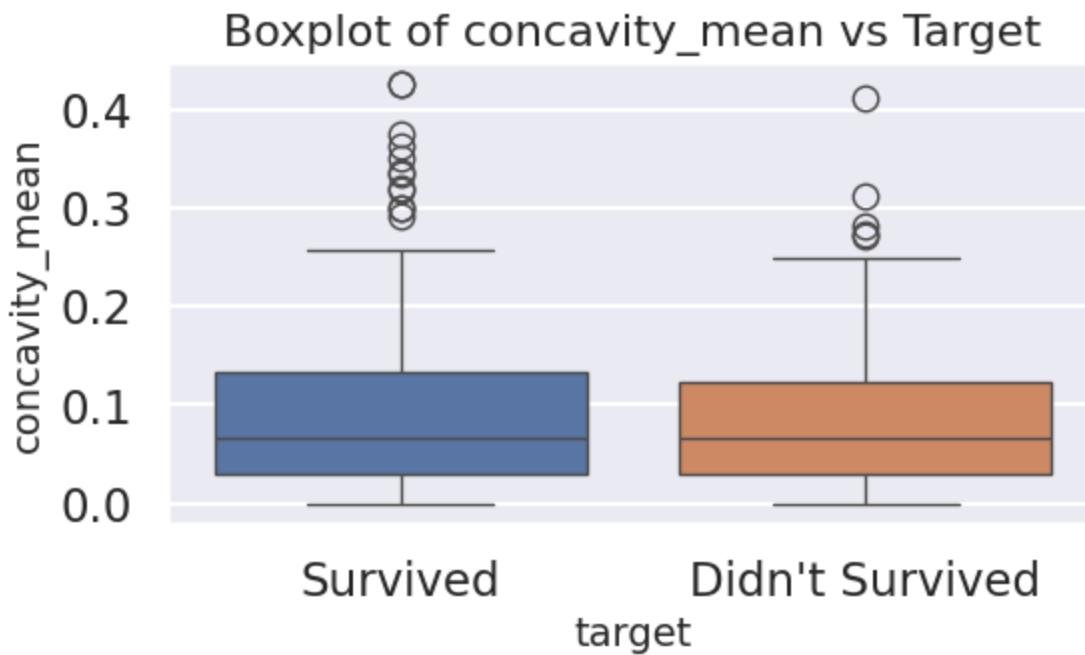


```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='compactness_mean'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of compactness_mean vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'compactness_mean')
```



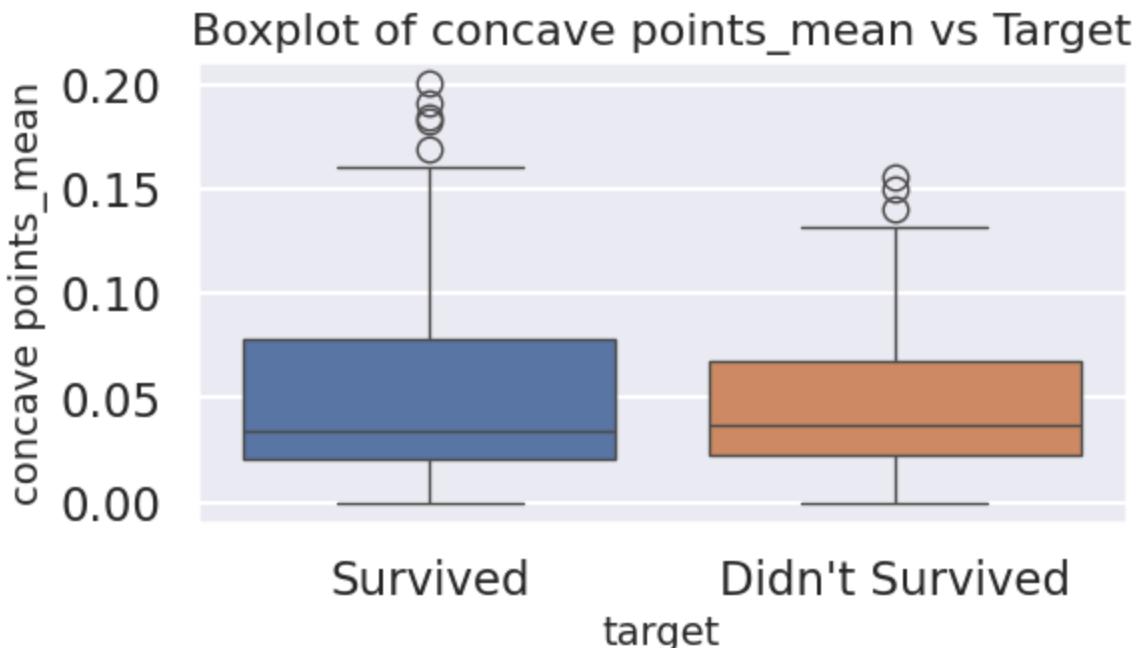
```
Out[12]: <Figure size 600x300 with 0 Axes>  
  
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='concavity_mean'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of concavity_mean vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'concavity_mean')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

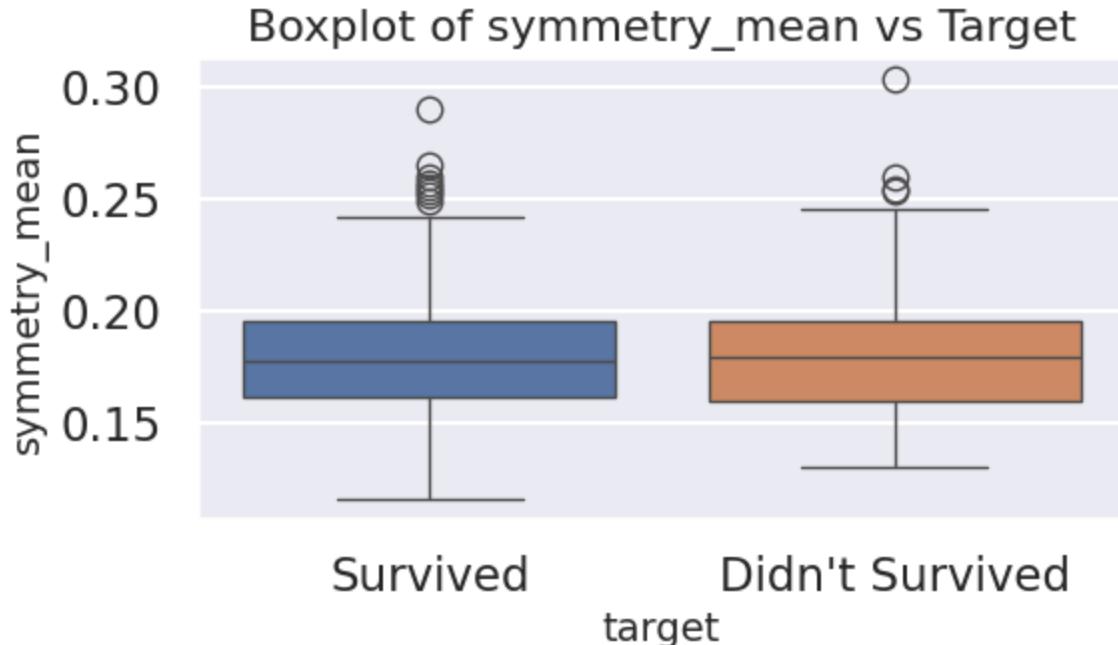
```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
<Axes: xlabel='target', ylabel='concave points_mean'>  
Text(0.5, 1.0, 'Boxplot of concave points_mean vs Target')  
Text(0.5, 0, 'target')  
Text(0, 0.5, 'concave points_mean')
```



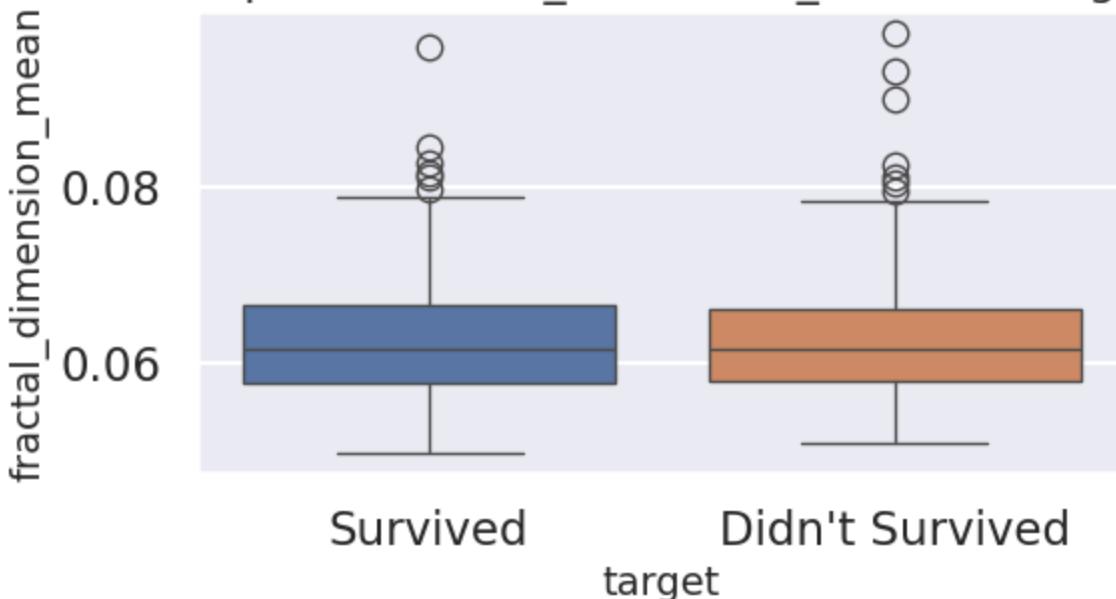
```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='symmetry_mean'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of symmetry_mean vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'symmetry_mean')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>  
  
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='fractal_dimension_mean'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of fractal_dimension_mean vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'fractal_dimension_mean')
```

### Boxplot of fractal\_dimension\_mean vs Target



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")
```

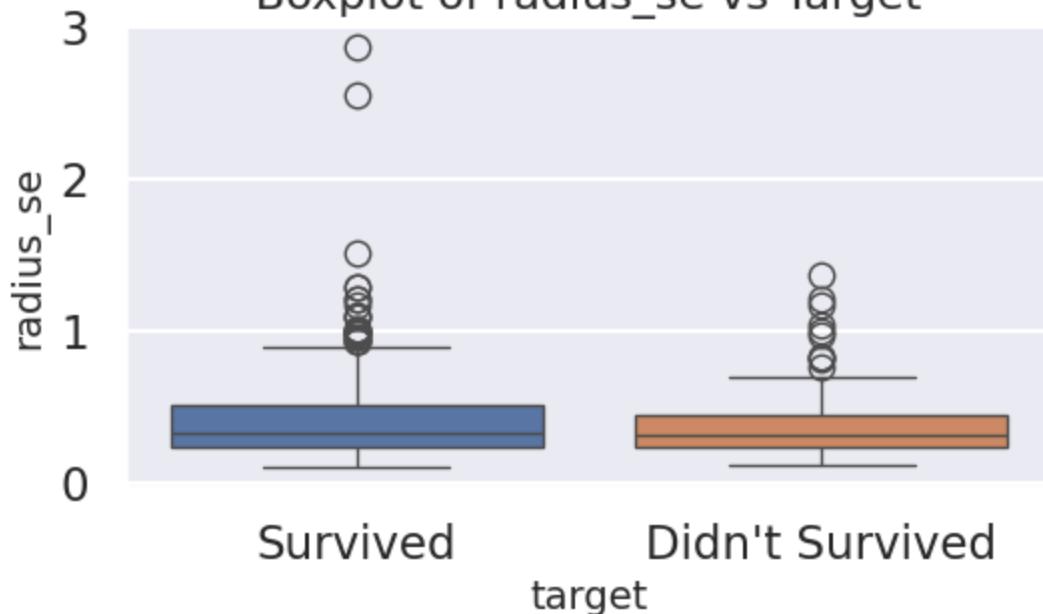
```
Out[12]: <Axes: xlabel='target', ylabel='radius_se'>
```

```
Out[12]: Text(0.5, 1.0, 'Boxplot of radius_se vs Target')
```

```
Out[12]: Text(0.5, 0, 'target')
```

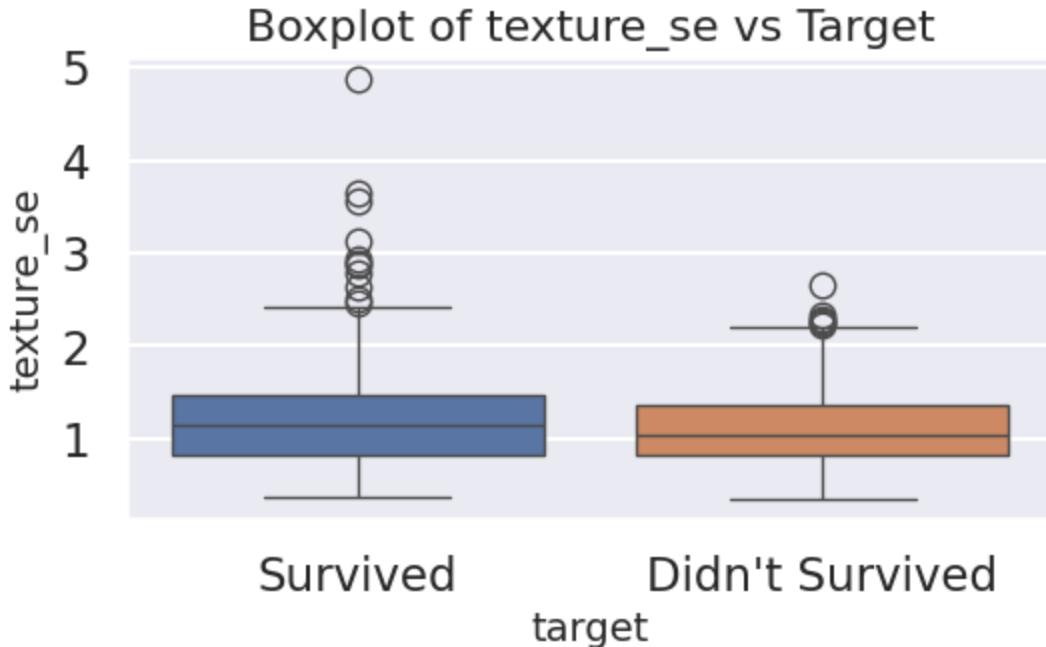
```
Out[12]: Text(0, 0.5, 'radius_se')
```

### Boxplot of radius\_se vs Target



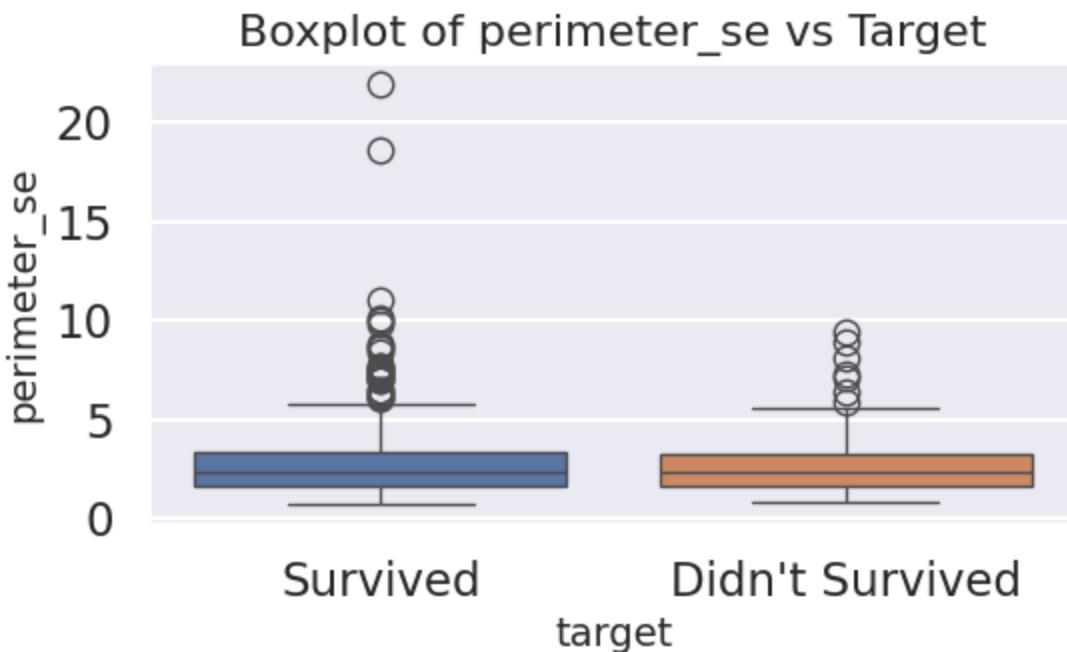
```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
<Axes: xlabel='target', ylabel='texture_se'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of texture_se vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'texture_se')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
<Axes: xlabel='target', ylabel='perimeter_se'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of perimeter_se vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'perimeter_se')
```



Out[12]: <Figure size 600x300 with 0 Axes>

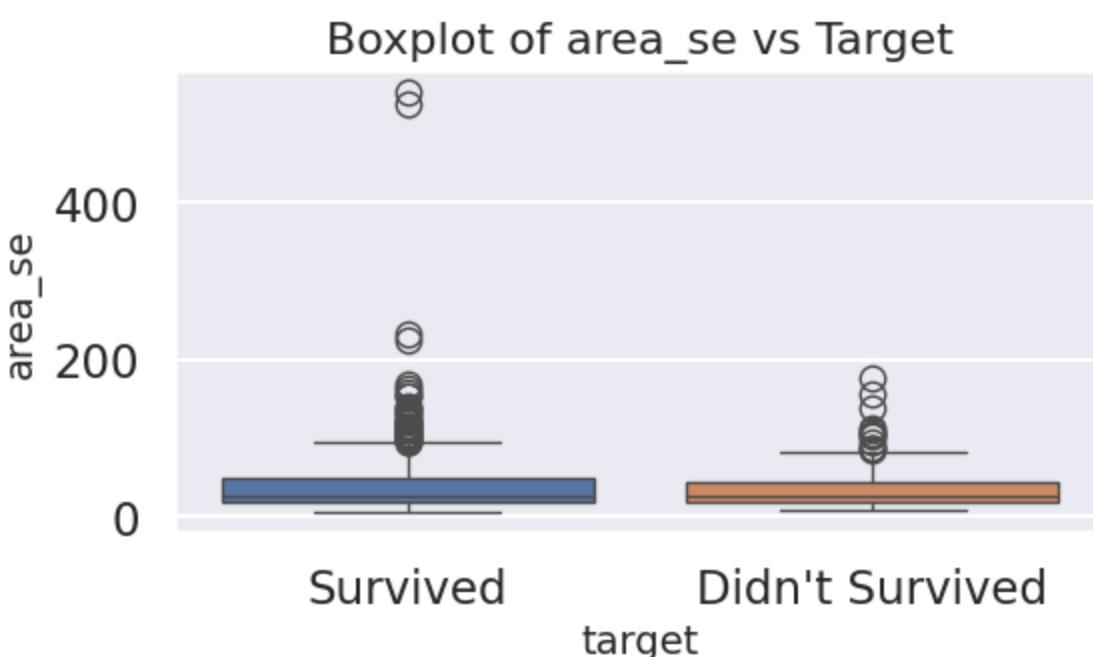
```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.boxplot(x='target', y=column, data=area_se)
Out[12]: <Axes: xlabel='target', ylabel='area_se'
```

```
Out[12]: Text(0.5, 1.0, 'Boxplot of area se vs Target')
```

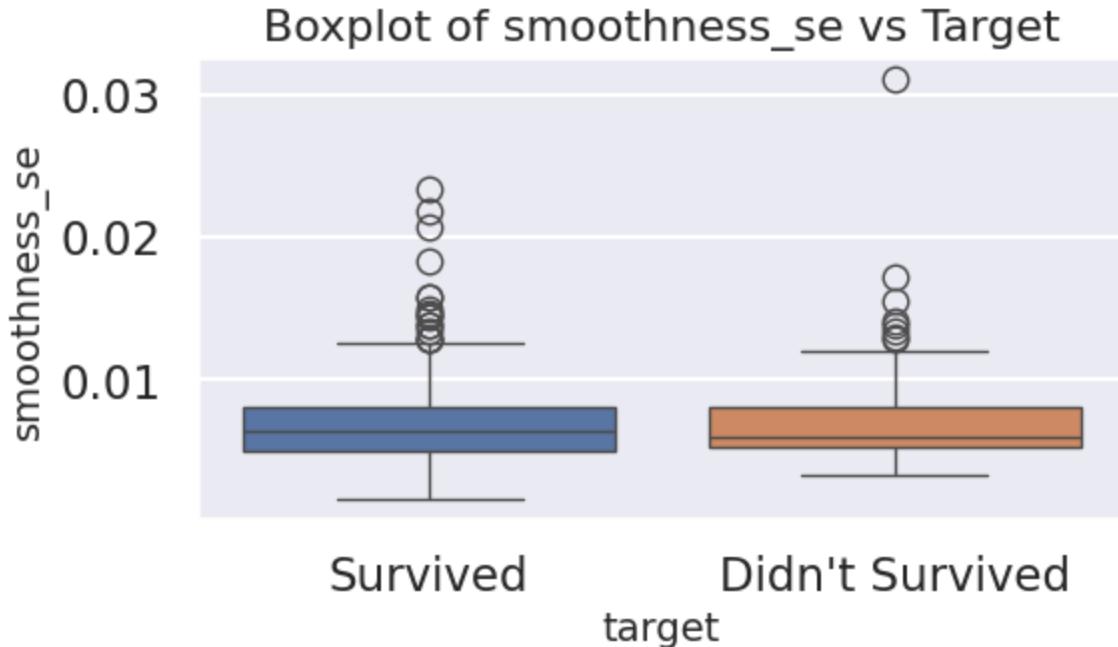
```
Out[12]: Text(0.5, 0, 'target')
```

```
Text(0, 0.5, 'area se')
```



Out[12]: <Figure size 600x300 with 0 Axes>

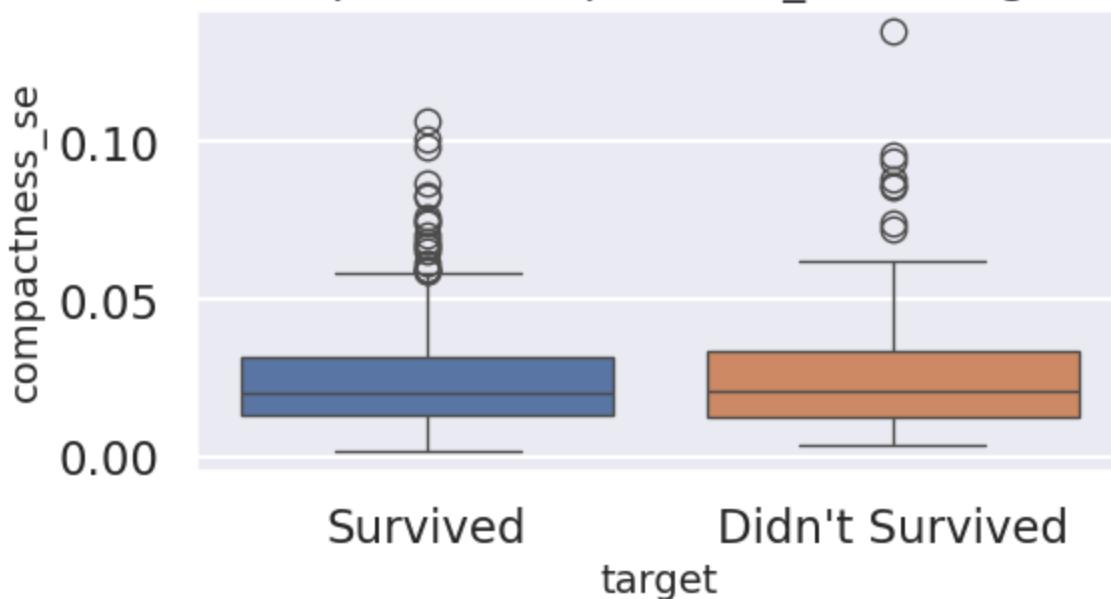
```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='smoothness_se'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of smoothness_se vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'smoothness_se')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='compactness_se'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of compactness_se vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'compactness_se')
```

### Boxplot of compactness\_se vs Target

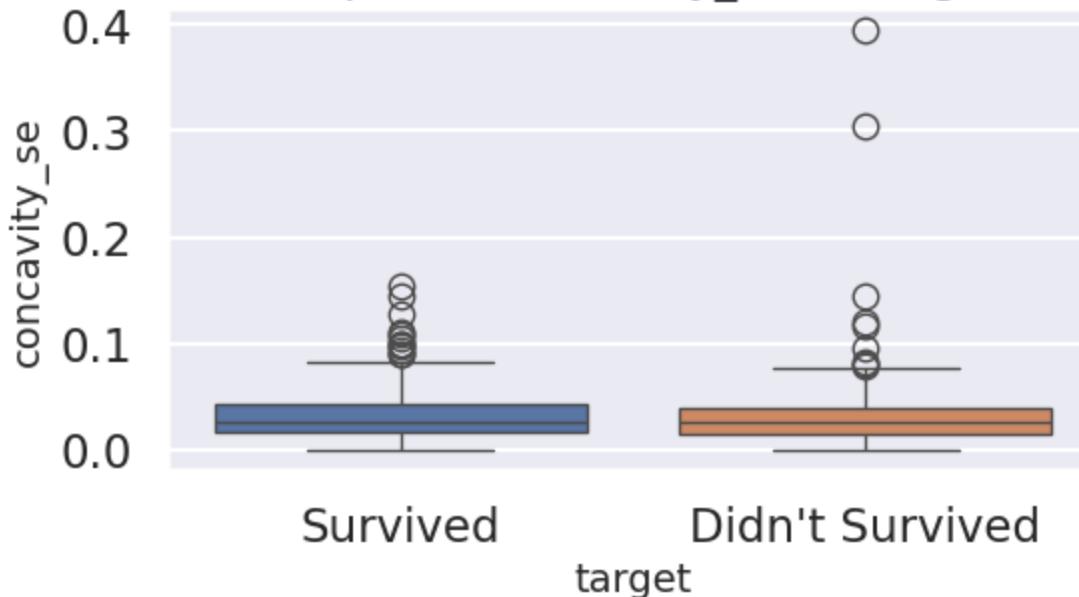


```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

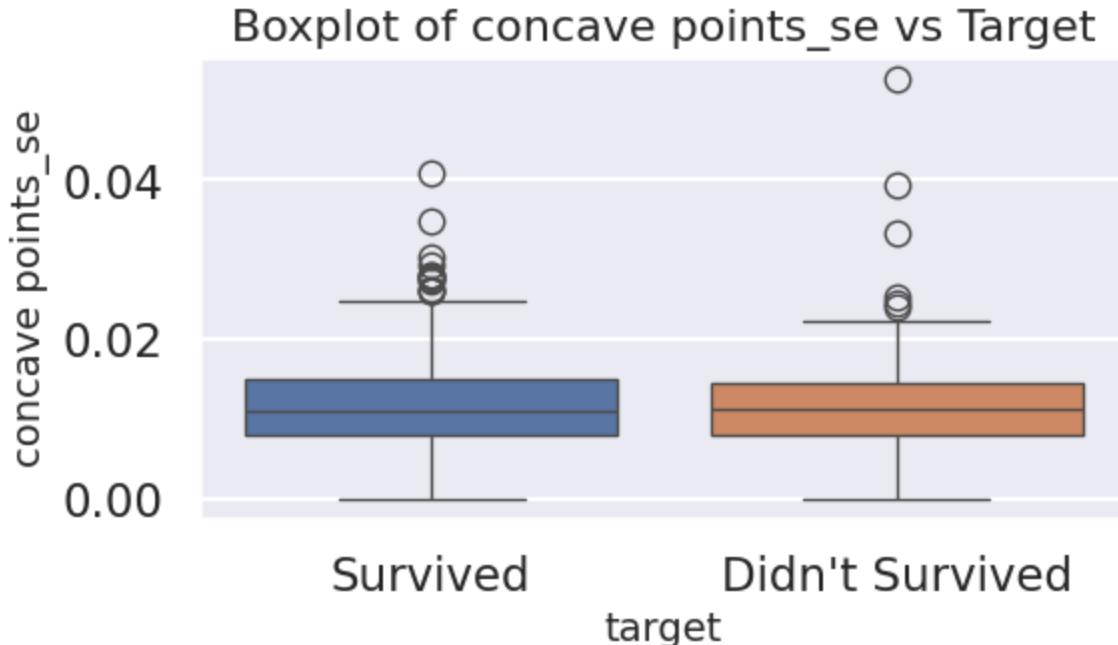
```
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
<Axes: xlabel='target', ylabel='concavity_se'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of concavity_se vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'concavity_se')
```

### Boxplot of concavity\_se vs Target



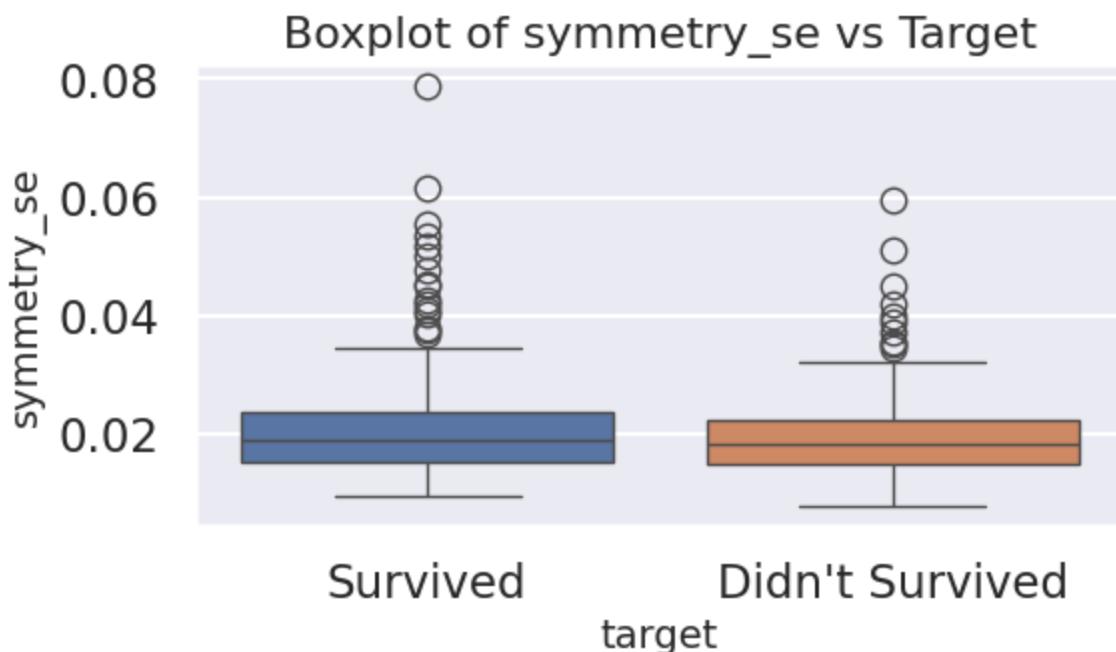
```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='concave points_se'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of concave points_se vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'concave points_se')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

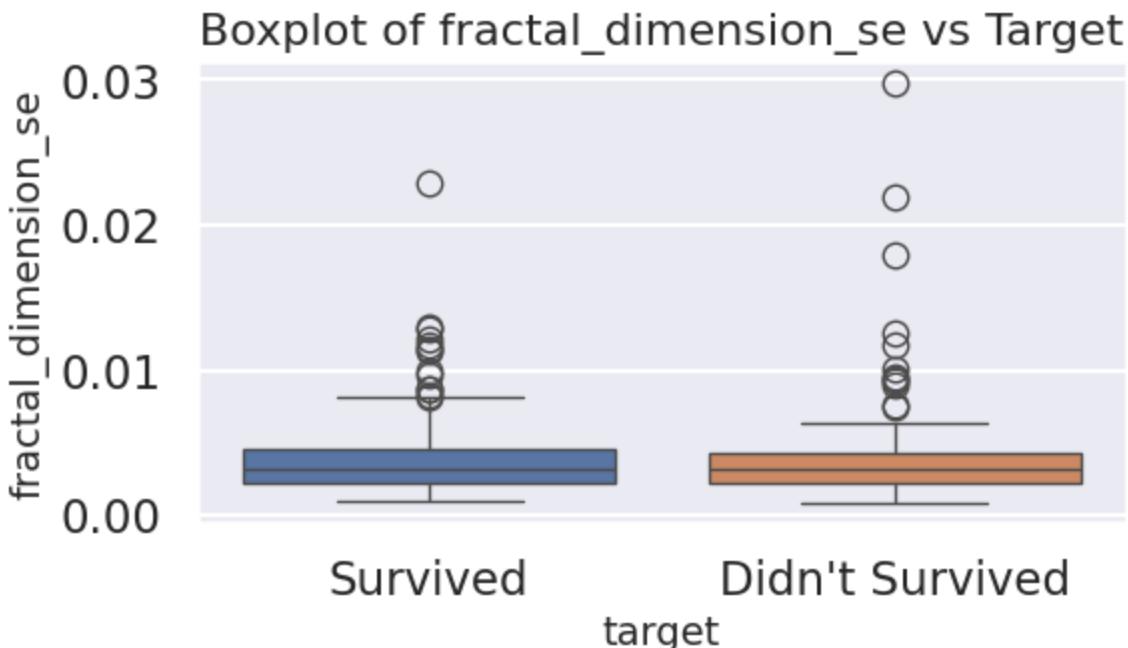
```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='symmetry_se'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of symmetry_se vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'symmetry_se')
```



Out[12]: <Figure size 600x300 with 0 Axes>

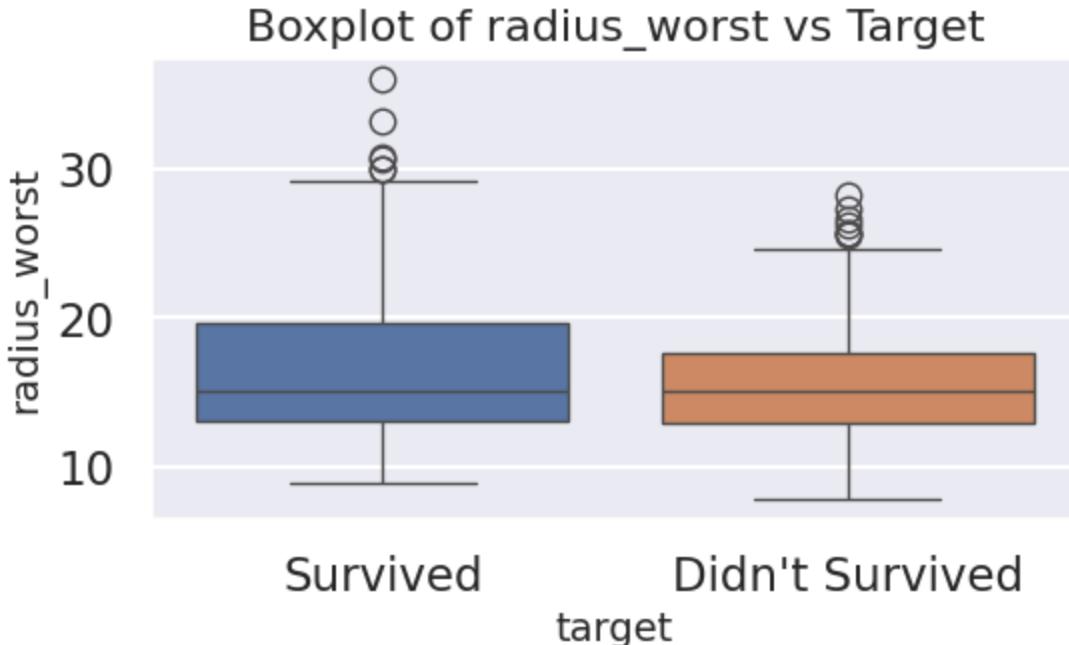
```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the ``x`` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")
Out[12]: <Axes: xlabel='target', ylabel='fractal_dimension_se'>
Out[12]: Text(0.5, 1.0, 'Boxplot of fractal_dimension_se vs Target')
Out[12]: Text(0.5, 0, 'target')
Out[12]: Text(0, 0.5, 'fractal_dimension_se')
```



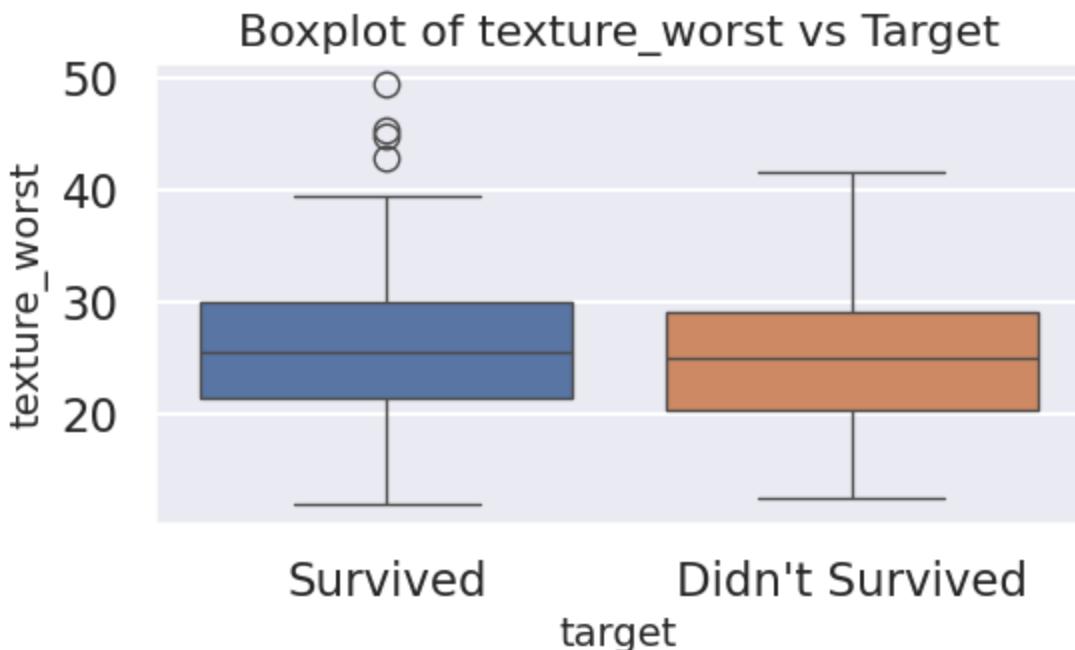
Out[12]: <Figure size 600x300 with 0 Axes>

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='radius_worst'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of radius_worst vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'radius_worst')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

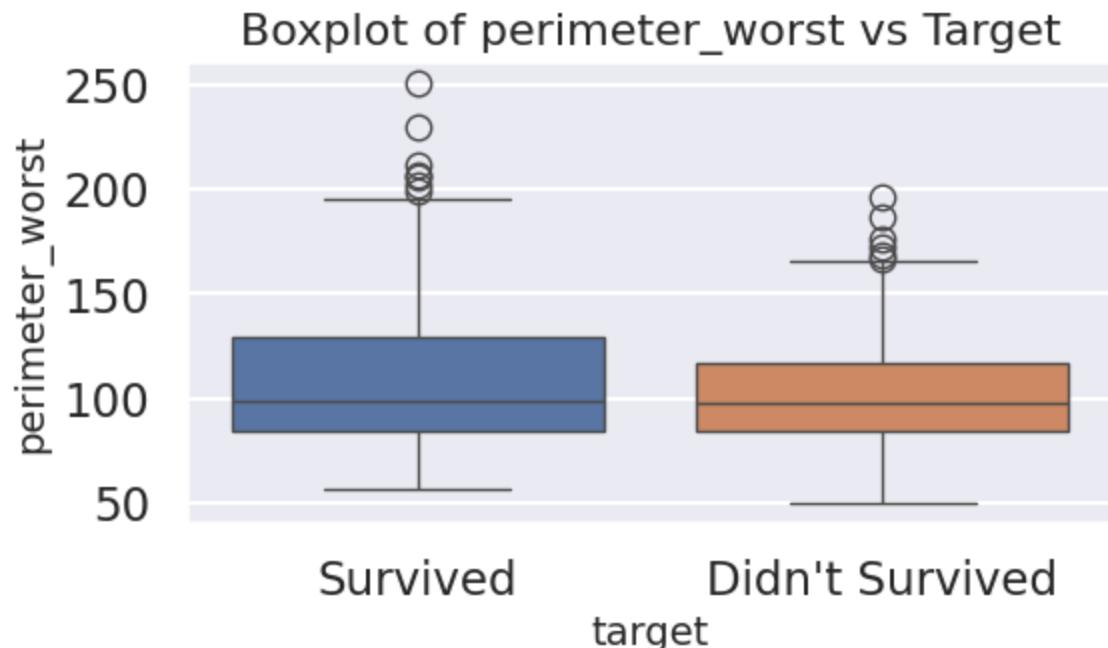
```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='texture_worst'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of texture_worst vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'texture_worst')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

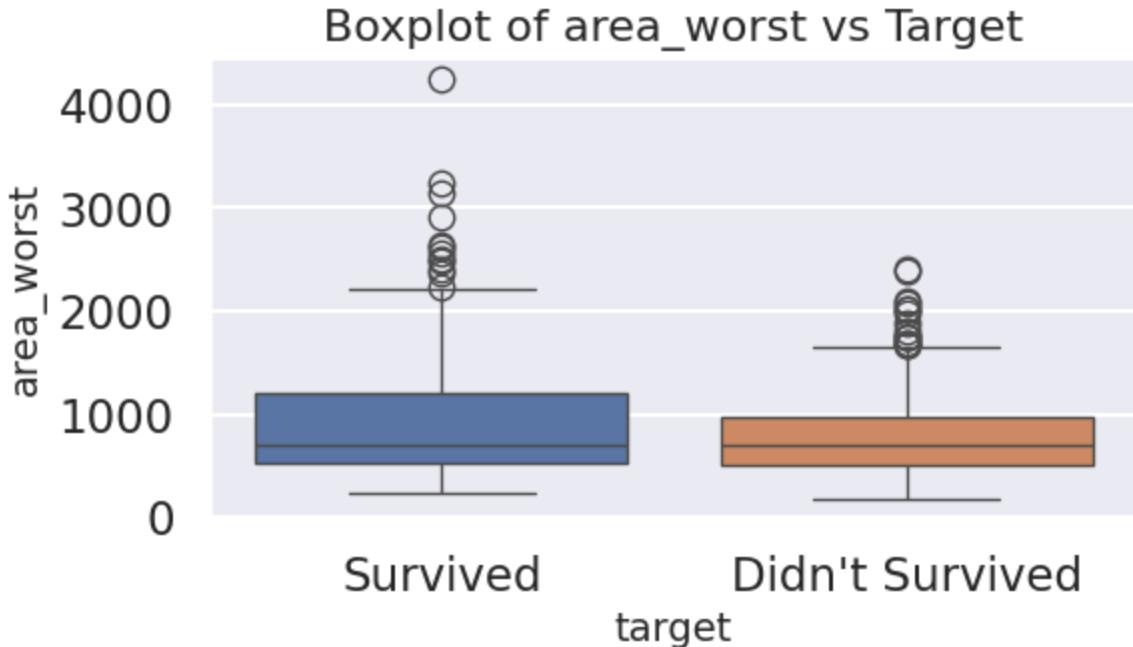
```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
<Axes: xlabel='target', ylabel='perimeter_worst'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of perimeter_worst vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'perimeter_worst')
```

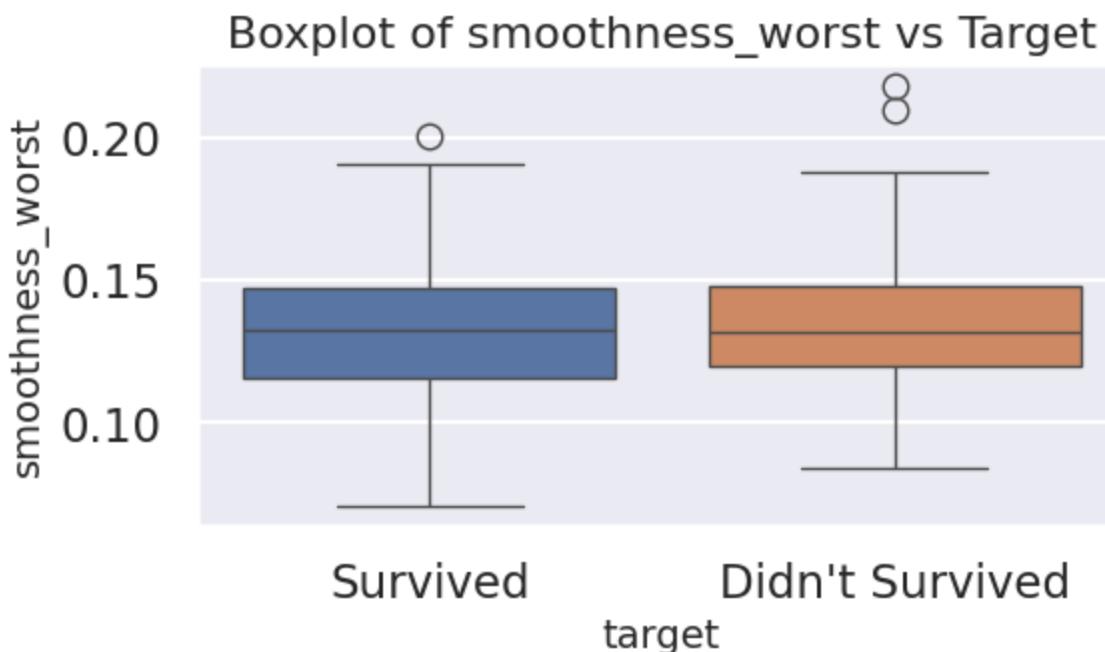


```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='area_worst'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of area_worst vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'area_worst')
```



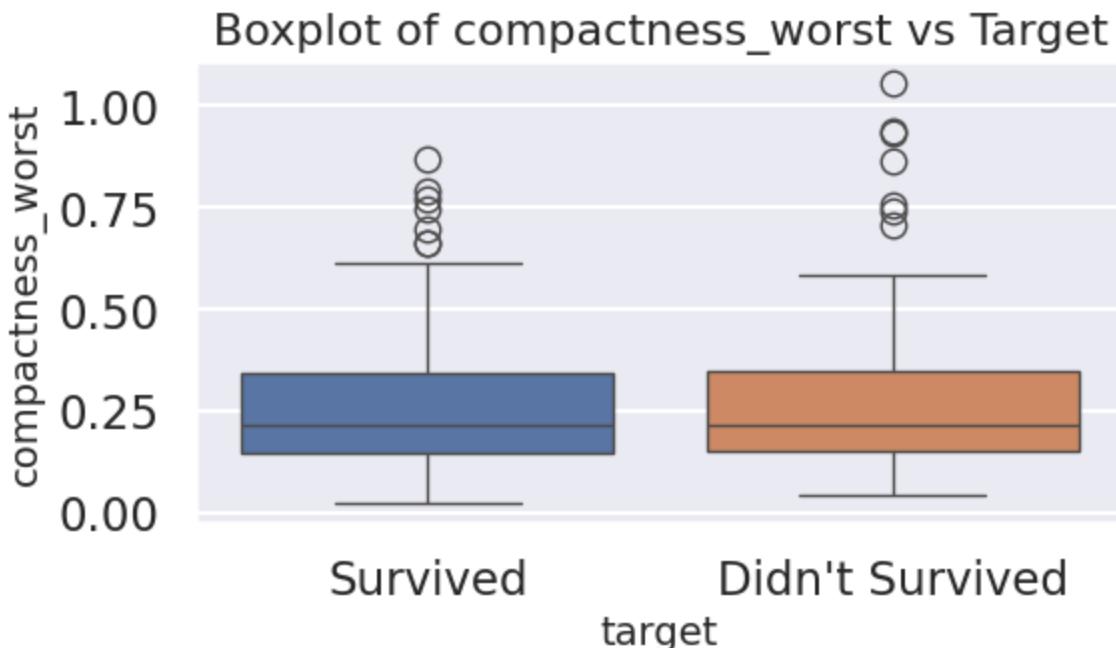
```
Out[12]: <Figure size 600x300 with 0 Axes>  
  
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='smoothness_worst'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of smoothness_worst vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'smoothness_worst')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

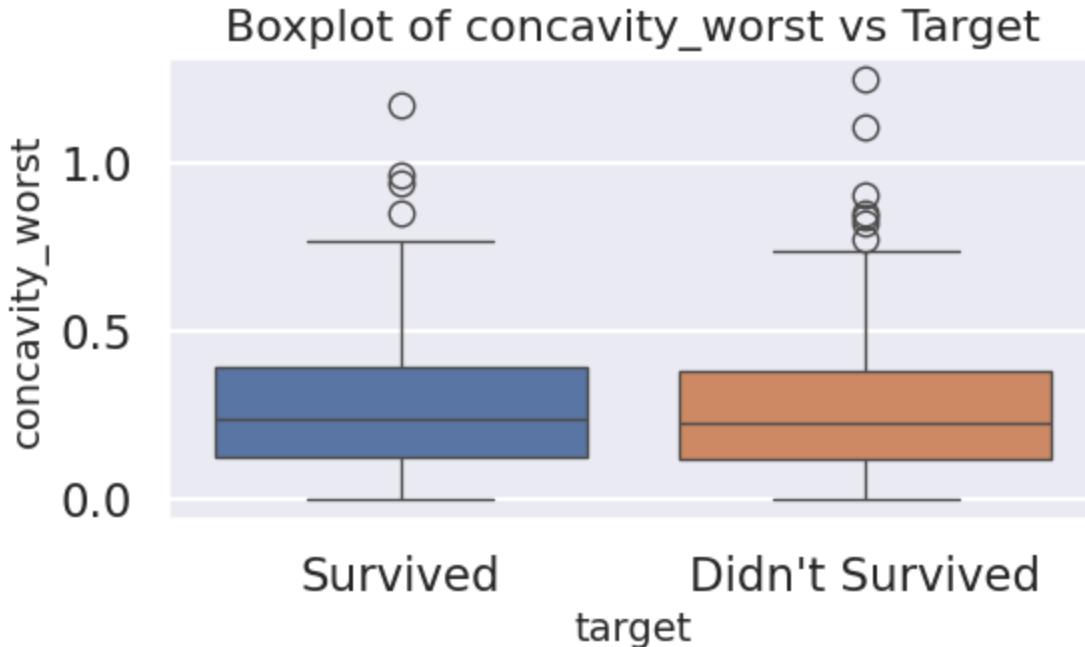
```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
<Axes: xlabel='target', ylabel='compactness_worst'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of compactness_worst vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'compactness_worst')
```

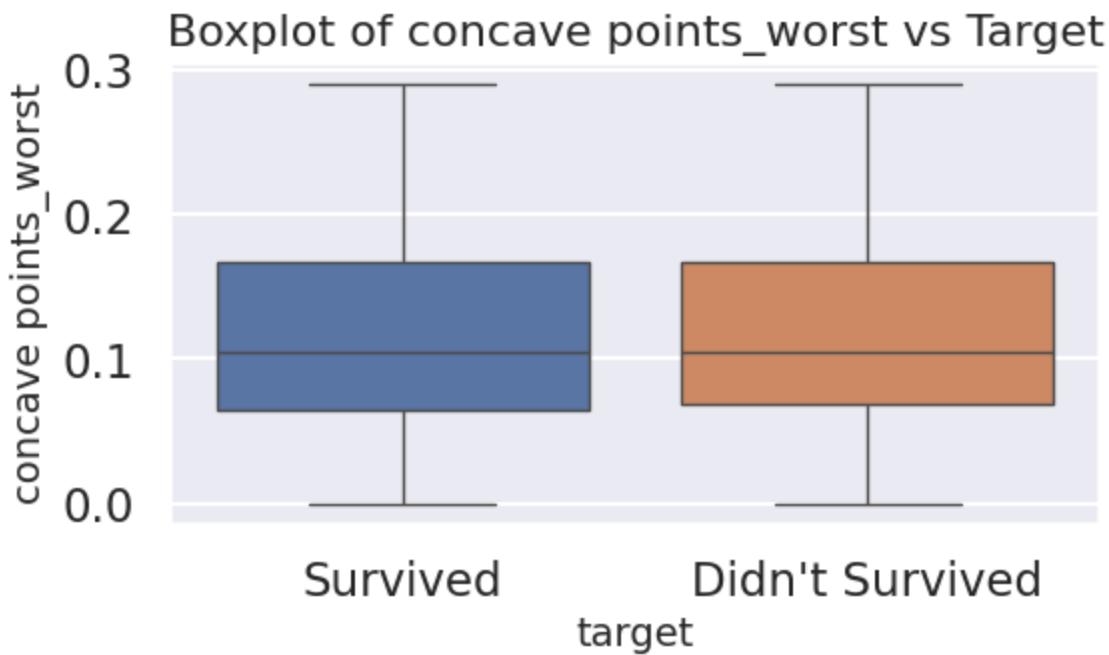


```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='concavity_worst'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of concavity_worst vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'concavity_worst')
```



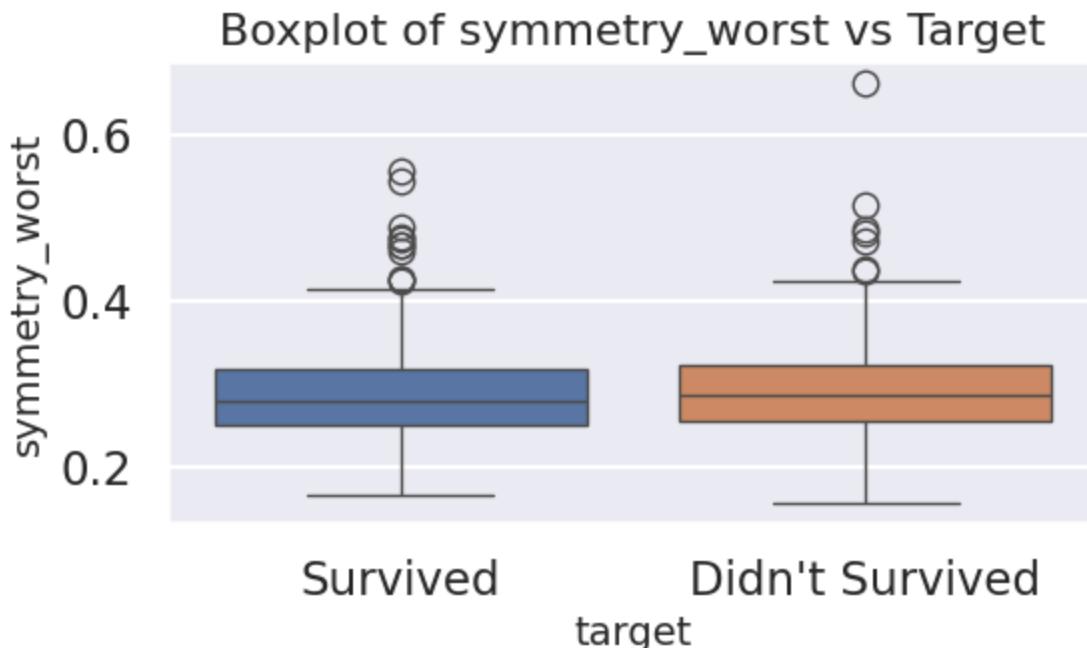
```
Out[12]: <Figure size 600x300 with 0 Axes>  
  
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
Out[12]: <Axes: xlabel='target', ylabel='concave points_worst'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of concave points_worst vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'concave points_worst')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

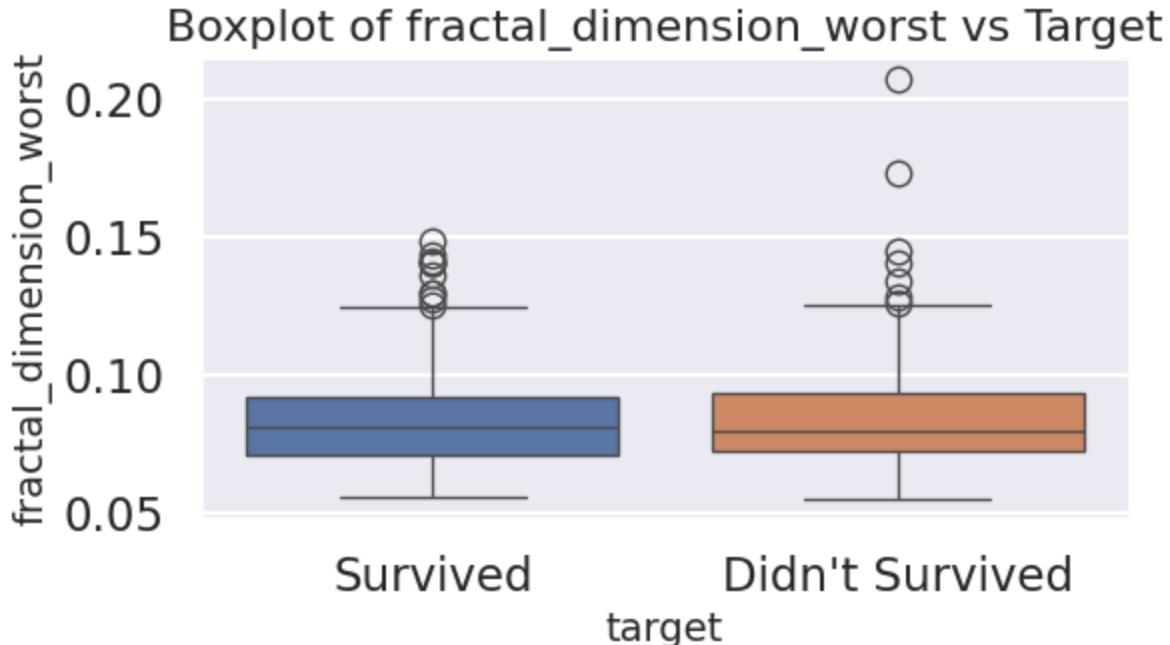
```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
<Axes: xlabel='target', ylabel='symmetry_worst'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of symmetry_worst vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'symmetry_worst')
```



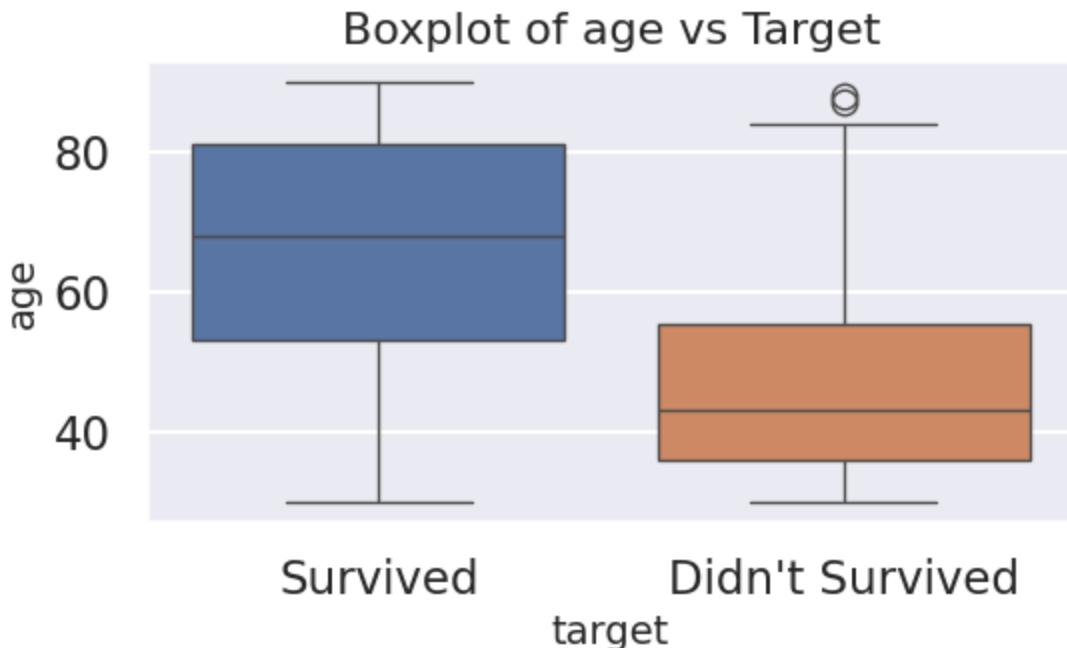
```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
<Axes: xlabel='target', ylabel='fractal_dimension_worst'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of fractal_dimension_worst vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'fractal_dimension_worst')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.  
  
sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")  
<Axes: xlabel='target', ylabel='age'>  
Out[12]: Text(0.5, 1.0, 'Boxplot of age vs Target')  
Out[12]: Text(0.5, 0, 'target')  
Out[12]: Text(0, 0.5, 'age')
```



```
Out[12]: <Figure size 600x300 with 0 Axes>
```

```
<ipython-input-12-c8da27f086f5>:11: FutureWarning:  
  Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.  
  0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

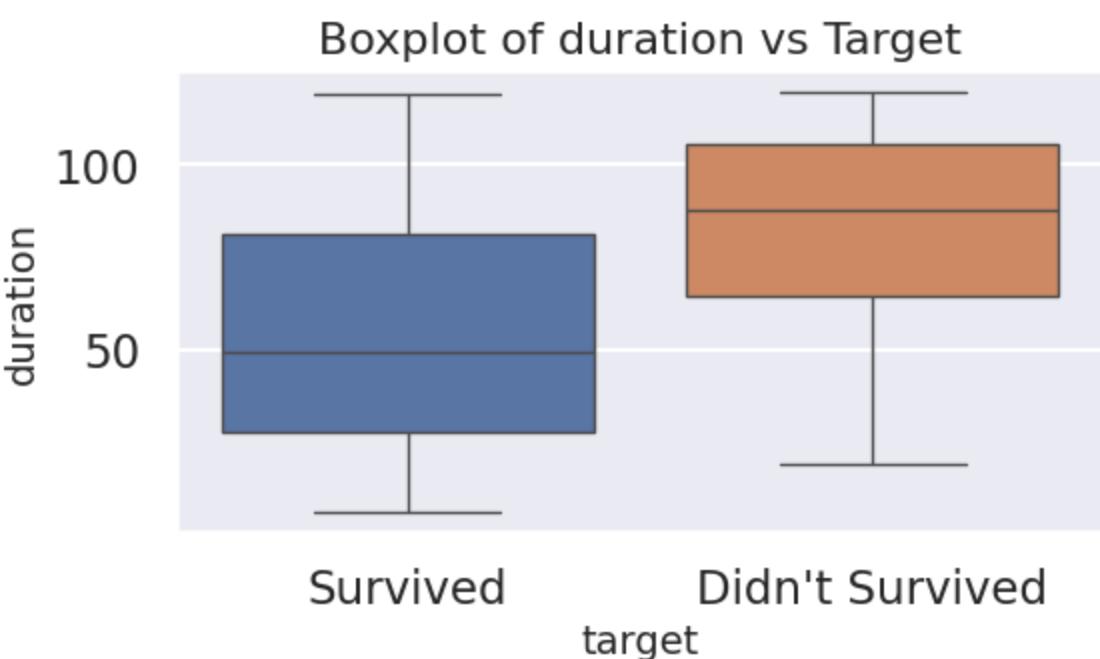
```
    sns.boxplot(x='target', y=column, data=train_data, orient="v", palette="deep")
```

```
Out[12]: <Axes: xlabel='target', ylabel='duration'>
```

```
Out[12]: Text(0.5, 1.0, 'Boxplot of duration vs Target')
```

```
Out[12]: Text(0.5, 0, 'target')
```

```
Out[12]: Text(0, 0.5, 'duration')
```



It appears that there are some patients with more than one outlier, we'll remove them from the data set.

```
In [13]: train_data = pd.read_csv('train_data.csv')

numerical_columns = train_data.select_dtypes(include=['int64', 'float64']).columns
numerical_columns = [col for col in numerical_columns if col != 'target']

outliers_mask = pd.Series([False] * len(train_data))

for column in numerical_columns:
    Q1 = train_data[column].quantile(0.25)
    Q3 = train_data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers_mask |= (train_data[column] < lower_bound) | (train_data[column] > upper_bound)

train_data = train_data[~outliers_mask]

rows_removed = outliers_mask.sum()
print(f"Number of patients removed: {rows_removed}")
train_data.to_csv('train_data_02.csv')
```

Number of patients removed: 138

## EDA

### Visualizing Features

Apply transformations based on the features's skewness and remove features that still have significant skewness after the transformations

```
In [14]: fig, axes = plt.subplots(10, 4, figsize=(40, 40))

axes = axes.flatten()

for index, columnName in enumerate(train_data.columns[1:]):
    ax = axes[index]
    if train_data[columnName].dtype == 'object':
        sns.countplot(x=columnName, data=train_data, ax=ax)
    else:
        sns.histplot(x=columnName, data=train_data, ax=ax)

    ax.set_title(columnName)

plt.tight_layout()
plt.show()

numerical_features = train_data.select_dtypes(include=['int64', 'float64']).columns[1:]

skewed_features = train_data[numerical_features].skew().sort_values(ascending=False)

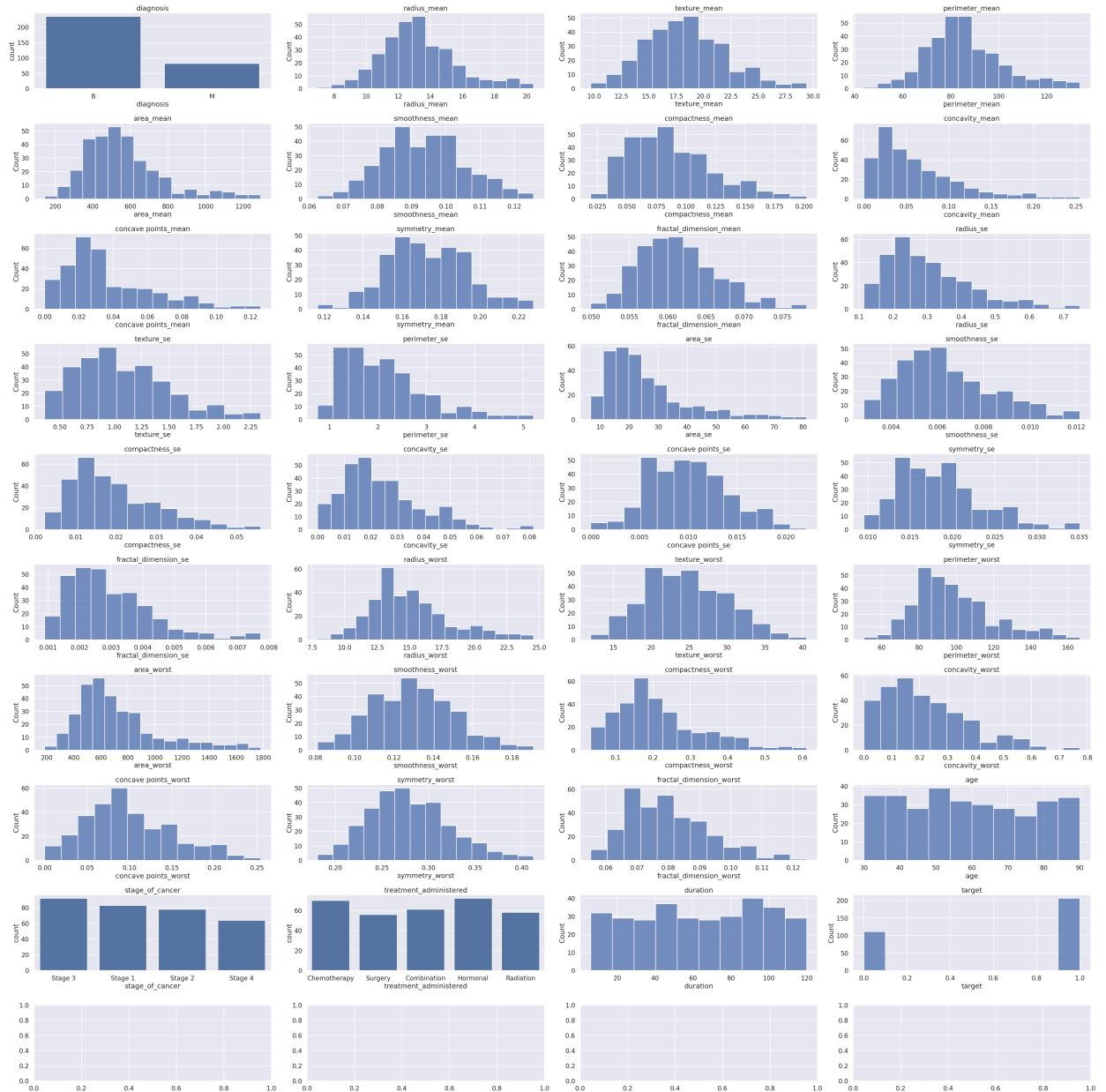
skewness_df = pd.DataFrame({'Skew': skewed_features})
```

```
print(skewness_df)

Out[14]: <Axes: xlabel='diagnosis', ylabel='count'>
Out[14]: Text(0.5, 1.0, 'diagnosis')
Out[14]: <Axes: xlabel='radius_mean', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'radius_mean')
Out[14]: <Axes: xlabel='texture_mean', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'texture_mean')
Out[14]: <Axes: xlabel='perimeter_mean', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'perimeter_mean')
Out[14]: <Axes: xlabel='area_mean', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'area_mean')
Out[14]: <Axes: xlabel='smoothness_mean', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'smoothness_mean')
Out[14]: <Axes: xlabel='compactness_mean', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'compactness_mean')
Out[14]: <Axes: xlabel='concavity_mean', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'concavity_mean')
Out[14]: <Axes: xlabel='concave points_mean', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'concave points_mean')
Out[14]: <Axes: xlabel='symmetry_mean', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'symmetry_mean')
Out[14]: <Axes: xlabel='fractal_dimension_mean', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'fractal_dimension_mean')
Out[14]: <Axes: xlabel='radius_se', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'radius_se')
Out[14]: <Axes: xlabel='texture_se', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'texture_se')
Out[14]: <Axes: xlabel='perimeter_se', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'perimeter_se')
Out[14]: <Axes: xlabel='area_se', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'area_se')
Out[14]: <Axes: xlabel='smoothness_se', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'smoothness_se')
```

```
Out[14]: <Axes: xlabel='compactness_se', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'compactness_se')
Out[14]: <Axes: xlabel='concavity_se', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'concavity_se')
Out[14]: <Axes: xlabel='concave points_se', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'concave points_se')
Out[14]: <Axes: xlabel='symmetry_se', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'symmetry_se')
Out[14]: <Axes: xlabel='fractal_dimension_se', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'fractal_dimension_se')
Out[14]: <Axes: xlabel='radius_worst', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'radius_worst')
Out[14]: <Axes: xlabel='texture_worst', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'texture_worst')
Out[14]: <Axes: xlabel='perimeter_worst', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'perimeter_worst')
Out[14]: <Axes: xlabel='area_worst', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'area_worst')
Out[14]: <Axes: xlabel='smoothness_worst', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'smoothness_worst')
Out[14]: <Axes: xlabel='compactness_worst', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'compactness_worst')
Out[14]: <Axes: xlabel='concavity_worst', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'concavity_worst')
Out[14]: <Axes: xlabel='concave points_worst', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'concave points_worst')
Out[14]: <Axes: xlabel='symmetry_worst', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'symmetry_worst')
Out[14]: <Axes: xlabel='fractal_dimension_worst', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'fractal_dimension_worst')
Out[14]: <Axes: xlabel='age', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'age')
Out[14]: <Axes: xlabel='stage_of_cancer', ylabel='count'>
```

```
Out[14]: Text(0.5, 1.0, 'stage_of_cancer')
Out[14]: <Axes: xlabel='treatment_administered', ylabel='count'>
Out[14]: Text(0.5, 1.0, 'treatment_administered')
Out[14]: <Axes: xlabel='duration', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'duration')
Out[14]: <Axes: xlabel='target', ylabel='Count'>
Out[14]: Text(0.5, 1.0, 'target')
```



Now we'll perform Log Transformation for the features that has right skewness and Square Root Transformation to the features that has left skewness.

```
In [15]: # Threshold for skewness
threshold = 0.75

def transform_features(train_data, skewed_features, threshold=0.75):
```

```

changed_features = {}
for feature in skewed_features.index:
    skewness = skewed_features[feature]
    original_values = train_data[feature].copy()

    if skewness > threshold:
        if (train_data[feature] > 0).all():
            train_data[feature] = np.log1p(train_data[feature])
        else:
            print(f"Skipping {feature} due to negative or zero values in the data.")

    elif skewness < -threshold:
        train_data[feature] = np.sqrt(train_data[feature].abs())

    if not train_data[feature].equals(original_values):
        changed_features[feature] = {
            'original_skewness': skewness,
            'new_skewness': train_data[feature].skew()
        }

return train_data, changed_features

train_data_transformed, changed_features = transform_features(train_data.copy(), skew)

if changed_features:
    print("\nFeatures with Skewness Changes:")
    for feature, skewness_info in changed_features.items():
        print(f"Feature: {feature}")
        print(f" Original Skewness: {skewness_info['original_skewness']}")
        print(f" New Skewness: {skewness_info['new_skewness']}\n")
else:
    print("No features had significant changes in skewness.")

if changed_features:
    fig, axes = plt.subplots(3, 4, figsize=(40, 40))
    axes = axes.flatten() if len(changed_features) > 1 else [axes]

    for ax, (feature, _) in zip(axes, changed_features.items()):
        sns.histplot(train_data_transformed[feature], ax=ax)
        ax.set_title(f"{feature} (Transformed)")

    plt.tight_layout()
    plt.show()

```

	Skew
area_se	1.500621
concavity_mean	1.341517
area_worst	1.259710
fractal_dimension_se	1.152364
area_mean	1.139781
concave_points_mean	1.109217
perimeter_se	1.056725
radius_se	1.045883
concavity_se	0.982360
compactness_se	0.938201
compactness_worst	0.927227
concavity_worst	0.905529
symmetry_se	0.823085
radius_worst	0.803780
perimeter_worst	0.779180
compactness_mean	0.727498
fractal_dimension_worst	0.726821
smoothness_se	0.694778
texture_se	0.665351
perimeter_mean	0.634102
radius_mean	0.609254
fractal_dimension_mean	0.545257
concave_points_worst	0.535519
symmetry_worst	0.455974
texture_mean	0.377752
texture_worst	0.264952
smoothness_worst	0.241018
concave_points_se	0.228033
smoothness_mean	0.184778
symmetry_mean	0.141812
age	0.088668
duration	-0.069092
target	-0.631235

Skipping concavity\_mean due to negative or zero values in the data.

Skipping concave points\_mean due to negative or zero values in the data.

Skipping concavity\_se due to negative or zero values in the data.

Skipping concavity\_worst due to negative or zero values in the data.

#### Features with Skewness Changes:

Feature: area\_se

Original Skewness: 1.500620894091251

New Skewness: 0.37384820311316624

Feature: area\_worst

Original Skewness: 1.2597098098356303

New Skewness: 0.20756307817678538

Feature: fractal\_dimension\_se

Original Skewness: 1.1523641192918657

New Skewness: 1.14801861775621

Feature: area\_mean

Original Skewness: 1.139780952552101

New Skewness: -0.010332358714100688

Feature: perimeter\_se

Original Skewness: 1.056724920701799

New Skewness: 0.43460137178308555

```
Feature: radius_se
Original Skewness: 1.045883338017319
New Skewness: 0.8134414067927772

Feature: compactness_se
Original Skewness: 0.938200774575379
New Skewness: 0.9140320773551073

Feature: compactness_worst
Original Skewness: 0.9272273016618426
New Skewness: 0.7074258537878015

Feature: symmetry_se
Original Skewness: 0.8230851527881472
New Skewness: 0.8096950632228345

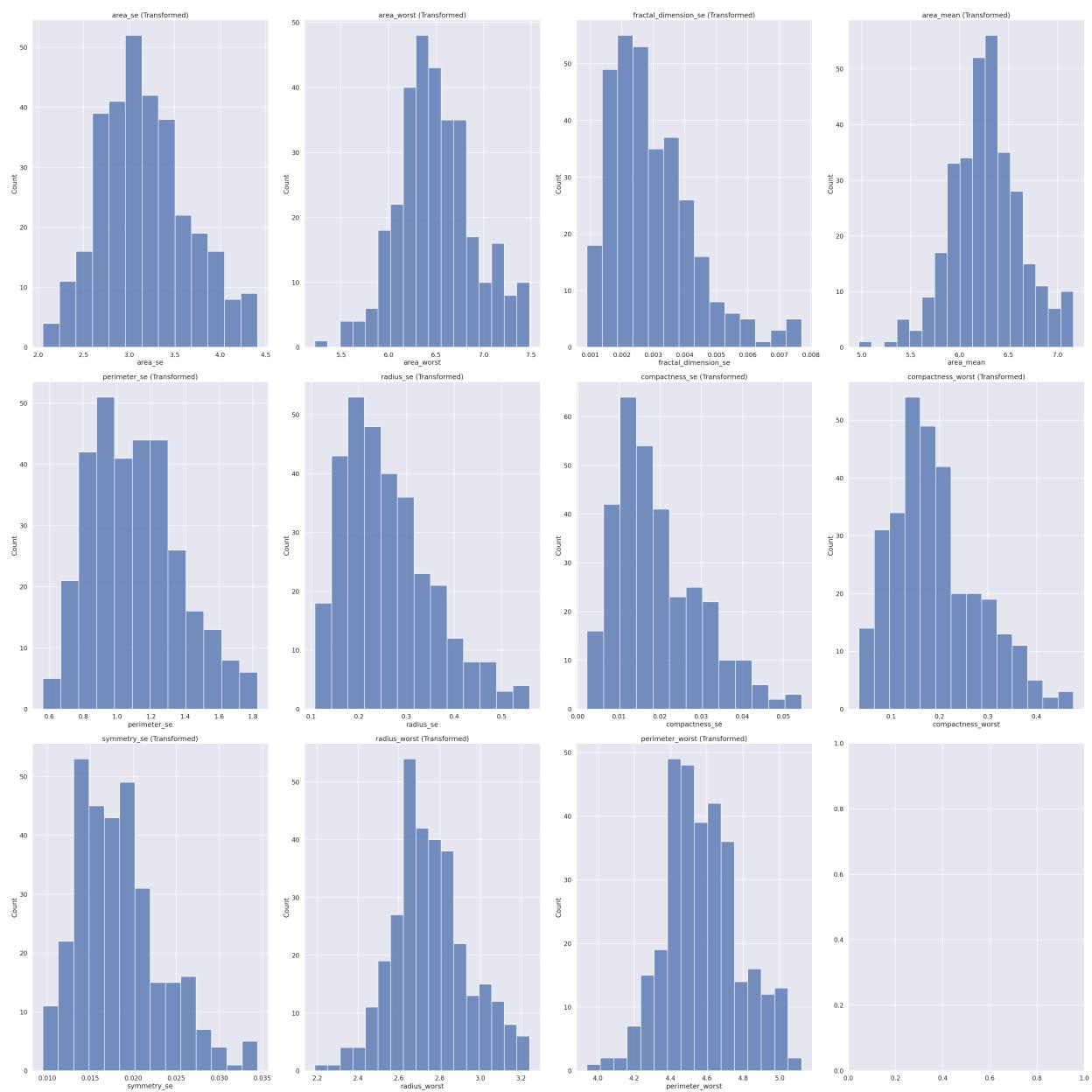
Feature: radius_worst
Original Skewness: 0.8037804580212827
New Skewness: 0.2781036839218181

Feature: perimeter_worst
Original Skewness: 0.779180092711607
New Skewness: 0.22718500225809612

Out[15]: <Axes: xlabel='area_se', ylabel='Count'>
Out[15]: Text(0.5, 1.0, 'area_se (Transformed)')
Out[15]: <Axes: xlabel='area_worst', ylabel='Count'>
Out[15]: Text(0.5, 1.0, 'area_worst (Transformed)')
Out[15]: <Axes: xlabel='fractal_dimension_se', ylabel='Count'>
Out[15]: Text(0.5, 1.0, 'fractal_dimension_se (Transformed)')
Out[15]: <Axes: xlabel='area_mean', ylabel='Count'>
Out[15]: Text(0.5, 1.0, 'area_mean (Transformed)')
Out[15]: <Axes: xlabel='perimeter_se', ylabel='Count'>
Out[15]: Text(0.5, 1.0, 'perimeter_se (Transformed)')
Out[15]: <Axes: xlabel='radius_se', ylabel='Count'>
Out[15]: Text(0.5, 1.0, 'radius_se (Transformed)')
Out[15]: <Axes: xlabel='compactness_se', ylabel='Count'>
Out[15]: Text(0.5, 1.0, 'compactness_se (Transformed)')
Out[15]: <Axes: xlabel='compactness_worst', ylabel='Count'>
Out[15]: Text(0.5, 1.0, 'compactness_worst (Transformed)')
Out[15]: <Axes: xlabel='symmetry_se', ylabel='Count'>
Out[15]: Text(0.5, 1.0, 'symmetry_se (Transformed)')
Out[15]: <Axes: xlabel='radius_worst', ylabel='Count'>
Out[15]: Text(0.5, 1.0, 'radius_worst (Transformed)')
```

```
Out[15]: <Axes: xlabel='perimeter_worst', ylabel='Count'>
```

```
Out[15]: Text(0.5, 1.0, 'perimeter_worst (Transformed)')
```



After the transformations, we'll remove the features that still has high skewness

```
In [16]: final_skewness = train_data_transformed[numerical_features].skew().sort_values(ascending=True)

features_to_remove = final_skewness[final_skewness.abs() > threshold].index

if len(features_to_remove) > 0:
    print(f"\nFeatures to be removed due to high skewness (>{threshold}):")
    for feature in features_to_remove:
        print(f" - {feature}")
else:
    print("\nNo features to remove. All features are within the skewness threshold.")

train_data_cleaned = train_data_transformed.drop(columns=features_to_remove)

print(f"\nNumber of features removed: {len(features_to_remove)})")
```

```
train_data_cleaned.to_csv('train_data_03.csv', index=False)
```

Features to be removed due to high skewness (>0.75):

- concavity\_mean
- fractal\_dimension\_se
- concave\_points\_mean
- concavity\_se
- compactness\_se
- concavity\_worst
- radius\_se
- symmetry\_se

Number of features removed: 8

## Feature Selection Based on Correlation Matrix

Iterative process where we remove one feature at a time, re-calculate the correlation matrix, and then reassess which features to remove next. This approach can help address any changes in correlation dynamics after each feature is removed.

```
In [17]: numerical_data = train_data_cleaned.select_dtypes(include=['float64', 'int64'])

correlation_threshold = 0.8

def find_most_correlated(data, threshold):
    correlation_matrix = data.corr()
    features = correlation_matrix.columns
    for i in range(len(features)):
        for j in range(i):
            if abs(correlation_matrix.iloc[i, j]) > threshold:
                return (features[i], features[j])
    return None

while True:
    pair = find_most_correlated(numerical_data, correlation_threshold)
    if pair is None:
        break
    feature_to_remove = pair[0]
    numerical_data = numerical_data.drop(columns=[feature_to_remove])
    print(f"Removed {feature_to_remove} due to high correlation with {pair[1]}")

plt.figure(figsize=(20, 20))
sns.heatmap(numerical_data.corr(), annot=True, cmap='coolwarm', fmt=".2f", linewidths=1)
plt.title('Final Correlation Matrix for Numerical Features')
plt.show()

numerical_data.to_csv('reduced_data.csv', index=False)
```

Removed perimeter\_mean due to high correlation with radius\_mean

Removed area\_mean due to high correlation with radius\_mean

Removed area\_se due to high correlation with perimeter\_se

Removed radius\_worst due to high correlation with radius\_mean

Removed texture\_worst due to high correlation with texture\_mean

Removed perimeter\_worst due to high correlation with radius\_mean

Removed area\_worst due to high correlation with radius\_mean

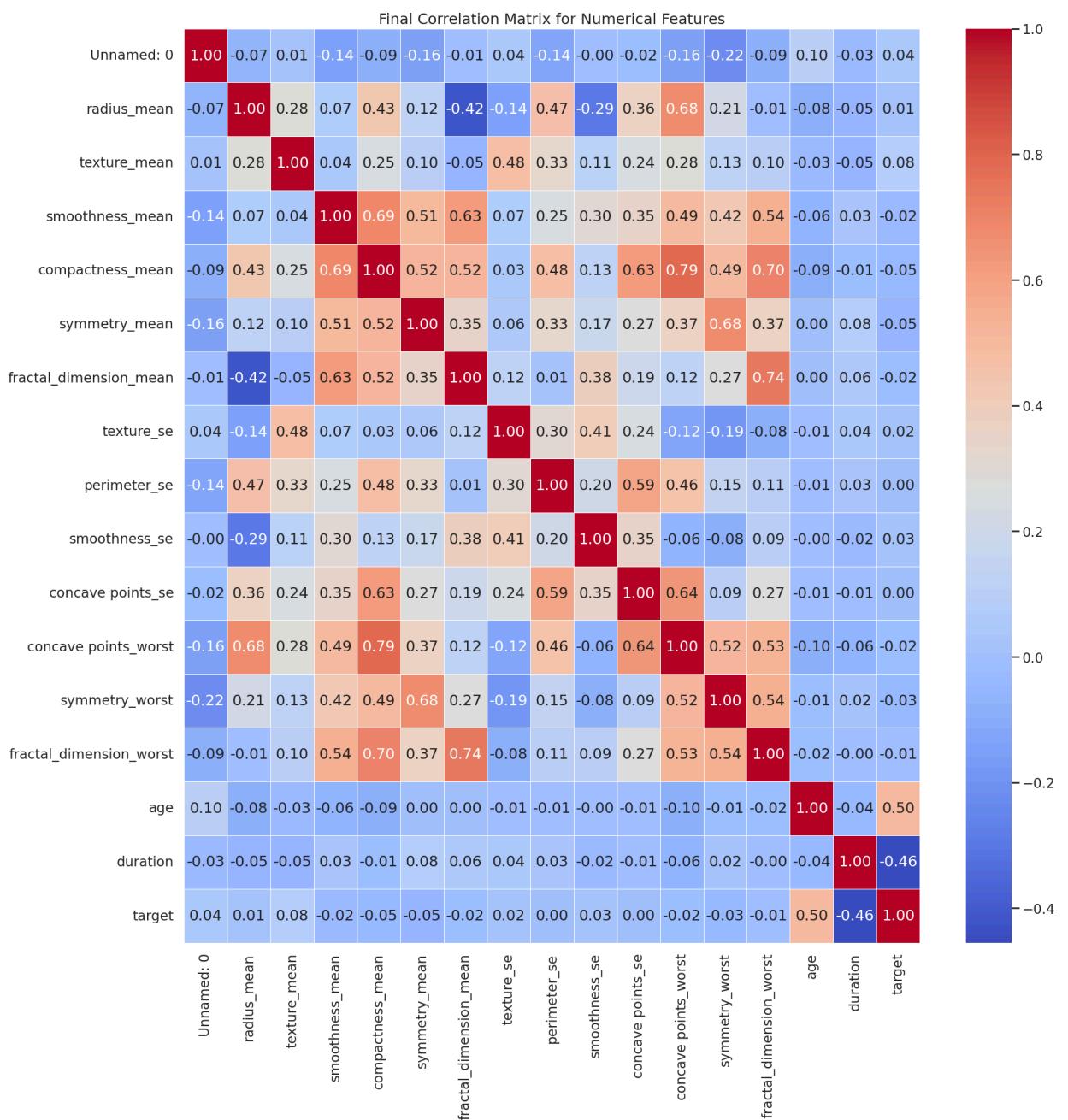
Removed smoothness\_worst due to high correlation with smoothness\_mean

Removed compactness\_worst due to high correlation with compactness\_mean

Out[17]: &lt;Figure size 2000x2000 with 0 Axes&gt;

Out[17]: &lt;Axes: &gt;

Out[17]: Text(0.5, 1.0, 'Final Correlation Matrix for Numerical Features')



## Feature Scaling

Standardize or normalize your data especially for ANN.

```
In [18]: scaler = StandardScaler()
data[data.columns] = scaler.fit_transform(data[data.columns])
```

## Modularized Data Processing Functions for Predictive Models:

In [19]:

```

def create_target_column(dataset):
    dataset['target'] = (dataset['survival_probability'] > 0.5).astype(int)
    dataset.drop(columns=['survival_probability'], inplace=True)
    return dataset

def get_outliers_mask(dataset):

    numerical_columns = dataset.select_dtypes(include=['int64', 'float64']).columns
    outliers_mask = pd.Series([False] * len(dataset), index=dataset.index)

    for column in numerical_columns:
        Q1 = dataset[column].quantile(0.25)
        Q3 = dataset[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        outliers_mask |= (dataset[column] < lower_bound) | (dataset[column] > upper_bound)

    return outliers_mask

def outliers_removal(X, y):

    outliers_mask = get_outliers_mask(X)
    X = X[~outliers_mask]
    y = y.loc[X.index]
    return X, y

def impute_values(X):

    X.replace('?', pd.NA, inplace=True)
    X.dropna(inplace=True)
    return X

def transform_features(X, skewness_threshold=0.75):
    numerical_columns = X.select_dtypes(include=['int64', 'float64']).columns

    skewed_features = X[numerical_columns].skew().sort_values(ascending=False)

    for feature in skewed_features.index:
        skewness = skewed_features[feature]

        if skewness > skewness_threshold:
            if (X[feature] > 0).all():
                X[feature] = np.log1p(X[feature])
            else:
                print(f"Skipping log transformation for {feature} due to zero or negative values")
        elif skewness < -skewness_threshold:
            X[feature] = np.sqrt(np.abs(X[feature]))

    return X

def remove_skewed_features(X, skewness_threshold=0.75):
    numerical_columns = X.select_dtypes(include=['int64', 'float64']).columns
    skewed_features = X[numerical_columns].skew().sort_values(ascending=False)

    features_to_remove = skewed_features[abs(skewed_features) > skewness_threshold].index

    if len(features_to_remove) > 0:
        X.drop(features_to_remove, axis=1, inplace=True)

```

```

X = X.drop(columns=features_to_remove)

return X

def one_hot_encode(X):
    categorical_columns = ['diagnosis', 'stage_of_cancer', 'treatment_administered']
    X = pd.get_dummies(X, columns=categorical_columns, drop_first=True)
    return X

def feature_selection_correlation(X, correlation_threshold=0.8):
    def find_most_correlated(data, threshold):
        correlation_matrix = data.corr()
        features = correlation_matrix.columns
        max_correlation = threshold
        feature_to_remove = None
        for i in range(len(features)):
            for j in range(i):
                if abs(correlation_matrix.iloc[i, j]) > max_correlation:
                    max_correlation = abs(correlation_matrix.iloc[i, j])
                    feature_to_remove = features[i] if abs(correlation_matrix.iloc[i,
    return feature_to_remove

while True:
    feature_to_remove = find_most_correlated(X, correlation_threshold)
    if feature_to_remove is None:
        break
    X = X.drop(columns=[feature_to_remove])

return X

def split_data(dataset):
    X = dataset.drop(columns=['target'])
    y = dataset['target']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)

    return X_train, X_test, y_train, y_test

def apply_transformations(X, y):
    X, y = outliers_removal(X, y)
    X = impute_values(X)
    X = transform_features(X)
    X = remove_skewed_features(X)
    X = one_hot_encode(X)
    X = feature_selection_correlation(X)

    return X, y

def process_data(dataset):
    dataset = create_target_column(dataset)
    X_train, X_test, y_train, y_test = split_data(dataset)
    X_train, y_train = apply_transformations(X_train, y_train)
    X_test, y_test = apply_transformations(X_test, y_test)
    common_columns = X_train.columns.intersection(X_test.columns)
    X_train = X_train[common_columns]
    X_test = X_test[common_columns]

    return X_train, X_test, y_train, y_test

```

## Breakdown of the Functions

1. Create Target Column and Remove Survival Probability
  2. Outliers Removal: Identifies and removes outliers using the IQR method for all numerical features.
  3. Impute Missing Values: Handles missing values by replacing ? with NaN and dropping rows with missing values.
  4. Transformation for Skewness Correction: Applies log transformation for right-skewed data and square root transformation for left-skewed data.
  5. Remove Highly Skewed Features: Removes features with high skewness that remain even after transformations.
  6. One-Hot Encoding: Converts categorical variables into dummy variables (binary format), excluding the first category to avoid multicollinearity.
  7. Feature Selection Based on Correlation: Identifies and removes features that have a correlation higher than the specified threshold (e.g., 0.8).
  8. Split Data into Train and Test Sets (with Stratification).

```
In [20]: td = pd.read_csv('data.csv')
X_train, X_test, y_train, y_test = process_data(td)
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
results = {}
```

```
<ipython-input-19-b0ac60b7fdca>:31: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
X.replace('?', pd.NA, inplace=True)
```

```
<ipython-input-19-b0ac60b7fdca>:32: SettingWithCopyWarning:
```

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
x.dropna(inplace=True)
```

```
<ipython-input-19-b0ac60b7fdca>:45: SettingWithCopyWarning:
```

A value is trying to be set on a copy of a slice from a DataFrame.

Try using `.loc[row_indexer,col_indexer] = value` instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
x[feature] = np.log1p(x[feature])
```

```

Skipping log transformation for concavity_mean due to zero or negative values.
Skipping log transformation for concave points_mean due to zero or negative values.
Skipping log transformation for concavity_se due to zero or negative values.
Skipping log transformation for concavity_worst due to zero or negative values.
Skipping log transformation for concavity_mean due to zero or negative values.
Skipping log transformation for concave points_mean due to zero or negative values.
Skipping log transformation for concavity_se due to zero or negative values.
Skipping log transformation for concavity_worst due to zero or negative values.
X_train shape: (255, 21)
X_test shape: (61, 21)
y_train shape: (255,)
y_test shape: (61,)

<ipython-input-19-b0ac60b7fdca>:31: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
    X.replace('?', pd.NA, inplace=True)
<ipython-input-19-b0ac60b7fdca>:32: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
    X.dropna(inplace=True)
<ipython-input-19-b0ac60b7fdca>:45: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
    X[feature] = np.log1p(X[feature])

```

# Logistic Regression

## Basic Training Function

This function will just train the logistic regression model on the entire training set without any cross-validation.

```
In [21]: def train_logistic_regression(X_train, y_train):
    model = LogisticRegression(max_iter=1000, random_state=42)
    model.fit(X_train, y_train)
    return model
```

## Cross-Validation Training Function

This function will perform cross-validation using 5 folds.

```
In [22]: def train_and_evaluate_logistic_regression_with_cv(X_train, y_train, n_splits=5):
    model = LogisticRegression(max_iter=1000, random_state=42)
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)

    cv_results = []
    for train_index, val_index in kf.split(X_train):
```

```

X_train_k, X_val_k = X_train.iloc[train_index], X_train.iloc[val_index]
y_train_k, y_val_k = y_train.iloc[train_index], y_train.iloc[val_index]
model.fit(X_train_k, y_train_k)
results = evaluate_logistic_regression(model, X_val_k, y_val_k, f"Validation F
cv_results.append(results)

# Calculate average of the results
average_results = pd.DataFrame(cv_results).mean().to_dict()
print(f"Average K-Folds results: {average_results}")

return model, average_results

```

## Evaluation Function

This function evaluates the model and returns the results.

```

In [23]: def evaluate_logistic_regression(model, X, y, dataset_name):
    y_pred = model.predict(X)
    y_prob = model.predict_proba(X)[:, 1]

    accuracy = accuracy_score(y, y_pred)
    roc_auc = roc_auc_score(y, y_prob)
    cm = confusion_matrix(y, y_pred)
    sensitivity = cm[1, 1] / (cm[1, 1] + cm[1, 0])
    specificity = cm[0, 0] / (cm[0, 0] + cm[0, 1])

    plt.figure(figsize=(6, 3))
    fpr, tpr, _ = roc_curve(y, y_prob)
    plt.plot(fpr, tpr, label=f'{dataset_name} (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC Curve - {dataset_name}')
    plt.legend()
    plt.show()

    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Greens", cbar=False, square=True)
    plt.title(f'Confusion Matrix - {dataset_name}')
    plt.ylabel('Actual label')
    plt.xlabel('Predicted label')
    plt.show()

    return {'Accuracy': accuracy, 'ROC AUC': roc_auc, 'Sensitivity': sensitivity, 'Sp

```

```

In [24]: initial_lr_model = train_logistic_regression(X_train, y_train)

original_train_results = evaluate_logistic_regression(initial_lr_model, X_train, y_train)

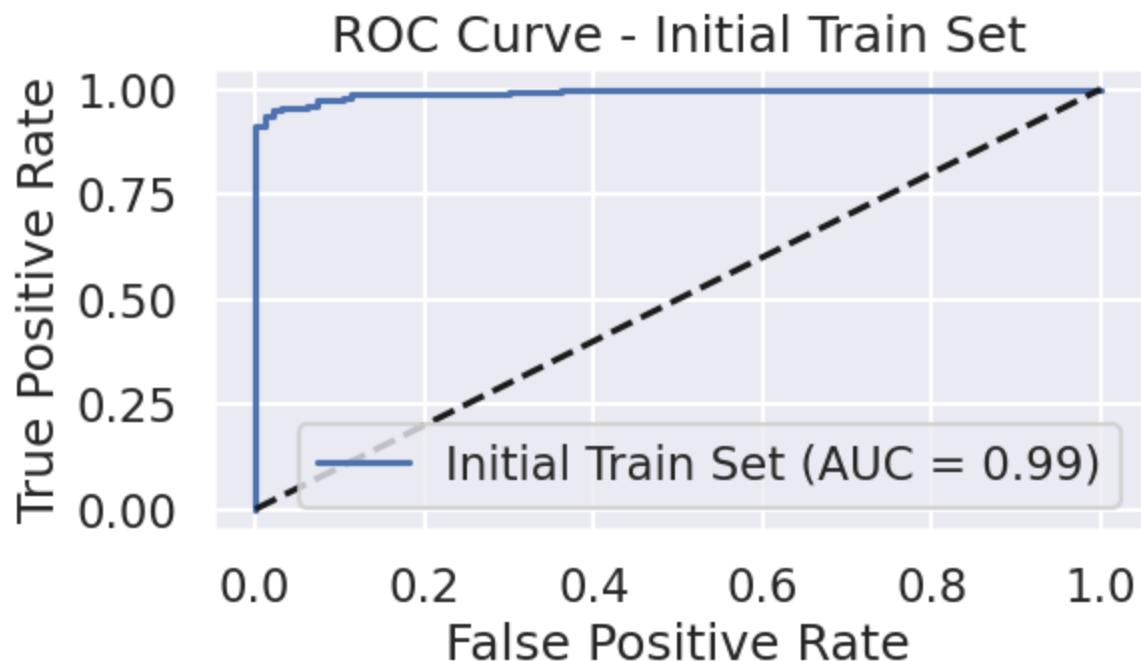
cv_lr_model, cv_train_results = train_and_evaluate_logistic_regression_with_cv(X_train, y_train)

test_results = evaluate_logistic_regression(cv_lr_model, X_test, y_test, "Test Set")
test_results = evaluate_logistic_regression(cv_lr_model, X_test, y_test, "Test Set")

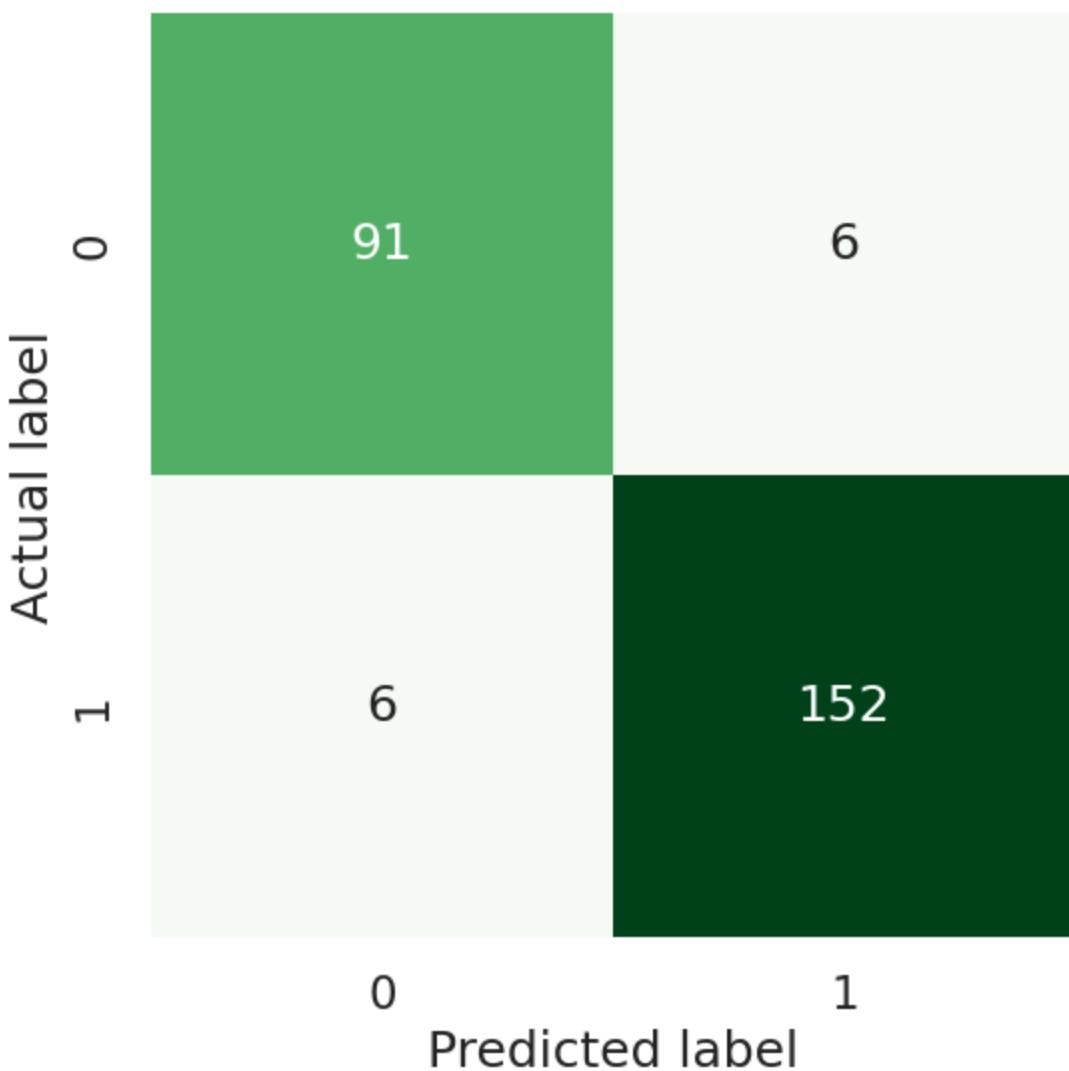
results_df = pd.DataFrame({
    'Logistic Regression': original_train_results,
    'Logistic Regression (CV)': cv_train_results,
})

```

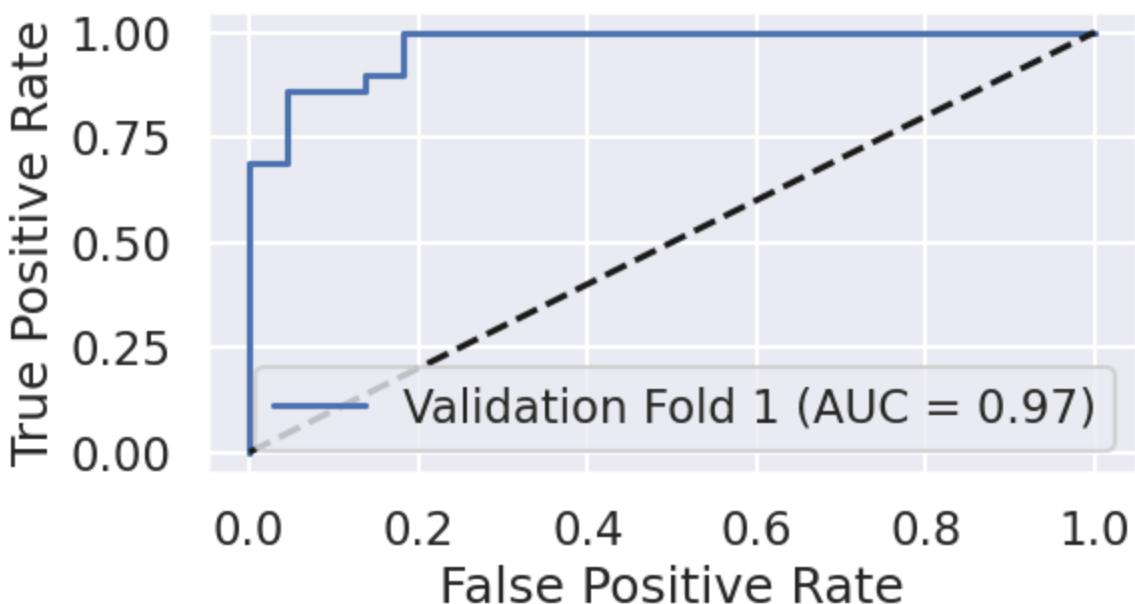
```
'Logistic Regression (Test)': test_results  
}).T  
  
display(results_df)
```



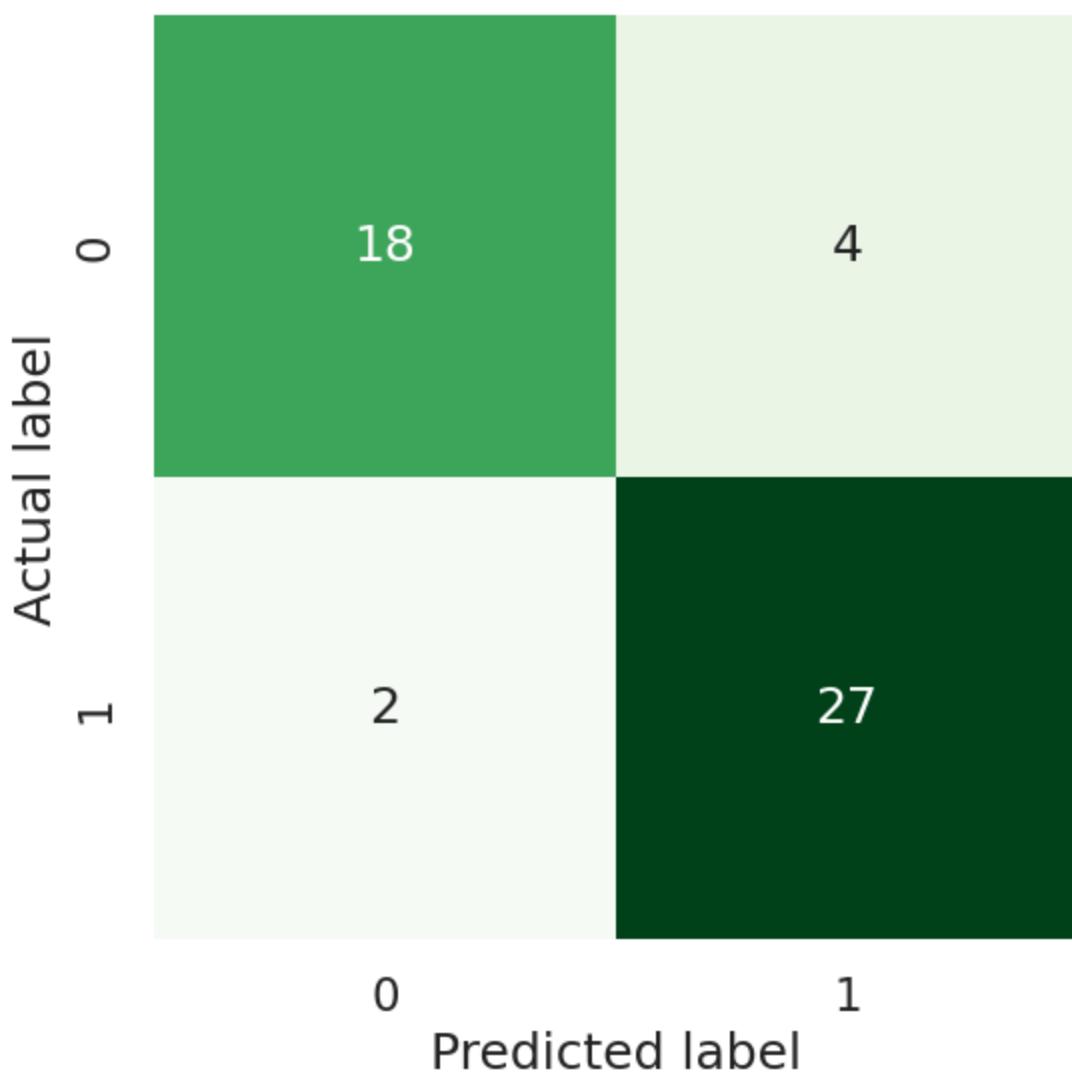
## Confusion Matrix - Initial Train Set



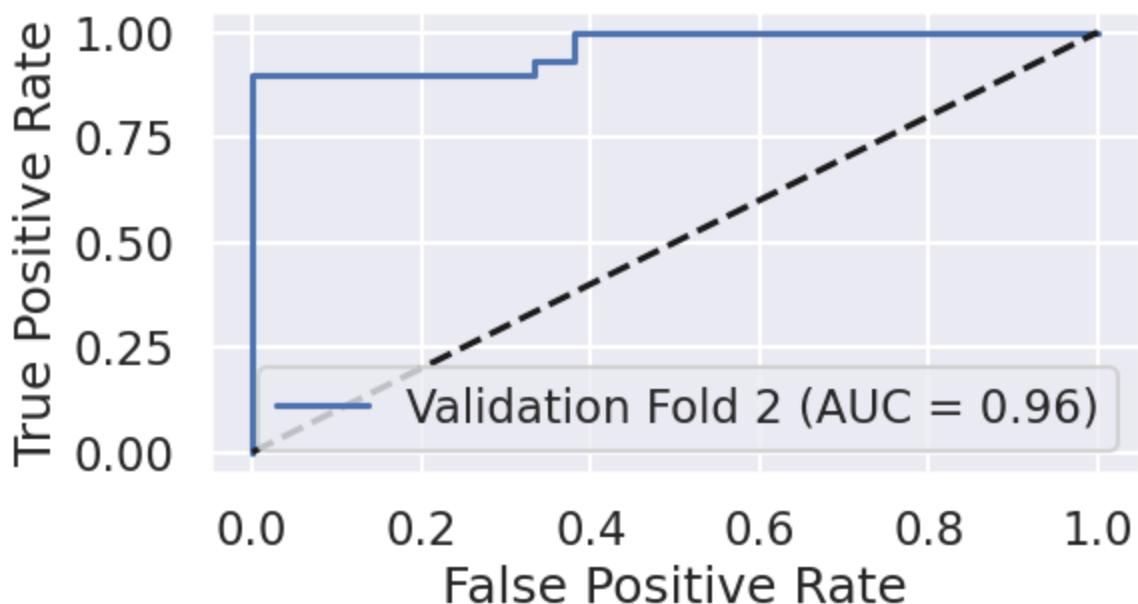
ROC Curve - Validation Fold 1



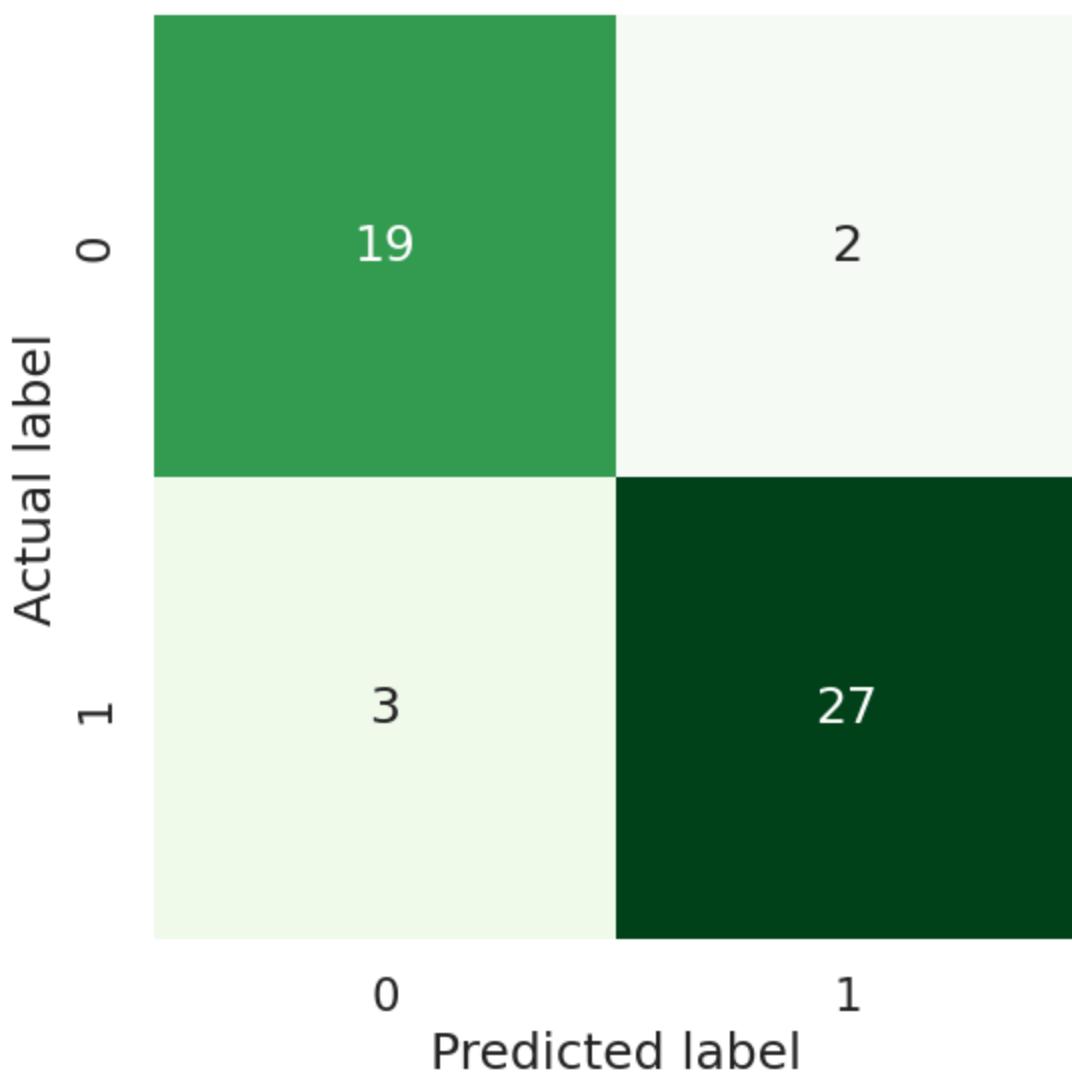
## Confusion Matrix - Validation Fold 1



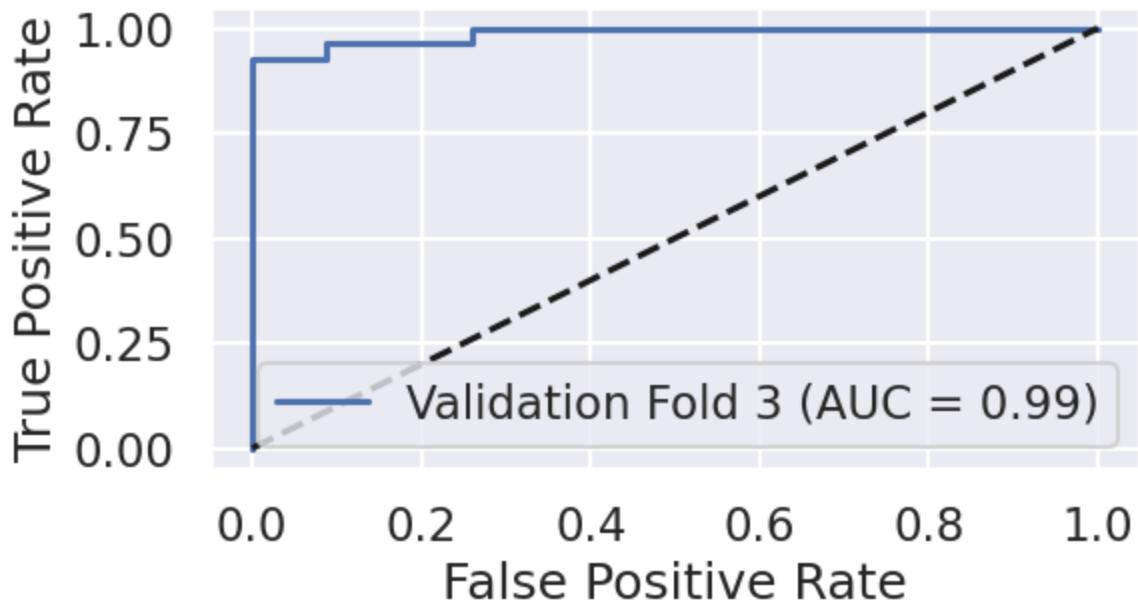
## ROC Curve - Validation Fold 2



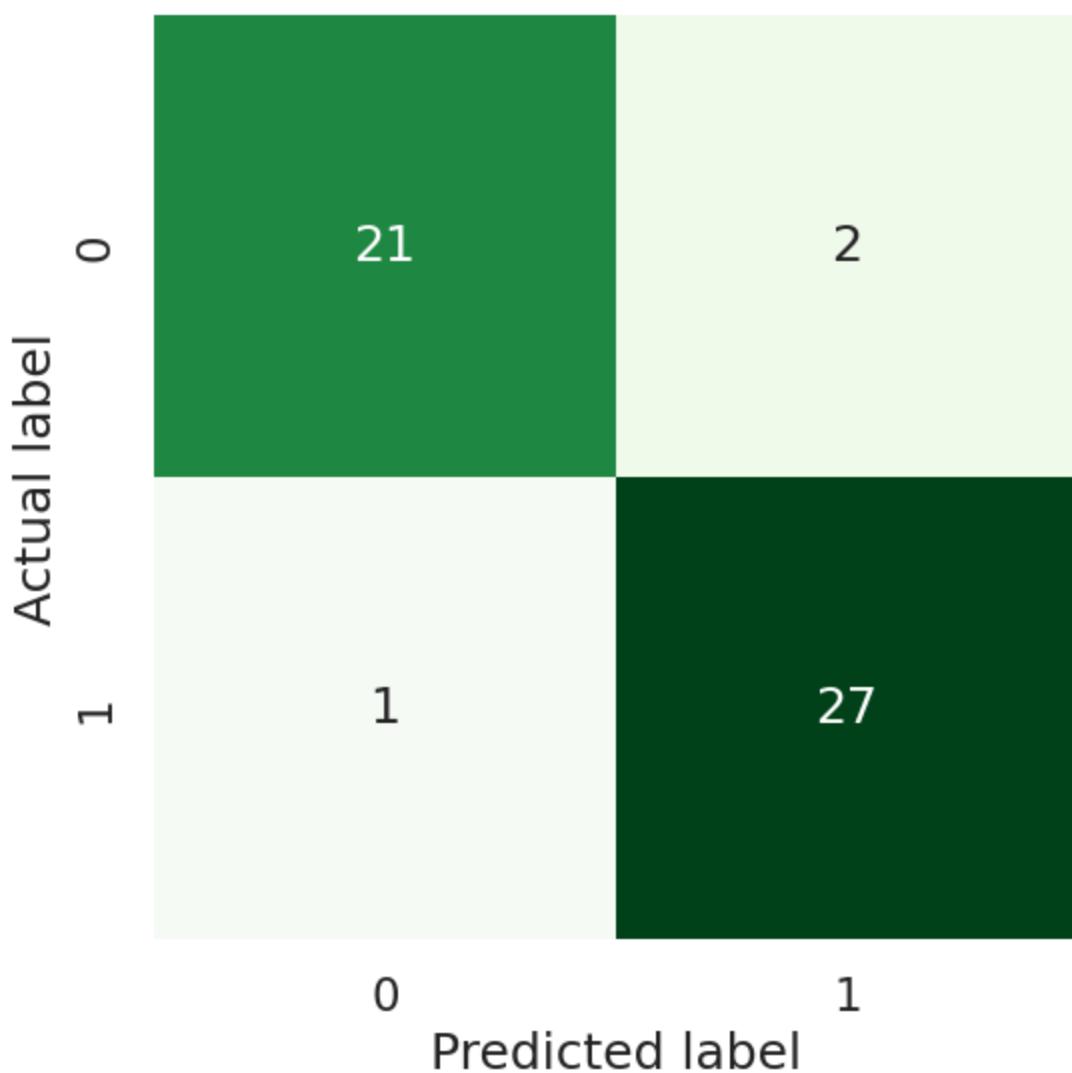
## Confusion Matrix - Validation Fold 2



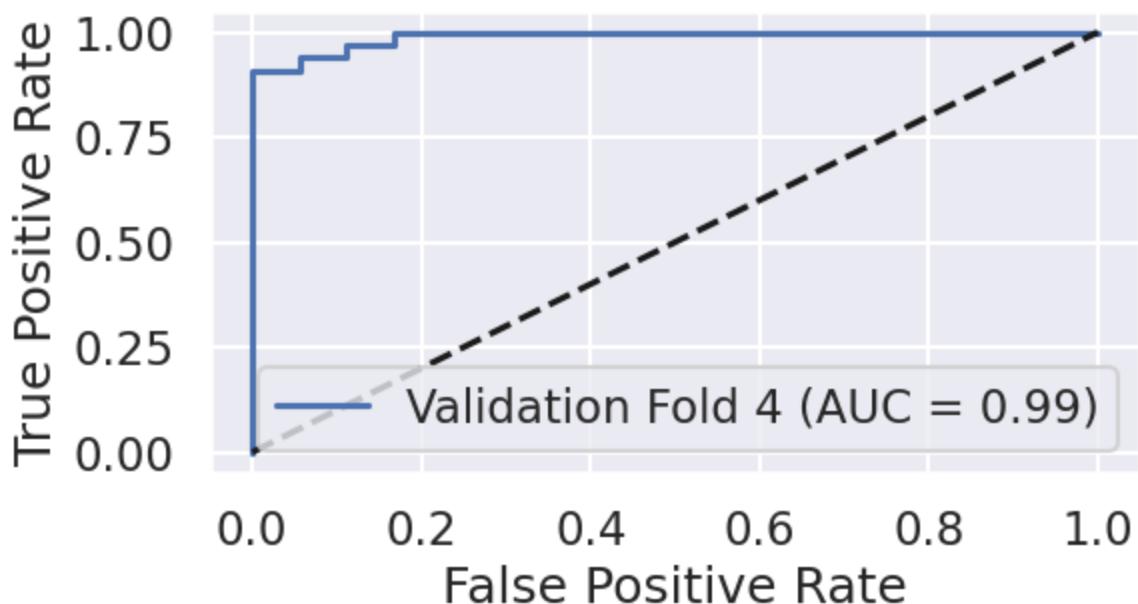
## ROC Curve - Validation Fold 3



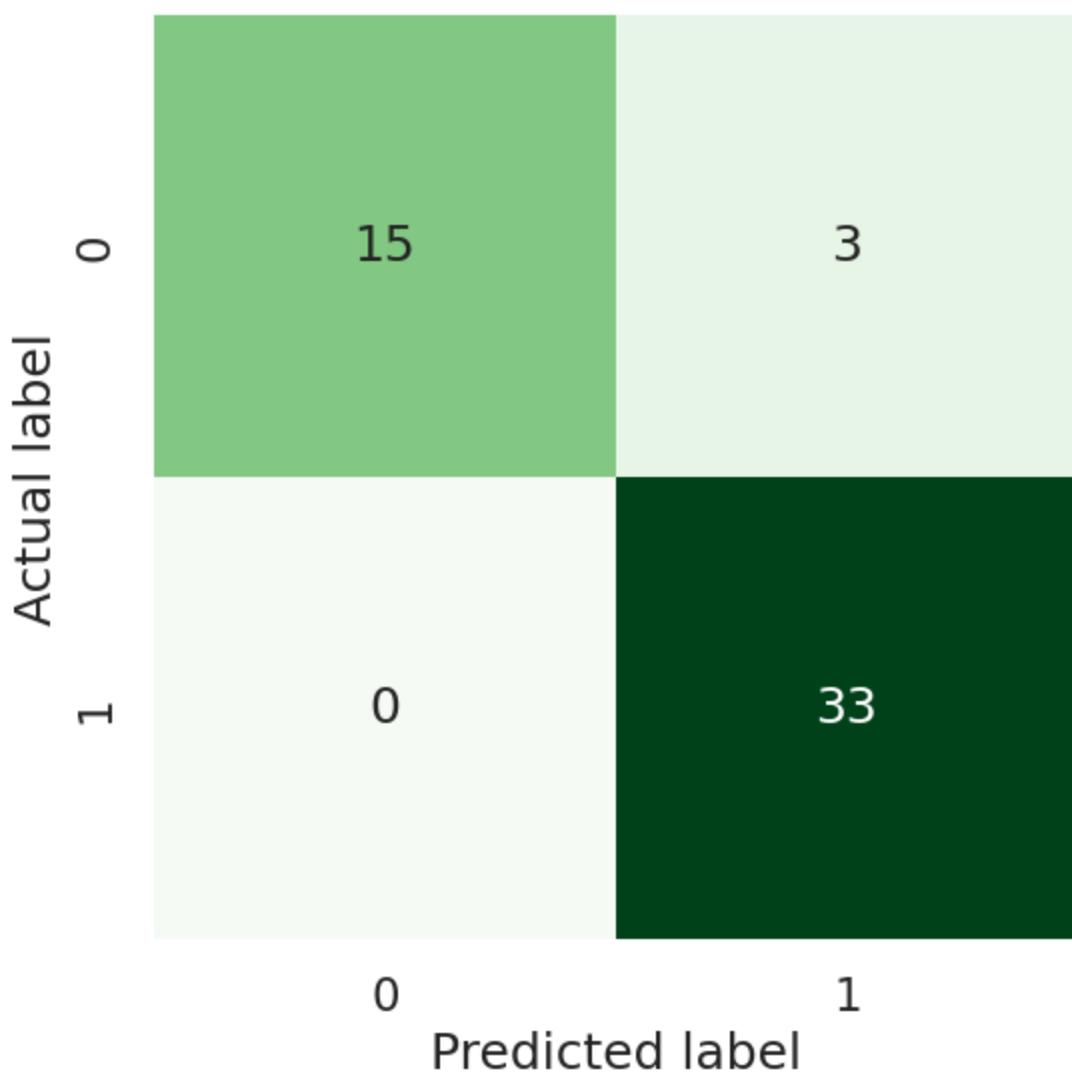
### Confusion Matrix - Validation Fold 3



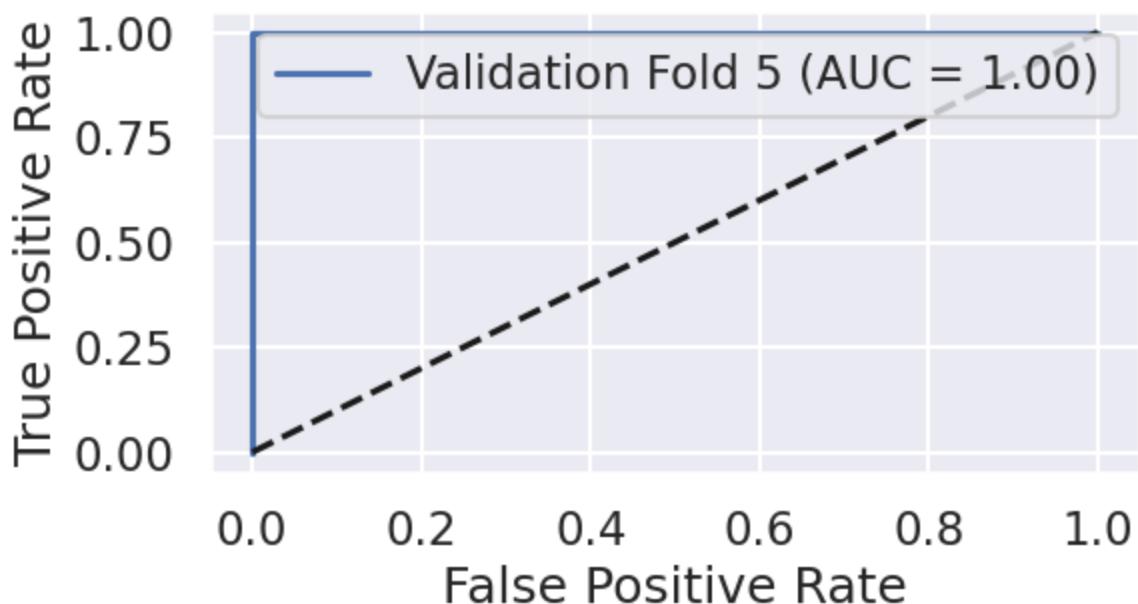
### ROC Curve - Validation Fold 4



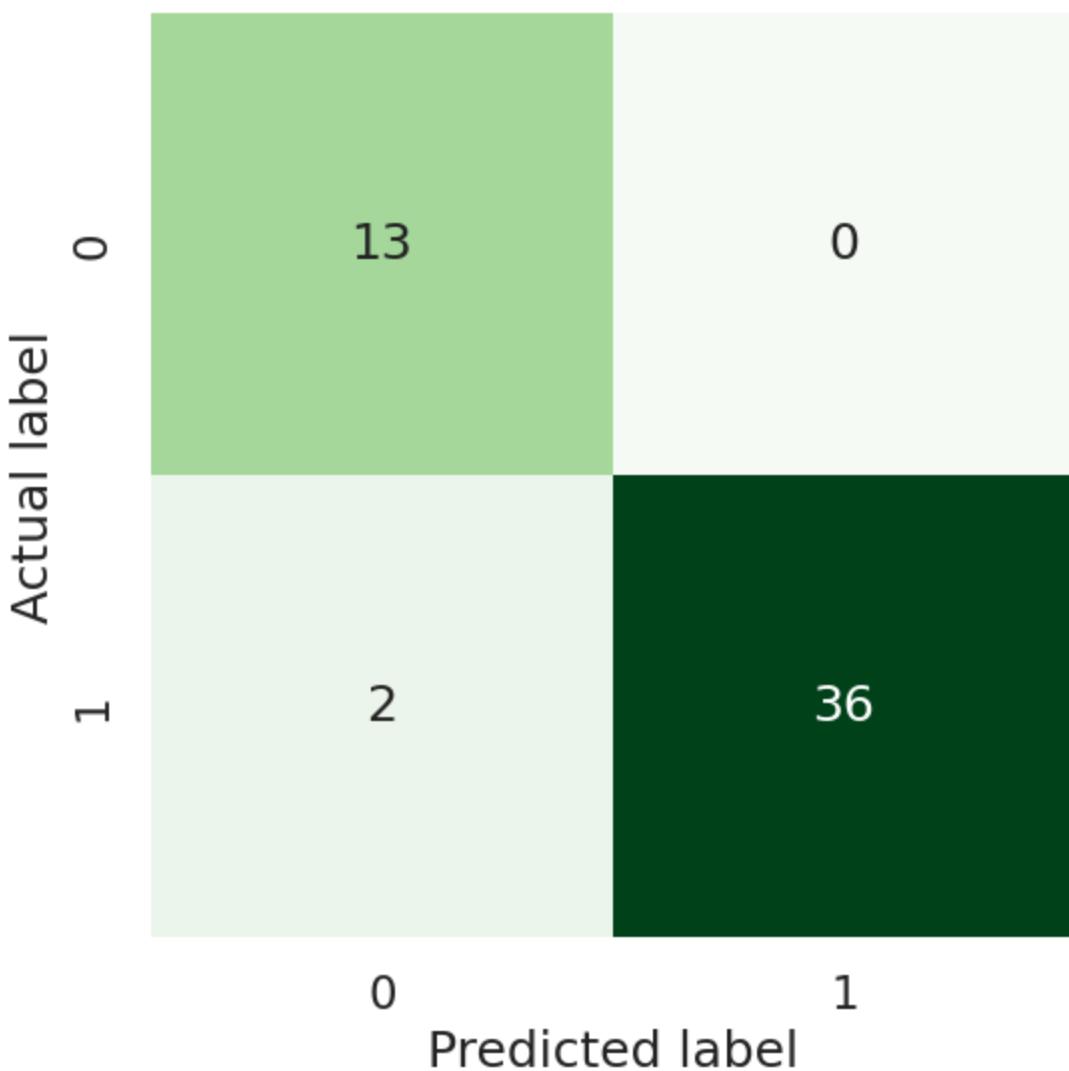
### Confusion Matrix - Validation Fold 4



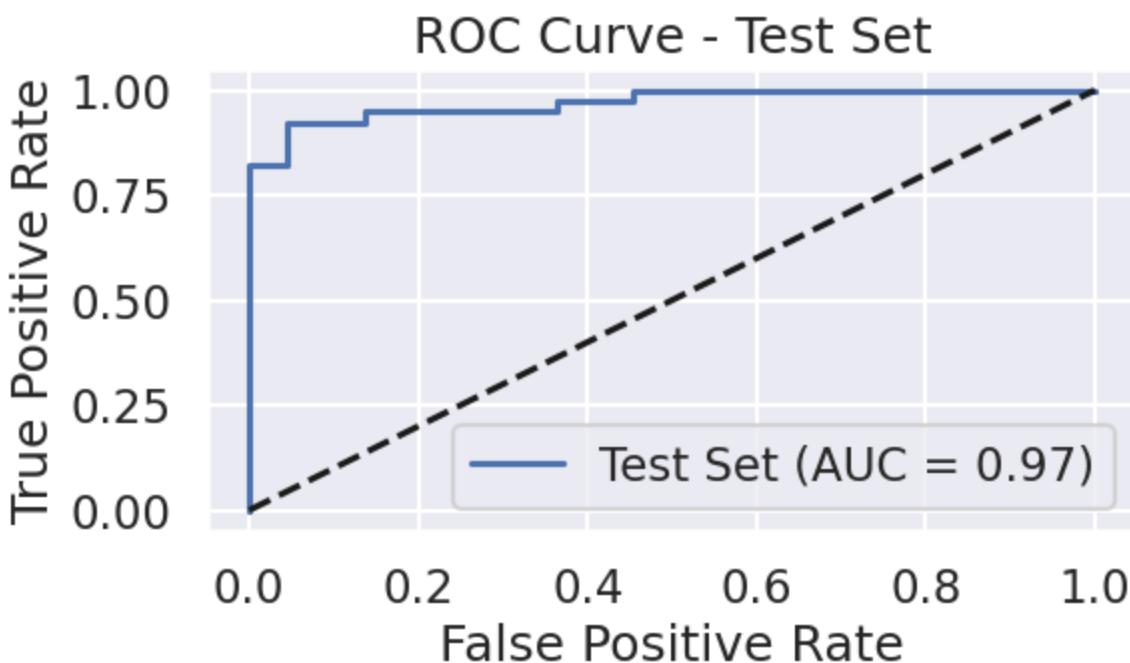
### ROC Curve - Validation Fold 5



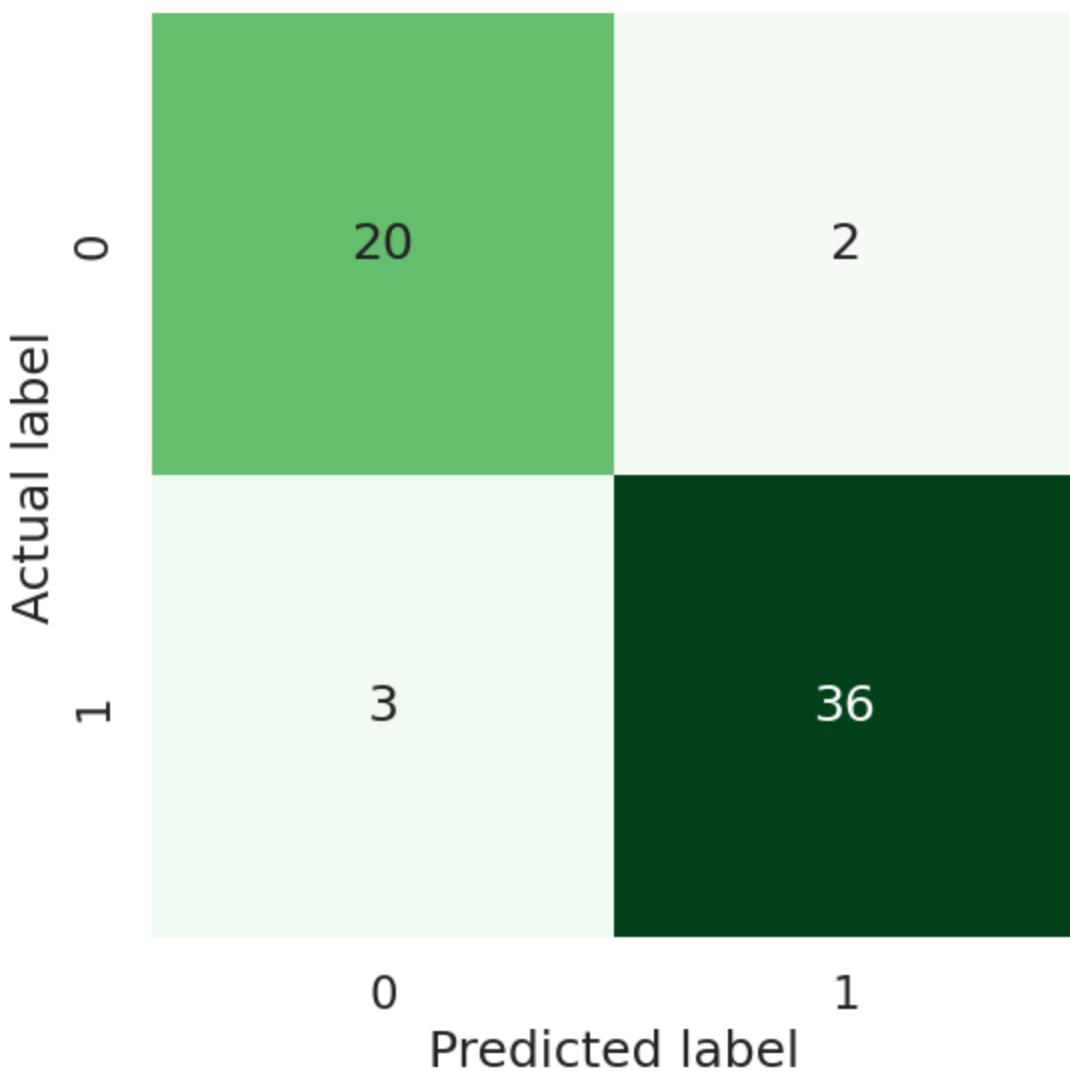
## Confusion Matrix - Validation Fold 5



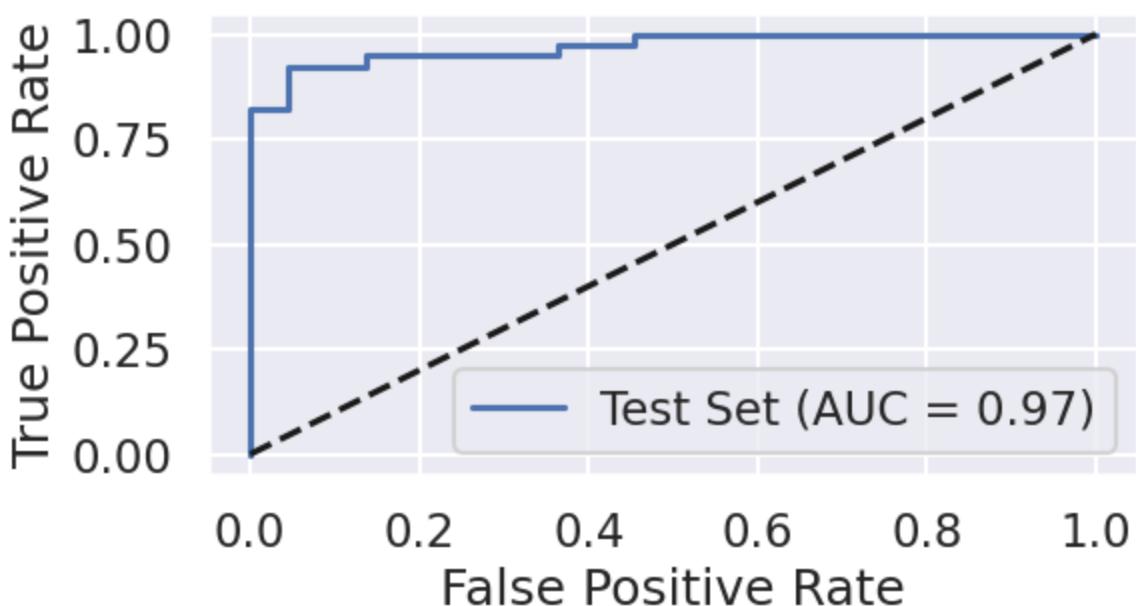
Average K-Folds results: {'Accuracy': 0.9254901960784314, 'ROC AUC': 0.9819241461520323, 'Sensitivity': 0.9485377236193934, 'Specificity': 0.8938641069075853}



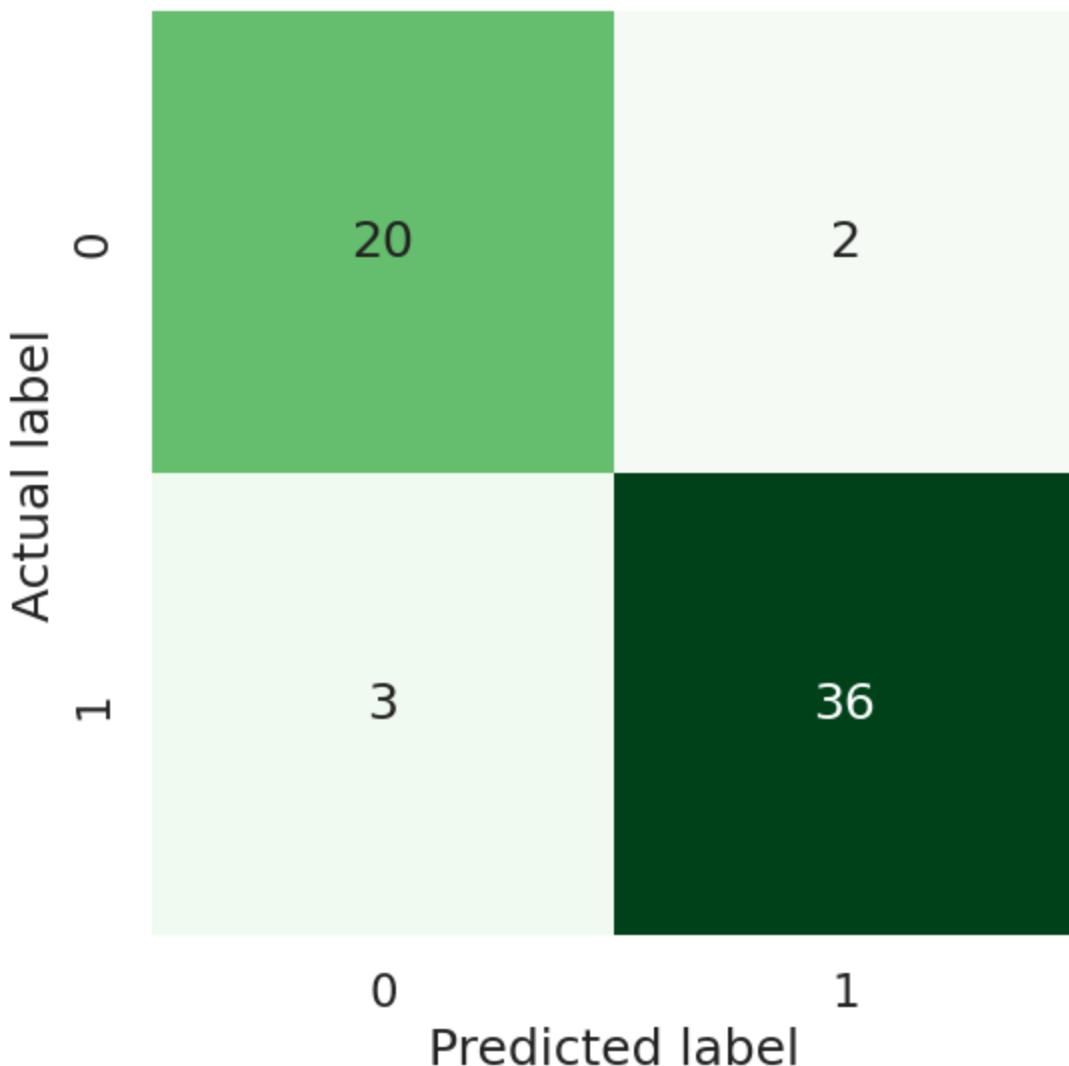
## Confusion Matrix - Test Set



## ROC Curve - Test Set



## Confusion Matrix - Test Set



	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.952941	0.992431	0.962025	0.938144
<b>Logistic Regression (CV)</b>	0.925490	0.981924	0.948538	0.893864
<b>Logistic Regression (Test)</b>	0.918033	0.970862	0.923077	0.909091

## Decision Tree

### Basic train of the Decision Tree Model

```
In [25]: def train_decision_tree(X_train, y_train, criterion='gini', max_depth=3, min_samples_s
dt_model = DecisionTreeClassifier(criterion=criterion, max_depth=max_depth,
min_samples_split=min_samples_split,
min_samples_leaf=min_samples_leaf,
random_state=42)
dt_model.fit(X_train, y_train)
return dt_model
```

## Train the Desicion Tree with 5-Folds Cross-Validation

```
In [26]: def train_decision_tree_with_cv(X_train, y_train, n_splits=5, criterion='gini', max_depth=None):
    model = DecisionTreeClassifier(criterion=criterion, max_depth=max_depth,
                                    min_samples_split=min_samples_split,
                                    min_samples_leaf=min_samples_leaf,
                                    random_state=42)
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)
    for train_index, val_index in kf.split(X_train):
        X_train_k, _ = X_train.iloc[train_index], X_train.iloc[val_index]
        y_train_k, _ = y_train.iloc[train_index], y_train.iloc[val_index]
        model.fit(X_train_k, y_train_k)

    return model
```

## Evaluate the Decision Tree Model

```
In [27]: def visualize_tree(dt_model, feature_names):
    plt.figure(figsize=(20, 10))
    plot_tree(dt_model, filled=True, feature_names=feature_names, class_names=['0', '1'])
    plt.title('Decision Tree Visualization')
    plt.show()

def evaluate_decision_tree(model, X, y, dataset_name):
    y_pred = model.predict(X)
    y_prob = model.predict_proba(X)[:, 1]

    accuracy = accuracy_score(y, y_pred)
    roc_auc = roc_auc_score(y, y_prob)
    cm = confusion_matrix(y, y_pred)
    sensitivity = cm[1, 1] / (cm[1, 1] + cm[1, 0])
    specificity = cm[0, 0] / (cm[0, 0] + cm[0, 1])

    plt.figure(figsize=(6, 3))
    fpr, tpr, _ = roc_curve(y, y_prob)
    plt.plot(fpr, tpr, label=f'{dataset_name} (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC Curve - {dataset_name}')
    plt.legend()
    plt.show()

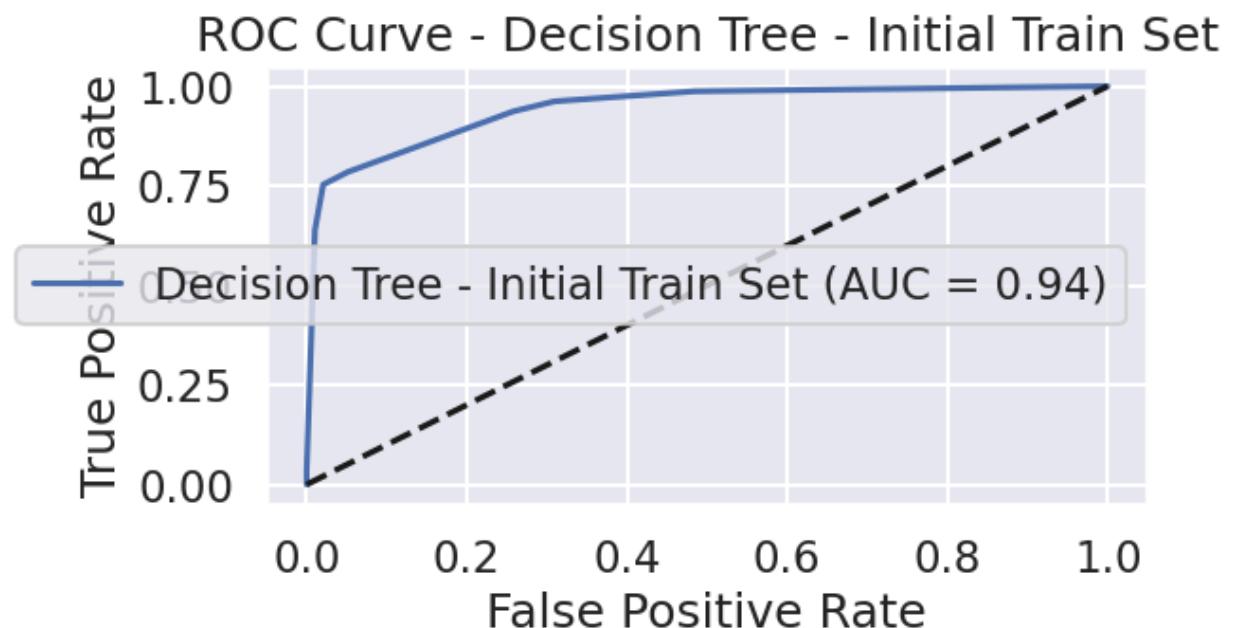
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Greens", cbar=False, square=True)
    plt.title(f'Confusion Matrix - {dataset_name}')
    plt.ylabel('Actual label')
    plt.xlabel('Predicted label')
    plt.show()

    return {'Accuracy': accuracy, 'ROC AUC': roc_auc, 'Sensitivity': sensitivity, 'Specificity': specificity}
```

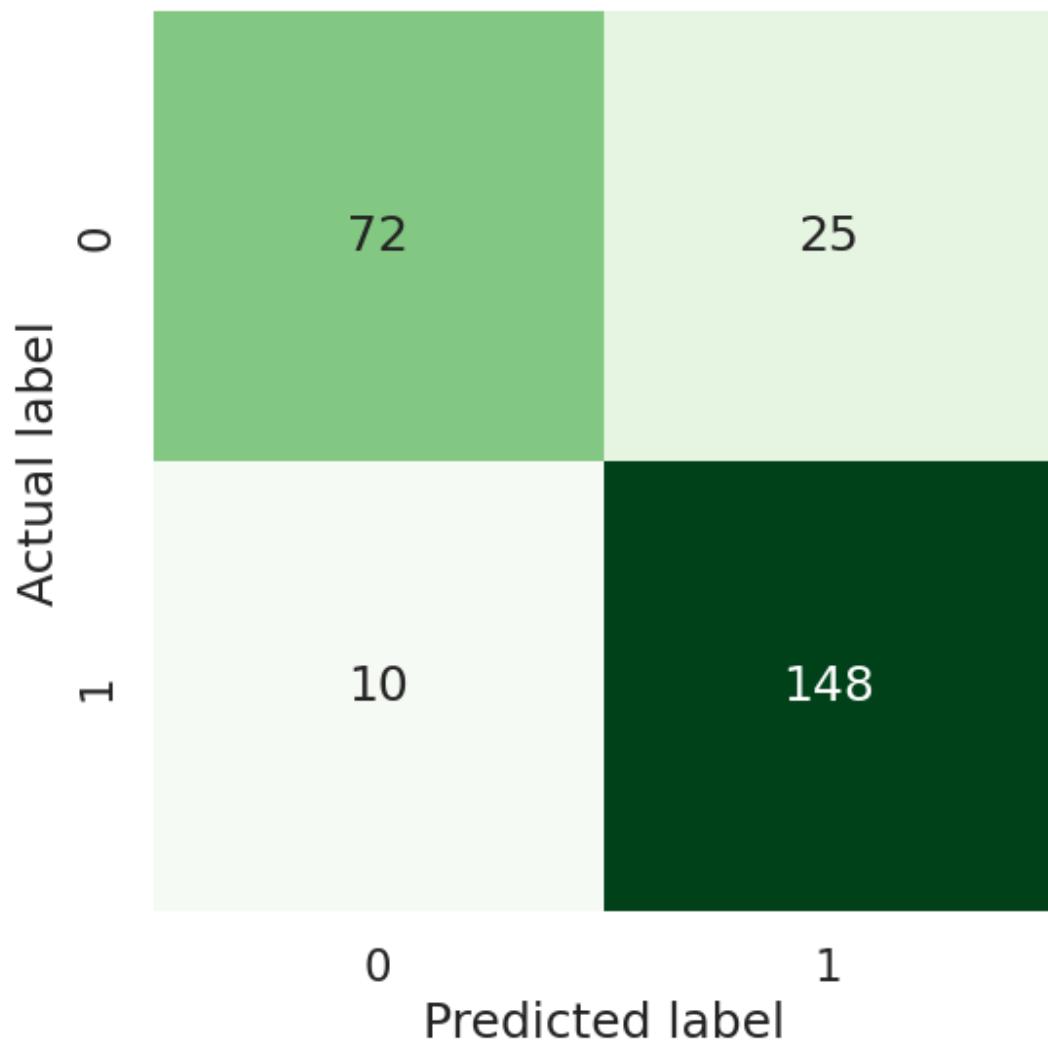
```
In [28]: dt_initial = train_decision_tree(X_train, y_train)
initial_train_results_dt = evaluate_decision_tree(dt_initial, X_train, y_train, "Decision Tree")
visualize_tree(dt_initial, X_train.columns)
```

```
results["Decision Tree - Initial Train Set"] = initial_train_results_dt

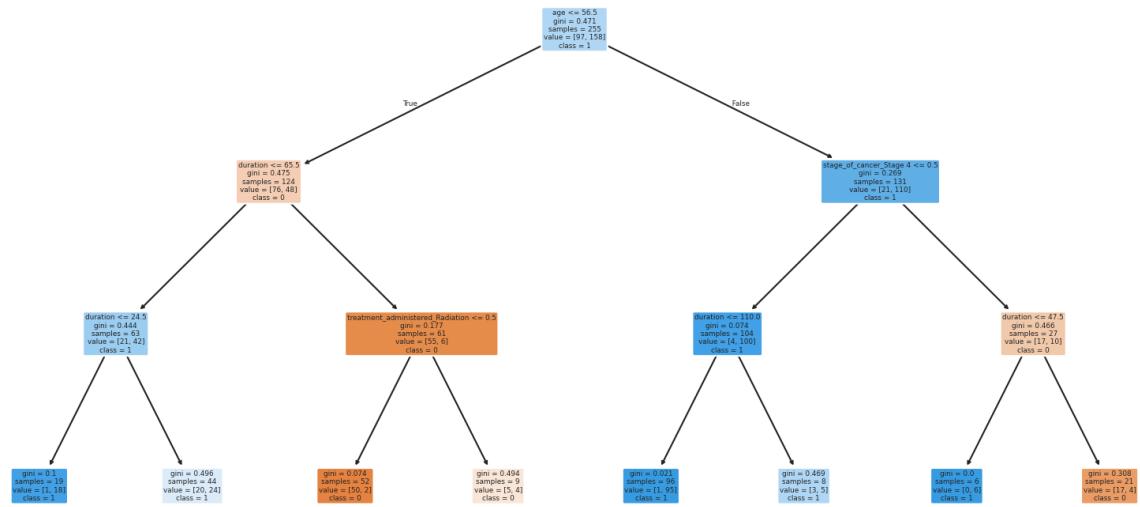
results_df = pd.DataFrame({
    'Logistic Regression': original_train_results,
    'Logistic Regression (CV)': cv_train_results,
    'Logistic Regression (Test)': test_results,
    'Decision Tree': initial_train_results_dt,
}).T
display(results_df)
```



# Confusion Matrix - Decision Tree - Initial Train Set



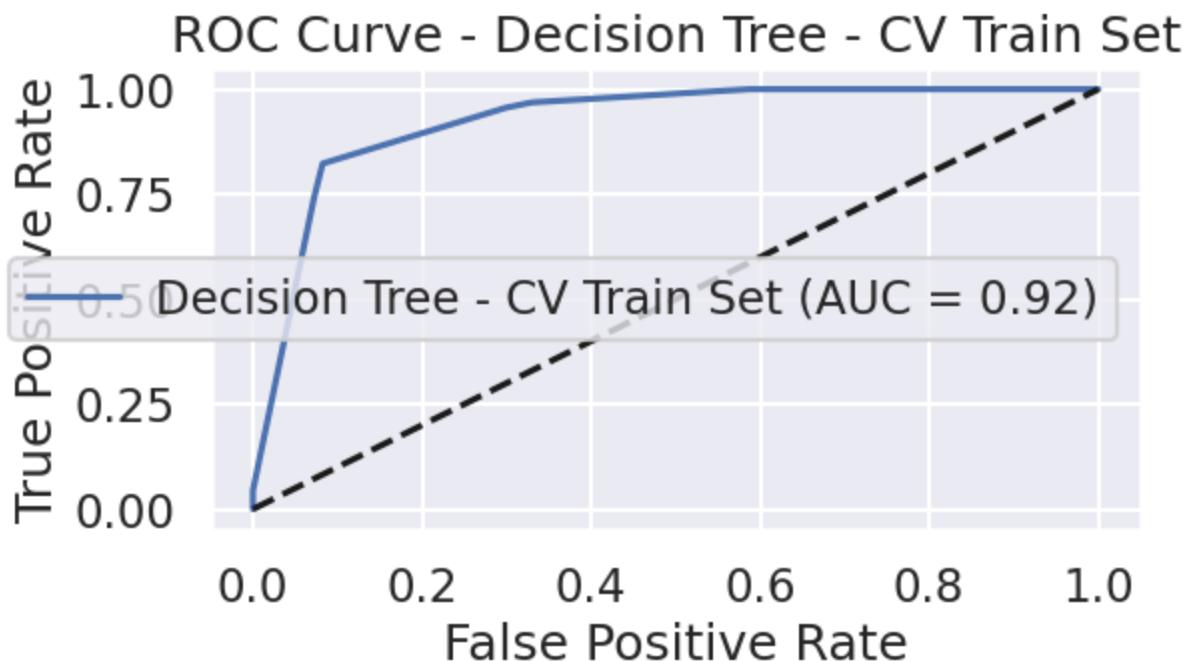
Decision Tree Visualization



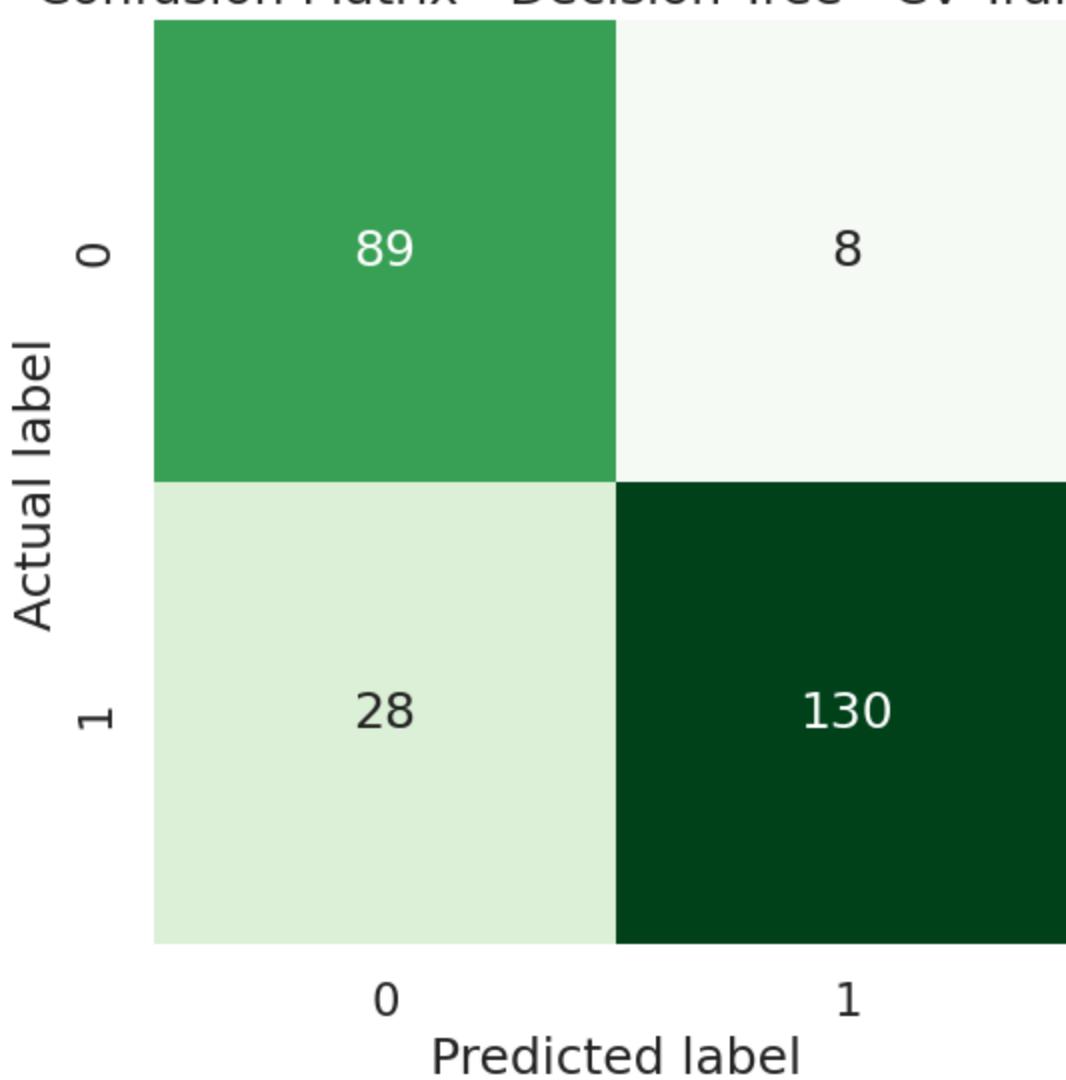
	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.952941	0.992431	0.962025	0.938144
<b>Logistic Regression (CV)</b>	0.925490	0.981924	0.948538	0.893864
<b>Logistic Regression (Test)</b>	0.918033	0.970862	0.923077	0.909091
<b>Decision Tree</b>	0.862745	0.943886	0.936709	0.742268

```
In [29]: dt_cv = train_decision_tree_with_cv(X_train, y_train)
cv_train_results_dt = evaluate_decision_tree(dt_cv, X_train, y_train, "Decision Tree - CV Train Set")
visualize_tree(dt_cv, X_train.columns)
results["Decision Tree - CV Train Set"] = cv_train_results_dt

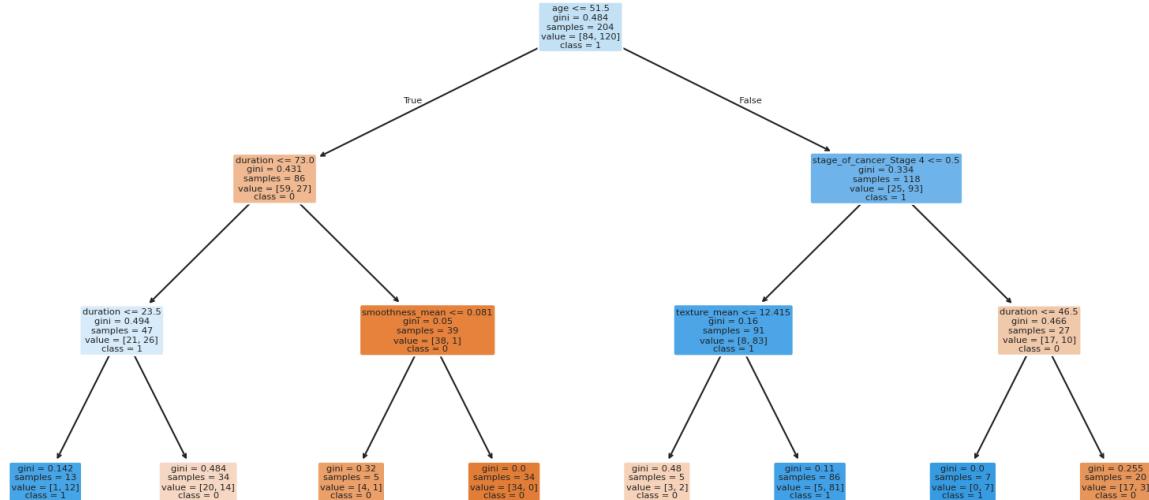
results_df = pd.DataFrame({
    'Logistic Regression': original_train_results,
    'Logistic Regression (CV)': cv_train_results,
    'Logistic Regression (Test)': test_results,
    'Decision Tree': initial_train_results_dt,
    'Decision Tree (CV)': cv_train_results_dt
}).T
display(results_df)
```



# Confusion Matrix - Decision Tree - CV Train Set



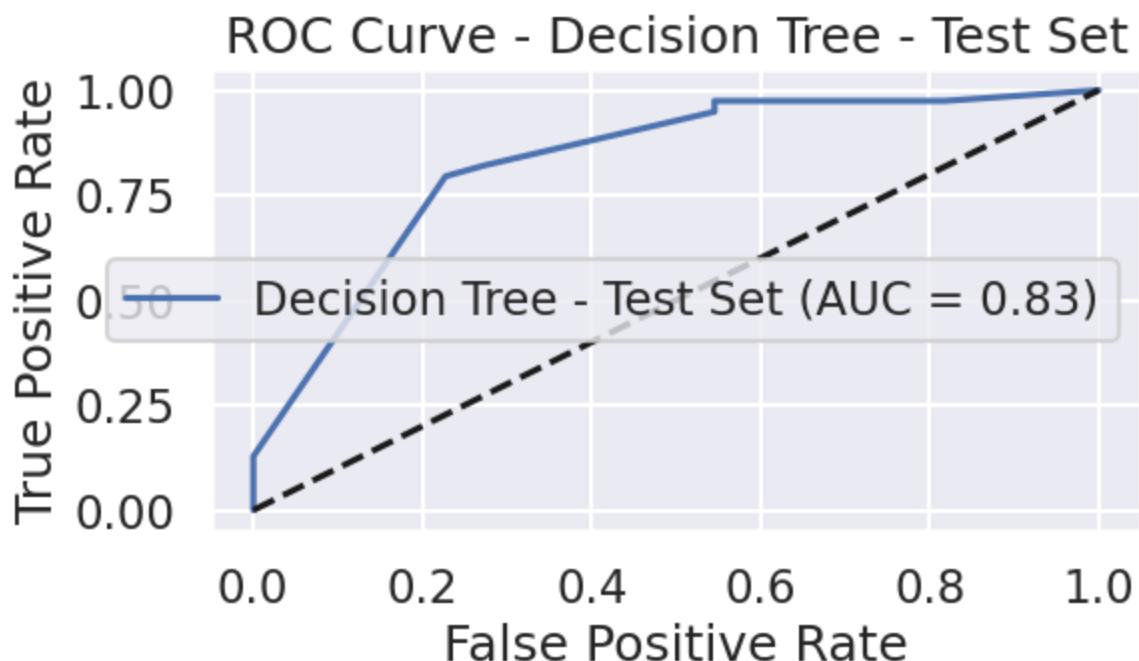
Decision Tree Visualization



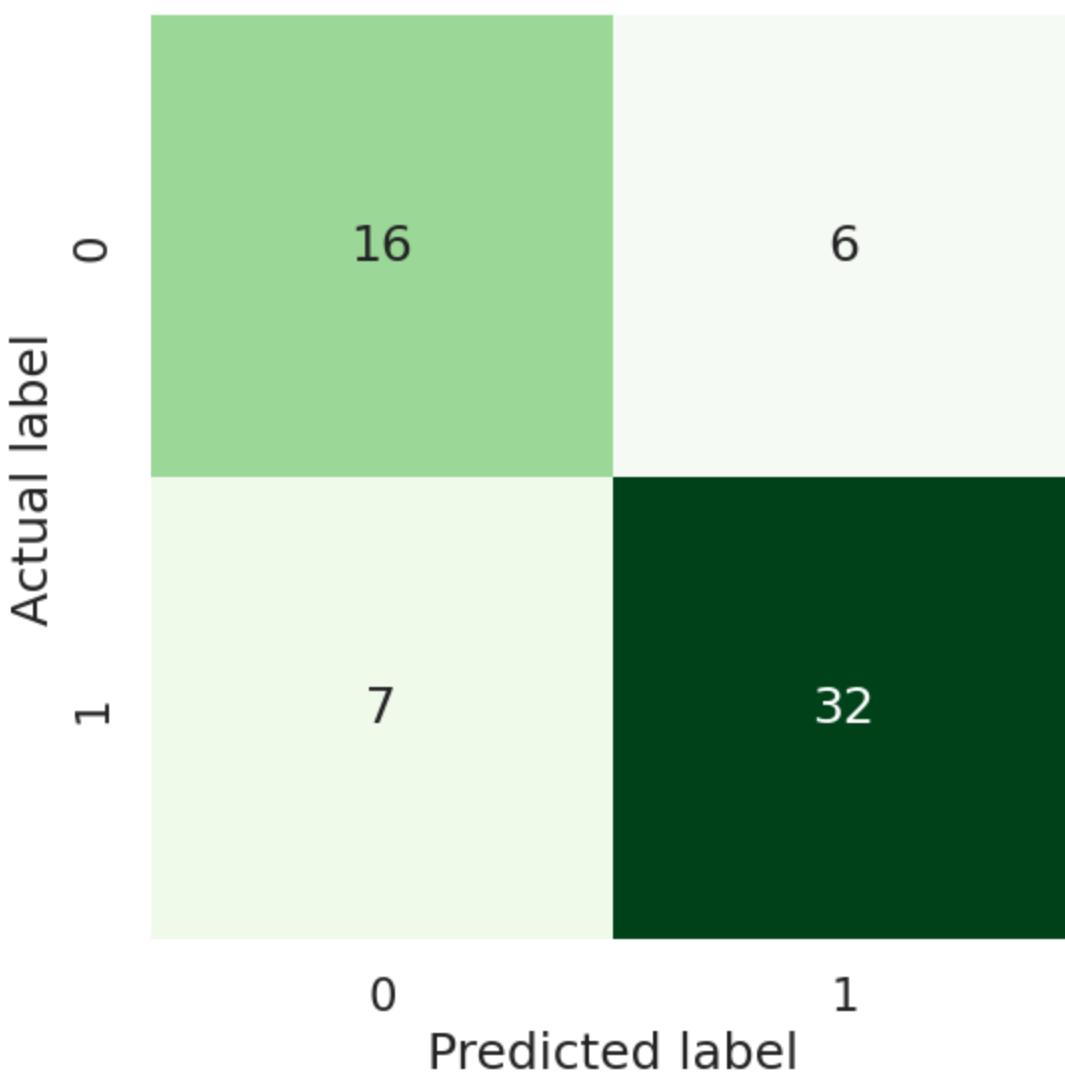
	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.952941	0.992431	0.962025	0.938144
<b>Logistic Regression (CV)</b>	0.925490	0.981924	0.948538	0.893864
<b>Logistic Regression (Test)</b>	0.918033	0.970862	0.923077	0.909091
<b>Decision Tree</b>	0.862745	0.943886	0.936709	0.742268
<b>Decision Tree (CV)</b>	0.858824	0.924670	0.822785	0.917526

```
In [30]: test_results_dt = evaluate_decision_tree(dt_cv, X_test, y_test, "Decision Tree - Test Set")
results["Decision Tree - Test Set"] = test_results_dt

results_df = pd.DataFrame({
    'Logistic Regression': original_train_results,
    'Logistic Regression (CV)': cv_train_results,
    'Logistic Regression (Test)': test_results,
    'Decision Tree': initial_train_results_dt,
    'Decision Tree (CV)': cv_train_results_dt,
    'Decision Tree (Test)': test_results_dt
}).T
display(results_df)
```



## Confusion Matrix - Decision Tree - Test Set



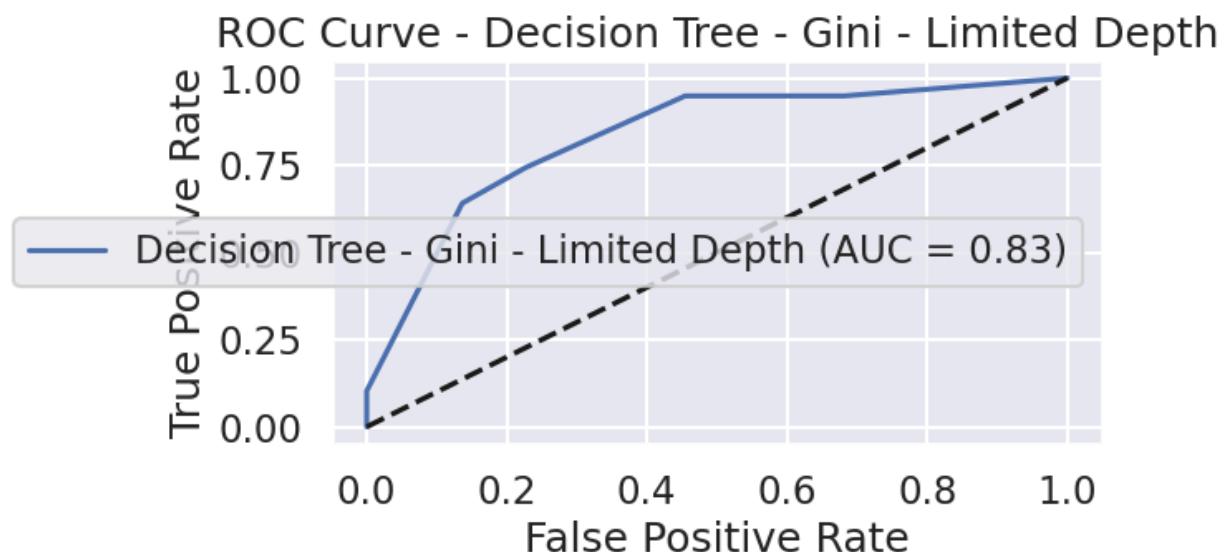
	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.952941	0.992431	0.962025	0.938144
<b>Logistic Regression (CV)</b>	0.925490	0.981924	0.948538	0.893864
<b>Logistic Regression (Test)</b>	0.918033	0.970862	0.923077	0.909091
<b>Decision Tree</b>	0.862745	0.943886	0.936709	0.742268
<b>Decision Tree (CV)</b>	0.858824	0.924670	0.822785	0.917526
<b>Decision Tree (Test)</b>	0.786885	0.828089	0.820513	0.727273

Simulating CHAID

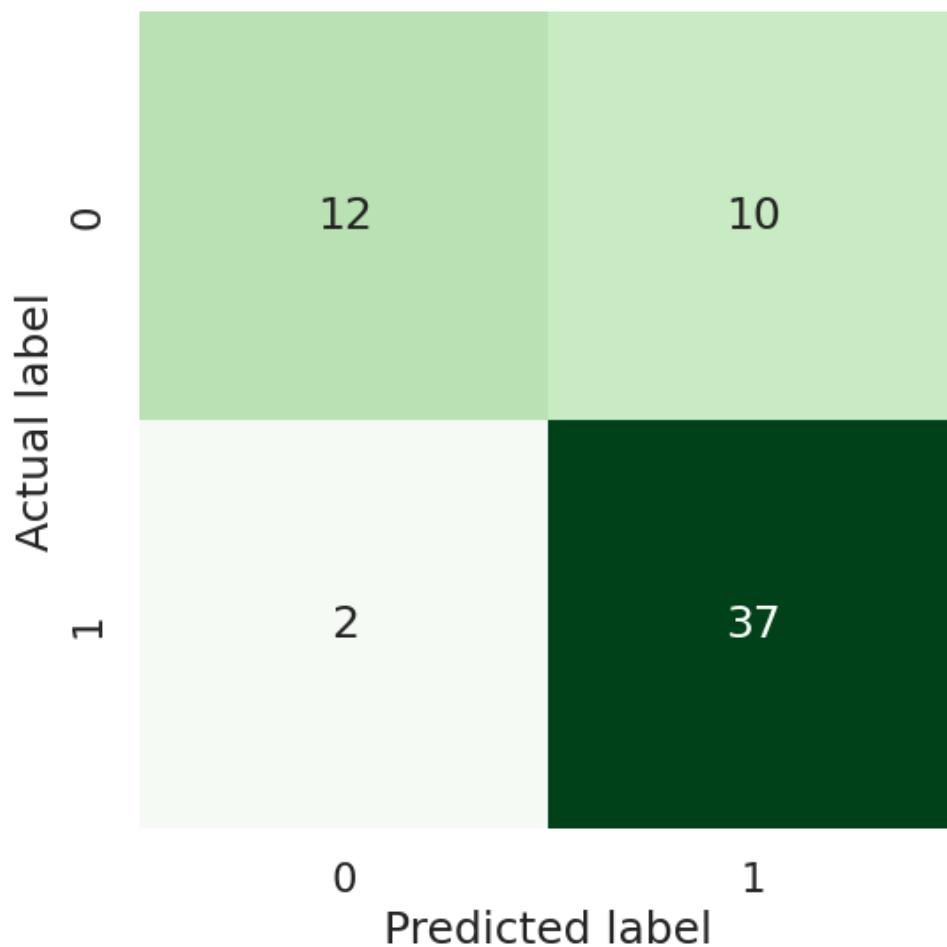
```
In [31]: dt_model_gini = train_decision_tree(X_train, y_train, criterion='gini', max_depth=3)
evaluate_decision_tree(dt_model_gini, X_test, y_test, 'Decision Tree - Gini - Limited')
visualize_tree(dt_model_gini, X_train.columns)

dt_model_entropy = train_decision_tree(X_train, y_train, criterion='entropy', max_dept
evaluate_decision_tree(dt_model_entropy, X_test, y_test, 'Decision Tree - Entropy - Li
visualize_tree(dt_model_entropy, X_train.columns)
```

```
results_df = pd.DataFrame({
    'Logistic Regression': original_train_results,
    'Logistic Regression (CV)': cv_train_results,
    'Logistic Regression (Test)': test_results,
    'Decision Tree': initial_train_results_dt,
    'Decision Tree (CV)': cv_train_results_dt,
    'Decision Tree (Test)': test_results_dt,
    'Decision Tree - Gini - Limited Depth': evaluate_decision_tree(dt_model_gini, X_te
    'Decision Tree - Entropy - Limited Depth': evaluate_decision_tree(dt_model_entropy
}).T
display(results_df)
```

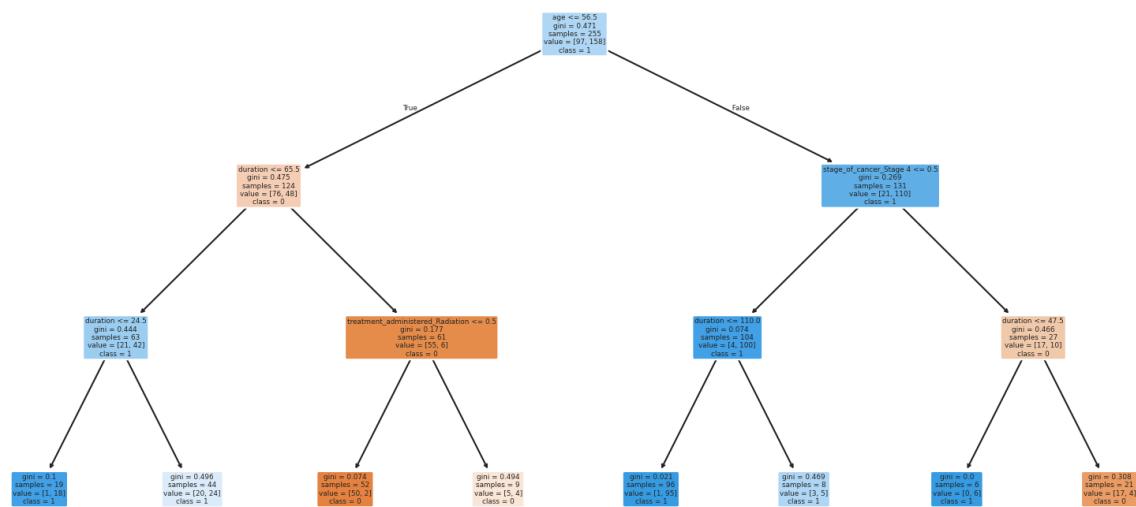


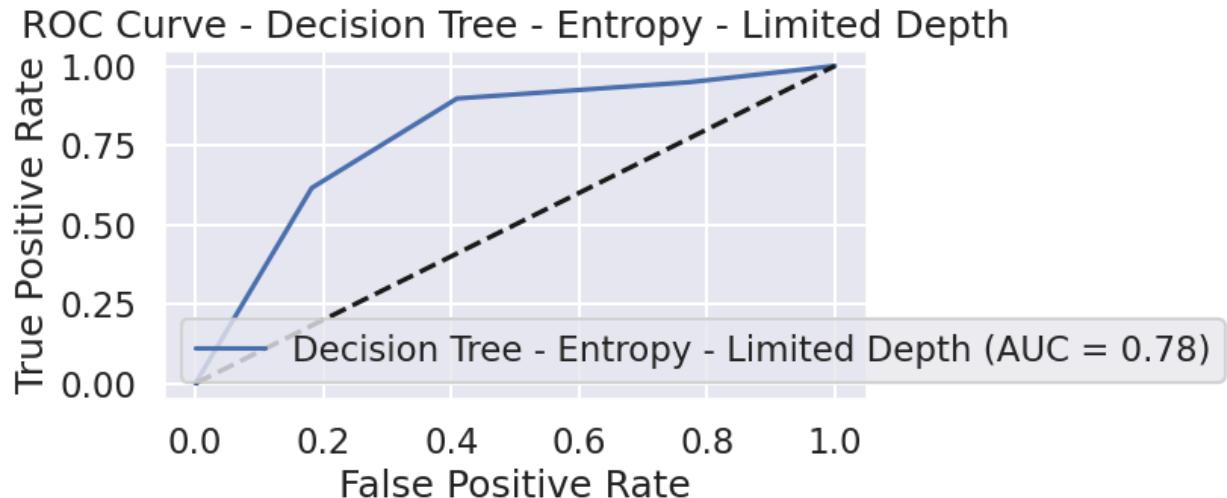
# Confusion Matrix - Decision Tree - Gini - Limited Depth



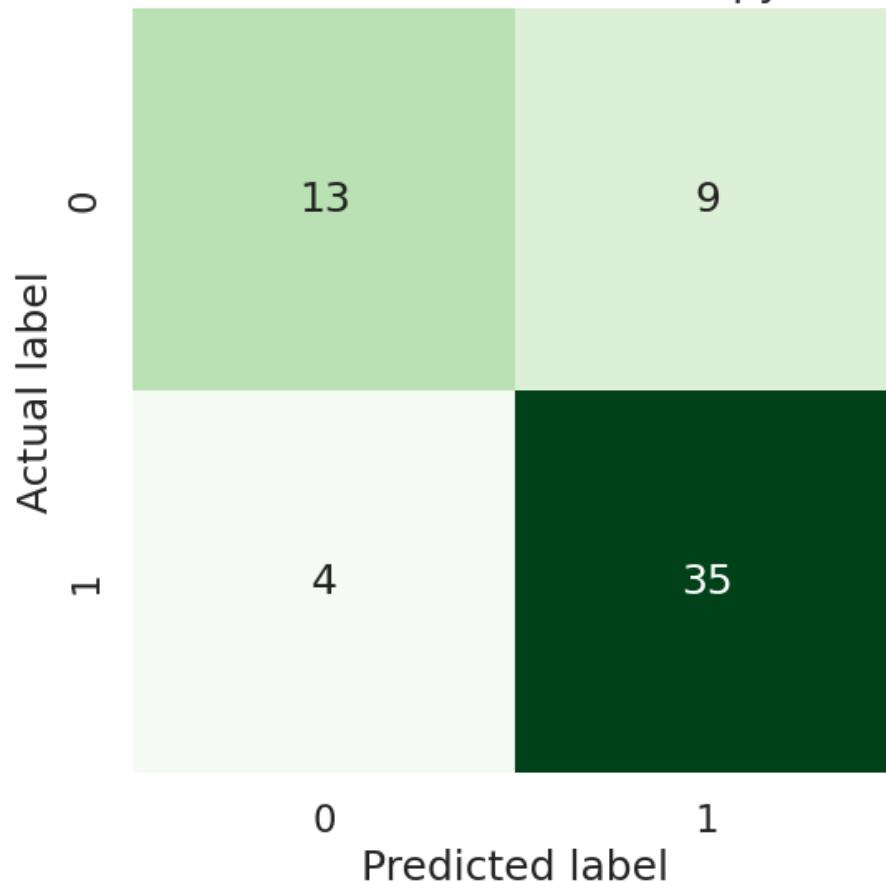
```
Out[31]: {'Accuracy': 0.8032786885245902,
'ROC AUC': 0.8315850815850816,
'Sensitivity': 0.9487179487179487,
'Specificity': 0.5454545454545454}
```

Decision Tree Visualization



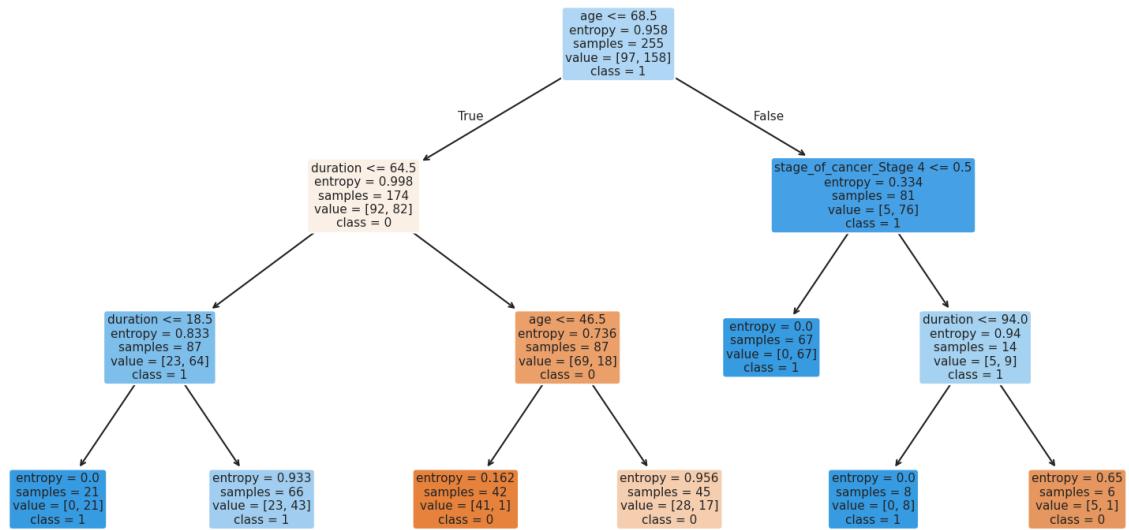


### Confusion Matrix - Decision Tree - Entropy - Limited Depth

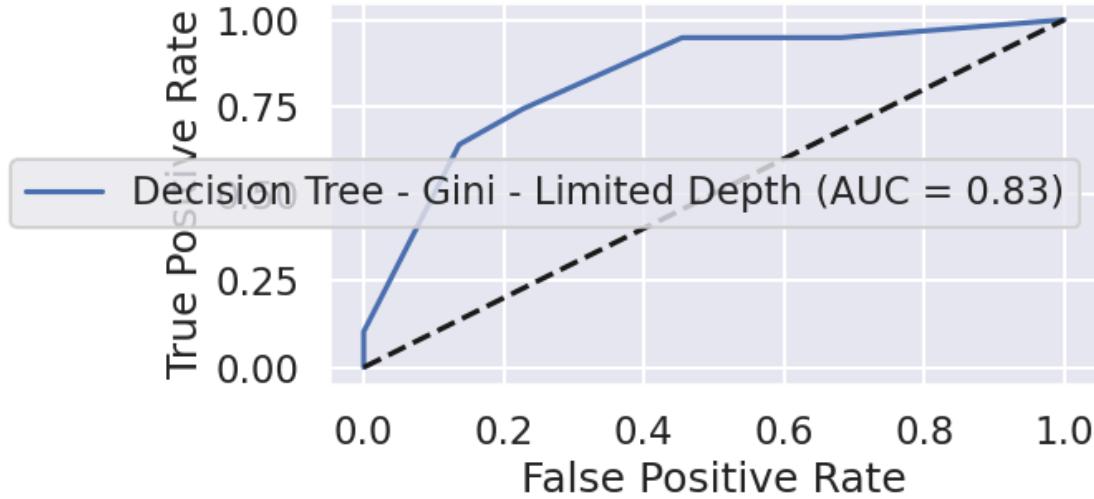


```
Out[31]: {'Accuracy': 0.7868852459016393,  
          'ROC AUC': 0.784965034965035,  
          'Sensitivity': 0.8974358974358975,  
          'Specificity': 0.5909090909090909}
```

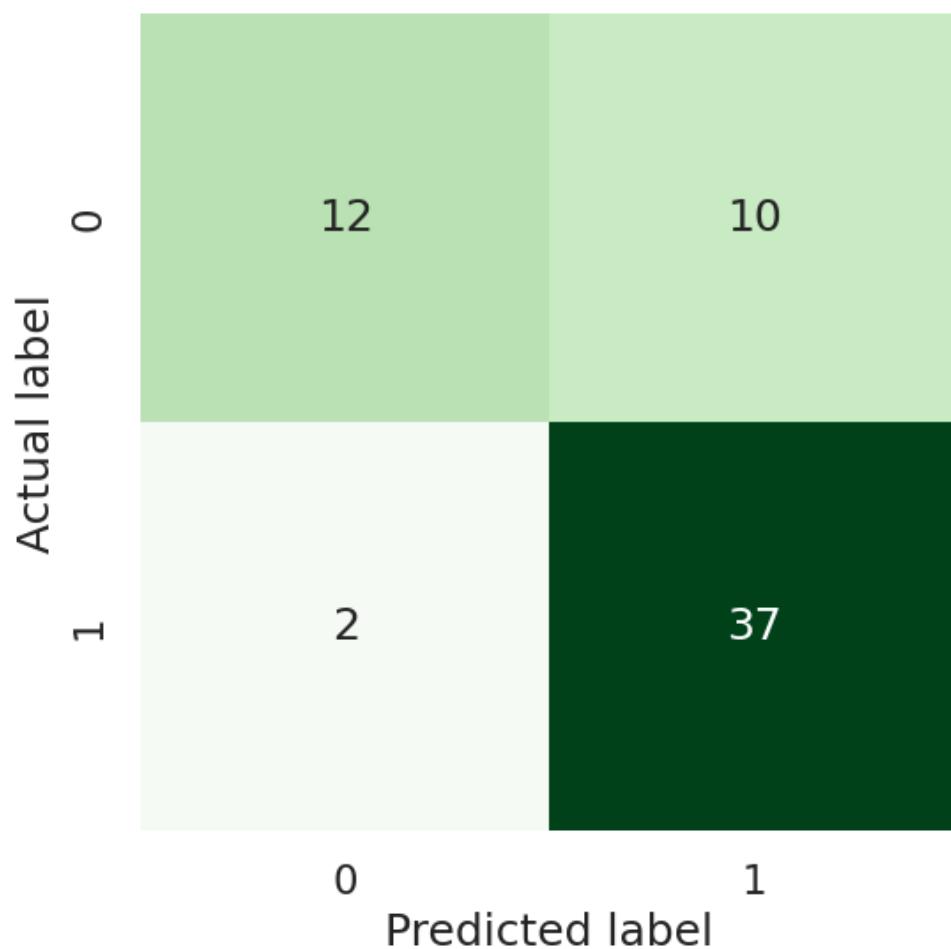
## Decision Tree Visualization



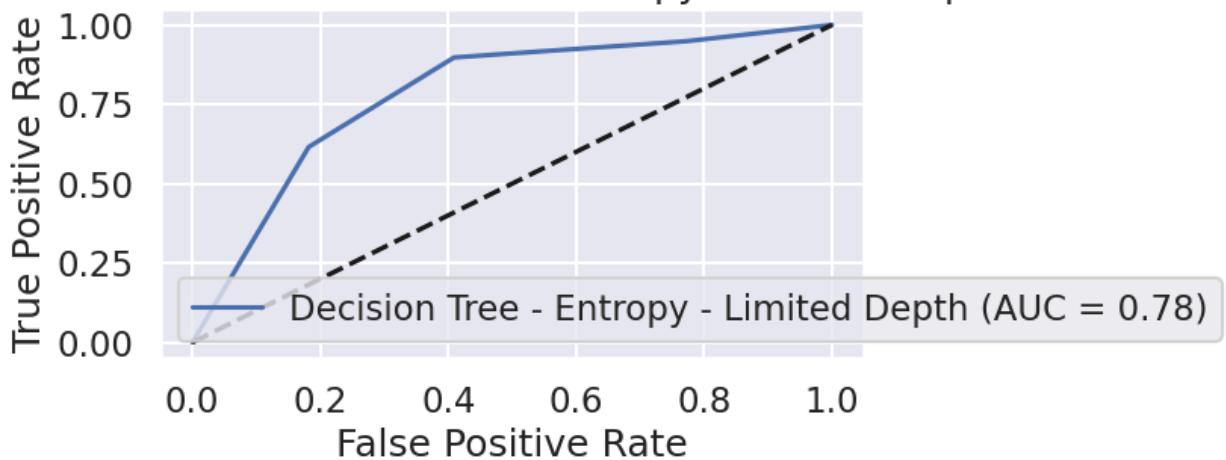
ROC Curve - Decision Tree - Gini - Limited Depth



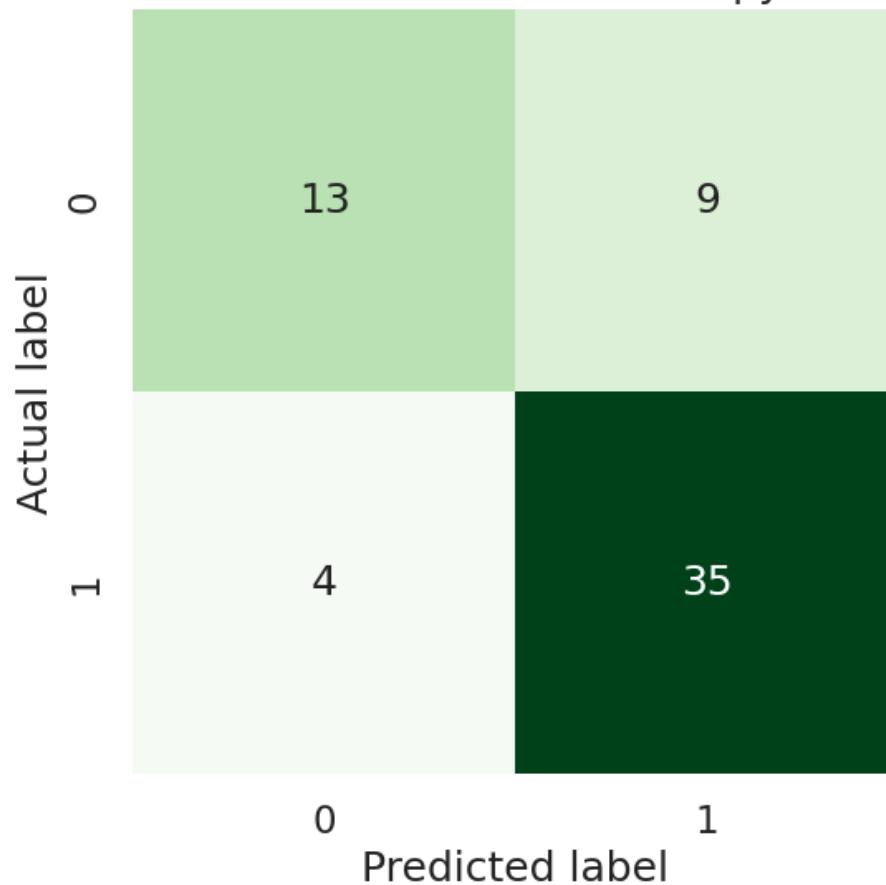
## Confusion Matrix - Decision Tree - Gini - Limited Depth



## ROC Curve - Decision Tree - Entropy - Limited Depth



## Confusion Matrix - Decision Tree - Entropy - Limited Depth



	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.952941	0.992431	0.962025	0.938144
<b>Logistic Regression (CV)</b>	0.925490	0.981924	0.948538	0.893864
<b>Logistic Regression (Test)</b>	0.918033	0.970862	0.923077	0.909091
<b>Decision Tree</b>	0.862745	0.943886	0.936709	0.742268
<b>Decision Tree (CV)</b>	0.858824	0.924670	0.822785	0.917526
<b>Decision Tree (Test)</b>	0.786885	0.828089	0.820513	0.727273
<b>Decision Tree - Gini - Limited Depth</b>	0.803279	0.831585	0.948718	0.545455
<b>Decision Tree - Entropy - Limited Depth</b>	0.786885	0.784965	0.897436	0.590909

# ANN (Artificial Neural Networks)

1. Model Setup: We'll use MLPClassifier from sklearn, which is a feedforward neural network implementation.
2. Network Architecture: We'll start with a simple architecture (one hidden layer) and gradually increase complexity if needed.
3. Training and Evaluation: The process will be similar to what we've done for Logistic Regression and Decision Trees, using ROC AUC, Sensitivity, and Specificity for evaluation.

Train the Neural Network (ANN)

```
In [32]: def train_neural_network(X_train, y_train, hidden_layer_sizes=(100,), activation='relu'):
    ann_model = MLPClassifier(hidden_layer_sizes=hidden_layer_sizes, activation=activation)
    ann_model.fit(X_train, y_train)
    return ann_model
```

## Train ANN with K-Folds Cross-Validation

```
In [33]: def train_neural_network_with_kfolds(X, y, n_splits=5, hidden_layer_sizes=(100,), activation='relu'):
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)
    models = []
    for train_index, val_index in kf.split(X):
        X_train, X_val = X.iloc[train_index], X.iloc[val_index]
        y_train, y_val = y.iloc[train_index], y.iloc[val_index]

        ann_model = MLPClassifier(hidden_layer_sizes=hidden_layer_sizes, activation=activation)
        ann_model.fit(X_train, y_train)

        models.append(ann_model)

    return models
```

Evaluate the Neural Network

```
In [34]: def evaluate_neural_network(ann_model, X_train, y_train, model_name):
    y_pred = ann_model.predict(X_train)
    y_prob = ann_model.predict_proba(X_train)[:, 1]

    roc_auc = roc_auc_score(y_train, y_prob)

    fpr, tpr, _ = roc_curve(y_train, y_prob)
    plt.figure()
    plt.plot(fpr, tpr, label=f'Neural Network (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve - Neural Network')
    plt.legend()
    plt.show()

    cm = confusion_matrix(y_train, y_pred)
    sensitivity = cm[1, 1] / (cm[1, 1] + cm[0, 1])
```

```
specificity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
accuracy = accuracy_score(y_train, y_pred)

sns.heatmap(cm, annot=True, fmt="d", cmap="Greens")
plt.title('Confusion Matrix - Neural Network')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

return {'Accuracy': accuracy, 'ROC AUC': roc_auc, 'Sensitivity': sensitivity, 'Spe
```

```
In [35]: ann_model = train_neural_network(X_train, y_train, hidden_layer_sizes=(100), activation='relu')
initial_train_results_ann = evaluate_neural_network(ann_model, X_train, y_train, 'Artificial Neural Network - Initial Train Set')
results["ANN - Initial Train Set"] = initial_train_results_ann

results_df = pd.DataFrame({
    'Logistic Regression': original_train_results,
    'Logistic Regression (CV)': cv_train_results,
    'Logistic Regression (Test)': test_results,
    'Decision Tree - Initial Train Set': initial_train_results_dt,
    'Decision Tree - CV Train Set': cv_train_results_dt,
    'Decision Tree - Test Set': test_results_dt,
    'Decision Tree - Gini - Limited Depth': evaluate_decision_tree(dt_model_gini, X_test),
    'Decision Tree - Entropy - Limited Depth': evaluate_decision_tree(dt_model_entropy, X_test),
    'ANN': initial_train_results_ann
}).T
display(results_df)
```

```
Iteration 1, loss = 4.02230292
Iteration 2, loss = 2.10454022
Iteration 3, loss = 0.86947080
Iteration 4, loss = 0.63834363
Iteration 5, loss = 0.92990311
Iteration 6, loss = 1.12074851
Iteration 7, loss = 1.02213408
Iteration 8, loss = 0.80638496
Iteration 9, loss = 0.64397050
Iteration 10, loss = 0.61286203
Iteration 11, loss = 0.65718838
Iteration 12, loss = 0.70880343
Iteration 13, loss = 0.71580951
Iteration 14, loss = 0.67234684
Iteration 15, loss = 0.62725891
Iteration 16, loss = 0.58560556
Iteration 17, loss = 0.57560980
Iteration 18, loss = 0.57066244
Iteration 19, loss = 0.56298485
Iteration 20, loss = 0.54736251
Iteration 21, loss = 0.52878874
Iteration 22, loss = 0.51326551
Iteration 23, loss = 0.49312804
Iteration 24, loss = 0.48197188
Iteration 25, loss = 0.47281459
Iteration 26, loss = 0.46184911
Iteration 27, loss = 0.44970322
Iteration 28, loss = 0.43596365
Iteration 29, loss = 0.41821720
Iteration 30, loss = 0.40762723
Iteration 31, loss = 0.39689920
Iteration 32, loss = 0.38930702
Iteration 33, loss = 0.38296267
Iteration 34, loss = 0.37670067
Iteration 35, loss = 0.36881320
Iteration 36, loss = 0.36026601
Iteration 37, loss = 0.36009198
Iteration 38, loss = 0.35783499
Iteration 39, loss = 0.34957359
Iteration 40, loss = 0.34125632
Iteration 41, loss = 0.33902456
Iteration 42, loss = 0.33582467
Iteration 43, loss = 0.32950126
Iteration 44, loss = 0.32532578
Iteration 45, loss = 0.32451970
Iteration 46, loss = 0.32281269
Iteration 47, loss = 0.32066127
Iteration 48, loss = 0.31576226
Iteration 49, loss = 0.31280710
Iteration 50, loss = 0.31048232
Iteration 51, loss = 0.30959967
Iteration 52, loss = 0.30675096
Iteration 53, loss = 0.30470100
Iteration 54, loss = 0.30382580
Iteration 55, loss = 0.30156411
Iteration 56, loss = 0.30085709
Iteration 57, loss = 0.29825943
Iteration 58, loss = 0.29573218
Iteration 59, loss = 0.29429188
Iteration 60, loss = 0.29269111
```

Iteration 61, loss = 0.29140750  
Iteration 62, loss = 0.29282380  
Iteration 63, loss = 0.29039945  
Iteration 64, loss = 0.28653339  
Iteration 65, loss = 0.28746691  
Iteration 66, loss = 0.28743537  
Iteration 67, loss = 0.28227269  
Iteration 68, loss = 0.28445777  
Iteration 69, loss = 0.28291903  
Iteration 70, loss = 0.27880415  
Iteration 71, loss = 0.27871387  
Iteration 72, loss = 0.27958179  
Iteration 73, loss = 0.27518067  
Iteration 74, loss = 0.27381807  
Iteration 75, loss = 0.27147686  
Iteration 76, loss = 0.27001834  
Iteration 77, loss = 0.26881486  
Iteration 78, loss = 0.26853387  
Iteration 79, loss = 0.26798311  
Iteration 80, loss = 0.26576272  
Iteration 81, loss = 0.26483172  
Iteration 82, loss = 0.26545454  
Iteration 83, loss = 0.26324829  
Iteration 84, loss = 0.26074457  
Iteration 85, loss = 0.26010622  
Iteration 86, loss = 0.25794394  
Iteration 87, loss = 0.25850639  
Iteration 88, loss = 0.25669207  
Iteration 89, loss = 0.25484294  
Iteration 90, loss = 0.25658803  
Iteration 91, loss = 0.25209274  
Iteration 92, loss = 0.25700625  
Iteration 93, loss = 0.25482831  
Iteration 94, loss = 0.24928817  
Iteration 95, loss = 0.25103163  
Iteration 96, loss = 0.24688681  
Iteration 97, loss = 0.24745477  
Iteration 98, loss = 0.25403423  
Iteration 99, loss = 0.24920378  
Iteration 100, loss = 0.24413673  
Iteration 101, loss = 0.24500362  
Iteration 102, loss = 0.24273261  
Iteration 103, loss = 0.24267903  
Iteration 104, loss = 0.24122914  
Iteration 105, loss = 0.23857052  
Iteration 106, loss = 0.23781333  
Iteration 107, loss = 0.23669402  
Iteration 108, loss = 0.23579680  
Iteration 109, loss = 0.23448541  
Iteration 110, loss = 0.23481622  
Iteration 111, loss = 0.23363717  
Iteration 112, loss = 0.23239794  
Iteration 113, loss = 0.23182923  
Iteration 114, loss = 0.23044262  
Iteration 115, loss = 0.22943965  
Iteration 116, loss = 0.23016530  
Iteration 117, loss = 0.22802171  
Iteration 118, loss = 0.22735506  
Iteration 119, loss = 0.23126033  
Iteration 120, loss = 0.22780703

Iteration 121, loss = 0.22509048  
Iteration 122, loss = 0.22728015  
Iteration 123, loss = 0.22363601  
Iteration 124, loss = 0.22647265  
Iteration 125, loss = 0.22786131  
Iteration 126, loss = 0.22139982  
Iteration 127, loss = 0.22084346  
Iteration 128, loss = 0.21927932  
Iteration 129, loss = 0.21913891  
Iteration 130, loss = 0.21673575  
Iteration 131, loss = 0.21834780  
Iteration 132, loss = 0.22036976  
Iteration 133, loss = 0.21580011  
Iteration 134, loss = 0.21415451  
Iteration 135, loss = 0.21484682  
Iteration 136, loss = 0.21173361  
Iteration 137, loss = 0.21456477  
Iteration 138, loss = 0.21810526  
Iteration 139, loss = 0.21252006  
Iteration 140, loss = 0.21039078  
Iteration 141, loss = 0.21035409  
Iteration 142, loss = 0.20973249  
Iteration 143, loss = 0.20748699  
Iteration 144, loss = 0.20696330  
Iteration 145, loss = 0.20592567  
Iteration 146, loss = 0.20578461  
Iteration 147, loss = 0.20528378  
Iteration 148, loss = 0.20517077  
Iteration 149, loss = 0.20504235  
Iteration 150, loss = 0.20473887  
Iteration 151, loss = 0.20418614  
Iteration 152, loss = 0.20357433  
Iteration 153, loss = 0.20130895  
Iteration 154, loss = 0.20459066  
Iteration 155, loss = 0.20086647  
Iteration 156, loss = 0.20362074  
Iteration 157, loss = 0.20427172  
Iteration 158, loss = 0.19801184  
Iteration 159, loss = 0.19825854  
Iteration 160, loss = 0.19987893  
Iteration 161, loss = 0.19808404  
Iteration 162, loss = 0.19706006  
Iteration 163, loss = 0.19545960  
Iteration 164, loss = 0.19198142  
Iteration 165, loss = 0.19874646  
Iteration 166, loss = 0.19975094  
Iteration 167, loss = 0.18969855  
Iteration 168, loss = 0.19592014  
Iteration 169, loss = 0.20318988  
Iteration 170, loss = 0.18861323  
Iteration 171, loss = 0.19687122  
Iteration 172, loss = 0.19971058  
Iteration 173, loss = 0.18872662  
Iteration 174, loss = 0.19407785  
Iteration 175, loss = 0.18840658  
Iteration 176, loss = 0.18642795  
Iteration 177, loss = 0.20257853  
Iteration 178, loss = 0.19036916  
Iteration 179, loss = 0.18954640  
Iteration 180, loss = 0.19262127

```
Iteration 181, loss = 0.18555495
Iteration 182, loss = 0.18221644
Iteration 183, loss = 0.18179603
Iteration 184, loss = 0.18384272
Iteration 185, loss = 0.18038333
Iteration 186, loss = 0.18837300
Iteration 187, loss = 0.18552340
Iteration 188, loss = 0.17813300
Iteration 189, loss = 0.17841459
Iteration 190, loss = 0.17746224
Iteration 191, loss = 0.17615905
Iteration 192, loss = 0.17706448
Iteration 193, loss = 0.17445022
Iteration 194, loss = 0.17870040
Iteration 195, loss = 0.17945170
Iteration 196, loss = 0.17474412
Iteration 197, loss = 0.17325657
Iteration 198, loss = 0.17223660
Iteration 199, loss = 0.17192794
Iteration 200, loss = 0.17387431
Iteration 201, loss = 0.17361957
Iteration 202, loss = 0.17248654
Iteration 203, loss = 0.17401968
Iteration 204, loss = 0.17101373
Iteration 205, loss = 0.17081506
Iteration 206, loss = 0.17097379
Iteration 207, loss = 0.16758544
Iteration 208, loss = 0.16972054
Iteration 209, loss = 0.17010854
Iteration 210, loss = 0.16597442
Iteration 211, loss = 0.16795522
Iteration 212, loss = 0.16823160
Iteration 213, loss = 0.16418037
Iteration 214, loss = 0.16697008
Iteration 215, loss = 0.16582084
Iteration 216, loss = 0.16382480
Iteration 217, loss = 0.16484471
Iteration 218, loss = 0.16243197
Iteration 219, loss = 0.16218383
Iteration 220, loss = 0.16236868
Iteration 221, loss = 0.16106752
Iteration 222, loss = 0.16034033
Iteration 223, loss = 0.16019903
Iteration 224, loss = 0.15939317
Iteration 225, loss = 0.15812903
Iteration 226, loss = 0.15967220
Iteration 227, loss = 0.15972957
Iteration 228, loss = 0.15869792
Iteration 229, loss = 0.15701914
Iteration 230, loss = 0.15660028
Iteration 231, loss = 0.15681639
Iteration 232, loss = 0.15677736
Iteration 233, loss = 0.15779404
Iteration 234, loss = 0.16085464
Iteration 235, loss = 0.17011883
Iteration 236, loss = 0.15488930
Iteration 237, loss = 0.16842559
Iteration 238, loss = 0.15835579
Iteration 239, loss = 0.15875102
Iteration 240, loss = 0.16379257
```

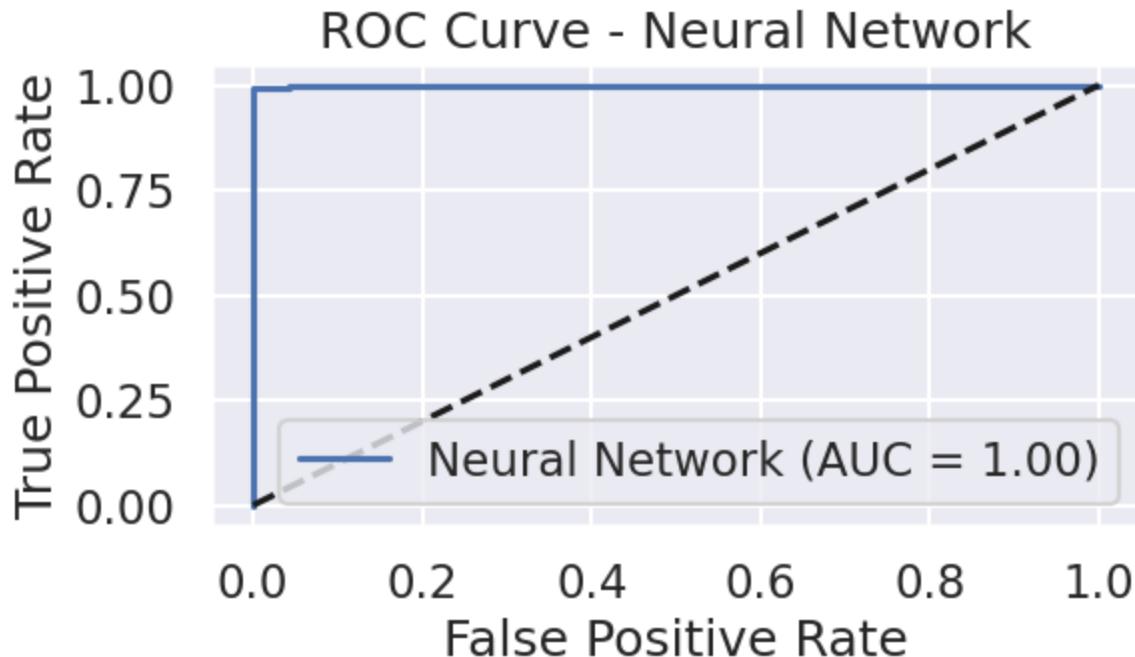
```
Iteration 241, loss = 0.15261567
Iteration 242, loss = 0.15714449
Iteration 243, loss = 0.15278826
Iteration 244, loss = 0.15089522
Iteration 245, loss = 0.15671092
Iteration 246, loss = 0.15103484
Iteration 247, loss = 0.15231768
Iteration 248, loss = 0.15242954
Iteration 249, loss = 0.14729689
Iteration 250, loss = 0.14944591
Iteration 251, loss = 0.15086011
Iteration 252, loss = 0.14731480
Iteration 253, loss = 0.14643971
Iteration 254, loss = 0.14697819
Iteration 255, loss = 0.14535753
Iteration 256, loss = 0.14441778
Iteration 257, loss = 0.14441094
Iteration 258, loss = 0.14371784
Iteration 259, loss = 0.14410804
Iteration 260, loss = 0.14373181
Iteration 261, loss = 0.14281049
Iteration 262, loss = 0.14266847
Iteration 263, loss = 0.14150942
Iteration 264, loss = 0.14135549
Iteration 265, loss = 0.14306235
Iteration 266, loss = 0.13924119
Iteration 267, loss = 0.14683114
Iteration 268, loss = 0.14989169
Iteration 269, loss = 0.14164417
Iteration 270, loss = 0.14145437
Iteration 271, loss = 0.13924703
Iteration 272, loss = 0.13965202
Iteration 273, loss = 0.14137038
Iteration 274, loss = 0.14434584
Iteration 275, loss = 0.13998019
Iteration 276, loss = 0.14006837
Iteration 277, loss = 0.13590659
Iteration 278, loss = 0.14019764
Iteration 279, loss = 0.14451119
Iteration 280, loss = 0.13560581
Iteration 281, loss = 0.13674836
Iteration 282, loss = 0.13751512
Iteration 283, loss = 0.13359945
Iteration 284, loss = 0.13567546
Iteration 285, loss = 0.13645231
Iteration 286, loss = 0.13285221
Iteration 287, loss = 0.13333664
Iteration 288, loss = 0.13635857
Iteration 289, loss = 0.13312503
Iteration 290, loss = 0.13223208
Iteration 291, loss = 0.13965799
Iteration 292, loss = 0.13122649
Iteration 293, loss = 0.13519012
Iteration 294, loss = 0.13540466
Iteration 295, loss = 0.13035279
Iteration 296, loss = 0.13180443
Iteration 297, loss = 0.12740952
Iteration 298, loss = 0.13376283
Iteration 299, loss = 0.13312770
Iteration 300, loss = 0.12937742
```

```
Iteration 301, loss = 0.13417229
Iteration 302, loss = 0.12919634
Iteration 303, loss = 0.12877167
Iteration 304, loss = 0.12688507
Iteration 305, loss = 0.12518773
Iteration 306, loss = 0.12507342
Iteration 307, loss = 0.12510165
Iteration 308, loss = 0.12449899
Iteration 309, loss = 0.12375685
Iteration 310, loss = 0.12734862
Iteration 311, loss = 0.12557102
Iteration 312, loss = 0.12164985
Iteration 313, loss = 0.13707716
Iteration 314, loss = 0.12867239
Iteration 315, loss = 0.12589512
Iteration 316, loss = 0.13253958
Iteration 317, loss = 0.12346435
Iteration 318, loss = 0.12208080
Iteration 319, loss = 0.12245356
Iteration 320, loss = 0.11961748
Iteration 321, loss = 0.12394614
Iteration 322, loss = 0.12021753
Iteration 323, loss = 0.12164048
Iteration 324, loss = 0.12626440
Iteration 325, loss = 0.12032976
Iteration 326, loss = 0.12178975
Iteration 327, loss = 0.12213310
Iteration 328, loss = 0.11803913
Iteration 329, loss = 0.12114994
Iteration 330, loss = 0.11725703
Iteration 331, loss = 0.12076094
Iteration 332, loss = 0.12026022
Iteration 333, loss = 0.11602327
Iteration 334, loss = 0.11995465
Iteration 335, loss = 0.11718524
Iteration 336, loss = 0.11609747
Iteration 337, loss = 0.11760468
Iteration 338, loss = 0.11490976
Iteration 339, loss = 0.11496963
Iteration 340, loss = 0.11422677
Iteration 341, loss = 0.11375234
Iteration 342, loss = 0.11340566
Iteration 343, loss = 0.11351552
Iteration 344, loss = 0.11356643
Iteration 345, loss = 0.11311543
Iteration 346, loss = 0.11259733
Iteration 347, loss = 0.11124089
Iteration 348, loss = 0.11185655
Iteration 349, loss = 0.11200215
Iteration 350, loss = 0.11118142
Iteration 351, loss = 0.11030712
Iteration 352, loss = 0.11021561
Iteration 353, loss = 0.10982264
Iteration 354, loss = 0.10987060
Iteration 355, loss = 0.10924542
Iteration 356, loss = 0.10947071
Iteration 357, loss = 0.10895485
Iteration 358, loss = 0.10889616
Iteration 359, loss = 0.10804792
Iteration 360, loss = 0.10791342
```

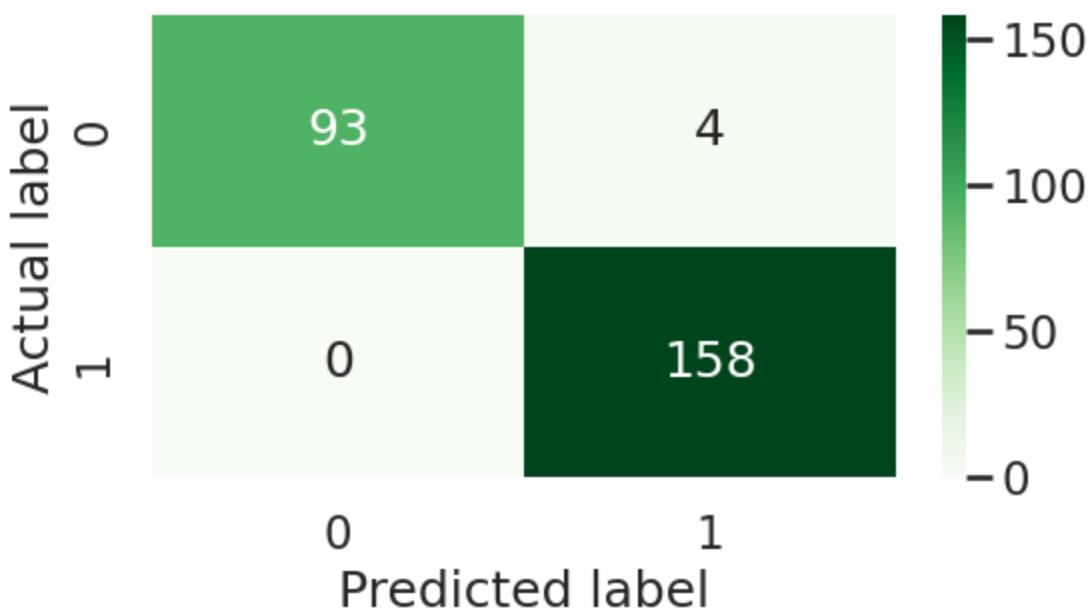
```
Iteration 361, loss = 0.10907126
Iteration 362, loss = 0.10781227
Iteration 363, loss = 0.10740163
Iteration 364, loss = 0.10885671
Iteration 365, loss = 0.10664161
Iteration 366, loss = 0.11066026
Iteration 367, loss = 0.10595491
Iteration 368, loss = 0.10856119
Iteration 369, loss = 0.11647613
Iteration 370, loss = 0.10547562
Iteration 371, loss = 0.11319035
Iteration 372, loss = 0.11453445
Iteration 373, loss = 0.10660037
Iteration 374, loss = 0.10672690
Iteration 375, loss = 0.10351444
Iteration 376, loss = 0.10486834
Iteration 377, loss = 0.10980695
Iteration 378, loss = 0.10328256
Iteration 379, loss = 0.10588789
Iteration 380, loss = 0.10731047
Iteration 381, loss = 0.10215224
Iteration 382, loss = 0.10181712
Iteration 383, loss = 0.10125605
Iteration 384, loss = 0.10228608
Iteration 385, loss = 0.10076854
Iteration 386, loss = 0.10212498
Iteration 387, loss = 0.10126292
Iteration 388, loss = 0.10246588
Iteration 389, loss = 0.10508157
Iteration 390, loss = 0.10176577
Iteration 391, loss = 0.10079252
Iteration 392, loss = 0.09880238
Iteration 393, loss = 0.10208954
Iteration 394, loss = 0.09894385
Iteration 395, loss = 0.10188564
Iteration 396, loss = 0.10125334
Iteration 397, loss = 0.09814275
Iteration 398, loss = 0.09847286
Iteration 399, loss = 0.09768785
Iteration 400, loss = 0.09746975
Iteration 401, loss = 0.09705566
Iteration 402, loss = 0.09770101
Iteration 403, loss = 0.09765786
Iteration 404, loss = 0.09823282
Iteration 405, loss = 0.09750841
Iteration 406, loss = 0.09622743
Iteration 407, loss = 0.09536445
Iteration 408, loss = 0.09492604
Iteration 409, loss = 0.09453060
Iteration 410, loss = 0.09465540
Iteration 411, loss = 0.09405552
Iteration 412, loss = 0.09404265
Iteration 413, loss = 0.09435809
Iteration 414, loss = 0.09354805
Iteration 415, loss = 0.09284339
Iteration 416, loss = 0.09329854
Iteration 417, loss = 0.09286583
Iteration 418, loss = 0.09342240
Iteration 419, loss = 0.09173622
Iteration 420, loss = 0.09311403
```

Iteration 421, loss = 0.09077651  
Iteration 422, loss = 0.09321873  
Iteration 423, loss = 0.09357174  
Iteration 424, loss = 0.09166461  
Iteration 425, loss = 0.09352065  
Iteration 426, loss = 0.08989276  
Iteration 427, loss = 0.09230079  
Iteration 428, loss = 0.09229090  
Iteration 429, loss = 0.08877926  
Iteration 430, loss = 0.09419020  
Iteration 431, loss = 0.09059947  
Iteration 432, loss = 0.09079556  
Iteration 433, loss = 0.09054734  
Iteration 434, loss = 0.08770177  
Iteration 435, loss = 0.09359509  
Iteration 436, loss = 0.08819395  
Iteration 437, loss = 0.09153621  
Iteration 438, loss = 0.09317750  
Iteration 439, loss = 0.08635020  
Iteration 440, loss = 0.09001044  
Iteration 441, loss = 0.09009227  
Iteration 442, loss = 0.08569725  
Iteration 443, loss = 0.09009921  
Iteration 444, loss = 0.08791455  
Iteration 445, loss = 0.08970940  
Iteration 446, loss = 0.09254000  
Iteration 447, loss = 0.08717615  
Iteration 448, loss = 0.08879322  
Iteration 449, loss = 0.08509210  
Iteration 450, loss = 0.08938803  
Iteration 451, loss = 0.08714110  
Iteration 452, loss = 0.08362052  
Iteration 453, loss = 0.08622018  
Iteration 454, loss = 0.08347077  
Iteration 455, loss = 0.08590590  
Iteration 456, loss = 0.08621327  
Iteration 457, loss = 0.08234254  
Iteration 458, loss = 0.08418872  
Iteration 459, loss = 0.08382589  
Iteration 460, loss = 0.08233419  
Iteration 461, loss = 0.08205337  
Iteration 462, loss = 0.08343911  
Iteration 463, loss = 0.08199110  
Iteration 464, loss = 0.08120543  
Iteration 465, loss = 0.08116426  
Iteration 466, loss = 0.08080704  
Iteration 467, loss = 0.08046211  
Iteration 468, loss = 0.08026456  
Iteration 469, loss = 0.08064470  
Iteration 470, loss = 0.07995165  
Iteration 471, loss = 0.08244801  
Iteration 472, loss = 0.08266426  
Iteration 473, loss = 0.08347053  
Iteration 474, loss = 0.08072649  
Iteration 475, loss = 0.07981491  
Iteration 476, loss = 0.08162995  
Iteration 477, loss = 0.07846417  
Iteration 478, loss = 0.08384241  
Iteration 479, loss = 0.08154267  
Iteration 480, loss = 0.07840825

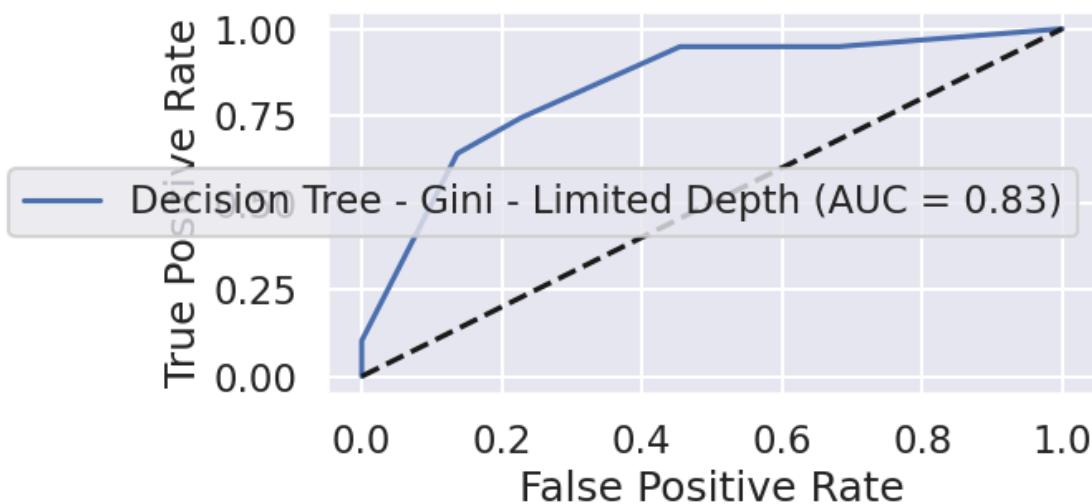
```
Iteration 481, loss = 0.08353738
Iteration 482, loss = 0.08160882
Iteration 483, loss = 0.07852621
Iteration 484, loss = 0.07844408
Iteration 485, loss = 0.07728118
Iteration 486, loss = 0.07703614
Iteration 487, loss = 0.07705122
Iteration 488, loss = 0.07667738
Iteration 489, loss = 0.07577607
Iteration 490, loss = 0.07723084
Iteration 491, loss = 0.07596730
Iteration 492, loss = 0.07628094
Iteration 493, loss = 0.07777272
Iteration 494, loss = 0.07465312
Iteration 495, loss = 0.07940364
Iteration 496, loss = 0.08560379
Iteration 497, loss = 0.07329538
Iteration 498, loss = 0.08452327
Iteration 499, loss = 0.08268718
Iteration 500, loss = 0.07561424
Iteration 501, loss = 0.07888863
Iteration 502, loss = 0.07339273
Iteration 503, loss = 0.07714437
Iteration 504, loss = 0.07403349
Iteration 505, loss = 0.07521156
Iteration 506, loss = 0.08397237
Iteration 507, loss = 0.07553793
Iteration 508, loss = 0.07679296
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```



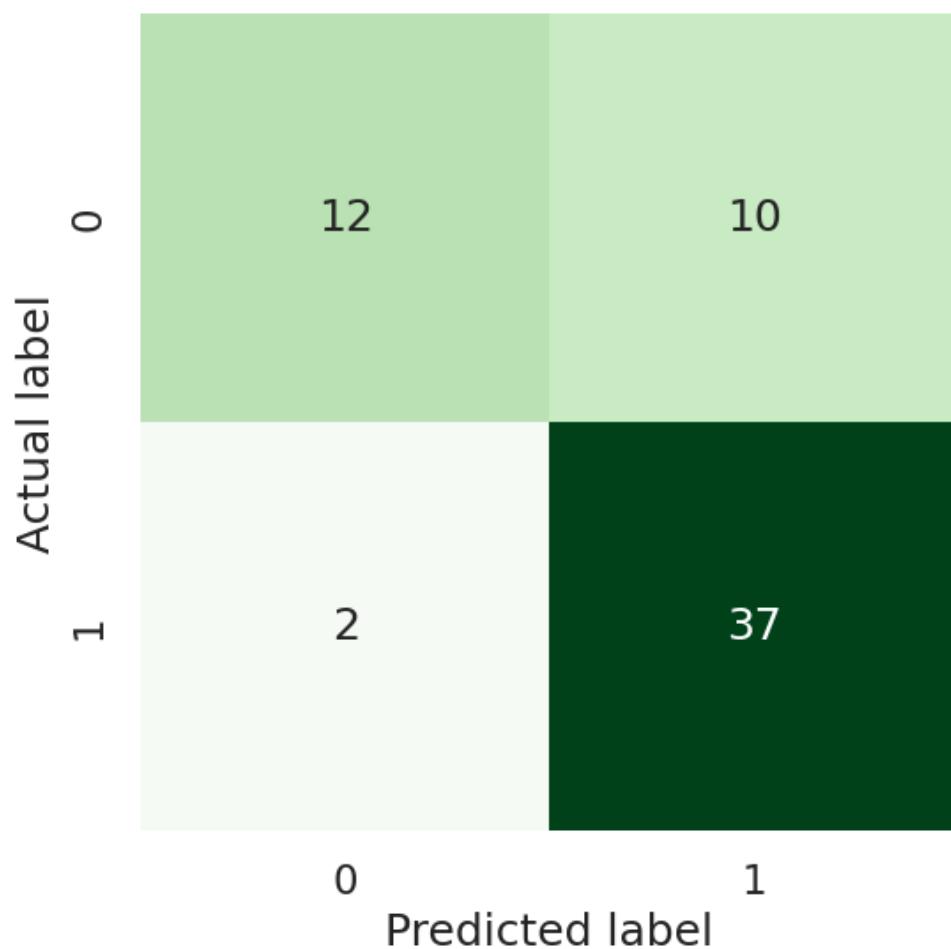
## Confusion Matrix - Neural Network



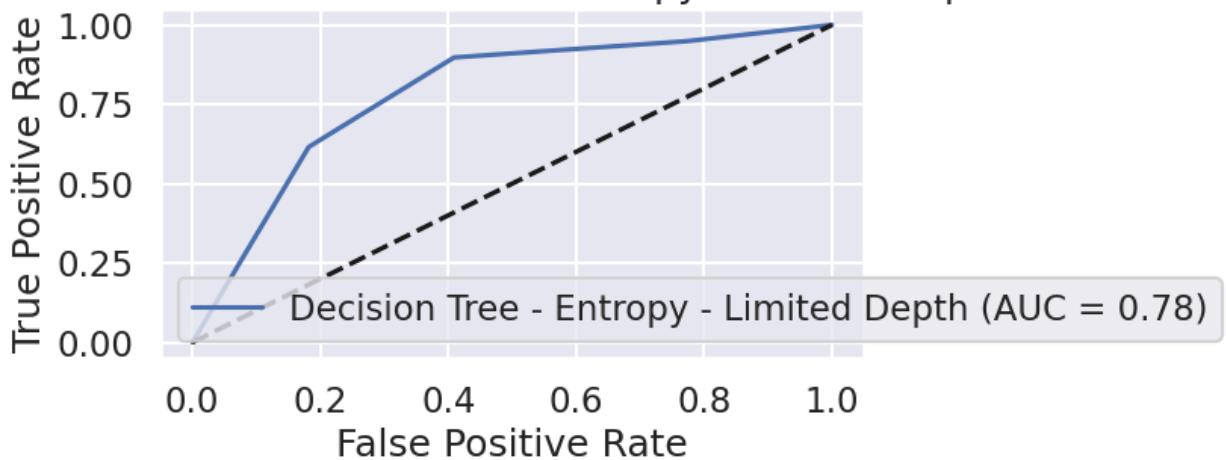
ROC Curve - Decision Tree - Gini - Limited Depth



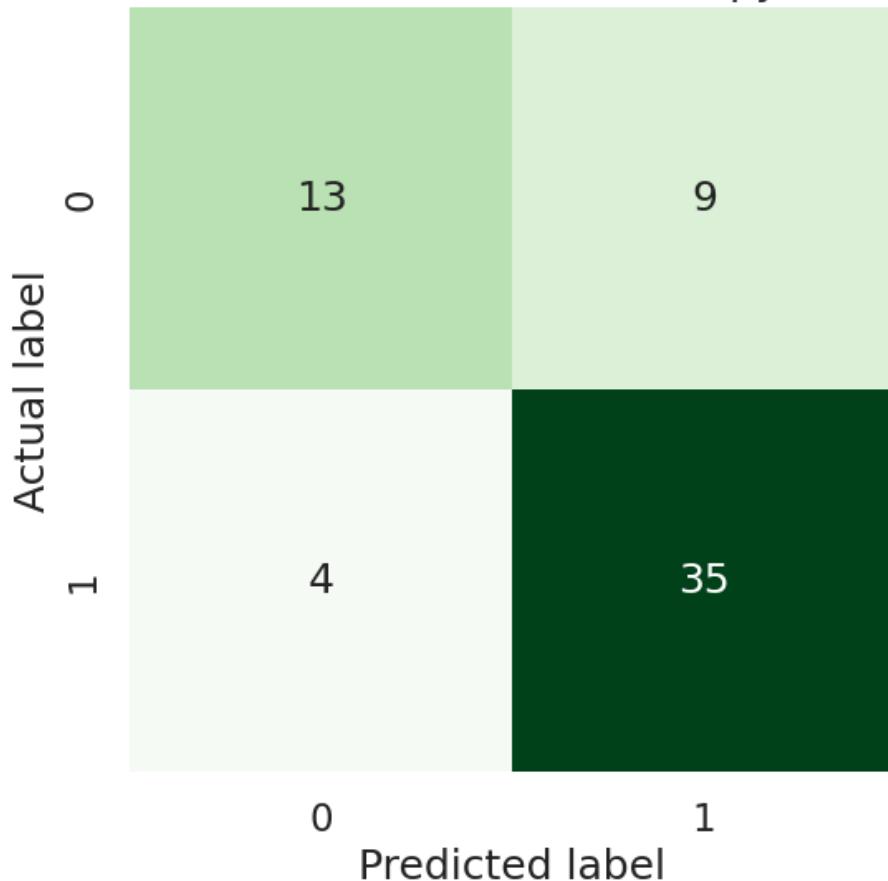
## Confusion Matrix - Decision Tree - Gini - Limited Depth



## ROC Curve - Decision Tree - Entropy - Limited Depth



## Confusion Matrix - Decision Tree - Entropy - Limited Depth



	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.952941	0.992431	0.962025	0.938144
<b>Logistic Regression (CV)</b>	0.925490	0.981924	0.948538	0.893864
<b>Logistic Regression (Test)</b>	0.918033	0.970862	0.923077	0.909091
<b>Decision Tree - Initial Train Set</b>	0.862745	0.943886	0.936709	0.742268
<b>Decision Tree - CV Train Set</b>	0.858824	0.924670	0.822785	0.917526
<b>Decision Tree - Test Set</b>	0.786885	0.828089	0.820513	0.727273
<b>Decision Tree - Gini - Limited Depth</b>	0.803279	0.831585	0.948718	0.545455
<b>Decision Tree - Entropy - Limited Depth</b>	0.786885	0.784965	0.897436	0.590909
<b>ANN</b>	0.984314	0.999739	1.000000	0.958763

```
In [36]: models = train_neural_network_with_kfolds(X_train, y_train, hidden_layer_sizes=(100,), cv_train_results_ann = evaluate_neural_network(models[-1], X_train, y_train, 'Artificial results["ANN - CV Train Set"] = cv_train_results_ann
```

```
results_df = pd.DataFrame({
    'Logistic Regression': original_train_results,
    'Logistic Regression (CV)': cv_train_results,
    'Logistic Regression (Test)': test_results,
    'Decision Tree - Initial Train Set': initial_train_results_dt,
    'Decision Tree - CV Train Set': cv_train_results_dt,
```

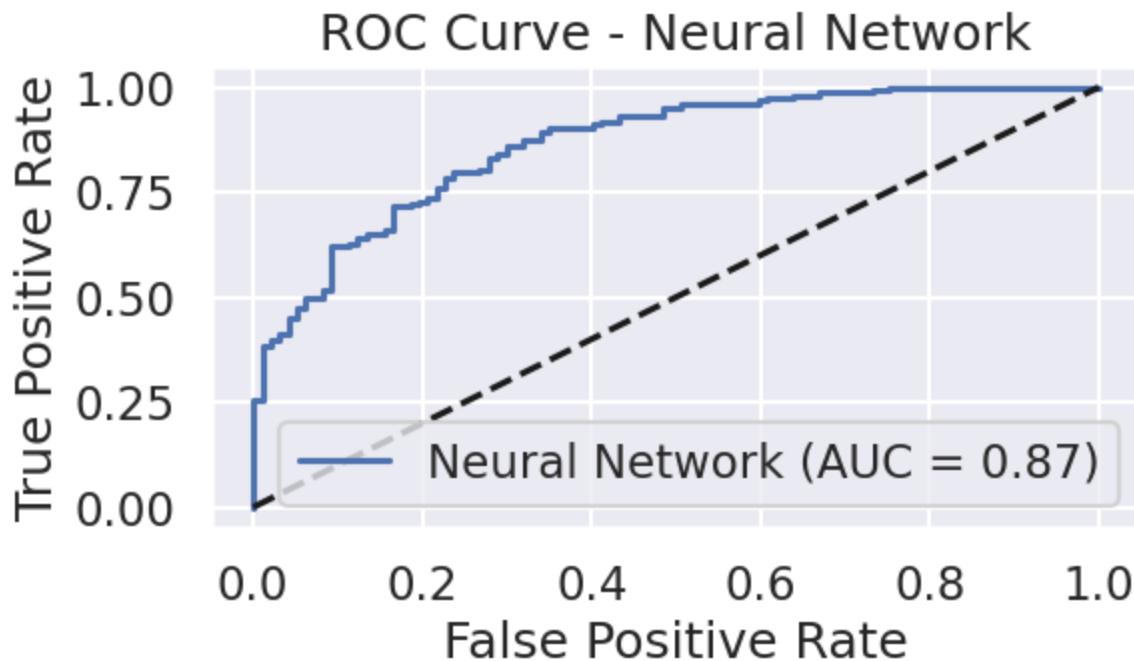
```
'Decision Tree - Test Set': test_results_dt,
'Decision Tree - Gini - Limited Depth': evaluate_decision_tree(dt_model_gini, X_te
'Decision Tree - Entropy - Limited Depth': evaluate_decision_tree(dt_model_entropy
'ANN': initial_train_results_ann,
'Ann (CV)': cv_train_results_ann
}).T
display(results_df)
```

```
Iteration 1, loss = 4.22163960
Iteration 2, loss = 2.23266392
Iteration 3, loss = 0.98520147
Iteration 4, loss = 0.54672483
Iteration 5, loss = 0.66943191
Iteration 6, loss = 0.93194218
Iteration 7, loss = 1.02349321
Iteration 8, loss = 0.96440416
Iteration 9, loss = 0.79275934
Iteration 10, loss = 0.62363374
Iteration 11, loss = 0.56077809
Iteration 12, loss = 0.57959426
Iteration 13, loss = 0.61640017
Iteration 14, loss = 0.64929655
Iteration 15, loss = 0.65909269
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
Iteration 1, loss = 4.22354979
Iteration 2, loss = 2.25010972
Iteration 3, loss = 0.97161520
Iteration 4, loss = 0.58184682
Iteration 5, loss = 0.69279621
Iteration 6, loss = 0.80113225
Iteration 7, loss = 0.81207305
Iteration 8, loss = 0.73692669
Iteration 9, loss = 0.62985174
Iteration 10, loss = 0.58411131
Iteration 11, loss = 0.59810975
Iteration 12, loss = 0.62565651
Iteration 13, loss = 0.63808097
Iteration 14, loss = 0.62662879
Iteration 15, loss = 0.59222289
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
Iteration 1, loss = 4.40308327
Iteration 2, loss = 2.35128481
Iteration 3, loss = 0.96430909
Iteration 4, loss = 0.58045854
Iteration 5, loss = 0.67514544
Iteration 6, loss = 0.82212449
Iteration 7, loss = 0.88910858
Iteration 8, loss = 0.81014449
Iteration 9, loss = 0.66751640
Iteration 10, loss = 0.57471490
Iteration 11, loss = 0.57928092
Iteration 12, loss = 0.63068796
Iteration 13, loss = 0.65174381
Iteration 14, loss = 0.62220160
Iteration 15, loss = 0.56242003
Iteration 16, loss = 0.52825348
Iteration 17, loss = 0.55181935
Iteration 18, loss = 0.62242423
Iteration 19, loss = 0.67718384
Iteration 20, loss = 0.68012165
Iteration 21, loss = 0.61506530
Iteration 22, loss = 0.53279130
Iteration 23, loss = 0.49080675
Iteration 24, loss = 0.48420794
Iteration 25, loss = 0.49438687
Iteration 26, loss = 0.51639462
```

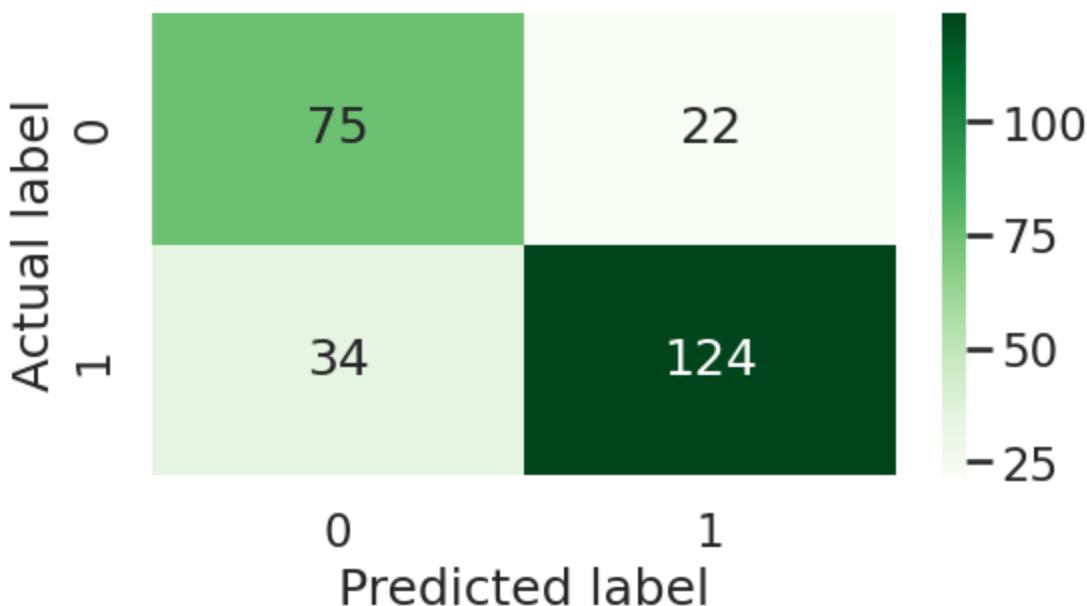
```
Iteration 27, loss = 0.51440801
Iteration 28, loss = 0.48609095
Iteration 29, loss = 0.47502913
Iteration 30, loss = 0.47100567
Iteration 31, loss = 0.46709770
Iteration 32, loss = 0.47838106
Iteration 33, loss = 0.50439703
Iteration 34, loss = 0.46787340
Iteration 35, loss = 0.40034390
Iteration 36, loss = 0.39182700
Iteration 37, loss = 0.40217608
Iteration 38, loss = 0.38545705
Iteration 39, loss = 0.36846385
Iteration 40, loss = 0.36162610
Iteration 41, loss = 0.36822233
Iteration 42, loss = 0.38895401
Iteration 43, loss = 0.36801980
Iteration 44, loss = 0.35042417
Iteration 45, loss = 0.34514174
Iteration 46, loss = 0.33672332
Iteration 47, loss = 0.33139784
Iteration 48, loss = 0.32938864
Iteration 49, loss = 0.33058718
Iteration 50, loss = 0.34757384
Iteration 51, loss = 0.35010392
Iteration 52, loss = 0.32402906
Iteration 53, loss = 0.35111728
Iteration 54, loss = 0.39375771
Iteration 55, loss = 0.36079342
Iteration 56, loss = 0.31791175
Iteration 57, loss = 0.32537116
Iteration 58, loss = 0.32398032
Iteration 59, loss = 0.31360946
Iteration 60, loss = 0.31106698
Iteration 61, loss = 0.31525581
Iteration 62, loss = 0.33791280
Iteration 63, loss = 0.34376663
Iteration 64, loss = 0.31549679
Iteration 65, loss = 0.30202004
Iteration 66, loss = 0.30222907
Iteration 67, loss = 0.32165497
Iteration 68, loss = 0.35633922
Iteration 69, loss = 0.35763595
Iteration 70, loss = 0.30702865
Iteration 71, loss = 0.30646405
Iteration 72, loss = 0.41109411
Iteration 73, loss = 0.38481325
Iteration 74, loss = 0.29196668
Iteration 75, loss = 0.35370177
Iteration 76, loss = 0.49051039
Iteration 77, loss = 0.51066222
Iteration 78, loss = 0.37842278
Iteration 79, loss = 0.29594399
Iteration 80, loss = 0.33059990
Iteration 81, loss = 0.35648972
Iteration 82, loss = 0.31689687
Iteration 83, loss = 0.30439458
Iteration 84, loss = 0.34936877
Iteration 85, loss = 0.39807018
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopp
```

ing.

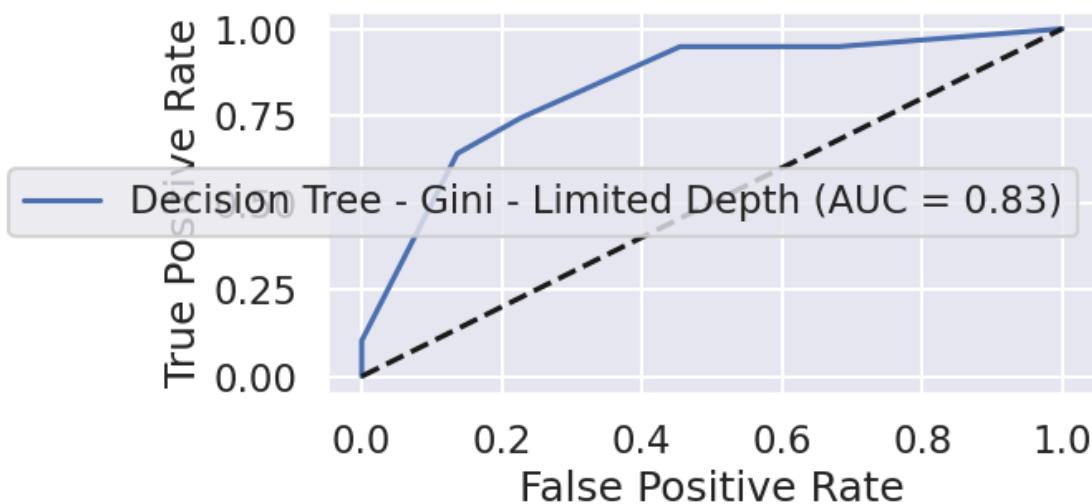
Iteration 1, loss = 4.21620790  
Iteration 2, loss = 2.24222414  
Iteration 3, loss = 0.93538134  
Iteration 4, loss = 0.61455033  
Iteration 5, loss = 0.86172328  
Iteration 6, loss = 1.10095280  
Iteration 7, loss = 1.13128692  
Iteration 8, loss = 1.05086702  
Iteration 9, loss = 0.91318579  
Iteration 10, loss = 0.76235227  
Iteration 11, loss = 0.66191068  
Iteration 12, loss = 0.63935631  
Iteration 13, loss = 0.69674533  
Iteration 14, loss = 0.81829953  
Iteration 15, loss = 0.93396626  
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.  
Iteration 1, loss = 3.92051852  
Iteration 2, loss = 2.13573007  
Iteration 3, loss = 0.99513238  
Iteration 4, loss = 0.68912846  
Iteration 5, loss = 0.90766105  
Iteration 6, loss = 1.15997858  
Iteration 7, loss = 1.12201840  
Iteration 8, loss = 0.91680068  
Iteration 9, loss = 0.76407906  
Iteration 10, loss = 0.71199174  
Iteration 11, loss = 0.73502274  
Iteration 12, loss = 0.77686128  
Iteration 13, loss = 0.80032925  
Iteration 14, loss = 0.79393632  
Iteration 15, loss = 0.77233785  
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.



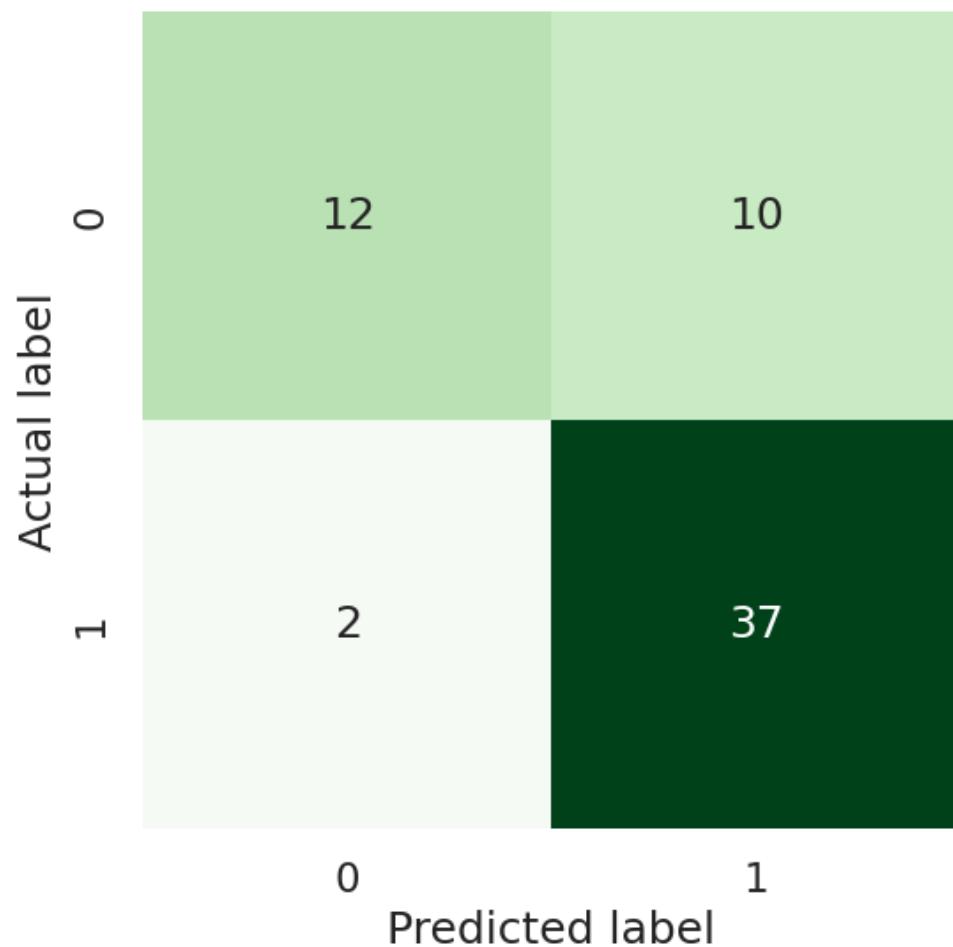
## Confusion Matrix - Neural Network



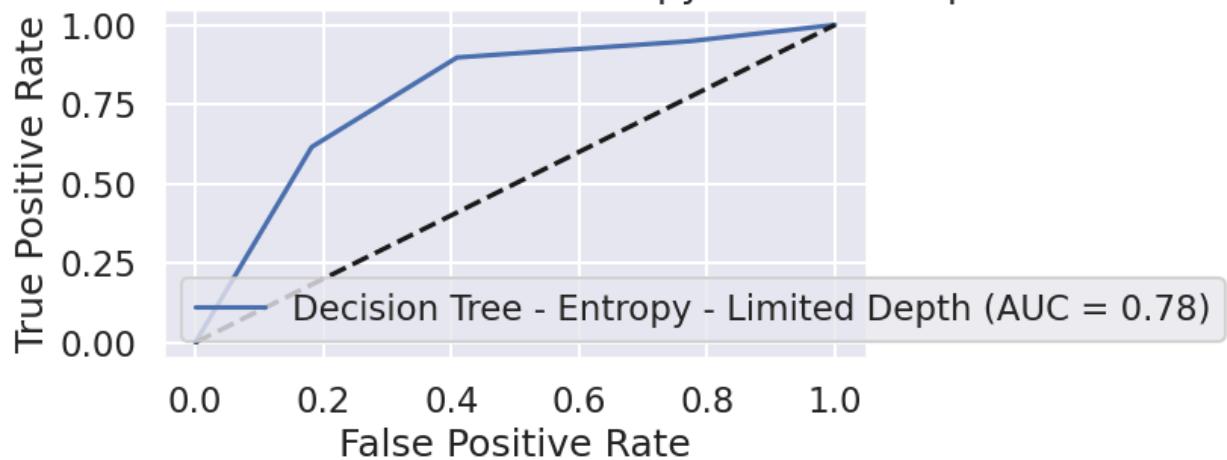
ROC Curve - Decision Tree - Gini - Limited Depth



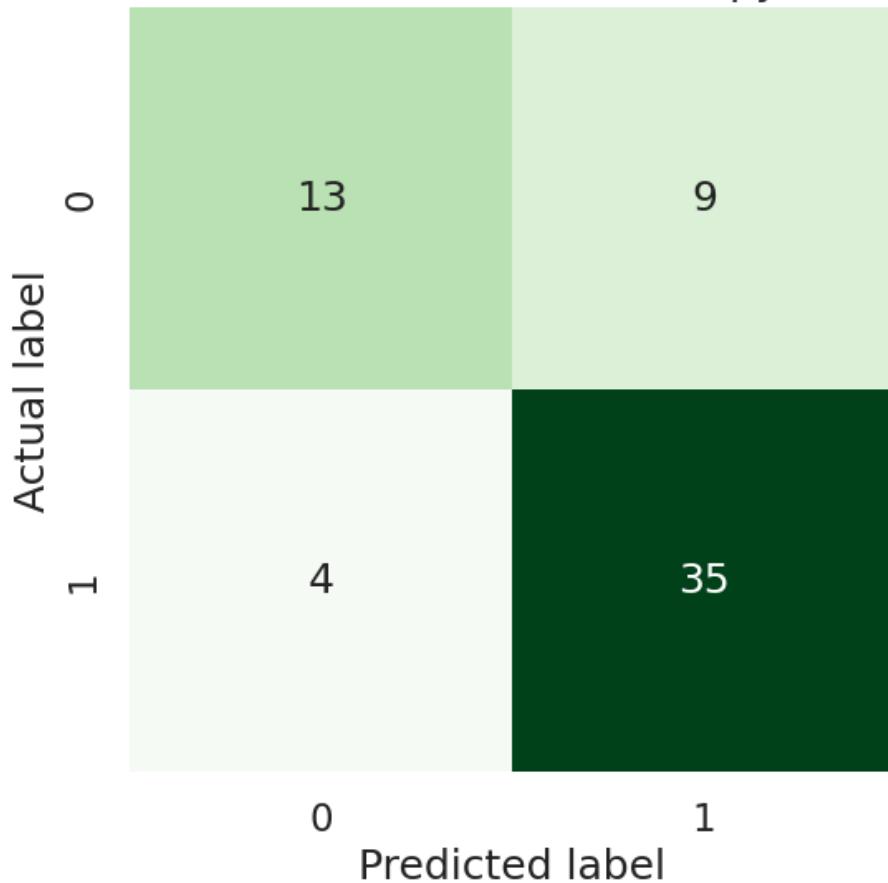
## Confusion Matrix - Decision Tree - Gini - Limited Depth



## ROC Curve - Decision Tree - Entropy - Limited Depth



## Confusion Matrix - Decision Tree - Entropy - Limited Depth



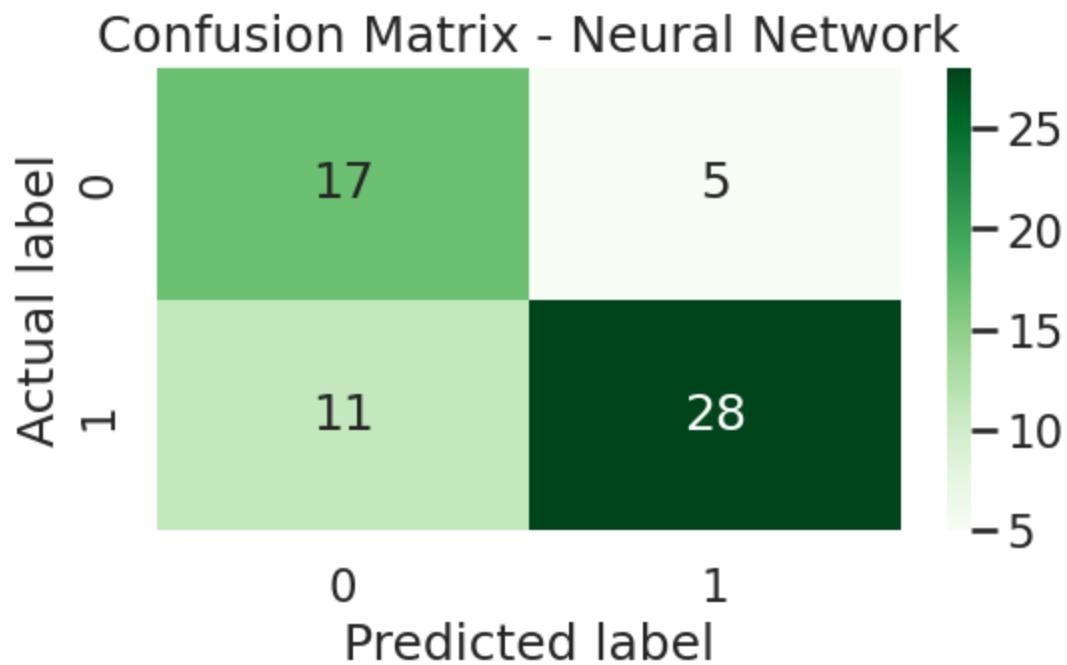
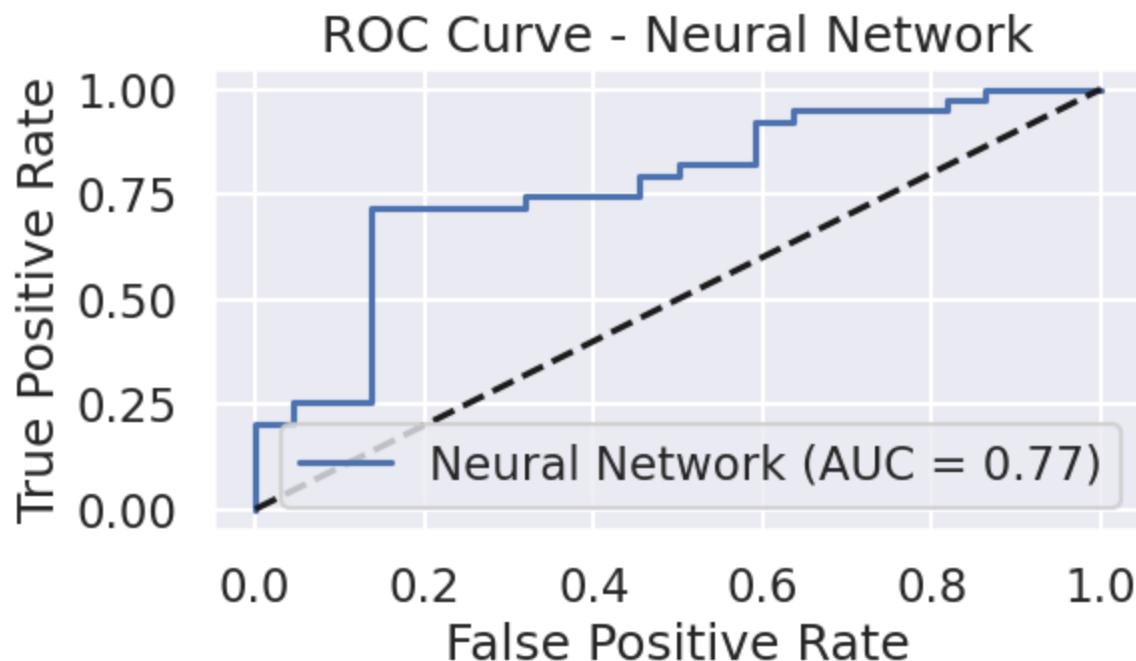
	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.952941	0.992431	0.962025	0.938144
<b>Logistic Regression (CV)</b>	0.925490	0.981924	0.948538	0.893864
<b>Logistic Regression (Test)</b>	0.918033	0.970862	0.923077	0.909091
<b>Decision Tree - Initial Train Set</b>	0.862745	0.943886	0.936709	0.742268
<b>Decision Tree - CV Train Set</b>	0.858824	0.924670	0.822785	0.917526
<b>Decision Tree - Test Set</b>	0.786885	0.828089	0.820513	0.727273
<b>Decision Tree - Gini - Limited Depth</b>	0.803279	0.831585	0.948718	0.545455
<b>Decision Tree - Entropy - Limited Depth</b>	0.786885	0.784965	0.897436	0.590909
<b>ANN</b>	0.984314	0.999739	1.000000	0.958763
<b>Ann (CV)</b>	0.780392	0.865718	0.784810	0.773196

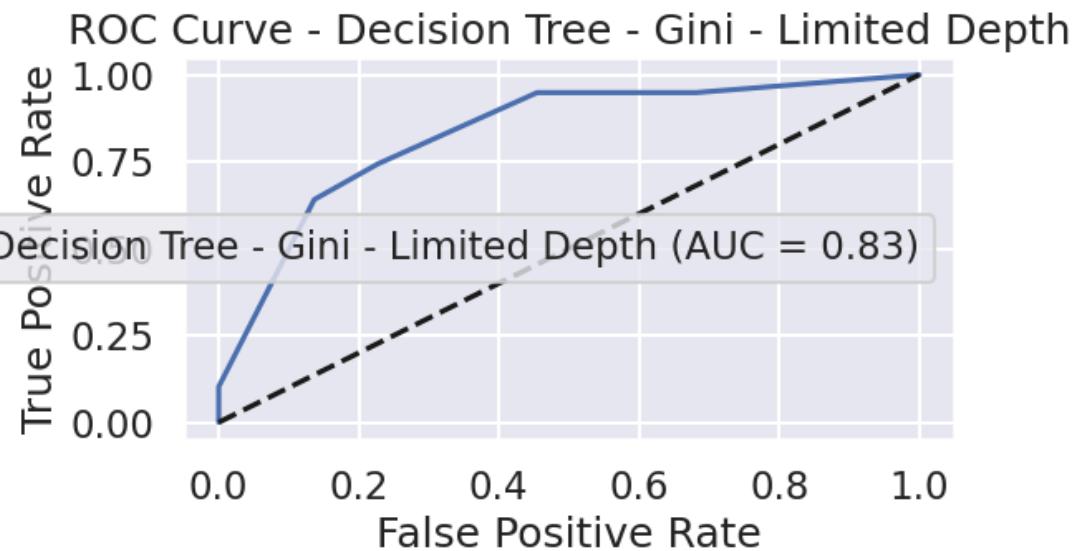
```
In [37]: test_results_ann = evaluate_neural_network(models[-1], X_test, y_test, 'ANN - Test Set'
results["ANN - Test Set"] = test_results_ann
```

```
results_df = pd.DataFrame({
    'Logistic Regression': original_train_results,
    'Logistic Regression (CV)': cv_train_results,
    'Logistic Regression (Test)': test_results,
    'Decision Tree': initial_train_results_dt,
```

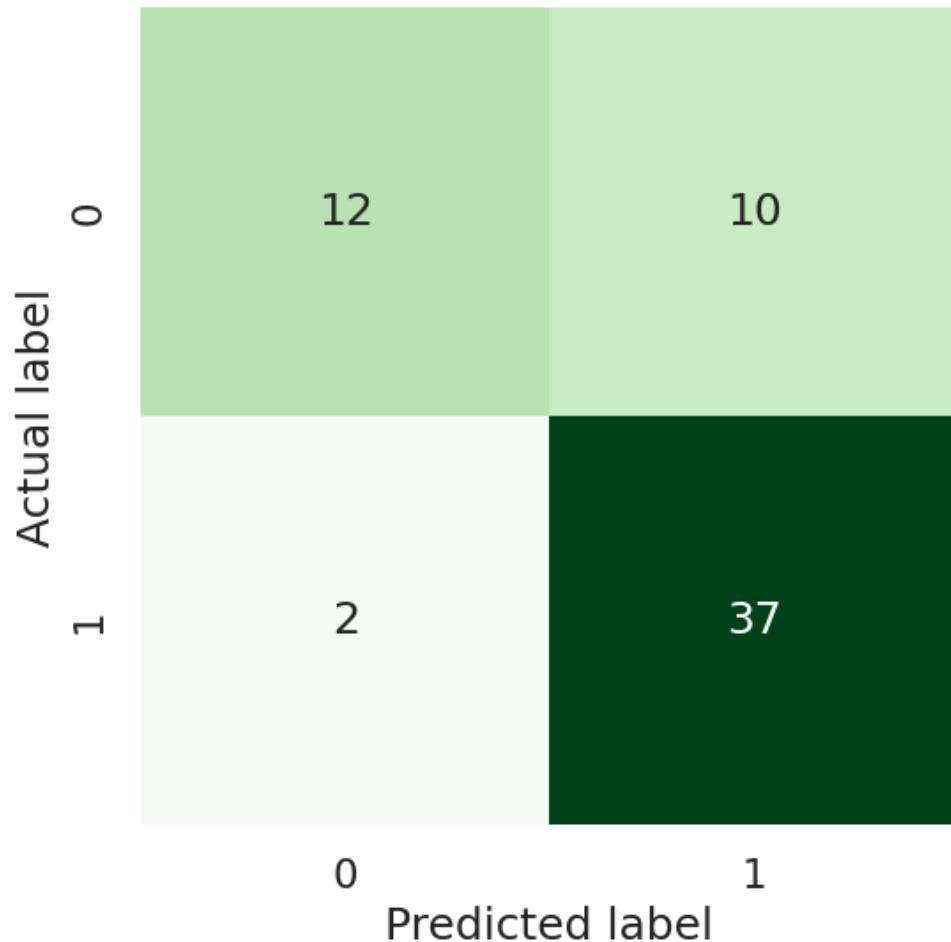
```
'Decision Tree (CV)': cv_train_results_dt,
'Decision Tree (Test)': test_results_dt,
'Decision Tree - Gini - Limited Depth': evaluate_decision_tree(dt_model_gini, X_te
'Decision Tree - Entropy - Limited Depth': evaluate_decision_tree(dt_model_entropy
'ANN': initial_train_results_ann,
'ANN (CV)': cv_train_results_ann,
'ANN (Test)': test_results_ann
}).T

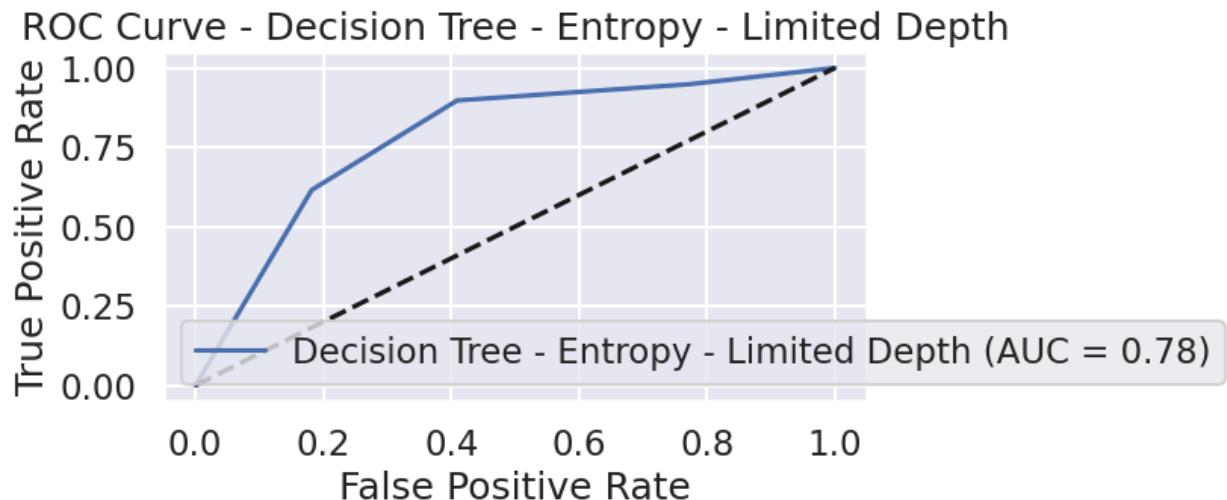
display(results_df)
```



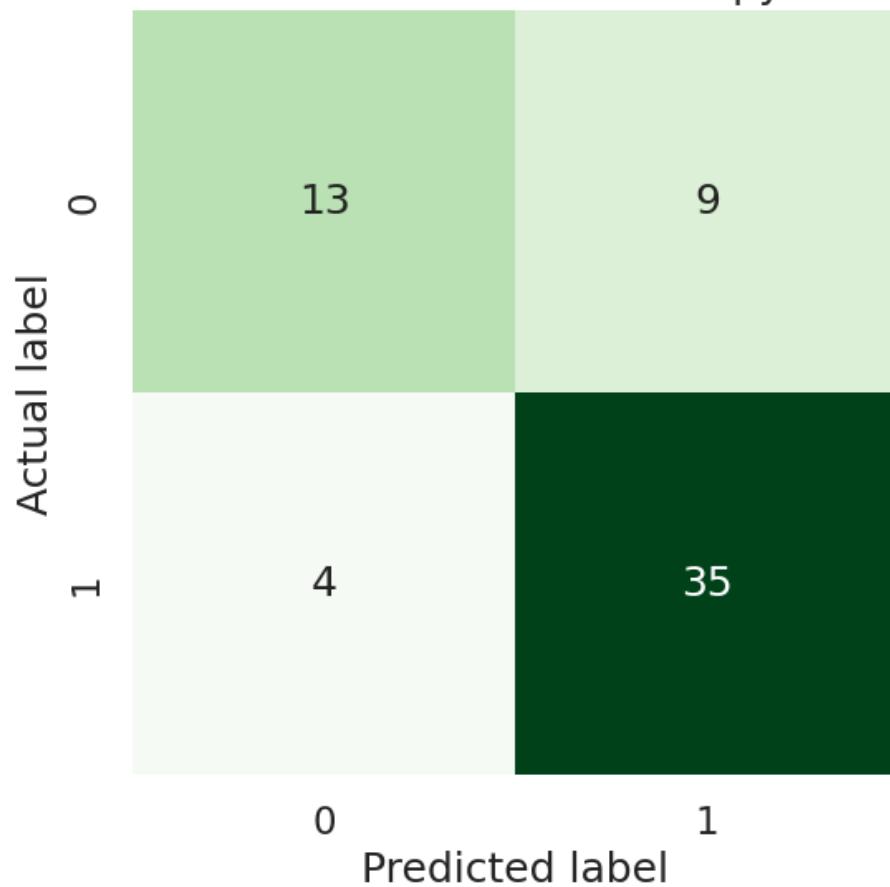


Confusion Matrix - Decision Tree - Gini - Limited Depth





### Confusion Matrix - Decision Tree - Entropy - Limited Depth



	Accuracy	ROC AUC	Sensitivity	Specificity
<b>Logistic Regression</b>	0.952941	0.992431	0.962025	0.938144
<b>Logistic Regression (CV)</b>	0.925490	0.981924	0.948538	0.893864
<b>Logistic Regression (Test)</b>	0.918033	0.970862	0.923077	0.909091
<b>Decision Tree</b>	0.862745	0.943886	0.936709	0.742268
<b>Decision Tree (CV)</b>	0.858824	0.924670	0.822785	0.917526
<b>Decision Tree (Test)</b>	0.786885	0.828089	0.820513	0.727273
<b>Decision Tree - Gini - Limited Depth</b>	0.803279	0.831585	0.948718	0.545455
<b>Decision Tree - Entropy - Limited Depth</b>	0.786885	0.784965	0.897436	0.590909
<b>ANN</b>	0.984314	0.999739	1.000000	0.958763
<b>ANN (CV)</b>	0.780392	0.865718	0.784810	0.773196
<b>ANN (Test)</b>	0.737705	0.770396	0.717949	0.772727

In [38]:

```

y_pred_lr = initial_lr_model.predict(X_test)
y_pred_lr_cv = cv_lr_model.predict(X_test)

y_pred_dt = dt_initial.predict(X_test)
y_pred_dt_cv = dt_cv.predict(X_test)

y_pred_dt_gini = dt_model_gini.predict(X_test)
y_pred_dt_entropy = dt_model_entropy.predict(X_test)

y_pred_ann = ann_model.predict(X_test)
y_pred_ann_cv = models[-1].predict(X_test)

precision_scores = {
    "Logistic Regression": precision_score(y_test, y_pred_lr),
    "Logistic Regression - CV": precision_score(y_test, y_pred_lr_cv),
    "Decision Tree": precision_score(y_test, y_pred_dt),
    "Decision Tree - CV": precision_score(y_test, y_pred_dt_cv),
    "Decision Tree - Gini - Limited Depth": precision_score(y_test, y_pred_dt_gini),
    "Decision Tree - Entropy - Limited Depth": precision_score(y_test, y_pred_dt_entropy),
    "Artificial Neural Network": precision_score(y_test, y_pred_ann),
    "Artificial Neural Network - CV": precision_score(y_test, y_pred_ann_cv)
}
model_names = list(precision_scores.keys())
precision_values = list(precision_scores.values())
norm = plt.Normalize(min(precision_values), max(precision_values))
colors = [plt.cm.Greens(norm(value)) for value in precision_values]
plt.figure(figsize=(10, 8))
bar = plt.bar(model_names, precision_values, color=colors)
plt.xlabel('Model')
plt.ylabel('Precision Score')
plt.title('Comparison of Model Precision Scores')
plt.xticks(rotation=45, fontsize=6)
sm = cm.ScalarMappable(cmap=plt.cm.Greens, norm=norm)
sm.set_array([])
plt.colorbar(sm, label='Precision Score')
plt.tight_layout()
plt.show()

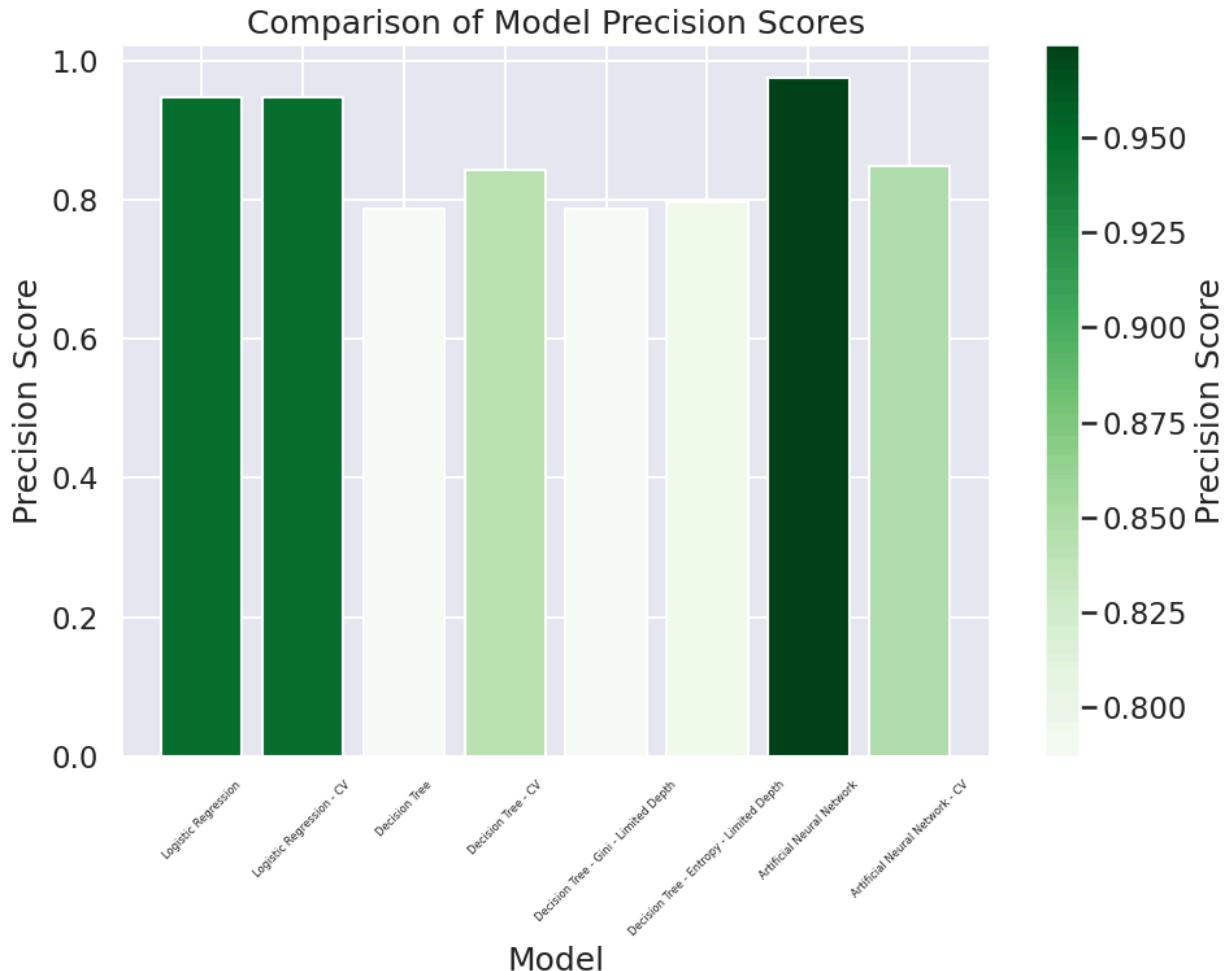
```

```

Out[38]: <Figure size 1000x800 with 0 Axes>
Out[38]: Text(0.5, 0, 'Model')
Out[38]: Text(0, 0.5, 'Precision Score')
Out[38]: Text(0.5, 1.0, 'Comparison of Model Precision Scores')
Out[38]: ([0, 1, 2, 3, 4, 5, 6, 7],
           [Text(0, 0, 'Logistic Regression'),
            Text(1, 0, 'Logistic Regression - CV'),
            Text(2, 0, 'Decision Tree'),
            Text(3, 0, 'Decision Tree - CV'),
            Text(4, 0, 'Decision Tree - Gini - Limited Depth'),
            Text(5, 0, 'Decision Tree - Entropy - Limited Depth'),
            Text(6, 0, 'Artificial Neural Network'),
            Text(7, 0, 'Artificial Neural Network - CV')])

<ipython-input-38-c9fb05c09afa>:35: MatplotlibDeprecationWarning: Unable to determine
Axes to steal space for Colorbar. Using gca(), but will raise in the future. Either p
rovide the *cax* argument to use as the Axes for the Colorbar, provide the *ax* argum
ent to steal space from it, or add *mappable* to an Axes.
    plt.colorbar(sm, label='Precision Score')
Out[38]: <matplotlib.colorbar.Colorbar at 0x7ddd520bc190>

```



```
In [39]: from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
In [40]: ! pwd
```

```
/content
```

```
In [41]: %%shell  
jupyter nbconvert --to html //content/Breast_Cancer_Survival_01.ipynb
```

[NbConvertApp] WARNING | pattern '//content/Breast\_Cancer\_Survival\_Prediction\_01.ipynb' matched no files

This application is used to convert notebook files (\*.ipynb)  
to various other formats.

WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

#### Options

=====

The options below are convenience aliases to configurable class-options,  
as listed in the "Equivalent to" description-line of the aliases.

To see all configurable class-options for some <cmd>, use:  
  <cmd> --help-all

--debug

  set log level to logging.DEBUG (maximize logging output)  
  Equivalent to: [--Application.log\_level=10]

--show-config

  Show the application's configuration (human-readable format)  
  Equivalent to: [--Application.show\_config=True]

--show-config-json

  Show the application's configuration (json format)  
  Equivalent to: [--Application.show\_config\_json=True]

--generate-config

  generate default config file  
  Equivalent to: [--JupyterApp.generate\_config=True]

-y

  Answer yes to any questions instead of prompting.  
  Equivalent to: [--JupyterApp.answer\_yes=True]

--execute

  Execute the notebook prior to export.  
  Equivalent to: [--ExecutePreprocessor.enabled=True]

--allow-errors

  Continue notebook execution even if one of the cells throws an error and include  
the error message in the cell output (the default behaviour is to abort conversion).  
This flag is only relevant if '--execute' was specified, too.

  Equivalent to: [--ExecutePreprocessor.allow\_errors=True]

--stdin

  read a single notebook file from stdin. Write the resulting notebook with default  
basename 'notebook.\*'

  Equivalent to: [--NbConvertApp.from\_stdin=True]

--stdout

  Write notebook output to stdout instead of files.  
  Equivalent to: [--NbConvertApp.writer\_class=StdoutWriter]

--inplace

  Run nbconvert in place, overwriting the existing notebook (only  
relevant when converting to notebook format)

  Equivalent to: [--NbConvertApp.use\_output\_suffix=False --NbConvertApp.export\_form  
at=notebook --FileWriter.build\_directory=]

--clear-output

  Clear output of current file and save in place,  
  overwriting the existing notebook.

  Equivalent to: [--NbConvertApp.use\_output\_suffix=False --NbConvertApp.export\_form  
at=notebook --FileWriter.build\_directory= --ClearOutputPreprocessor.enabled=True]

--no-prompt

  Exclude input and output prompts from converted document.

  Equivalent to: [--TemplateExporter.exclude\_input\_prompt=True --TemplateExporter.e  
xclude\_output\_prompt=True]

--no-input

  Exclude input cells and output prompts from converted document.

This mode is ideal for generating code-free reports.

Equivalent to: [--TemplateExporter.exclude\_output\_prompt=True --TemplateExporter.exclude\_input=True --TemplateExporter.exclude\_input\_prompt=True]  
--allow-chromium-download  
Whether to allow downloading chromium if no suitable version is found on the system.

Equivalent to: [--WebPDFExporter.allow\_chromium\_download=True]  
--disable-chromium-sandbox  
Disable chromium security sandbox when converting to PDF..

Equivalent to: [--WebPDFExporter.disable\_sandbox=True]  
--show-input  
Shows code input. This flag is only useful for dejavu users.

Equivalent to: [--TemplateExporter.exclude\_input=False]  
--embed-images  
Embed the images as base64 dataurls in the output. This flag is only useful for the HTML/WebPDF/Slides exports.

Equivalent to: [--HTMLExporter.embed\_images=True]  
--sanitize-html  
Whether the HTML in Markdown cells and cell outputs should be sanitized..

Equivalent to: [--HTMLExporter.sanitize\_html=True]  
--log-level=<Enum>  
Set the log level by value or name.  
Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICAL']  
Default: 30  
Equivalent to: [--Application.log\_level]  
--config=<Unicode>  
Full path of a config file.  
Default: ''  
Equivalent to: [--JupyterApp.config\_file]  
--to=<Unicode>  
The export format to be used, either one of the built-in formats  
['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides', 'webpdf']  
or a dotted object name that represents the import path for an  
`Exporter` class  
Default: ''  
Equivalent to: [--NbConvertApp.export\_format]  
--template=<Unicode>  
Name of the template to use  
Default: ''  
Equivalent to: [--TemplateExporter.template\_name]  
--template-file=<Unicode>  
Name of the template file to use  
Default: None  
Equivalent to: [--TemplateExporter.template\_file]  
--theme=<Unicode>  
Template specific theme(e.g. the name of a JupyterLab CSS theme distributed as prebuilt extension for the lab template)  
Default: 'light'  
Equivalent to: [--HTMLExporter.theme]  
--sanitize\_html=<Bool>  
Whether the HTML in Markdown cells and cell outputs should be sanitized.This should be set to True by nbviewer or similar tools.  
Default: False  
Equivalent to: [--HTMLExporter.sanitize\_html]  
--writer=<DottedObjectName>  
Writer class used to write the results of the conversion  
Default: 'FileWriter'

```

Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
PostProcessor class used to write the
                                         results of the conversion
Default: ''
Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
overwrite base name use for output files.
                                         can only be used when converting one notebook at a time.
Default: ''
Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
Directory to write output(s) to. Defaults
                                         to output to the directory of each notebook. To rec
over
                                         previous default behaviour (outputting to the curre
nt
                                         working directory) use . as the flag value.
Default: ''
Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
The URL prefix for reveal.js (version 3.x).
This defaults to the reveal CDN, but can be any url pointing to a copy
of reveal.js.
For speaker notes to work, this must be a relative path to a local
copy of reveal.js: e.g., "reveal.js".
If a relative path is given, it must be a subdirectory of the
current directory (from which the server is run).
See the usage documentation
(https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-sli
deshow)
                                         for more details.
Default: ''
Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
The nbformat version to write.
                                         Use this to downgrade notebooks.
Choices: any of [1, 2, 3, 4]
Default: 4
Equivalent to: [--NotebookExporter.nbformat_version]

```

## Examples

-----

The simplest way to use nbconvert is

```
> jupyter nbconvert mynotebook.ipynb --to html
```

Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides', 'wevpdf'].

```
> jupyter nbconvert --to latex mynotebook.ipynb
```

Both HTML and LaTeX support multiple output templates. LaTeX includes 'base', 'article' and 'report'. HTML includes 'basic', 'lab' and 'classic'. You can specify the flavor of the format used.

```
> jupyter nbconvert --to html --template lab mynotebook.ipynb
```

You can also pipe the output to stdout, rather than a file

```
> jupyter nbconvert mynotebook.ipynb --stdout  
PDF is generated via latex  
> jupyter nbconvert mynotebook.ipynb --to pdf  
You can get (and serve) a Reveal.js-powered slideshow  
> jupyter nbconvert myslides.ipynb --to slides --post serve  
Multiple notebooks can be given at the command line in a couple of  
different ways:  
> jupyter nbconvert notebook*.ipynb  
> jupyter nbconvert notebook1.ipynb notebook2.ipynb  
or you can specify the notebooks list in a config file, containing::  
c.NbConvertApp.notebooks = ["my_notebook.ipynb"]
```

```
> jupyter nbconvert --config mycfg.py
```

To see all available configurables, use `--help-all`.

```
CalledProcessError                                     Traceback (most recent call last)
<ipython-input-41-4ad7f0406dcb> in <cell line: 1>()
----> 1 get_ipython().run_cell_magic('shell', '', 'jupyter nbconvert --to html //content/Breast_Cancer_Survival_Prediction_01.ipynb\n')

/usr/local/lib/python3.10/dist-packages/google/colab/_shell.py in run_cell_magic(self, magic_name, line, cell)
    332     if line and not cell:
    333         cell = ''
--> 334     return super().run_cell_magic(magic_name, line, cell)
    335
    336

/usr/local/lib/python3.10/dist-packages/IPython/core/interactiveshell.py in run_cell_magic(self, magic_name, line, cell)
    2471             with self.builtin_trap:
    2472                 args = (magic_arg_s, cell)
-> 2473                 result = fn(*args, **kwargs)
    2474             return result
    2475

/usr/local/lib/python3.10/dist-packages/google/colab/_system_commands.py in _shell_cell_magic(args, cmd)
    110     result = _run_command(cmd, clear_streamed_output=False)
    111     if not parsed_args.ignore_errors:
--> 112         result.check_returncode()
    113     return result
    114

/usr/local/lib/python3.10/dist-packages/google/colab/_system_commands.py in check_returncode(self)
    135     def check_returncode(self):
    136         if self.returncode:
--> 137             raise subprocess.CalledProcessError(
    138                 returncode=self.returncode, cmd=self.args, output=self.output
    139             )

CalledProcessError: Command 'jupyter nbconvert --to html //content/Breast_Cancer_Survival_Prediction_01.ipynb
' returned non-zero exit status 255.
```