ML – עבודת גמר

‏2023

תוכן עניינים

[הסבר על הקוד – טיטניק : 2](#_Toc140354812)

[הסבר על הספריות והמודולים שמיובאים: 2](#_Toc140354813)

[הסבר על הפונקציות: 3](#_Toc140354814)

[הסבר MENE הקוד: 4](#_Toc140354815)

[ויזואליזציות של הנתונים 6](#_Toc140354816)

[סיכום : 10](#_Toc140354817)

[הסבר על הקוד – Android 13](#_Toc140354818)

[הסבר על הספריות והמודולים שמיובאים: 13](#_Toc140354819)

[הסבר על הפונקציות: 13](#_Toc140354820)

[הסבר MENE הקוד: 15](#_Toc140354821)

[ויזואליזציות של הנתונים 17](#_Toc140354822)

# הסבר על הקוד – טיטניק :

## הסבר על הספריות והמודולים שמיובאים:

1. sklearn.preprocessing.LabelEncoder: מחלקה זו משמשת להמרת משתנים קטגוריים למספריים. היא מייבאת פונקציות שמסייעות בהמרה של משתנים קטגוריים כמו תווים (strings) לקודים מספריים.

2. sklearn.preprocessing.OneHotEncoder: מחלקה זו משמשת להמרת משתנים קטגוריים למשתנים מורכבים. היא מפרקת משתנה קטגורי עם n ערכים ל-n משתנים בינאריים, בהם כל אחד מהם מייצג ערך אחד מהערכים האפשריים של המשתנה המקורי.

3. sklearn.preprocessing.StandardScaler: מחלקה זו משמשת לביצוע סטנדרטיזציה על הנתונים. הסטנדרטיזציה היא תהליך שבו נערכים את הנתונים על ידי הפרמטרים "ממוצע" ו-"סטיית התקן". באמצעות הסטנדרטיזציה, הנתונים מתחלקים בסטיית התקן שלהם ומועברים להיות ממוצעים 0 וסטיית תקן 1. זה עשוי לשפר את ביצועי המודל ולעזור לטפל בבעיות של מידע שקולט מוצל וחריג.

4. sklearn.linear\_model.LogisticRegression: מחלקה זו מייצגת מודל לוגיסטי רגרסיה. זהו מודל למידת מכונה שמשמש לחיזוי ערכים קטגוריים, בדרך כלל בין שני ערכים אפשריים (לדוגמה, חיים/מתים).

5. sklearn.ensemble.RandomForestClassifier: מחלקה זו מייצגת מודל של עץ החלטה מבוסס על הרכבות של מחלקות עצים החלטה. המודל משמש לחיזוי תוצאות על פי המידע שנלמד באופן אוטומטי.

6. sklearn.model\_selection.train\_test\_split: פונקציה זו משמשת לחלוקת הנתונים לקבוצת אימון וקבוצת בדיקה. היא מקבלת את הנתונים ומחזירה את הנתונים המחולקים בהתאם ליחס שנקבע מראש.

7. sklearn.metrics.accuracy\_score: פונקציה זו משמשת לחישוב הדיוק של מודל חיזוי על נתוני בדיקה. היא משווה בין התוצאות הצפויות לבין התוצאות הנחזות על ידי המודל ומחזירה את האחוז הנכון.

8. matplotlib.pyplot: מודול זה מספק כלים ליצירת גרפים ותרשימים. הוא נכלל בתוך matplotlib ומשמש לגרפיקה איכותית.

9. seaborn: ספריית גרפיקה סטטיסטית המבוססת על matplotlib. היא מספקת ממשק גרפי מתקדם ליצירת תרשימים וויזואליזציה של נתונים סטטיסטיים.

10. Pandas: ספריה לניתוח וניהול נתונים טבלאיים. היא מספקת מבנים נתונים יעילים כמו DataFrame, המאפשרים עיבוד וניתוח נתונים בצורה נוחה ויעילה. פנדס משתמשת ב-Numpy לביצוע החישובים המתמטיים שבסיסם.

## הסבר על הפונקציות:

1. load\_data(path): הפונקציה מקבלת נתיב לקובץ CSV ומחזירה DataFrame שמייצג את הנתונים המצויים בקובץ.

2. data\_info(data): הפונקציה מקבלת DataFrame ומדפיסה מידע על הנתונים, כולל גודל הדאטהפריים, חמישה שורות ראשונות, מידע על עמודות הנתונים, תיאור סטטיסטי של הנתונים ומספר הערכים החסרים בכל עמודה.

3. data\_visualization(data, save\_path=None): הפונקציה מקבלת DataFrame ומציגה ויזואליזציות שונות של הנתונים. היא מציגה היסטוגרמות של עמודות הנתונים, גרפים של התפלגות הנתונים לפי המחלקה של "ניצול" ו"לא ניצול", גרפים של התפלגות הנתונים לפי מין הנוסע, גרף של התפלגות הנתונים לפי מין הנוסע והמחלקה שלו, גרף של התפלגות הנתונים לפי גיל הנוסעים, מפת חום של התפלגות הנתונים לפי המחלקה והמחלקה של הנוסעים, וגרפים נוספים. הפונקציה יכולה לשמור את התרשימים בקובץ אם נתת נתיב לשמירה (save\_path).

4. ticket\_check(tic): הפונקציה מקבלת רשימה של כרטיסים ומבצעת ניקוי של מידע הקשור לכרטיסים. היא עוברת על כל כרטיס, מפרקת את המידע על פי רווח וכתובת הכרטיס ומבצעת ניקוי של המידע. היא מחזירה רשימה חדשה עם המידע המעודכן.

5. pre\_processing(data): הפונקציה מקבלת DataFrame של הנתונים ומבצעת עיבוד מקדים על הנתונים. היא מפקסת ערכים חסרים בעמודת הגיל על ידי מילוי בממוצע, משנה את ערכי העמודה 'Cabin' להיות ערכים יחידים של האות הראשונה בשם הקבינה, ממירה את ערכי המין למספריים וממירה עמודת היציאה 'Embarked' למספריים גם כן. היא מסירה עמודות לא רלוונטיות ומבצעת One-Hot Encoding על עמודות המחלקה והקבינה. היא מחזירה DataFrame מעודכן.

6. train\_lr\_model(data): הפונקציה מבצעת אימון של מודל לוגיסטי רגרסיה על הנתונים. היא מחלקת את הנתונים למערכות אימון ובדיקה, מבצעת סטנדרטיזציה של הנתונים, ומאמן את המודל. היא מדפיסה את דיוק האימון והבדיקה ומחזירה את המודל המאומן.

7. train\_rf\_model(data): הפונקציה מבצעת אימון של מודל מחלקת עץ החלטה על הנתונים. היא מחלקת את הנתונים למערכות אימון ובדיקה, מבצעת סטנדרטיזציה של הנתונים, ומאמן את המודל. היא מדפיסה את דיוק האימון והבדיקה ומחזירה את המודל המאומן.

תוך כדי, הקוד טוען נתונים מקובץ CSV, מציג את המידע והויזואליזציות של הנתונים, מבצע עיבוד מקדים על הנתונים, ומאמן ומבצע חיזוי על מודלי למידת המכונה Logistic Regression ו-Random Forest Classifier. בנוסף, הקוד מציג גם את התוצאות של החיזוי על קבוצת בדיקה ושומר את התוצאות בקובץ CSV.

## הסבר MENE הקוד:

1. df = load\_data("\\Titanic\_train.csv"): השורה מטעין את קובץ הנתונים ומאתחלת את המשתנה df עם הנתונים מהקובץ.

2. data\_info(df): השורה מפעילה את הפונקציה data\_info על הנתונים על מנת להדפיס מידע על הנתונים.

3. women = df.loc[df.Sex == 'female']["Survived"]: השורה מבצעת סינון של הנתונים לפי המין "female" ומשמיעה עמודה ספציפית "Survived". התוצאה היא סדרה שמכילה את הערכים של העמודה "Survived" לנשים בלבד.

4. rate\_women = sum(women) / len(women): השורה מחשבת את אחוז הנשים ששרדו על פי הסדרה שנוצרה בשורה הקודמת. היא סוכמת את כל הערכים בסדרה ומחלקת אותם באורך הסדרה על מנת לקבל את הממוצע.

5. men = df.loc[df.Sex == 'male']["Survived"]: השורה מבצעת סינון דומה עבור הגברים ויוצרת סדרה עם הערכים של העמודה "Survived" עבור הגברים בלבד.

6. rate\_men = sum(men) / len(men): השורה מחשבת את אחוז הגברים ששרדו על פי הסדרה שנוצרה בשורה הקודמת, דומה לשורה 4.

7. data\_visualization(df, save\_path='visualization.png'): השורה מפעילה את הפונקציה data\_visualization על הנתונים על מנת ליצור ויזואליזציות של הנתונים. היא מציגה את התרשימים ויכולה לשמור אותם בקובץ בשם visualization.png.

8. train = pre\_processing(df): השורה מפעילה את הפונקציה pre\_processing על הנתונים על מנת לבצע עיבוד מקדים ולהכין את הנתונים לאימון.

9. lr\_regressor = train\_lr\_model(train): השורה מפעילה את הפונקציה train\_lr\_model על הנתונים מעובדים על מנת לאמן מודל לוגיסטי רגרסיה ולהכין את המודל המאומן.

10. rf\_clf = train\_rf\_model(train): השורה מפעילה את הפונקציה train\_rf\_model על הנתונים מעובדים על מנת לאמן מודל מחלקת עץ החלטה ולהכין את המודל המאומן.

11. df = load\_data("\\Titanic\_test.csv"): השורה מטעין את קובץ הנתונים ומאתחלת את המשתנה df עם הנתונים מהקובץ.

12. test\_df = df: השורה מעתיקה את הנתונים מdf למשתנה test\_df.

13. data\_visualization(df, save\_path='visualization\_test.png'): השורה מפעילה את הפונקציה data\_visualization על הנתונים על מנת ליצור ויזואליזציות של הנתונים מקובץ הבדיקה. היא מציגה את התרשימים ויכולה לשמור אותם בקובץ בשם visualization\_test.png.

14. test\_df = pre\_processing(test\_df): השורה מפעילה את הפונקציה pre\_processing על

הנתונים מקובץ הבדיקה על מנת לבצע עיבוד מקדים ולהכין את הנתונים לבדיקה.

15. test\_df.drop('LR\_y', inplace=True, axis=1): השורה מסירה את העמודה 'LR\_y' ממשתנה test\_df.

16. test\_df.drop('RF\_y', inplace=True, axis=1): השורה מסירה את העמודה 'RF\_y' ממשתנה test\_df.

17. sc = StandardScaler(): השורה מפעילה את המפעיל StandardScaler, שימושי עבור הסטנדרטיזציה של הנתונים.

18. X\_1 = sc.fit\_transform(test\_df): השורה מבצעת סטנדרטיזציה על הנתונים במשתנה test\_df.

19. LR\_y = lr\_regressor.predict(X\_1): השורה מבצעת חיזוי עבור הנתונים במשתנה test\_df באמצעות המודל המאומן lr\_regressor.

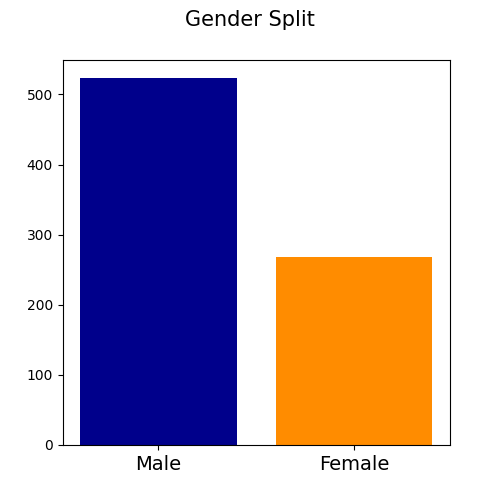
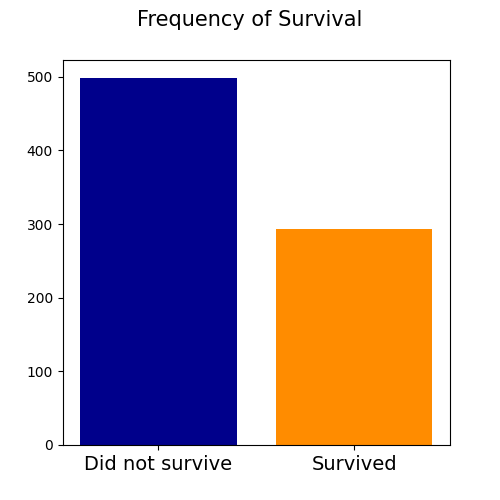
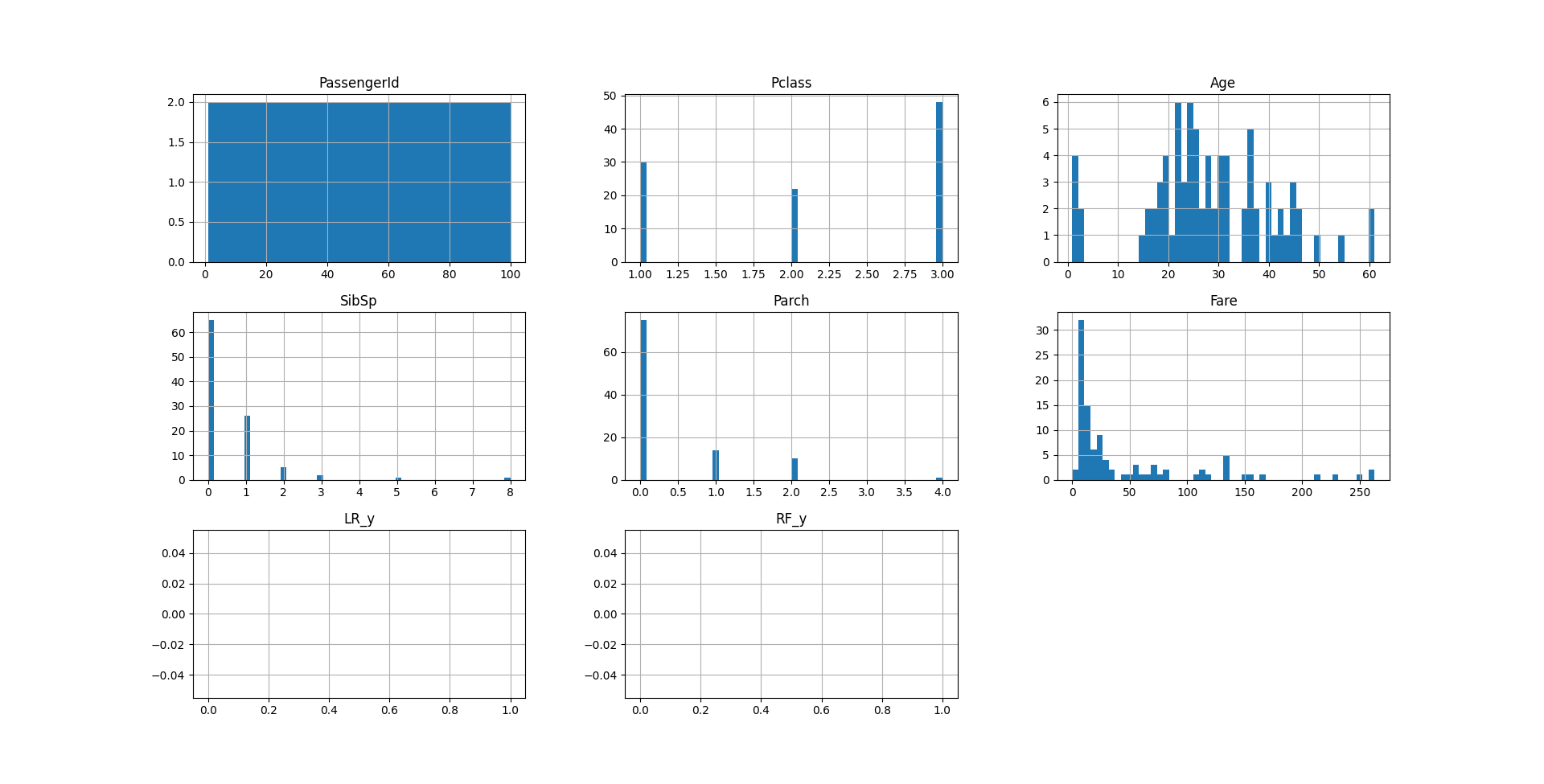
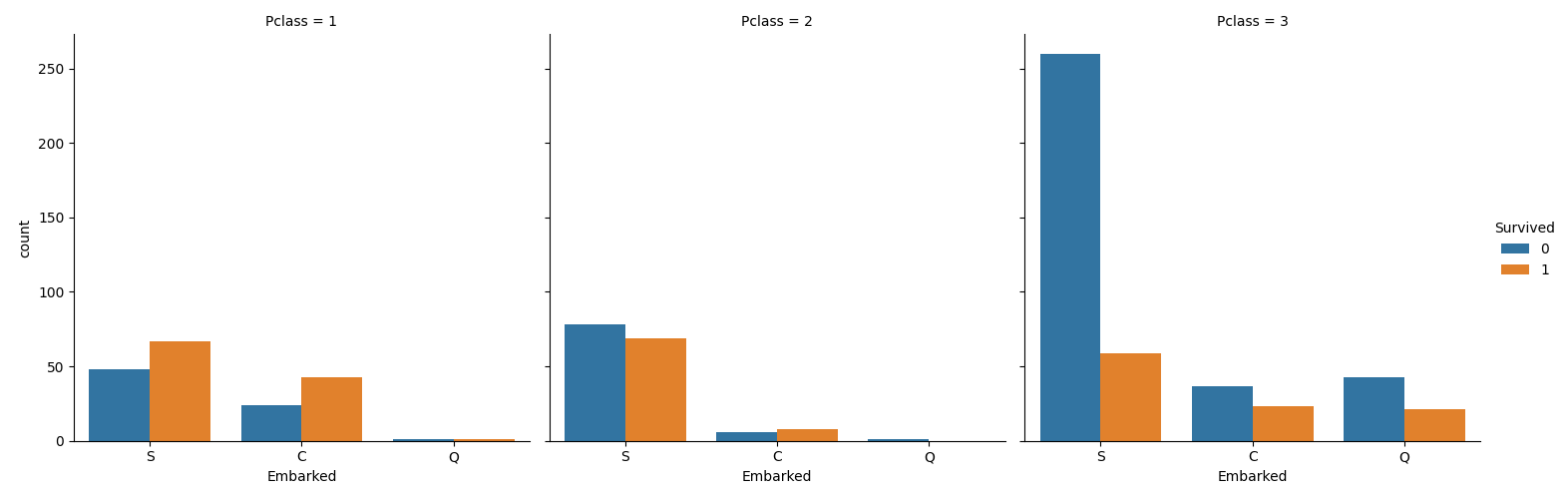
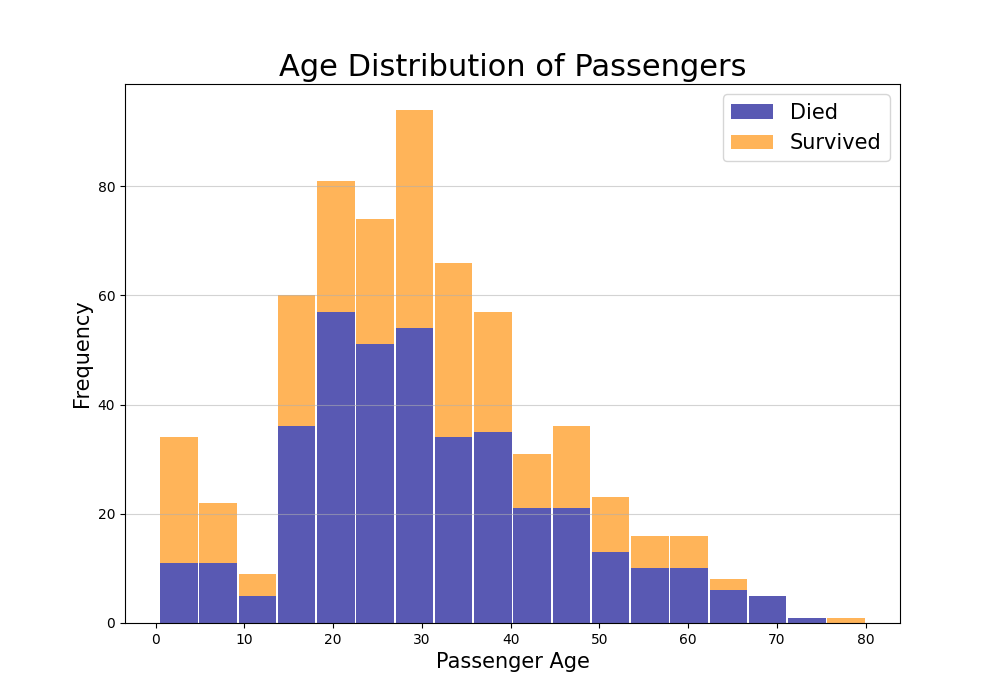
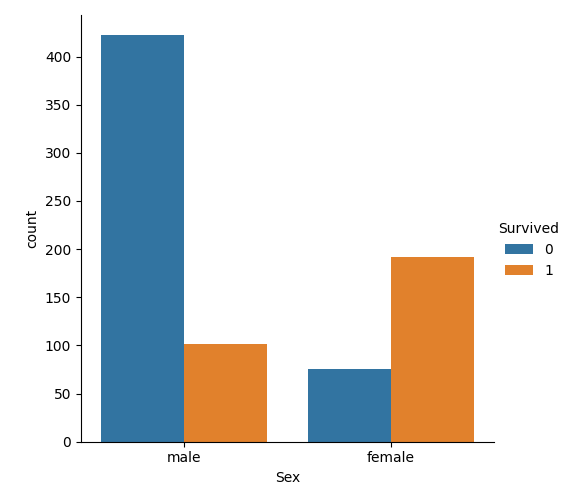
20. RF\_y = rf\_clf.predict(X\_1): השורה מבצעת חיזוי עבור הנתונים במשתנה test\_df באמצעות המודל המאומן rf\_clf.

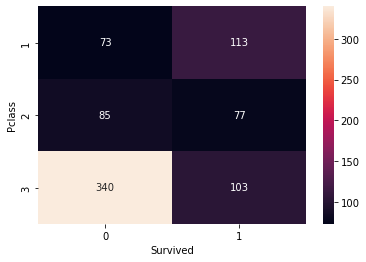
21. df['LR\_y'] = LR\_y: השורה מוסיפה את התוצאות החיזוי של lr\_regressor למשתנה df בעמודה בשם 'LR\_y'.

22. df['RF\_y'] = RF\_y: השורה מוסיפה את התוצאות החיזוי של rf\_clf למשתנה df בעמודה בשם 'RF\_y'.

23. df.to\_csv("Titanic\_test.csv", index=False): השורה שומרת את המשתנה df כקובץ CSV בשם Titanic\_test.csv, ללא כתובת שורת האינדקס.

## ויזואליזציות של הנתונים





## סיכום :

ניתן לראות מהנתונים :

1. שיש פחות נשים מגברים על הטיטניק (268 נשים לעומת 523) , האחוז נשים הינו 34%
2. אחוז השורדות הינו 72% , לעומת הגברים שהינו 19%.
3. אחוז הנשים ששרדו בClass 3 נמוך בהרבה מאחוז הנשים ששרדו ב Class 1 (47% מול 96% , אצל הגברים יש את אותו הכיוון אך לא בהפרש אחוזים גבוהה כמו אצל הנשים ( 14% מול 38%)

כדי לשרוד הכי טוב על הטיטניק עדיף להיות אישה ממחלקה ראשונה 😊

השתמשנו ב-2 אלגוריתמים :

1. Logistic Regression (רגרסיה לוגיסטית) : נחשבת לאחד מהאלגוריתמים הקלאסיים והמועילים ביותר בתחום המידע.

היתרונות של רגרסיה לוגיסטית:

* 1. פשוט להבנה וליישום: הרגרסיה לוגיסטית היא יחסית פשוטה להבנה וליישום. היא משתמשת בפונקצית לוגיסטית להסתברות סיווג, שהיא פונקציה חלקה וקשיחה בו זמנית.
  2. טוב למידת סיווג בינארי: רגרסיה לוגיסטית נפוצה במיוחד במקרים של מידת סיווג בינארי, כלומר כאשר יש רק שני קטגוריות סיווג (למשל כן/לא, חי/מת). היא מתאימה באופן מיוחד לסיווג מבוסס תכונות.
  3. עמידות גבוהה למערכות קטנות של נתונים: רגרסיה לוגיסטית מתאימה טוב למערכות קטנות של נתונים. היא עמידה למערכות נתונים שכוללות מספר יחידות נתונים נמוך.
  4. ניתוח יחסיות בין משתנים: רגרסיה לוגיסטית מאפשרת ניתוח יחסיות בין משתנים באמצעות מערכת היחסים שמופיעה במודל. ניתוח זה מאפשר להבין את ההשפעה של כל משתנה על התוצאה ולזהות את המשתנים המשפיעים ביותר על התוצאה.
  5. יכולת להתמודד עם פיצ'רים לינאריים ולא לינאריים: רגרסיה לוגיסטית יכולה להתמודד עם פיצ'רים לינאריים וגם עם פיצ'רים לא לינאריים, בעזרת עיבוד מתאים של הנתונים.
  6. יכולת לקבוע סיווג סיכון: רגרסיה לוגיסטית מאפשרת להעריך הסתברות סיווג סיכון בצורה מדויקת ולקבוע גבול סיווג מתאים, כך שניתן להפעיל מנגנונים לניהול סיכונים ולקבל החלטות מבוססות נתונים.

חסרונות :

* 1. תלות לינארית: רגרסיה לוגיסטית מניחה תלות לינארית בין המשתנים הסבירים לסיווג התוצאה. זה יכול להיות מגביל במציאת הקשרים המורכבים ולא לינאריים בנתונים.
  2. רגישות לערכים חריגים: רגרסיה לוגיסטית רגישה לערכים חריגים בנתונים. ערכים חריגים יכולים להשפיע על פרמטרים החשובים במודל ולהטעות את התוצאות הסופיות.
  3. רגישות לקולינאריות: אם ישנה קולינאריות גבוהה בין המשתנים המשמשים במודל, זה עשוי לגרום לבעיות בסבירות הסיווג.
  4. קשיים בפיתוח מודל מורכב: במקרים שבהם ישנן יותר מדי משתנים משמעותיים או התכונות אינן קלות להבנה ולתיאור, ייתכן שקשה לפתח מודל רגרסיה לוגיסטית אמין ויעיל.
  5. תלות מונוטונית: רגרסיה לוגיסטית מניחה תלות מונוטונית בין המשתנים הסבירים לסיווג התוצאה. זה אומר שהיא מניחה כי שינוי חיובי או שלילי בערכי המשתנים הסבירים ישפיע תמיד באותה כיוון על התוצאה. במציאות ייתכן שישנם יחסים מורכבים ולא לינאריים שאינם עומדים בהנחה זו.

1. Random Forest Classifier (מחלק עץ החלטה רנדומלי):הוא אלגוריתם מבוסס על קבוצה של עצי החלטה.

הייתרונות של Random Forest Classifier:

* 1. יעילות במידת סיווג: Random Forest Classifier מספק תוצאות מדויקות ויעילות במידת סיווג נתונים. הוא מסוגל לגרום לעצים המרכזיים ביותר לעבוד על הפיצ'רים הרלוונטיים ביותר ולסיווג את הנתונים באופן מדויק ומהיר.
  2. יכולת לסיווג במקרים מורכבים: Random Forest Classifier יכול להתמודד עם נתונים מורכבים ומגוונים, שכוללים פיצ'רים לינאריים ולא לינאריים. הוא מסוגל לזהות רעשים ופרטים חשובים בנתונים כדי ליצור דגימות עצים מדויקות.
  3. יכולת לקבוע חשיבות פיצ'רים: Random Forest Classifier מספק חשיבות פיצ'רים, שכן הוא מסוגל למדוד את התרומה של כל פיצ'ר לתהליך הסיווג. זה מאפשר לזהות פיצ'רים מרכזיים ולהבין אילו פיצ'רים יכולים להשפיע יותר על תהליך הסיווג.
  4. עמידות לערכים חריגים: Random Forest Classifier מתמודד בצורה טובה עם ערכים חריגים בנתונים, מכיוון שהוא מבצע סיווג על ידי ממוצע והצבעת מספר עצים. זה מפחית את ההשפעה של ערכים חריגים מופרזים על תהליך הסיווג.
  5. אין צורך בניתוח מוקדם של הנתונים: בניגוד למספר אלגוריתמים אחרים, Random Forest Classifier אינו דורש ניתוח מוקדם של הנתונים, כמו סיווג פיצ'רים או חיפוש ערכים חסרים. זה מאפשר ניצול מהיר של הנתונים ותוצאות מדויקות.
  6. עמידות למשתנים חסרים: Random Forest Classifier עמיד בפני משתנים חסרים בנתונים, ויכול להתמודד עם נתונים חסרים באופן אוטומטי.

החסרונות של Random Forest Classifier:

* 1. קומפלקסיות וזמן ריצה: Random Forest Classifier עשוי להיות מורכב ממספר עצים החלטה, ולכן עשוי להיות זמן ריצה יחסית ארוך יותר מאלגוריתמים יחידים אחרים. זה יכול להשפיע על ביצועים בזמן אמת או על משאבי המחשב הנדרשים.
  2. קשיים בהבנת המודל: Random Forest Classifier אינו תמיד קל להבנה ולתיאור, במיוחד כאשר המודל מורכב ממספר עצים החלטה. זה יכול לקשור לקומפלקסיות המודל ולקשיים בהבנת התהליכים המובנים במסגרת המודל.
  3. תלות בהגדרות הפרמטרים: כדי להשיג את הביצועים הטובים ביותר של Random Forest Classifier, יש צורך בהגדרת הפרמטרים הנכונה. זה יכול לכלול את מספר העצים באנסמבל, עומק העצים, מספר הפיצ'רים המשמשים בכל עץ ועוד. ניהול הפרמטרים עשוי להיות אתגר ויכול להשפיע על ביצועי המודל.
  4. קולינאריות: Random Forest Classifier מתמודד בצורה טובה עם קולינאריות בין הפיצ'רים, אך עדיין ייתכן שקולינאריות גבוהה תוביל לאובדן חשיבות של פיצ'רים או לדעיכת ביצועים במקרים מסוימים.
  5. קשיים בתיאום המודל: כאשר משתמשים ב Random Forest Classifier, ייתכן ויהיו קשיים בתיאום המודל עם דאטה חדש או נתונים מתעדכנים. כל עץ באנסמבל יכול להתייחס לפיצ'רים באופן שונה, ולכן ייתכן שיהיו קושיים בהתמודדות עם נתונים חדשים.
  6. אי יעילות במקרים עם מספר פיצ'רים גדול: כאשר ישנם מספר גדול של פיצ'רים בנתונים, Random Forest Classifier עשוי להיות פחות יעיל ולקחת יותר זמן להכשיל ולסווג את הנתונים. הזמן הנדרש לסיווג יכול לגדול באופן לא יחסי למספר הפיצ'רים.

# הסבר על הקוד – Android

## הסבר על הספריות והמודולים שמיובאים:

(לא הסברתי על מה שקיים בטיטניק אלא מה שבנוסף)

1. plotly.offline (מובנה כ-"py"): ספריית פייתון ליצירת תרשימים אינטראקטיביים. בקוד זה משמשת ליצירת תרשימים מתוך פרמטרים באופן מוקדם.
2. plotly.graph\_objs (מובנה כ-"go"): מודול מתוך ספריית plotly שמגדיר סוגי תרשימים ועיבוד נתונים עבור תרשימים אינטראקטיביים.
3. plotly.figure\_factory (מובנה כ-"ff"): מודול מתוך ספריית plotly שמאפשר בניית תרשימים מתקדמים יותר ומותאמים.
4. plotly.express (מובנה כ-"px"): מודול מתוך ספריית plotly שמאפשר בניית תרשימים עם פקודות פשוטות ומותאמות.
5. wordcloud.WordCloud ו-wordcloud.STOPWORDS: ספריית WordCloud מספקת כלים ליצירת ענני מילים לנתונים טקסטואליים. הספרייה ניתנת לשימוש כדי ליצור תרשימים מרהיבים שמראים את התדירות של מילים שונות בטקסט.
6. bs4.BeautifulSoup: מודול מסייע לניתוח ולניקוי קוד HTML ו-XML. בקוד זה משמש להסרת תגי HTML מטקסט.
7. re: מודול משמש לפעולות על ביטויים רגולריים, המתייחסות לטקסט וניתוחו.
8. string: מודול מכיל קבועים ופונקציות שקשורות למחרוזות
9. unicodedata: מודול מכיל פונקציות לעבודה עם תווים יוניקוד.
10. nltk: ספריית לעיבוד טקסט בפייתון. בקוד זה משמשת לניהול מילוני הפסוקים (stopwords) ולשימוש באלגוריתם הלמה.
11. keras.models.Sequential: מודול מתוך ספריית Keras, שמגדיר רכיבים ליצירת מודלים סיכוייים.
12. keras.layers.Dense: מודול מתוך ספריית Keras, המייצג שכבה רגילה ומלאה בפתחים. keras.optimizers.Adam: מודול מתוך ספריית Keras, המייצג אלגוריתם אופטימיזציה ליניארית כמו גם אלגוריתם תוכנות שונות.

## הסבר על הפונקציות:

1. `load\_data(path)`: פונקציה זו מקבלת נתיב לקובץ CSV ומחזירה טבלת נתונים מהקובץ המסופק. היא משתמשת בספריה pandas כדי לטעון את הנתונים מהקובץ ולהחזירם כטבלה.

2. `data\_info(data)`: פונקציה זו מקבלת את הטבלה שנטענה מהפונקציה הקודמת ומציגה מידע מקדים על הנתונים. היא מדפיסה את המידות של הטבלה (כמות השורות והעמודות), מדפיסה את החמש שורות הראשונות בטבלה, את מידע על כל עמודה (כמות הערכים בעמודה, טיפוס הנתונים), את התיאור הסטטיסטי של הנתונים (לדוגמה: ערכים ממוצעים, סטיית תקן וכו') ואת שמות העמודות בטבלה. כמו כן, הפונקציה בודקת עבור כל עמודה אם יש ערכים חסרים (NaN) ומדפיסה את העמודות שיש בהן ערכים חסרים.

3. `clean\_text(text, lemmatize=True)`: פונקציה זו מקבלת טקסט ובוצעים בה שורה של פעולות ניקוי ומועדון על הטקסט. פונקציה זו משמשת לניקוי הטקסט מתגי HTML, כמו גם הרחבת מילים קטנות (chatwords) ומילולון (contractions). לאחר מכן הפונקציה מבצעת ניקוי נוסף, כולל הסרת תווי אמוג'י, הוספת רווחים אחרי נקודות סיום והסרת קישורים (URLs). בסיום הפונקציה מבצעת עיבוד נוסף כגון הפיכת הטקסט לאותיות קטנות, סילוק פיסוק והפיכת המילים לקטנות בלבד

4. `data\_processing(data)`: פונקציה זו מקבלת את הטבלה שנטענה מהפונקציה `load\_data` ומבצעת עיבוד וניקוי נוספים על הנתונים. הפונקציה ממלאת ערכים חסרים בעמודת 'Dangerous permissions count' בממוצע של העמודה, וממלאת ערכים חסרים בעמודות 'Description' ו-'Related apps' בערך 'U'. לאחר מכן, הפונקציה מסירה ערכים כפולים ושורות עם ערכים חסרים. לבסוף, הפונקציה מבצעת ניקוי ומועדון על התיאורים והחבילות (packages) בעמודות 'Description' ו-'Package' באמצעות הפונקציה `clean\_text`.

5. `train\_data\_visualization(data)`: פונקציה זו מקבלת את הטבלה שכבר עברה עיבוד וניקוי מהפונקציה `data\_processing` ומבצעת חקירות והצגת נתונים ותרשימים שונים. הפונקציה מדפיסה כמה דברים כמו כמות התוכניות המאליציה והלא מאליציות בטבלה, מציגה תרשימה דו-ממדית שמראה כמה תוכניות בכל קטגוריה הן מאליציות וכמה הן לא מאליציות, וגם מציגה תרשים נוסף המראה כמה תוכניות בכל קטגוריה הן מזיקות וכמה הן לא מזיקות. הפונקציה יוצרת גם שני ענפי התרשימים האחרונים באמצעות ספריית Plotly, כדי להציג אותם בצורה יעילה ואינטראקטיבית. כמו כן, הפונקציה מציגה שני ענפי התרשימים האחרונים באמצעות ספריית Seaborn, להצגת תרשימים נוספים וניתוח נתונים בפונקציה.

6. `ANN\_model(X, X\_train, y\_train)`: פונקציה זו מקבלת מטריצת יכולות, מטריצת אימון ו- y מטריצת תוויות אימון ומאמצת מודל רשת נוירונים (ANN). היא בונה מודל של ANN ומבצעת אימון עליו באמצעות הנתונים של X\_train ו- y\_train. הפונקציה משתמשת במדריך Adam לאופטימיזציה, והיא מחזירה את המודל שהתקבל לאחר האימון.

7. `LR\_model(X\_test, y\_test, X\_train, y\_train)`: פונקציה זו מקבלת מטריצות אימון ובדיקה, ומכניסה את הנתונים למודל של רגרסיה לוגיסטית (Logistic Regression). היא מבצעת אימון על הנתונים של X\_train ו- y\_train, ומחזירה את המודל. לאחר מכן, הפונקציה משתמשת במודל לבצע חיזוי על הנתונים במטריצת X\_test ומחשבת דיוק באמצעות מדד ה- accuracy\_score.

8. `test\_pre\_processing(test\_ann)`: פונקציה זו מקבלת טבלת נתונים (בדרך כלל מטריצת בדיקה) ומבצעת עיבוד וניקוי על הנתונים. היא משתמשת בפונקציה `clean\_text` לניקוי התיאורים שבטבלה ומסירה עמודות לא רלוונטיות (לדוגמה: 'Package', 'Description', 'Related apps' ו-'App'). כמו כן, הפונקציה ממירה את העמודה 'Category' לערכים מספריים באמצעות הקידוד Label Encoding, ויוצרת משתנה דמי עבור כל ערך יחיד ב- Category כדי להוסיף אותו כעמודה נפרדת ולקבל תיקיית עוגיות שמכילה נתונים אפשריים עבור המודל.

## הסבר MENE הקוד:

1. `df = load\_data(...)`:נטענים הנתונים מקובץ ה-CSV עליו מתבצע האימון. המשתנה `df` מכיל טבלת נתונים.
2. `data\_info(df)`: מופעלת הפונקציה `data\_info`, שמדפיסה מידע על הנתונים בטבלה, כולל כמות השורות והעמודות, מידע על העמודות והערכים שבהן (כמות ערכים, טיפוס הנתונים וכו'), ממוצעים, וכמה ערכים חסרים יש בכל עמודה.
3. `df = data\_processing(df)`: מפעיל את הפונקציה `data\_processing` על הנתונים בטבלה, כדי לבצע עיבוד נוסף שלהם, לכלול אימות (lemmatization) של התיאורים, ולמחוק שורות או עמודות שבהן יש נתונים חסרים. המשתנה `df` מכיל את הטבלה לאחר העיבוד.
4. ממשיכים עם תהליך האימון:
   * 1. מקודדים את העמודה "Category" כערכים מספריים באמצעות הקידוד Label Encoding.
     2. יוצרים עמודות נפרדות עבור ערכים ייחודיים בעמודה "Category" כעמודות נפרדות עם One-Hot Encoding באמצעות `pd.get\_dummies`.
     3. משחררים את העמודה "Category\_29", שנוצרה על ידי הקידוד הייחודי, מהטבלה באמצעות הפונקציה `result.drop(['Category\_29'], axis=1, inplace=True)`.
     4. מסירים את העמודה "Category" מהטבלה, כי נמצא, כי כל הערכים הם אפשריים ואין צורך בשורה נפרדת לכל ערך בעמודה.
5. `s = result`: מאתחלים את המשתנה `s` כטבלת הנתונים אחרי העיבוד.
6. סינון של עמודות מיותרות: מסירים את העמודות "App", "Package", "Description" ו-"Related apps" מהטבלה ומעידים אותן ל-S
7. חילוץ תכונות ותוויות: הפרד את הנתונים לתכונות X (כל העמודות מלבד "Class") ותוויות y (העמודה "Class").
8. `sc = StandardScaler()`: מאתחלים מודל StandardScaler כדי לנרמל את התכונות על ידי שינוי קנה המידה שלהן לממוצע של 0 וסטיית תקן של 1.
9. `X\_train, X\_test, y\_train, y\_test = train\_test\_split(...)`: מפצלים את הנתונים לשני סטים - הסט האימון והסט הבדיקה. הסט האימון ישמש להכשרת המודל, והסט הבדיקה ישמש לבדיקת דיוק המודל.
10. `classifier = ANN\_model(X, X\_train, y\_train)`: יוצרים ומאמנים את המודל מסוג ANN (רשת נוירונים). משתנה `classifier` מכיל את המודל לאחר האימון.
11. `print(classifier.evaluate(X\_test, y\_test)[1])`: בודקים את דיוק המודל בעזרת הסט הבדיקה ומדפיסים את התוצאה.
12. `test\_ann = load\_data(...)`: מטענים את הנתונים מקובץ ה-CSV של קבוצת הבדיקה (לא האימון) ומאחסנים אותם במשתנה `test\_ann`.
13. `test\_ann.isnull().sum()`: מחשבים את כמות הערכים החסרים בכל עמודה שבטבלה.
14. `test\_ann = test\_pre\_processing(test\_ann)`: מטפלים בנתוני הבדיקה באמצעות הפונקציה `test\_pre\_processing`, שמעבירה עליהם עיבוד דומה למה שנעשה עם נתוני האימון.
15. קידוד מחדש: משתמשים שוב במודל של `StandardScaler()` כדי לנרמל את נתוני הבדיקה.
16. `classifier.predict(X\_ann\_test)`: מזינים את נתוני הבדיקה למודל ANN כדי לבצע חיזוי עליהם. התוצאה מועברת למשתנה `ann\_pred`.
17. הופכים את התוצאות לתוויות: הפעולה מעבירה את תוצאות החיזוי של מודל ANN לתוויות סופיות באמצעות תנאי פשוט - אם ערך חיזוי הסיכוי הוא מעל 0.5, הוא יסומן כ-1, ואחרת יסומן כ-0.
18. `regressor = LR\_model(...)`: מאמנים את המודל מסוג Logistic Regression על הסט הבדיקה. באמצעות המודל, נבצע חיזוי נוסף על הסט הבדיקה, והתוצאות נשמרות במשתנה `LR\_y`.
19. מכניסים את התוצאות של החיזויים לטבלה עם הנתונים של קבוצת הבדיקה. מסירים את העמודה "predicted\_y" אם כבר קיימת, ואז מוסיפים אותה עם תוצאות החיזוי של המודל ANN ושל מודל הרגרסיה לוגיסטית.
20. בסופו של דבר, שומרים את הטבלה בקובץ חדש באמצעות הפונקציה `to\_csv`.

## ויזואליזציות של הנתונים

