בס"ד

Names: Avigail shekasta, 209104314

Ofri rom, 208891804

Dan monsonego, 313577595

## תיאור של מבנה הפרויקט:

הפרוייקט שלנו מבוסס על ממשק GUI אשר אותו אנו מפעילים דרך streamlit

ראשית כל, מכניסים את הניתוב לקובץ ה- data ועמודת הסיווג ובוחרים איך למלא ערכים חסרים, לאחר מכן אנו מבצעים מחיקת שורות עם עמודת סיווג ללא ערך, וממלאים את הערכים החסרים לפי בחירת המשתמש (ביחס לערך סיווג או ביחס לכלל הנתונים)למשתמש יש אפשרות לנרמול לפי min-max מתוך SKLEARN . משתמש יכול לבחור האם תתקיים דיסקרטיזציה ובמידה וכן, איזה(עומק שווה, רוחב שווה או מבוסס אנטרופיה).לאחר מכן יש ללחוץ על כפתור " save clean data " על מנת שה -data ישמר בקובץ. ולאחר מכן המשתמש בוחר איזה מודל הוא רוצה להפעיל(עץ שממומש על ידנו או עץ מוכן או naive bayes שממומש על ידנו או naive bayes מוכן או K-MEANS או KNN )ולבסוף המשתמש לוחץ על כפתור המטריצה שאותה ירצה לראות(ישנם 3 סוגים, 1-לפי קובץ האימון, 2- לפי קובץ הבדיקה, 3-לפי הביצועים של המודל(מציין גם את רמת הדיוק)).

## תפקיד של כל קובץ:

models_preprocess- בקובץ זה נשמר כל המידע שלנו( data, model ועוד), וגם רשומים כל הפונקציות שלנו(נרחיב בהמשך).

GUI- בקובץ זה ממומש ממשק ה-GUI שמפעיל את הפונקציות שבקובץ "models_preprocess" בהתאם לבחירה של המשתמש.

## תפקיד של כל פונקציה:

install_and_import- הורדת ספריות

get_df- יוצרת data frame מהנתיב שהיא מקבלת

drop_rows- מחיקת שורות עם עמודת סיווג ללא ערך

fill_mean_values- ממלא את הערכים החסרים בעמודה לפי ממוצע

fill_mode_values- ממלא את הערכים החסרים בעמודה לפי ערך שכיח

sub_fill_data- ממלא ערכים חסרים בכל העמודות(לא כולל עמודת סיווג) עבור ערכים רציפים משלים ממוצע, ועבור ערכים בדידים משלים שכיח.

main_fill_data- מפעיל את הפונקציה "sub_fill_data " בהתאם להשלמה ביחס לערך סיווג או ביחס לכלל הנתונים

Conversion_to_number- עובר על כל ה- data וממיר אותו מ-str ל-int

Normalization- מנרמל לפי minmax מתוך SKLEARN

Equal_width- דיסקרטיזציה רוחב שווה, ממומש על ידנו

Equal_frequency_discretization- דיסקרטיזציה עומק שווה, ממומש על ידנו

entrophy_based_binning- דיסקרטיזציה מבוסס אנטרופיה, מוכן

בס"ד

split_data- מחלק את ה-data ל- train ו-test

id3_by_us- מודל עץ החלטה שממומש על ידנו

Predict- מחשב את ה-predict  של המודל עץ החלטה שעשינו

Test-משמשת לאימות תוצאות הניבוי של המודל מול הערכים המקוריים של עמודת הסיווג(בשביל המודל עץ החלטה שעשינו)

Entropy-מחשב אנטרופיה של עמודה (בשביל המודל עץ החלטה שעשינו)

InfoGain- מחשב את ה- InfoGain  של עמודה (בשביל המודל עץ החלטה שעשינו)

id3- מודל עץ החלטה מוכן

Naive_bayes_by_us- מודל Naïve bayes שממומש על ידנו

Query- מחשב את הערך לפי מודל Naïve  bayes , שממומש על ידנו

Naive_bayes- מודל Naïve bayes , מוכן

Knn- מודל Knn , מוכן

Kmeans- מודל kmeans , מוכן

Save- שומר את התוצאה בקובץ (ה-data לאחר הניקיון)

pickl_model_save- שומר את המודל כקובץ (pickle)

pickl_matrix_etc- שומר את תוצאות האימון והבדיקה, נתוני "חוק הרוב" עבור הניסוי ונתוני העיבוד המקדים שנבחרו כקובץ (pickle)

matrix_performace- מטריצה לפי הביצועים של המודל מציין גם את רמת הדיוק

matrix_train- מטריצה לפי קובץ האימון

matrix_test- מטריצה לפי קובץ הבדיקה

acc-רמת דיוק

majority_test- מחשב לפי "חוק הרוב"

create_web_page- יוצר את ה-web

d- מילון ששומר את ה-data

model_dict- מילון ששומר את המודל, שם המודל,Predict  ועוד

result_dict- מילון ששומר את תוצאות האימון והבדיקה, נתוני "חוק הרוב" עבור הניסוי ונתוני העיבוד המקדים שנבחרו

בס"ד

# EDA:

להלן ה DB אשר איתו אנו נעבוד:

| | class | cap-shape | cap-surface | cap-color | bruises | odor | gill-attachment | gill-spacing | gill-size | gill-color | ... | stalk-surface-below-ring | stalk-color-above-ring | stalk-color-below-ring | veil-type | veil-color | ring-number | ring-type | spore-print-color | population | habitat |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | e | x | s | n | t | p | f | c | n | k | ... | s | w | w | p | w | o | p | k | s | u |
| **1** | e | x | s | y | t | a | f | c | b | k | ... | s | w | w | p | w | o | p | n | n | g |
| **2** | e | b | s | w | t | l | f | c | b | n | ... | s | w | w | p | w | o | p | n | n | m |
| **3** | e | x | y | w | t | p | f | c | n | n | ... | s | w | w | p | w | o | p | k | s | u |
| **4** | e | x | s | g | f | n | f | w | b | k | ... | s | w | w | p | w | o | e | n | a | g |

כעת נרצה לנתח וללמוד את מערך הנתונים אשר איתו נעבוד, ז"א נרצה לדעת אילו פעולות נצטרך

לבצע על מנת שנביא את ה DB שלנו למצב שבו נוכל לעבוד איתו, נתחיל בניתוח פשוט של אילו שדות

חסרים לנו, על מנת שנוכל להשלים אותם במידת הצורך

```
class                     0
cap-shape                 0
cap-surface               0
cap-color                 0
bruises                   0
odor                      0
gill-attachment           0
gill-spacing              0
gill-size                 0
gill-color                0
stalk-shape               0
stalk-root             2480
stalk-surface-above-ring  0
stalk-surface-below-ring  0
stalk-color-above-ring    0
stalk-color-below-ring    0
veil-type                 0
veil-color                0
ring-number               0
ring-type                 0
spore-print-color         0
population                0
habitat                   0
dtype: int64
```

בס"ד

כפי שניתן לראות בטבלה העמודה stalk-root עם 2480 ערכים חסרים, מכיוון שזו עמודה בדידה נמלא את התאים בערך השכיח.

במידה והיו לנו שורות עם ערכי class חסרים, היינו נדרשים למחוק אותן, מאחר ולא נוכל ללמוד

משורות אלו אף מידע, מאחר ולא נדע לאיזה סיווג לשייך מידע זה .

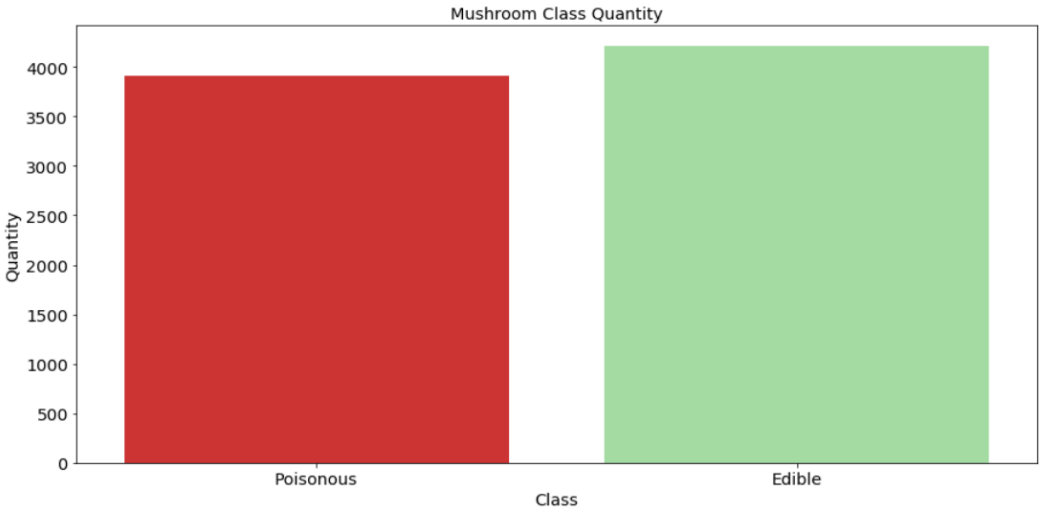
```
df["stalk-root"] = df["stalk-root"].fillna(df["stalk-root"].mode()[0])
```

וכעת נבדוק שוב את הDB

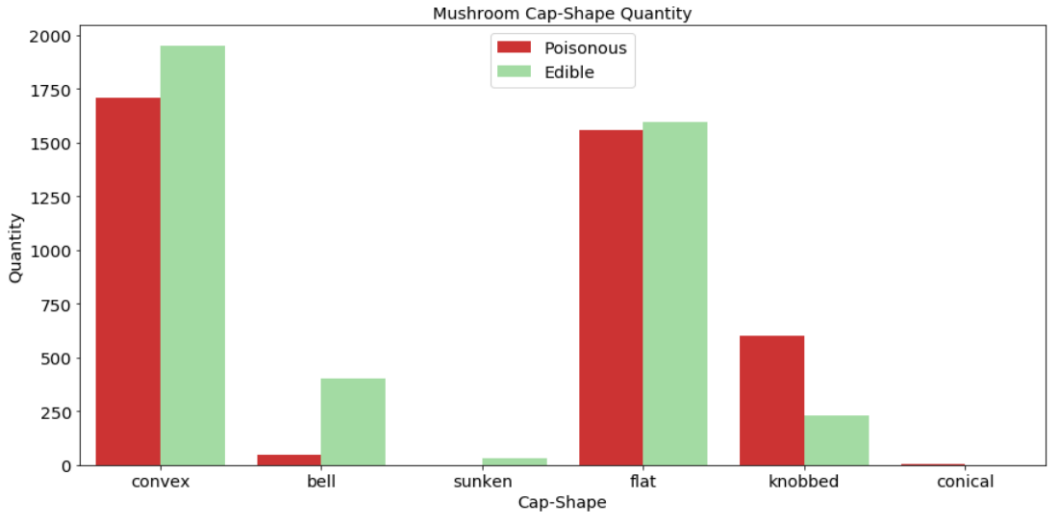
```
class                     0
cap-shape                 0
cap-surface               0
cap-color                 0
bruises                   0
odor                      0
gill-attachment           0
gill-spacing              0
gill-size                 0
gill-color                0
stalk-shape               0
stalk-root                0
stalk-surface-above-ring  0
stalk-surface-below-ring  0
stalk-color-above-ring    0
stalk-color-below-ring    0
veil-type                 0
veil-color                0
ring-number               0
ring-type                 0
spore-print-color         0
population                0
habitat                   0
dtype: int64
```

בס"ד

## גרפים וויזואליזציות:

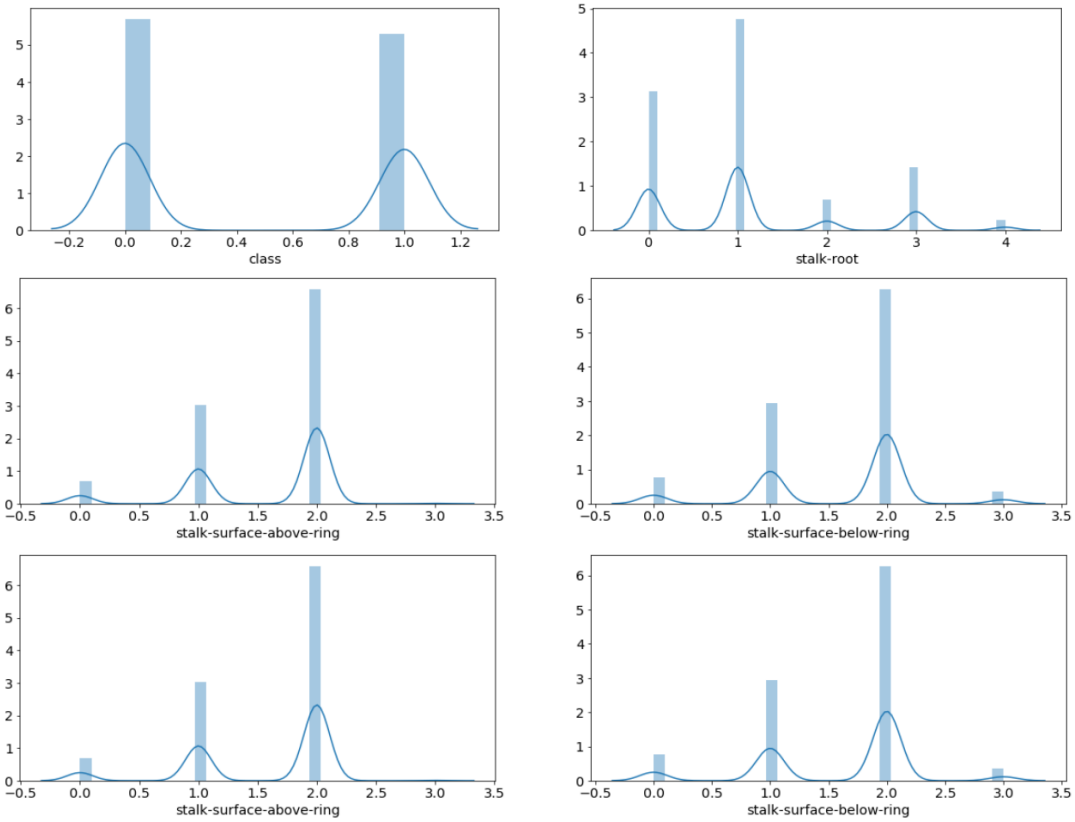



ניתן לראות כי יש יותר פטריות אכילות מאשר רעילות בdb שלנו, בנוסף ניתן לראות כי הDB שלנו מתפלג כמעט חצי חצי




ניתן לראות כי פטריה בעלת צורת sunkenהיא לא רעילה כלל

בס"ד



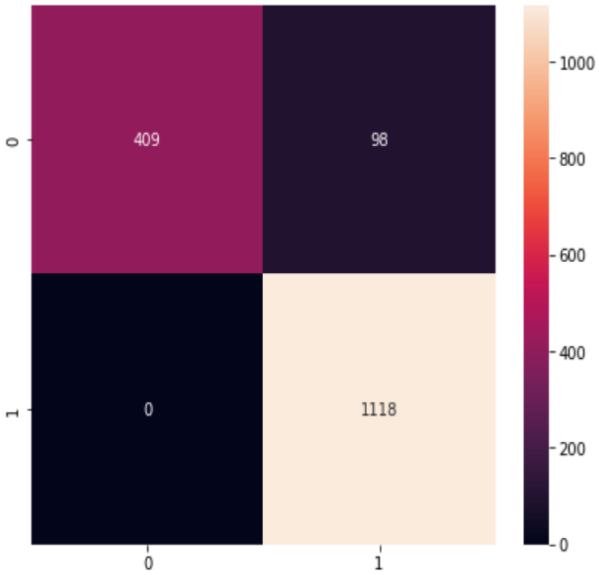

ניתן לראות את התפלגות הפטריות עבור כל סוג של גבעול

בס"ד

# Model:

כעת נרצה לבחון הגדרות עבור המודל שלנו, נציג הרצת מדדי הערכה לפי כמות הבינים בדיסקרטיזציה של עמודות נומריות במודלים שלנו :

מודל Naïve Bayes מימוש שלנו:




```
[ ] x = accuracy_score(model_dict["y_test"], model_dict["predict"])
    print("Result of accuracy test: " + str(x) )
```

```
Result of accuracy test: 0.9396923076923077
```

```
[ ] x = recall_score(model_dict["y_test"], model_dict["predict"])
    print("Result of recall test: " + str(x) )
```

```
Result of recall test: 1.0
```

```
[ ] x = precision_score(model_dict["y_test"], model_dict["predict"])
    print("Result of precision test: " + str(x) )
```

```
Result of precision test: 0.9194078947368421
```

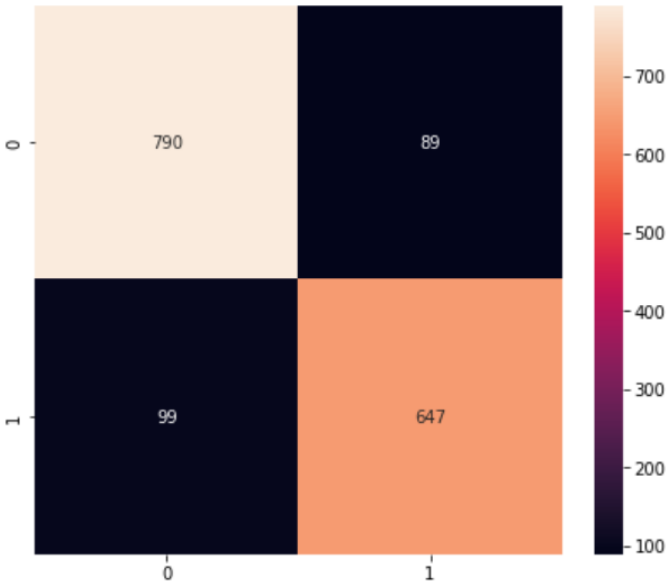
```
x = f1_score(model_dict["y_test"], model_dict["predict"])
print("Result of F-measure test: " + str(x) )
```

```
Result of F-measure test: 0.958011996572408
```

```
x = roc_auc_score(model_dict["y_test"], model_dict["predict"])
print("Result of auc_roc test: " + str(x) )
```

```
Result of auc_roc test: 0.903353057199211
```

מודל Naïve Bayes מוכן:




```
[ ] x = accuracy_score(model_dict["y_test"], model_dict["predict"])
    print("Result of accuracy test: " + str(x) )
```

```
Result of accuracy test: 0.8843076923076924
```

```
[ ] x = recall_score(model_dict["y_test"], model_dict["predict"])
    print("Result of recall test: " + str(x) )
```

```
Result of recall test: 0.8672922252010724
```

```
x = precision_score(model_dict["y_test"], model_dict["predict"])
print("Result of precision test: " + str(x) )
```

```
Result of precision test: 0.8790760869565217
```

```
[ ] x = f1_score(model_dict["y_test"], model_dict["predict"])
    print("Result of F-measure test: " + str(x) )
```

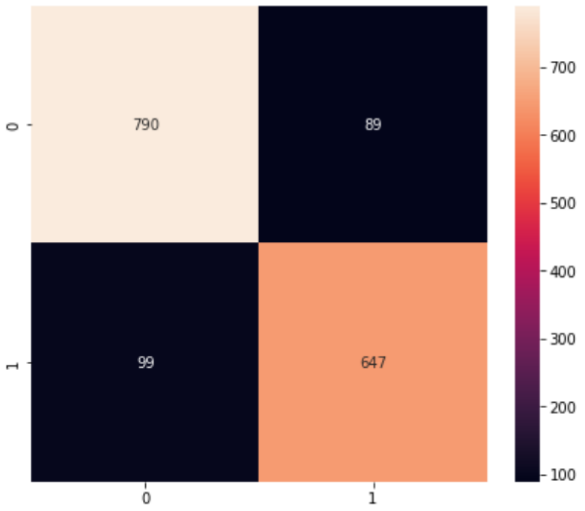
```
Result of F-measure test: 0.873144399460189
```

```
x = roc_auc_score(model_dict["y_test"], model_dict["predict"])
print("Result of auc_roc test: " + str(x) )
```

```
Result of auc_roc test: 0.8830204015652688
```

בס"ד

## מודל id3 מימוש שלנו:




```
[ ] x = accuracy_score(model_dict["y_test"], model_dict["predict"])
    print("Result of accuracy test: " + str(x) )
```

```
Result of accuracy test: 0.8843076923076924
```

```
x = recall_score(model_dict["y_test"], model_dict["predict"])
print("Result of recall test: " + str(x) )
```

```
Result of recall test: 0.8672922252010724
```

```
[ ] x = precision_score(model_dict["y_test"], model_dict["predict"])
    print("Result of precision test: " + str(x) )
```

```
Result of precision test: 0.8790760869565217
```

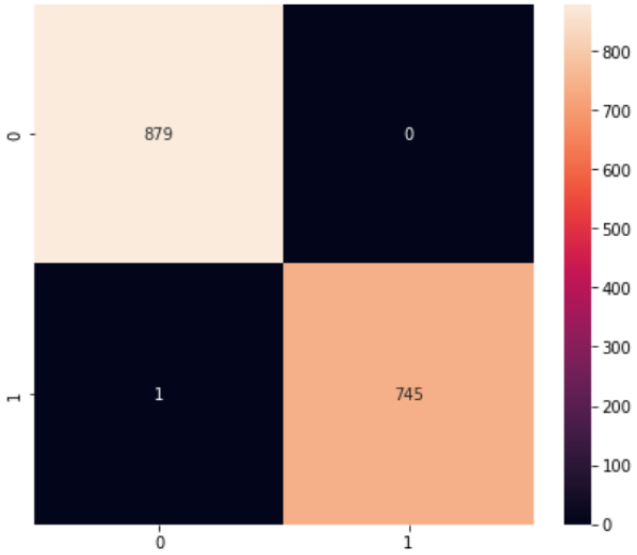
```
[ ] x = f1_score(model_dict["y_test"], model_dict["predict"])
    print("Result of F-measure test: " + str(x) )
```

```
Result of F-measure test: 0.873144399460189
```

```
x = roc_auc_score(model_dict["y_test"], model_dict["predict"])
print("Result of auc_roc test: " + str(x) )
```

```
Result of auc_roc test: 0.8830204015652688
```

## מודל id3 מוכן בעל עומק מרבי 5:




```
[ ] x = accuracy_score(model_dict["y_test"], model_dict["predict"])
    print("Result of accuracy test: " + str(x) )
```

```
Result of accuracy test: 0.9993846153846154
```

```
[ ] x = recall_score(model_dict["y_test"], model_dict["predict"])
    print("Result of recall test: " + str(x) )
```

```
Result of recall test: 0.9986595174262735
```

```
[ ] x = precision_score(model_dict["y_test"], model_dict["predict"])
    print("Result of precision test: " + str(x) )
```

```
Result of precision test: 1.0
```

```
[ ] x = f1_score(model_dict["y_test"], model_dict["predict"])
    print("Result of F-measure test: " + str(x) )
```

```
Result of F-measure test: 0.9993293091884642
```

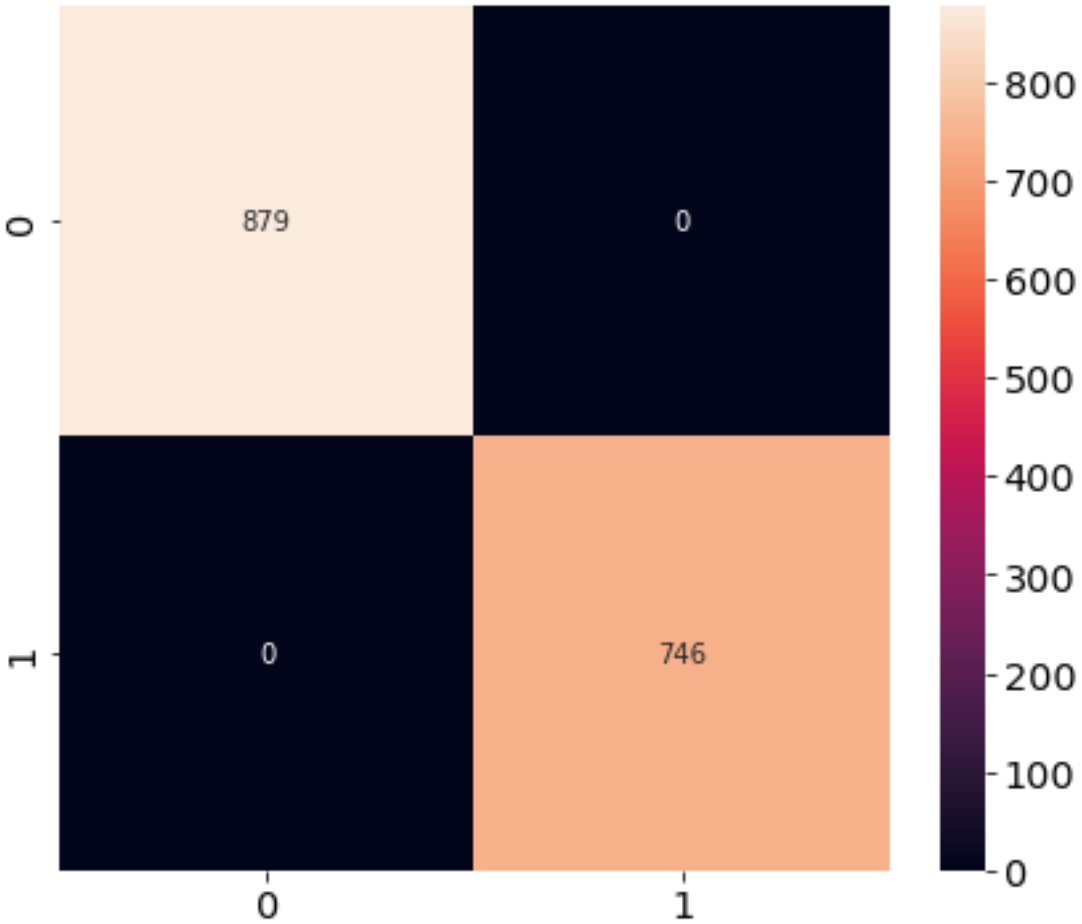
```
x = roc_auc_score(model_dict["y_test"], model_dict["predict"])
print("Result of auc_roc test: " + str(x) )
```

```
Result of auc_roc test: 1.0
```

בס"ד

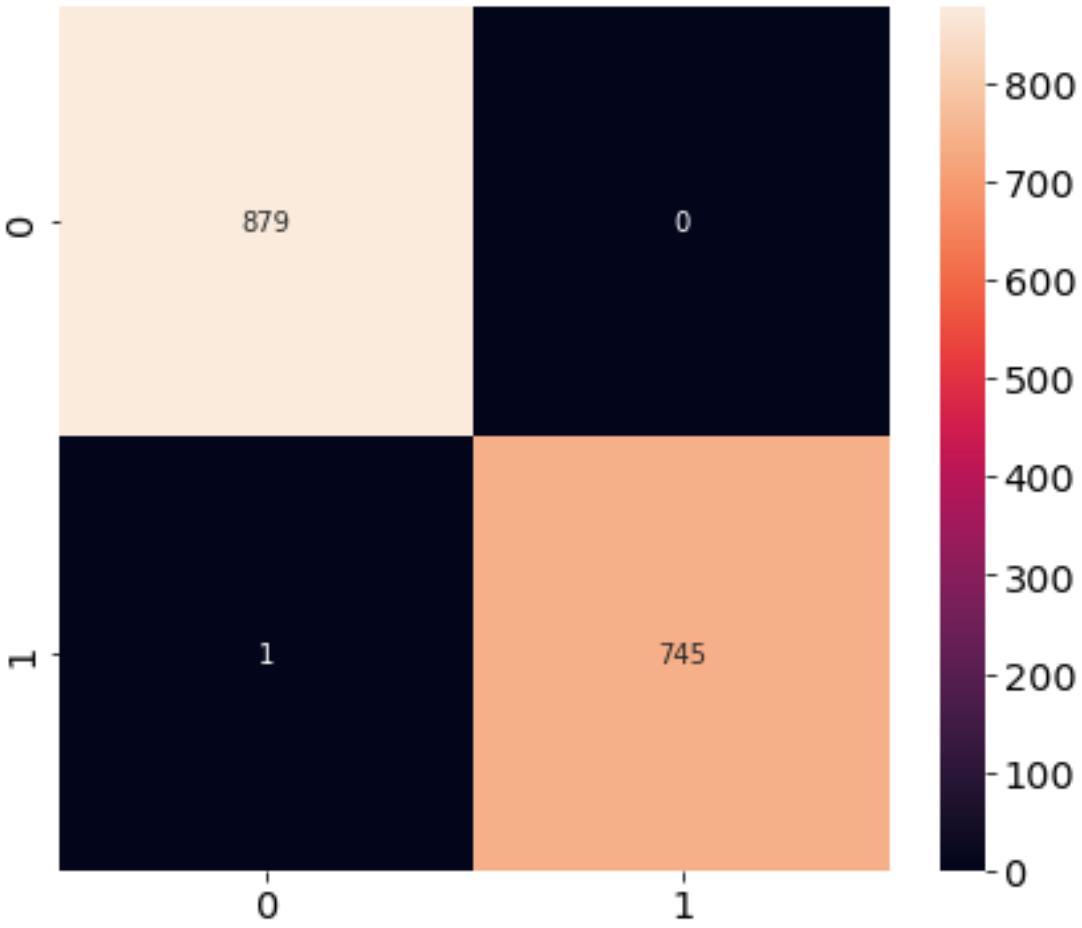
מודל KNN מימוש של sklearn : מימוש של 10 פרמטרים שונים בתור כמות החברים

```
Run with parameter: 1
```



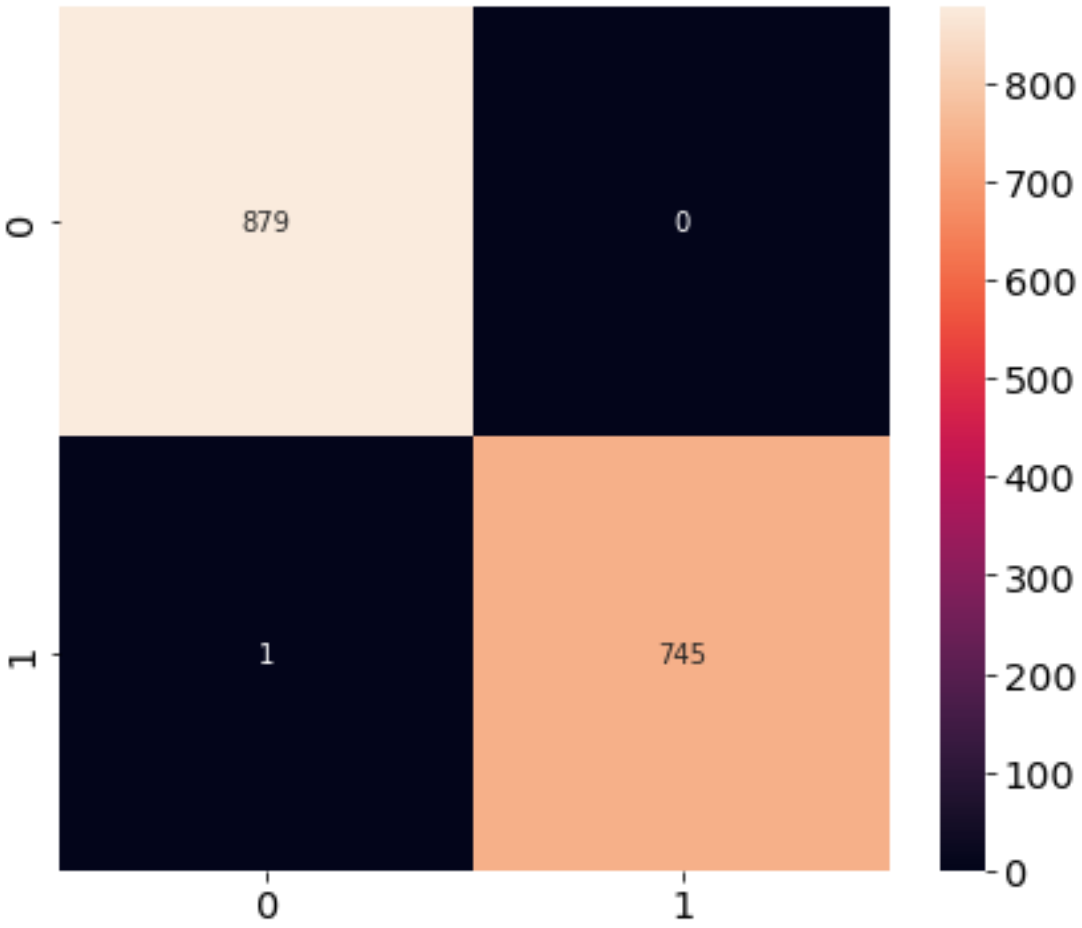

```
Result of accuracy test: 1.0
Result of recall test: 1.0
Result of precision test: 1.0
Result of F-measure test: 1.0
Result of auc_roc test: 1.0
End of run with parameter: 1
-------------------------------------------------------
-------------------------------------------------------
Run with parameter: 2
```

בס"ד



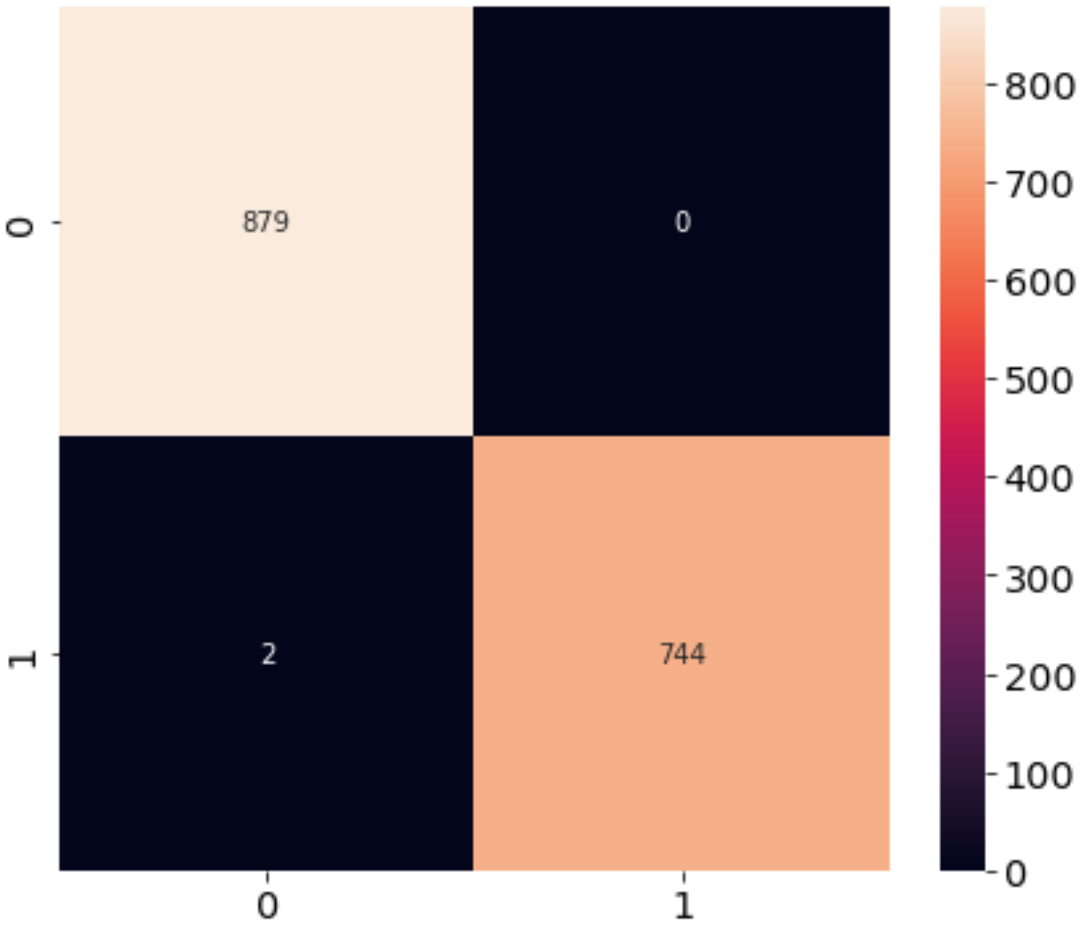

```
Result of accuracy test: 0.9993846153846154
Result of recall test: 0.9986595174262735
Result of precision test: 1.0
Result of F-measure test: 0.9993293091884642
Result of auc_roc test: 0.9993297587131367
End of run with parameter: 2
-------------------------------------------------------
-------------------------------------------------------
Run with parameter: 3
```

בס"ד



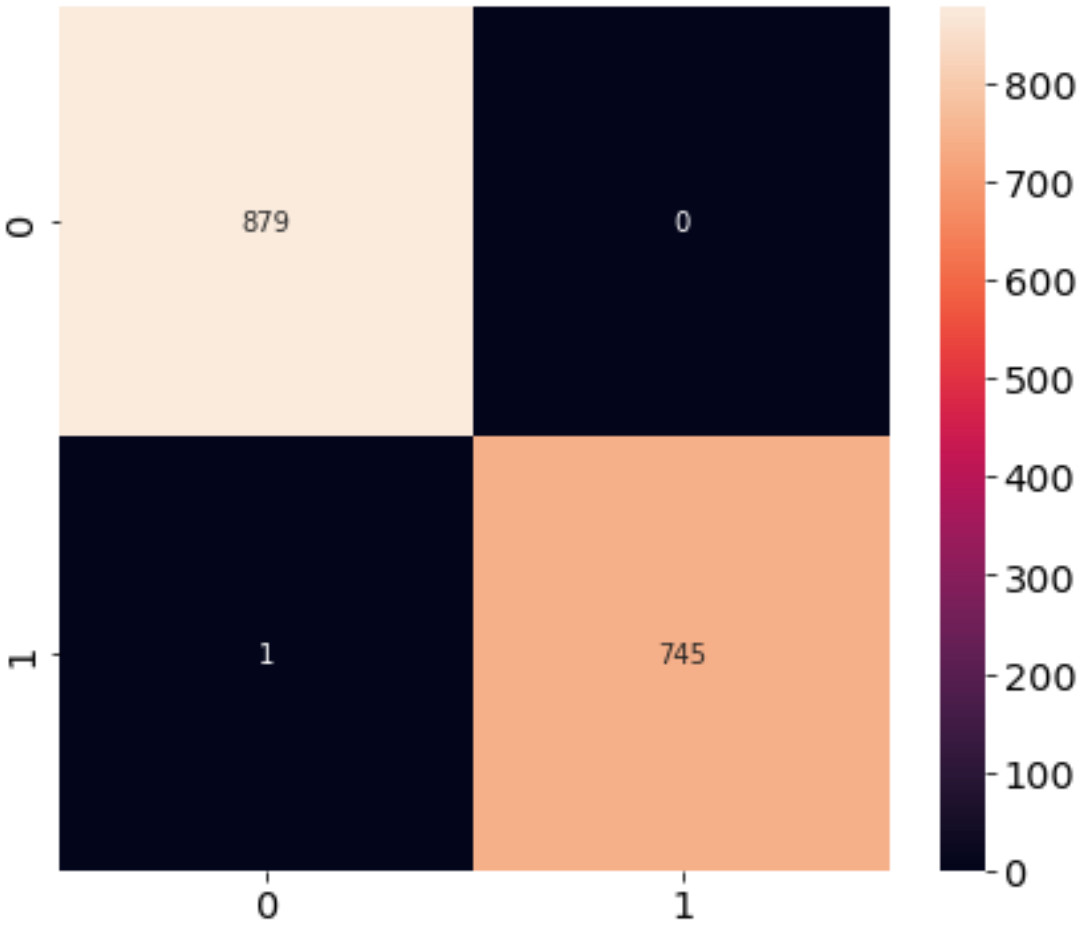

```
Result of accuracy test: 0.9993846153846154
Result of recall test: 0.9986595174262735
Result of precision test: 1.0
Result of F-measure test: 0.9993293091884642
Result of auc_roc test: 0.9993297587131367
End of run with parameter: 3
-------------------------------------------------------
-------------------------------------------------------
Run with parameter: 4
```

בס"ד



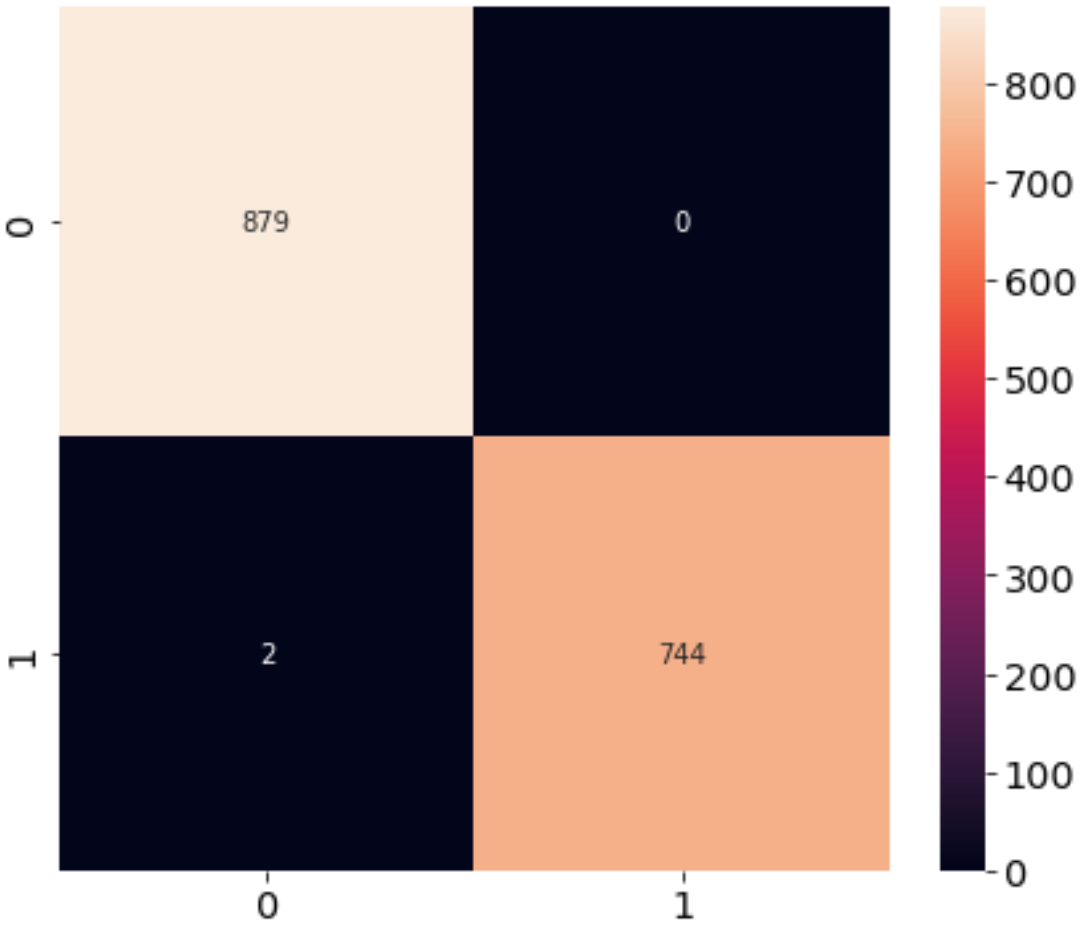

```
Result of accuracy test: 0.9987692307692307
Result of recall test: 0.9973190348525469
Result of precision test: 1.0
Result of F-measure test: 0.9986577181208054
Result of auc_roc test: 0.9986595174262735
End of run with parameter: 4
-------------------------------------------------------
-------------------------------------------------------
Run with parameter: 5
```

בס"ד



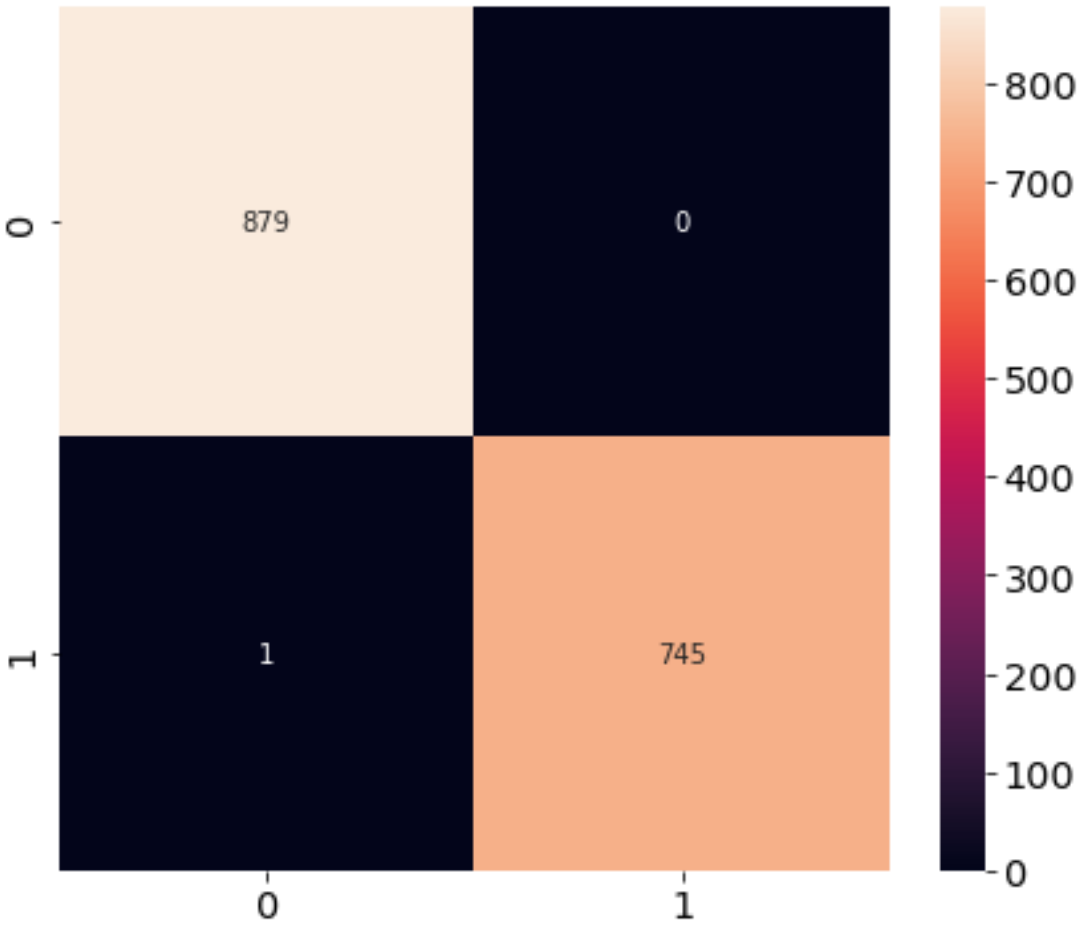

```
Result of accuracy test: 0.9993846153846154
Result of recall test: 0.9986595174262735
Result of precision test: 1.0
Result of F-measure test: 0.9993293091884642
Result of auc_roc test: 0.9993297587131367
End of run with parameter: 5
-------------------------------------------------------
-------------------------------------------------------
Run with parameter: 6
```

בס"ד



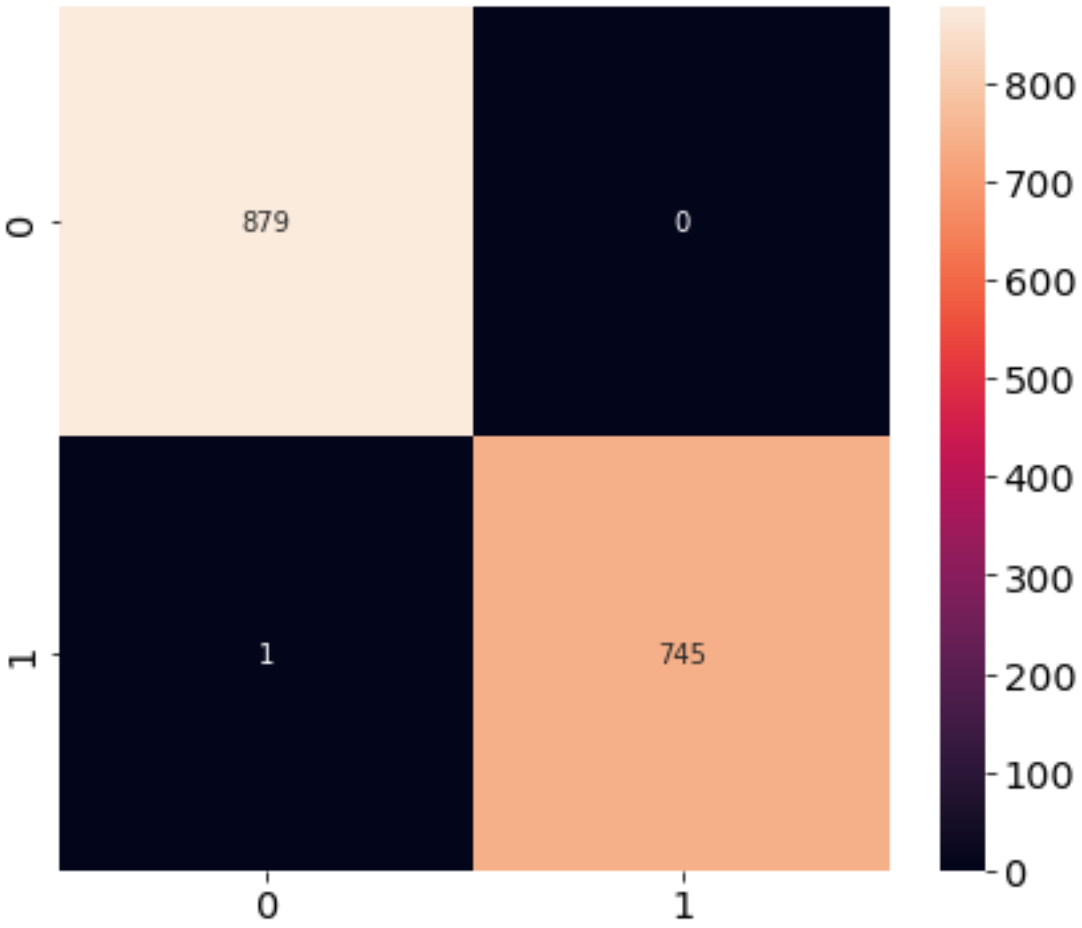

```
Result of accuracy test: 0.9987692307692307
Result of recall test: 0.9973190348525469
Result of precision test: 1.0
Result of F-measure test: 0.9986577181208054
Result of auc_roc test: 0.9986595174262735
End of run with parameter: 6
-------------------------------------------------------
-------------------------------------------------------
Run with parameter: 7
```

בס"ד



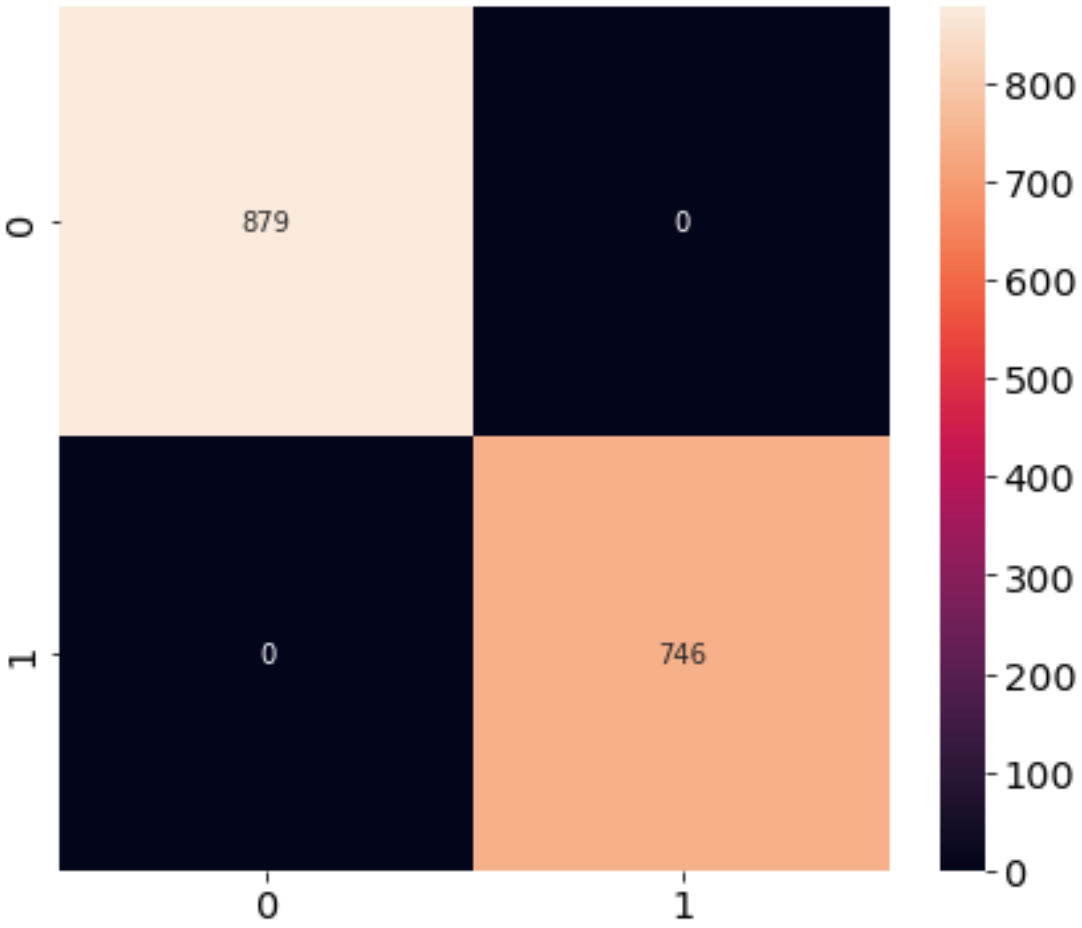

```
Result of accuracy test: 0.9993846153846154
Result of recall test: 0.9986595174262735
Result of precision test: 1.0
Result of F-measure test: 0.9993293091884642
Result of auc_roc test: 0.9993297587131367
End of run with parameter: 7
-------------------------------------------------------
-------------------------------------------------------
Run with parameter: 8
```

בס"ד




```
Result of accuracy test: 0.9993846153846154
Result of recall test: 0.9986595174262735
Result of precision test: 1.0
Result of F-measure test: 0.9993293091884642
Result of auc_roc test: 0.9993297587131367
End of run with parameter: 8
-------------------------------------------------------
-------------------------------------------------------
Run with parameter: 9
```

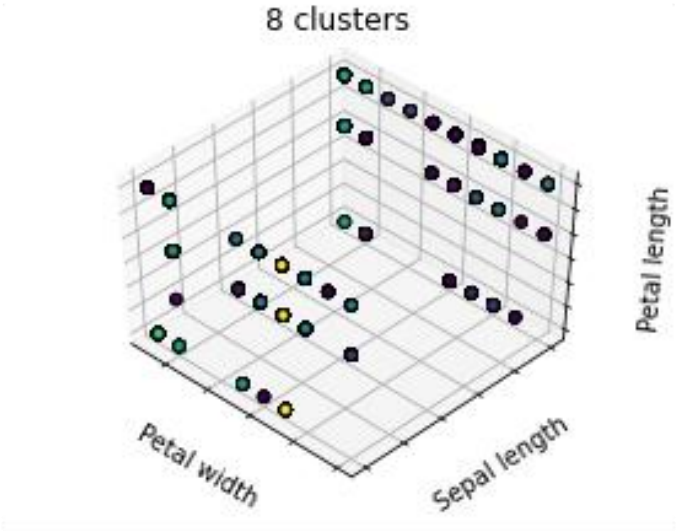
בס"ד

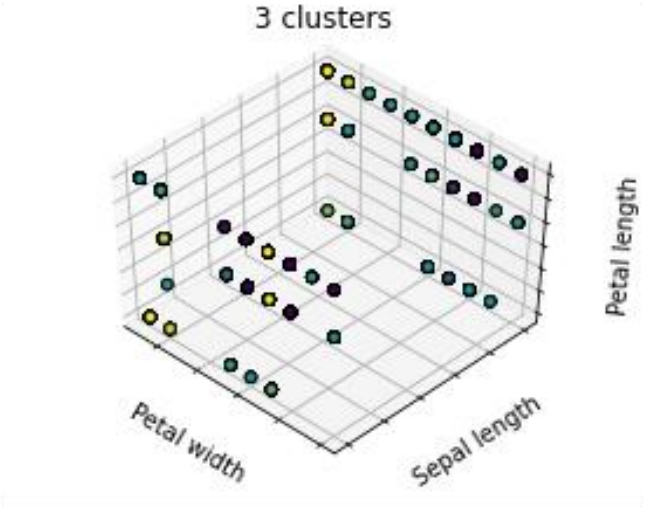


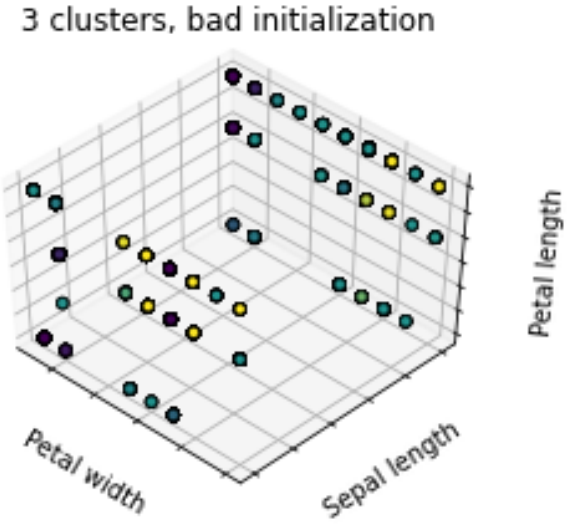

```
Result of accuracy test: 1.0
Result of recall test: 1.0
Result of precision test: 1.0
Result of F-measure test: 1.0
Result of auc_roc test: 1.0
End of run with parameter: 9
```

בס"ד

מודל kmeans מוכן:



בס"ד

## מסקנות:

**naive Bayes**:

המסווג הנאיבי של בייס קל לשימוש כאשר ישנו מספר רב של מאפיינים. המסווג הנאיבי של בייס יוצר מערך של הנחות פשטניות שסביר להניח שאינן נכונות לגמרי בפועל. עם זאת, נמצא שהמסווג הנאיבי של בייס מועיל ביותר במצבים שונים ומגוונים .

**Decision tree:**

עץ החלטה (Decision Tree) הינו אלגוריתם לסיווג או לחיזוי ערכו של משתנה, כאשר המאפיינים מסודרים לפי סדר החשיבות. לצורך סיווג, קיימים שני מדדים אלטרנטיביים לאי וודאות: מדד אנטרופי (Entropy) ומדד ג'יני .(Gini) כאשר ערכו של משתנה מסוים נחזה, אי הוודאות נמדדת באמצעות שורש השגיאה הריבועית הממוצעת (RMSE- Root Mean Squared Error) של התחזית. חשיבותו של מאפיין הינו הרווח מהמידע הצפוי שלו .(Expected Information Gain) הרווח מהמידע הצפוי נמדד על ידי הירידה באי הוודאות הצפויה אשר תתרחש כאשר יתקבל מידע אודות המאפיין.

KNN:

אלגוריתם KNN שמשמש לסיווג ולרגרסיה. אלגוריתם זה פשוט יחסית להבנה ומהיר למימוש. הוא לא מתאים לסטטיסטיקה עיסקית (אלא רק לחיזוי), והוא יכול להיות מאוד יעיל בבעיות בעלות מאפיינים מסוימים.

הרעיון שבבסיס שיטת KNN הוא שכאשר מגיעה תצפית חדשה, וצריך להעריך משתנה מטרה שלה או לסווג אותה לקטגוריה מסוימת, אז בוחנים לאילו תצפיות קיימות היא "קרובה" ומניחים שהערך שלה יהיה הסיווג הנפוץ בקרב התצפיות הקרובות (או ברגרסיה, יהיה ממוצע של ערך משתנה המטרה של השכנים הקרובים).

**K-MEANS:**

האלגוריתם הוא אינו מושפע מתצפיות חריגות בניגוד לרגרסיה ,זמן העיבוד לסווג דוגמה חדשה הוא קצר ביותר, יתרון נוסף המתבטא באחסון המודל הוא שאחרי שהוגדרו האשכולות ניתן לעבוד ללא נתוני המקור.

חסרונותיו הם בצורך בעיבוד מוקדם ולעיתים מרובה, הוא מושפע מבחירה של גבולות המשתנים, והוא אינו מאפשר לטפל בנתונים שאינם מספריים.

האלגוריתם תמיד ימצא פתרון כלשהו ולעיתים הפתרון אינו יציב או מייצג.