Find out your size- AI project

מגישות:
בר סלע 206902355
בר ויצמן 206492449

תיאור והגדרת בעיה:

עולם האופנה לנשים מעולם לא היה מגוון, מחמיא ומשתלם יותר מאשר בעידן המסחר באינטרנט.
בימינו קניות באינטרנט היא תופעה ידועה. אך אחד מהאתגרים בכך הוא בחירת מידה מתאימה עבורנו.
המרת מידה בהתאם לנתונים מסוימים יכולה להפוך את חווית הקנייה לפשוטה ומהנה יותר.
כיום אתרים שונים משתמשים במודל חיזוי מידה שיכולה להתאים ללקוחה לפי פרמטרים שונים הקובעים את מידתה ושהיא מתבקשת להזין - על מנת לקבל אינדיקציה עבור המידה המתאימה ביותר.

תיאור המערכת להתאמת מידה:

יש כמה פרמטרים המשפיעים על בחירת מידה מתאימה אצל נשים ואנחנו בחרנו להתמקד בגובה, משקל וגיל.
על ידי למידה מונחית (Supervised Learning) מאמנים את המערכת באמצעות נתונים ש"מתויגים" למידות..
זה אומר שחלק מהנתונים כבר מתויגים עם התשובה הנכונה.
כלומר, אלגוריתמי למידה מונחית מאפשרים לנבא labels (תוויות) מתוך features (מאפיינים) על סמך דוגמאות קודמות. במקרה שלנו, נוכל לנבא מידת בגדים (-label) לפי פרמטרים פיזיולוגיים, כמו גובה ומשקל על סמך דוגמאות של אנשים אחרים.

במערכת הנוכחית נצטרך למצוא שיטות על מנת למצוא בעיית Classification- זיהוי הקטגוריה הנכונה עבור הדוגמה החדשה מתוך סט הקטגוריות הנתונות. סט הקטגוריות שלנו הוא המידות. יש לנו קטגוריית מידת XXS ועד XXXL . בהתאם לפרמטרים הנתונים נוכל לסווג לכל אדם את הקטגוריית מידה(Label) אליו הוא שייך.
*נתייחס בהמשך לתוויות labels גם בתור תוצאות targets

Data Base

קובץ Size_DB.csv
Supervised
תוכן: מידות בגדים המחולקות לפי קטגוריות XXS-XXXL בהתאם לפרמטרים (תכונות): weight,age ו- height .
גודל: מכיל 119734 רשומות.

גישות לפתרון:

עבור Supervised machine learning נשתמש בטכניקות של קלסיפיקציה או רגרסיה בבנייה של המודל.
regression (רגרסיה) - ניסיון לזהות מס' מדויק , מול classification (סיווג) - זיהוי קבוצה.
טכניקות קלסיפיקציה (**classification**) מסווגות ומחלקות נתוני קלט לקטגוריות. כלומר, במידה והמידות מחולקות לפי סיווג מידה שאינה מיוצגת מספרית (XS/S/M/L וכו'...)
טכניקות רגרסיה (**Regression**) מנבאות תגובות מתמשכות.בטכניקה זו משתמשים כאשר עובדים עם טווח נתונים או אם אופי התוצאה הוא מספר ריאלי (כלומר כאשר המידה היא מיוצגת באמצעות טווח מספרים (34/34.5/35...וכו')

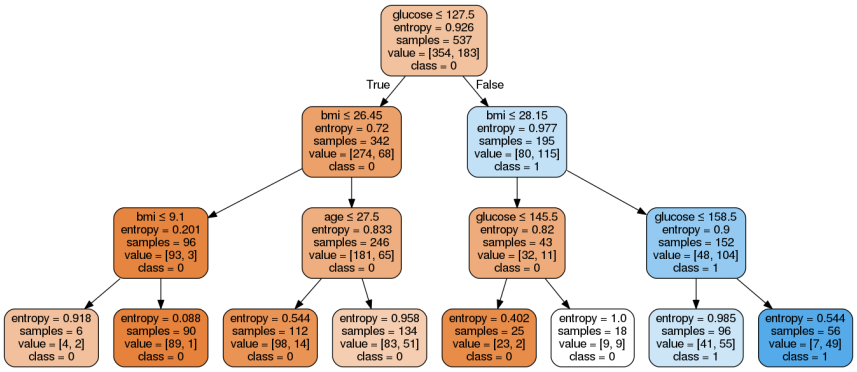
בחרנו להשתמש בטכניקת קלסיפיקציה מכיוון שה DB שלנו הוא סיווג הנתונים לקטגוריות ולא משתנה רציף.

שיטות:

מטרתנו היא לחזות מידות עבור פרמטרים חדשים באמצעות סיווג. נוכל להסיק מהפרמטרים גיל,משקל וגובה את המידה של האישה.במילים אחרות, לסווג לקבוצת מידה של בגדים אליה היא שייכת.
השיטות classification שאנחנו השתמשנו:

-Decision Tree

עצי החלטה (DTs) הם שיטת למידה מפוקחת המשמשת לסיווג (גם לרגרסיה).בכל נקודת החלטה נקבל אירוע מסוים (וקטור של נתונים), ננסה להחליט עבור הנקודה הנוכחית עד שנוכל לחזות לאיזה קטגוריה לשייך את המידע. המטרה היא ליצור מודל המנבא את הערך של משתנה יעד (במקרה שלנו מידה) על ידי לימוד כללי החלטה פשוטים המוסקים מתכונות הנתונים.
מבנה:עץ החלטה הוא עץ בינארי מלא המורכב מצמתי החלטה שבכל אחד מהם נבדק תנאי מסוים על מאפיין מסוים של התצפיות ועלים המכילים את הערך החזוי עבור התצפית המתאימה למסלול שמוביל אליהם בעץ.
כל עלה מוביל למחלקת סיווג אחרת-כלומר מידה אחרת.
תמונה להמחשה:



אלגוריתמים נפוצים לבניית עצי החלטה הם CART ,ID3, C4.5, C5.0, MARS, CHAID. אנחנו בחרנו להשתמש באלגוריתם 3ID, אלגוריתם איטרטיבי ולממש אותו באמצעות scikit-learn.

באלגוריתם 3ID הממומש בscikit-learn בונה את עץ ההחלטה בצורה איטרטיבית.

1. חישוב אנטרופיה של כל תכונה a של S-דגימת נתונים מקבוצת האימון
2. חלוקת קבוצת S לתת-קבוצות באמצעות התכונה שעבורה ממוזערת האנטרופיה המתקבלת לאחר הפיצול או רווח המידע (information gain) IG הוא מקסימלי והתחשבות במדד GINI שהוא אחת השיטות המשמשות באלגוריתמים של עצי החלטות כדי להחליט על הפיצול האופטימלי מצומת שורש, והפיצולים הבאים. מדד GINI יעזור לקבל החלטה זו ואומר מהי ההסתברות לסיווג שגוי.
3. יצירת צומת עץ החלטות המכיל את התכונה הזו.
4. חזרה על קבוצות משנה באמצעות התכונות הנותרות.

נעצור כאשר הגענו לסיווג מלא.

אנחנו מוצאים את חשיבות התכונות והשפעתן על סיווג המידה, ובכך נדע לבנות את עץ ההחלטות.

הבנייה תתבצע כך שתכונה בעלת IG מקסימלי או אנטרופיה מינימלית כלומר שואף לסדר וההסתברות לסיווג שגוי נמוך יהיה השורש (המפריד הטוב ביותר) וכך בהדרגה יבנה העץ מלמעלה למטה.

יתרונות של עץ החלטה:

- פשוט להבנה.
- פועל טוב עם מערכי נתונים גדולים .
- השימוש בו בזמן לינארי.
- עץ החלטה עובד טוב עם מידע שהוא יחסית רועש תוך שימוש בטכניקות מסוימות כדי לאפשר את זה.
- עץ החלטה יודע לעבוד עם מידע חסר. כלומר, מידע קודם לא חייב להיות שלם.

מגבלה: עץ ההחלטה יכול להיות עצום בגודלו.

-Logistic Regression

רגרסיה לוגיסטית הוא אלגוריתם Machine Learning לקלסיפיקציה (Classification). רגרסיה לוגיסטית, למרות שמה, היא מודל ליניארי לסיווג ולא לרגרסיה.

אלגוריתם רגרסיה לוגיסטית הוא בעל מספר יישומים אפשריים- זיהוי קשר בין משתנים או סיווג וחיזוי. בפתרון הבעיה שלנו אנחנו נתמקד ביישום של סיווג וחיזוי (ניבוי תוצאות לפי קטגוריה) . המודל פועל על מנת לשפר לנו את קבלת ההחלטות.

בנוסף קיימות מספר גישות לשימוש ברגרסיה לוגיסטית אך נשים לב שמה שמנחה אותנו בפתרון היא גישת המולטינום (Multinomial logistic regression) המאפשרת יותר משתי קטגוריות של המשתנה התלוי כלומר התוצאה. בגישה זו מסווגים נתונים לקבוצות על סמך טווח קטגוריות על מנת לחזות התנהגות (קידוד בעל משמעות של ערכי התלוי לקבוצות).

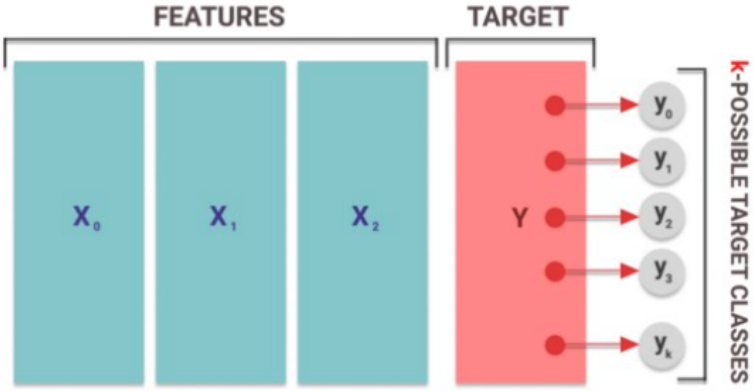
Multinomial logistic regression מחייבת שיקול זהיר של גודל המדגם ובדיקה למקרים חריגים. כמו נהלי ניתוח נתונים אחרים, ניתוח הנתונים הראשוני צריך להיות יסודי.

מבנה:

(מאפיינים features) משתנים בלתי תלויים - גיל, משקל, גובה.

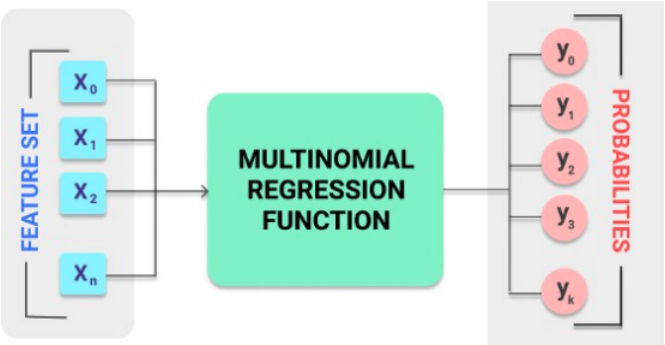
(תוצאה target) משתנה תלוי - מידה ,באשר התוצאה יכולה להיות XXS-XXXL או לאחר המיפוי שעשינו מספר שלם מ1-7.

רגרסיה לוגיסטית שמה דגש על המשתנה התלוי כך שיש לו יותר מ2 תוצאות ואין סדר לתוצאות המתקבלות. בשיטה זו אנו משתמשים במשתנים הבלתי תלויים על מנת לחזות את התוצאה. קיימות לנו k מחלקות תוצאה אפשריות(קטגוריות) כאשר k יכול להיות כל מספר מ1-7 שמייצג את המידה המתאימה.



הפונקציה של Multinomial regression היא אלגוריתם סטטיסטי לקלסיפיקציה

ברגע שאנו מפעילים את הפונקציה עם סט של features , המודל מבצע מספר פעולות מתמטיות על מנת לנרמל את ערכי הקלט לווקטור של ערכים המאפשרים חלוקת הסתברויות -



עבור ווקטור X שהוא ווקטור המאפיינים נפעיל את פונקציית ה Multinomial regression עבורה נקבל ווקטור Y בגודל k של של ערכים שהם ההסתברות עבור כל קטגוריה שסט המאפיינים זה שייך לקטגוריה זו.

בתוצאה הסופית אנחנו נבחר מתוך ווקטור Y את ההסתברות הכי גבוהה להיות הקטגוריה אליה סט המאפיינים שייך.

המודל בעת פעולתו עובר תהליך של אופטימיזציה ,על מנת להפחית מצב של איבוד מידע והוא נעזר בהגדרת פונקציית הפסד הנקראת cross-entropy-loss function ומחושבת בצורה הבאה:

$$L = -\frac{1}{n}\sum_{i=0}^{i=n} y_i \log(s_i)$$

L = Average cross-entropy loss for the model

y = Ground truth (1-hot encoded target variable; 1 for target outcome, 0 for all other possible outcomes)

s = probability vectors obtained from softmax function

n = Total number of feature sets

<u>בשימוש ברגרסיה לוגיסטית נשים לב להנחות ולדברים הבאים להצלחת המודל:</u>

1. ספציפיקציה – הכללת כל המשתנים הרלוונטיים במודל.
2. משתנים שגויים או לא שלמים עשויים לשבש את הניבוי של המודל, ולכן נצטרך לטפל בחוסרים של מידע ולשים לב שאין תלות בין הטעויות (אחת ההנחות אצלנו היא שאין טעויות מדידה). כלומר, אלגוריתם זה אינו עובד טוב עם מידע רועש או מידע חסר בניגוד לעץ החלטה.
3. העדר השפעות הדדיות בין המשתנים, נעדיף להשתמש בנתונים שלא קשורים זה לזה מאחר שאם תצפיות מסוימות קשורות זו לזו, המודל נוטה לעלות משקל יתר במשמעותן.
4. הימנעות מהתאמת יתר (overfitting) - כשאנחנו מוסיפים למודל פרמטרים נלמדים (יותר שכבות ויותר יחידות בכל שכבה) אנחנו מגבירים את היכולת שלו ללמוד בצורה מדויקת את סט האימון. הדיוק מועיל עד לגבול מסוים שמעבר לו המודל מתאים יותר מדי לסט האימון ופחות מדי לסט המבחן, שזו אינדיקציה ל-overfitting. נרצה להמנע מכך.
5. העדר טעויות קיצוניות (outliers) – טעויות מעל 3 ס"ת >> הן טעויות קיצוניות שיש להורידן.

הערה: נשים לב שאכן במהלך כל הפתרון שמנו דגש לכל אחד מדברים אלו. המודל קיבל את המשתנים הרלוונטים, טיפלנו בשגיאות וחוסרים של מידע, השתמשנו במשתנים לא תלויים זה בזה, נמנענו מהתאמת יתר (overfitting) ע"י פישוט המודל וחלוקת הנתונים לקבוצת אימון ובדיקה ושימוש בסט מידע גדול, ועשינו סינון של חריגים בעלי סטיית תקן קיצונית.

יתרונות של logistic regression:

- קל ליישום.
- פשוט יחסית להבנה.
- יעיל לאימון.

מגבלות:

- רגישות לחריגים עלולה להיות גדולה .
- עבודה עם מידע שאינו שלם עלול לפגוע בניבוי המודל.

Performance measurements

איך נדע להעריך האם המודל שלנו נכון? בדיקת נכונות המודלים. המידע בDB שלנו הוא מתויג. ולשם כך נשתמש בו בשביל להעריך את נכונות המודלים.

נמדוד את המודל בכך שנריץ על מידע שאנחנו כבר יודעים את התשובות.

לכן, נחלק את המידע שלנו ל2 קבוצות.

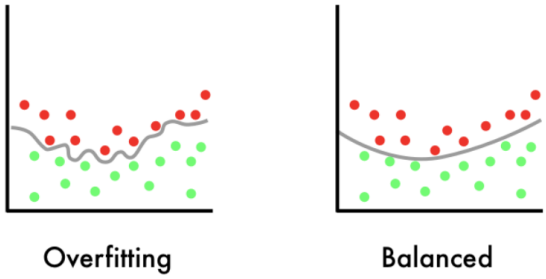
קבוצת אימון (training) - תשמש על מנת לבניית המודל ואימון האלגוריתם לזהות לפי הפרמטרים את המידה.

קבוצת אימות/בדיקה (validation/testing)- בעזרת קבוצה זו נעריך את נכונות המודלים ולבדוק את המודל בדיעבד. כלומר נריץ את המודל על הtesting data ונבדוק עד כמה הוא מדויק.

בשביל להגדיל את רמת הדיוק נקצה יותר מידע עבור קבוצת האימון ובכך נקטין את כמות ההכללות הלא נכונות אך תוך כדי גם נזהר מ overfitting .

מניעת **overfitting** בפתרון שלנו:

1. המידע מתפלג נורמלית .
2. סט הנתונים עימו אנחנו עובדים גדול ומספק .
3. שימוש בחלוקת המידע לשני Dataset - חלוקת המידע ל70% מידע עבור training ו30% עבור testing למשל .



ייעול תהליך אימון המודל:

שימוש ב feature engineering-תהליך של בחירה, מניפולציה והפיכת נתונים גולמיים לתכונות שניתן להשתמש בהן בלמידה מפוקחת.

1. Imputation: השלמת מידע חסר באמצעות פונקציית medians (הסבר בקוד)
2. Handling Outliers: איתור חריגים באמצעות סטיית תקן (z-score normalization)
3. קורלציה בין משתנים: תכונות עם קורלציה גבוהה תלויות לינארית ולכן יש להן כמעט אותה השפעה-עוזר להשמיט משתנים.

הסבר קוד:

בחלק זה נסביר באילו ספריות נעשה שימוש בקוד -

- pandas:ספרייה המשתמשת לניתוח מידע. מאפשרת לנו לקרוא את קובץ CSV המכיל את מאגר המידע שלנו עם הפרמטרים מידות בגדים בהתאם לפרמטרים שבאמצעותו נבצע את שלבי האימון וההרצה. מסדרת את המידע בתצורת dataset (כמו בטבלה עם ערכים ושמות עמודות).

- numpy: ספרייה הכוללת יכולת לבצע פעולות מתמטיות וחישוביות. ספרייה זו נפוצה בשימוש בלמידת מכונה. מאחר שאנו עובדים עם אימון מודלים, נשים לב שכאשר אנו מאמנים את אותו המודל על אותם הנתונים אנו מקבלים תוצאות שונות-בעצם היעדר החזרתיות נובע מהאופי האקראי של למידת המכונה. מאחר שאנו מעוניינות לקבל תוצאות עקביות נצטרך לאלץ בקוד להנפיק את אותן התוצאות בתנאי שאנחנו משתמשים באותו המודל ובאותם הנתונים- וזה מתאפשר באמצעות שימוש בפונקציה הבנויה בnumpy הנקראת random.seed .

- matplotlib: ספרייה להצגת נתונים באמצעות תרשימים כחלק מתהליך ניתוח המידע.

- Seaborn: ספרייה המבוססת על Matplotlib ועוזרת גם לחלק של הצגת תהליך ניתוח המידע.

- Time: ספרייה לעבודה עם זמנים. נשתמש עבור מדידת זמני ריצה.

- sklearn : מכילה בתוכה את המימוש לאלגוריתמים אותם אנו משווים. בנוסף,הפונקציה train_test_split משמשת לפיצול סט הנתונים בין קבוצת אימון לקבוצת בדיקה. (הערה: בפרויקט זה התבססנו על הAPI של sklearn והמידע שיש באתר

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time


# Models from Scikit-learn
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

# Model Evaluations
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn import tree
```

1. ייבוא של הData , לקחנו את קובץ ה csv. שלנו והצגנו אותו בתור סט נתונים (נקרא Dataframe) של עמודות ושורות עם ערכים

```
"""### Import data"""

df_raw = pd.read_csv("Size_DB.csv")
print(df_raw)

print(df_raw.info())
```

2. הוצאת נתונים סטטיסטיים כלליים על סט הנתונים שלנו - נוכל לראות את התפלגות המידות,גילאים,משקל וגובה שבתוך הDataframe בתצורה של גרף.

```
# Number of occurences for each size (target variable)-chart description
sns.countplot(x=df_raw["size"])
plt.show()
```

```
# Age distribution
sns.displot(df_raw["age"])
plt.show()

"""Large fraction of population seems to be around the ages of `25 to 35 years old`"""

# Weight distribution
sns.displot(df_raw["weight"])
plt.show()

# height distribution
sns.displot(df_raw["height"])
plt.show()

"""Population weight and height seem to show reasonable normal distributions
```

3. חישוב סטיית תקן לכל אחת מהמידות הקיימות בDataframe על מנת לתאר את הפיזור של נתונים מספריים סביב הממוצע שלהם. כלומר עבור כל מידה נחשב את סטיית התקן ברשומות של אותה המידה.

נשים לב שנקבל עבור כל מידה מעין טבלה המכילה את כל הרשומות הקשורות לאותה המידה כך שהן מכילות את סטיית התקן. נשים לב לסינון של הרשומות בצורה הבאה - נבחר את כל הרשומות כך שציון התקן בכל אחד מהערכים לא גבוה מ3 או קטן מ -3.

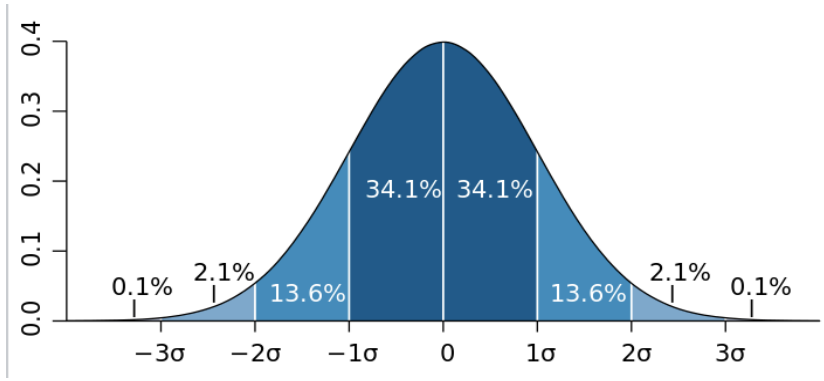
```
# Removing Outliers
dfs = []
sizes = []

for size_type in df_raw['size'].unique():
    sizes.append(size_type)
    ndf = df_raw[['age','height','weight']][df_raw['size'] == size_type]
    zscore = ((ndf - ndf.mean()) /ndf.std())
    dfs.append(zscore)

for i in range(len(dfs)):
    dfs[i]['age'] = dfs[i]['age'][(dfs[i]['age'] >-3) & (dfs[i]['age'] <3)]
    dfs[i]['height'] = dfs[i]['height'][(dfs[i]['height'] >-3) & (dfs[i]['height'] <3)]
    dfs[i]['weight'] = dfs[i]['weight'][(dfs[i]['weight'] >-3) & (dfs[i]['weight'] <3)]

for i in range(len(sizes)):
    dfs[i]['size'] = sizes[i]
df_raw = pd.concat(dfs)
print(df_raw.head())
```

פעמון גאוס : אחוזי ההתפלגות הנורמלית מסביב לממוצע (ציר הסימטריה) לפי סטיות תקן



4. השלמת מידע חסר- נעבור על כל סט הנתונים שלנו, ועבור כל תא חסר במידע נשלים בעזרת החציון של הקבוצה של הערך החסר כלומר נקודת האמצע של קבוצה של מספרים (לדוגמא נשים לב שאם חסר משקל, נשים את המשקל עפ"י החציון בהתאם לערך החסר). נשים לב שהפונקציה isna() השייכת לספריית pandas מסננת עבורנו את כל התאים הריקים ב dataframe ומסמנת אותם ב true כך שנבחין בדיוק באילו תאים חסר מידע ועלינו להשלים אותו.

```
"""### Filling missing data"""

# Check for missing values
df_raw.isna().sum()

# Filling missing data
df_raw["age"] = df_raw["age"].fillna(df_raw['age'].median())
df_raw["height"] = df_raw["height"].fillna(df_raw['height'].median())
df_raw["weight"] = df_raw["weight"].fillna(df_raw['weight'].median())

# Mapping clothes size from strings to numeric
df_raw['size'] = df_raw['size'].map({"XXS": 1,
                                     "S": 2,
                                     "M": 3,
                                     "L": 4,
                                     "XL": 5,
                                     "XXL": 6,
                                     "XXXL": 7})

# Check for missing values
df_raw.isna().sum()
```

5. ייעול תהליך אימון המודל, הכולל את הוספת מאפיין - bmi-מושפע משני פרמטרים גובה ומשקל.

```
"""###
We will create new feature to help model training effectiveness:
* `bmi` (body-mass index) - medically accepted measure of obesity
"""

df_raw["bmi"] = df_raw["height"] /df_raw["weight"]


print(df_raw)
```

ולאחר מכן נציג את הטבלה שמראה לנו את הקורלציה בין הנתונים

```
"""### Correlation matrix"""

corr = sns.heatmap(df_raw.corr(), annot=True)
plt.show()
```

6. חלוקת המידע לשני קבוצות (train_test_split):
   - קבוצת אימון (training) - תשמש על מנת לאמן את האלגוריתם לזהות לפי הפרמטרים את המידה.
   - קבוצת אימות (validation)- יהווה כחלק ב30% מהמידע .

```
# Features
X = df_raw.drop("size", axis=1)

# Target
y = df_raw["size"]

X.head()

y.head()

# Splitting data into training set and validation set

X_train, X_test, y_train, y_test, = train_test_split(X, y, test_size=0.30)

len(X_train), len(X_test)
```

7. הכנסת המודלים של האלגוריתמים שבחרנו להשתמש בהם לרשימה

```
# Put models in a dictionary
models = {"Logistic Regression": LogisticRegression(),
          "Decision Tree": DecisionTreeClassifier()}
```

8. הפונקציה fit_and _score לוקחת את המילון המכיל את האלגוריתמים ועל ידי fit() מתאימה את המודל למידע ושומרת ברשימה את תוצאת המודל. כאן נוכל לראות את האלגוריתם שנתן את הדיוק הגבוה ביותר. (תוך כדי נחשב את זמני הריצה ונדפיסם)

```
# Create a function to fit and score models
def fit_and_score(models, X_train, X_test, y_train, y_test):
    """..."""
    # Set random seed
    np.random.seed(18)
    # Make a dictionary to keep model scores
    model_scores = {}
    model_coplexity = {}
    # Loop through models
    for name, model in models.items():
        # Fit model to data
        start = time.time()
        model.fit(X_train, y_train)
        # Evaluate model and append its score to model_scores
        model_scores[name] = model.score(X_test, y_test)
        end = time.time()
        model_coplexity[name] = end - start
    print('model_coplexity runtime:', end=' ')
    print(model_coplexity)
    return model_scores


model_scores = fit_and_score(models, X_train, X_test, y_train, y_test)
```

9. הערכת המודל - בחלק זה נבדוק ונבחר את המודל המתאים מבחינה אידיאלית לפתרון הבעיה הנתונה ובהינתן סט המידע שהוזן.
נבחר את המודל בעל התוצאות הטובות ביותר, ואיתו נבצע חיזוי של התוצאות שהתקבלו באמצעות שימוש ב predict - בסיום הרצת האלגוריתם עם סט הנתונים, זהו השלב הסופי בפרויקט שהוא חיזוי התוצאות(פלט) של הפעלת המודל.
(נריץ את פונקציית predict גם על המודל השני על מנת לבדוק סיבוכיות זמן ריצה של חיזוי תוצאות)

```
"""### Model evaluation
The DecisionTreeClassifier model scored highest in initial tests with `99.9749%` accuracy.
"""
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
start = time.time()
y_pred = model.predict(X_test)
end = time.time()
print(y_pred)
print('Decision tree predict runtime:', end=' ')
print(end-start)

# Confusion matrix
print(confusion_matrix(y_test, y_pred))

# Classification report
print(classification_report(y_test, y_pred))
```

```
model1 = LogisticRegression()
model1.fit(X_train, y_train)
start = time.time()
y_pred1 = model.predict(X_test)
end = time.time()
print('Logistic regression predict runtime:', end=' ')
print(end-start)
```

10. מציאת חשיבות הפרמטרים. מי הפרמטר בעל ההשפעה הכי גדולה בעץ ההחלטות,מושפע מ2 מדדים:

1) מדד Gini
2) מדד Information Gain המוגדר על פי נוסחת האנטרופיה

Plot_features היא פונקציית עזר לשרטוט חשיבות פרמטר.

```
# Find feature importance of ideal model
len(model.feature_importances_)

model.feature_importances_

# Helper function for plotting feature importance
def plot_features(columns, importances ,n=20):
    df = (pd.DataFrame({"features": columns,
                        "feature_importances": importances})
          .sort_values("feature_importances", ascending=False)
          .reset_index(drop=True))
    # Plot dataframe
    fix, ax = plt.subplots()
    ax.barh(df["features"][:n], df["feature_importances"][:20])
    ax.set_ylabel("Features")
    ax.set_xlabel("Feature Importance")
    ax.invert_yaxis()

plot_features(X_train.columns, model.feature_importances_)
plt.show()
```

11. ייצוג עץ ההחלטות. נציג את עץ ההחלטות שנבנה עד עומק 1 ביוון שהאיבות תמונה ירודה כאשר העץ גדול יותר. לאחר שמצאנו מי הפרמטר המפצל הטוב ביותר נבדוק אם הוא אכן מופיע בשורש.

```
# Build Decision Tree
# Limit of poor quality of graph representation due to present to depth 1
clf = tree.DecisionTreeClassifier(max_depth=1)
clf = clf.fit(X_test, y_pred)
tree.plot_tree(clf)
plt.show()

# Order of the features
print(df_raw.columns)
# ['age', 'height', 'weight', 'size', 'bmi']
"""### x[2]=weight
the feature in the root is weight
"""
```

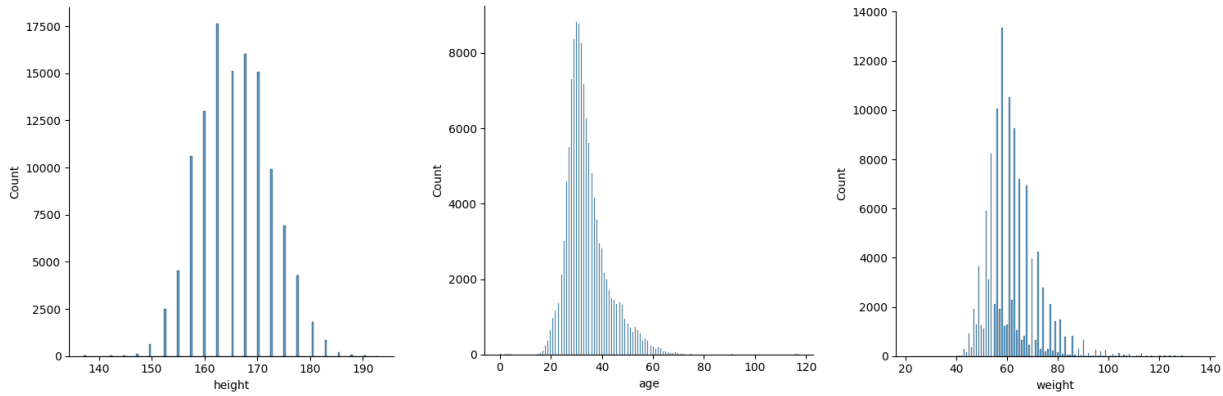
תוצאות שלב האימות ומסקנות:

בפרויקט שלנו אנו השתמשנו בשתי שיטות שונות על מנת לפתור את בעיית ההתאמת מידות:

1. Decision tree
2. Logistic regression

נסתכל על הנתונים מתוך הDB:

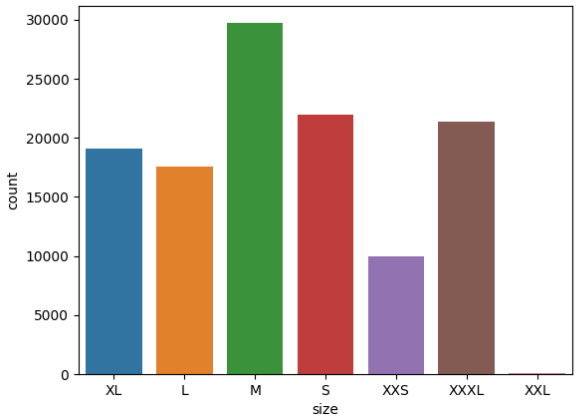
התפלגות מאפיינים:



גיל: נראה שחלק גדול מהטווח גילאים הוא בסביבות הגילאים 25 עד 35 שנים.

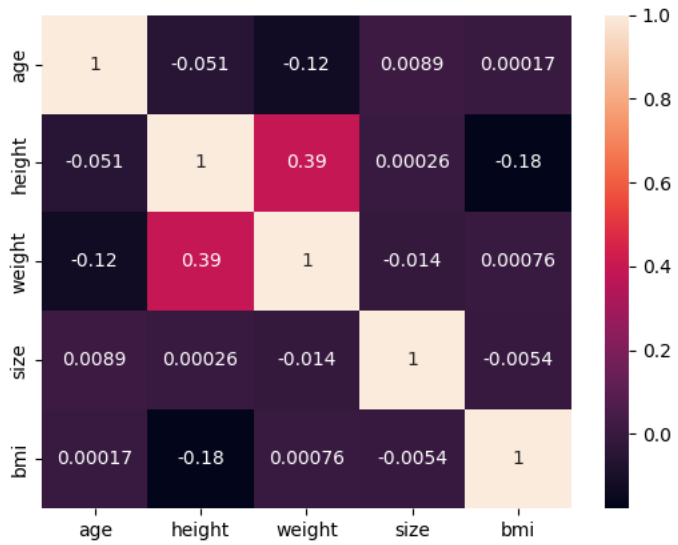
משקל וגובה: נראה כי משקל וגובה מראים יחסית התפלגויות נורמליות.

התפלגות מידות:



ציר X מייצג את הקטגוריות השונות (מידות) וציר Y מייצג את השורות המשתייכות לקטגוריה זו, ונשים לב שהמידה הנפוצה ביותר במאגר הנתונים היא M.

קורלציה בין מאפיינים:



ניתן לראות שהמידה תלויה הרבה יותר במשקל מאשר בגיל או גובה, ונראה שיש להם קורלציה חזקה עם BMI.

נכונות (דיוק) האלגוריתמים:

```
{'Logistic Regression': 0.2509674006848362, 'Decision Tree': 0.9995545781019459}
          Logistic Regression  Decision Tree
accuracy             0.250967       0.999555
```

1. Decision tree accuracy=99.95%
2. Logistic regression accuracy=25.09%

נשים לב שמבחינת דיוק בפער מאוד משמעותי עץ החלטות יותר טוב בחיזוי קטגוריית מידה.

ניתוח ומסקנות תוצאות דיוק:

אחוזי הדיוק של Logistic regression מאוד נמוכים מה שמעיד על כך שהאלגוריתם אינו מניב תוצאות חיזוי טובות מספיק. מאידך יש פה חשד ל overfitting בעץ החלטה. הדיוק מאוד גבוה. למרות שפעלנו בדרכים בשביל למנוע מצב של overfitting זה לא היה מספיק. יכול להיות שעקב כך שהמאגר נתונים גדול מאוד ו70% מכך מהווה נתח גדול הנתונים היה צריך לחלק אחרת את המידע ולהקצות פחות מידע לקבוצת האימון.

דיוק לפי מידות:

Decision tree:

	precision	recall	f1-score	support
1	1.00	1.00	1.00	2985
2	1.00	1.00	1.00	6618
3	1.00	1.00	1.00	8912
4	1.00	1.00	1.00	5269
5	1.00	1.00	1.00	5744
6	0.88	0.94	0.91	16
7	1.00	1.00	1.00	6377
accuracy			1.00	35921
macro avg	0.98	0.99	0.99	35921
weighted avg	1.00	1.00	1.00	35921

עבור 6- מידה XXL אחוזי הדיוק בהשוואה לשאר המידות נמוכים יותר.
אם נסתכל בגרף התפלגות המידות כמות הנתונים עבור מידה זו נמוכה יחסית בהשוואה לשאר המידות ב DB.
ולכן, יכול להיות שלא היה מספיק מידע להתאמן עליו ומכך נוכל להסביר את השוני באחוזי הדיוק בין המידות.

Logistic regression:

	precision	recall	f1-score	support
1	0.00	0.00	0.00	3011
2	0.67	0.01	0.02	6594
3	0.25	1.00	0.40	8910
4	0.00	0.00	0.00	5255
5	0.00	0.00	0.00	5746
6	0.00	0.00	0.00	20
7	0.00	0.00	0.00	6385
accuracy			0.25	35921
macro avg	0.13	0.14	0.06	35921
weighted avg	0.19	0.25	0.10	35921

מסקנה- שיטת העץ החלטה באמצעות מימוש אלגוריתם 3ID טוב ונכון יותר משמעותית מ logistic regression.

ניקוי/סידור הקלט והמידע עבור האלגוריתמים:

כמו שהזכרנו לפני כן אלגוריתם logistic regression אינו עובד טוב עם מידע רועש או מידע חסר בניגוד לעץ החלטה שבשימוש של אסטרטגיות מסוימות יכול להתגבר על מידע רועש ולא שלם.

בקוד כן "ליטשנו" את המידע לפני הרצת האלגוריתמים:

Imputation: השלמנו מידע חסר באמצעות פונקציית medians

וגם ביצענו Handling Outliers: איתור חריגים באמצעות סטיית תקן (z-score normalization) והסרתם.

ניכר לראות שפעולות אלו היטיבו את הדיוק של עץ החלטות אך עבור ה Logistic regression צריך להחמיר עם ניקוי במידע ובכך לקבל תוצאות יותר טובות בעתיד.

תהליך הרצה:

שימוש בAPI של ספריית sklearn הקלה על תהליך ההרצה של שני האלגוריתמים באותה מידה.

פונקציית fit בנויה כך שעבור כל מודל שממומש בsklearn יהיה ניתן להתאים את הdata למודל בקלות. גם אם היינו רוצים לשנות את המימוש של האלגוריתמים זה היה אפשרי וקל להרצה כיוון שקיימים מחסומי הפשטה והAPI נשאר זהה וכך גם ההרצה. פונקציית predict עובדת גם היא באופן זה.

שימוש ב sklearn חסך לנו לממש את האלגוריתמים בעצמנו, לא היינו צריכות להיכנס לאופן המימוש כל כך אלא רק להבין את צורת השימוש בהם. סה"כ הAPI הוא user friendly בשני האלגוריתמים.

יעילות האלגוריתמים:

זמן ריצה:

לקוח מתוך **SKlearn** שמימשה את האלגוריתמים:

ההנחה היא:

$$n_{\text{samples}} \geq n_{\text{features}}$$

Decision tree:

באופן כללי, עלות זמן הריצה לבניית עץ בינארי מאוזן היא $O(n_{samples} n_{features} \log(n_{samples}))$

קבוצה S היא samples- ו features זה המאפיינים שבחרנו.

כדי למצוא את התכונה שמציעה את ההפחתה הגדולה ביותר באנטרופיה. יש לזה עלות של

$O(n_{samples} n_{features} \log(n_{samples}))$ עבור כל צומת. נסתכל על המקרה גרוע שזה המסלול הארוך

ביותר בעץ כלומר -גובה העץ.

מה שמוביל לעלות כוללת על כל העצים (על ידי סיכום העלות בכל צומת) של

$$O(n_{features} n_{samples}^2 \log(n_{samples}))$$

.

Logistic regression:

X הוא מטריצה מהצורה (n_samples, n_features).

כאשר מדובר סיווג בינארי העלות היא $O(n_{\text{samples}} n_{\text{features}}^2)$,

מכיוון שאצלנו הסיווג הוא לפי קטגוריות נסמן בn_class את מס' הקטגוריות. ו n_class פעמים אנחנו מבצעים חישוב עבור כל קטגוריה.

לכן העלות הסופית תהיה

$$O(n_{\text{samples}} n_{\text{features}}^2 n^2{}_{\text{Classes}})$$

מדדנו בעצמנו את הזמנים.

תוצאות זמני ריצה:

```
model_coplexity
{'Logistic Regression': 10.760297536849976, 'Decision Tree': 0.4371907711029053}
```

Prediction time: הבדל של 0.00001 שניות. זניח.

ההבדל המהותי הוא בזמני הריצה של אימון המודל והתאמת המידע למודל.

נראה ש Decision tree מבחינת זמני ריצה Decision tree יותר יעיל מ .Logistic regression

מקום:

על מנת לחשב סיבוכיות מקום של כל אחת מההרצות של האלגוריתמים הכוללות אימון והרצה של Decision tree ו Logistic regression , נשתמש בספרייה הנקראת memory_profiler ממנה נייצא decorator בשם profile (נוסיף את זה בייבוא של כלל הספריות בפרויקט)

```
from memory_profiler import profile
```

נפעיל את @profile מעל הפונק' כדי לחשב memory usage עבור כל אחת מהפעולות -

```
@profile()
def fit_and_score(models, X_train, X_test, y_train, y_test):
```

Decision tree - Memory usage (בסיום הרצת **fit_and_score**):

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
   159    176.9 MiB    176.9 MiB           1   @profile()
   160                                         def fit_and_score(models, X_train, X_test, y_train, y_test):
   161                                             """
   162                                             Fits and evaluates given machine learning models.
   163                                             models: a dict of different Scikit_Learn machine learning models
   164                                             X_train: training data (no labels)
   165                                             X_test: testing data (no labels)
   166                                             y_train: training labels
   167                                             y_test: test labels
   168                                             """
   169                                             # Set random seed
   170    176.9 MiB      0.0 MiB           1       np.random.seed(18)
   171                                             # Make a dictionary to keep model scores
   172    176.9 MiB      0.0 MiB           1       model_scores = {}
   173                                             # Loop through models
   174    179.8 MiB      0.0 MiB           2       for name, model in models.items():
   175                                                 # Fit model to data
   176    179.8 MiB      2.9 MiB           1           model.fit(X_train, y_train)
   177                                                 # Evaluate model and append its score to model_scores
   178    179.8 MiB      0.1 MiB           1           model_scores[name] = model.score(X_test, y_test)
   179    179.8 MiB      0.0 MiB           1       return model_scores
```

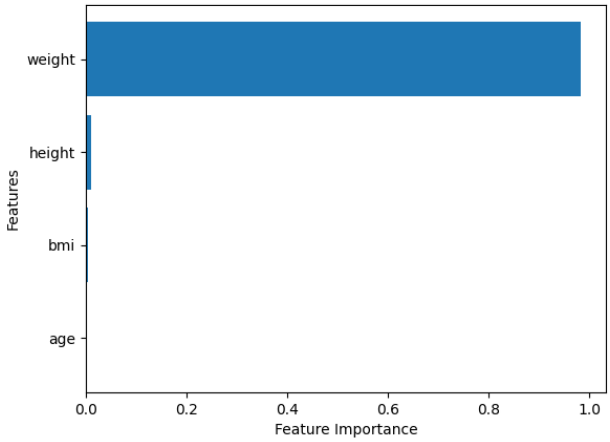
Logistic regression - Memory usage (בסיום הרצת **fit_and_score**):

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
   159    177.0 MiB    177.0 MiB           1   @profile()
   160                                         def fit_and_score(models, X_train, X_test, y_train, y_test):
   161                                             """
   162                                             Fits and evaluates given machine learning models.
   163                                             models: a dict of different Scikit_Learn machine learning models
   164                                             X_train: training data (no labels)
   165                                             X_test: testing data (no labels)
   166                                             y_train: training labels
   167                                             y_test: test labels
   168                                             """
   169                                             # Set random seed
   170    177.1 MiB      0.0 MiB           1       np.random.seed(18)
   171                                             # Make a dictionary to keep model scores
   172    177.1 MiB      0.0 MiB           1       model_scores = {}
   173                                             # Loop through models
   174    206.5 MiB      0.0 MiB           2       for name, model in models.items():
   175                                                 # Fit model to data
   176    206.3 MiB     29.3 MiB           1           model.fit(X_train, y_train)
   177                                                 # Evaluate model and append its score to model_scores
   178    206.5 MiB      0.2 MiB           1           model_scores[name] = model.score(X_test, y_test)
   179    206.5 MiB      0.0 MiB           1       return model_scores
```

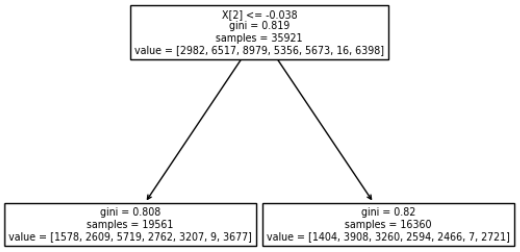
נראה שגם מבחינת סיבוכיות מקום עץ החלטות יעיל יותר ביוון שהוא תופס פחות מקום.

Feature Importance עץ החלטות :

כפי שהסברנו לפני כן על בניית העץ נראה כי לאחר ההרצה התוצאה היא שהמאפיין משקל (weight) מחלק טוב יותר לעומת שאר המאפיינים



ולכן יופיע בשורש העץ:




```
Index(['age', 'height', 'weight', 'size', 'bmi'], dtype='object')
```

X[2]=Weight

מה למדנו על האלגוריתמים ומהפרויקט ואילו תובנות הגענו:

ראשית, כתוצאה מהפרויקט למדנו גם על אלגוריתמים נוספים של מימוש עצי החלטה מלבד 3ID שנלמד בהרצאה. בנוסף, למדנו שיטות נוספות לקלסיפיקציה על המבנה שלהם היתרונות והחסרונות שלהם התנסינו בכתיבת קוד והבנת קוד כתוב של אלגוריתמים אלו.

מטרת הפרויקט על פי ההגדרה שלנו היא ניבוי כמה שיותר מוצלח של מידת הבגדים וקנה המידה של מאגר הנתונים יחסית סביר ולכן נעדיף את האלגוריתם עם אחוזי הדיוק הטובים יותר שבמקרה גם יעיל יותר מבחינת זמני ריצה וגם מבחינת מקום- Decision tree. נציין כי מחוץ למסגרת הפרויקט נבדקו עוד אלגוריתמים מלבד ה2 שיטות שנדרשו להשוות ומבחינת יעילות זמן ריצה RandomForest (המבוסס על Decision tree) יעיל יותר.

אך גם במידה ולא הייתה תשובה חד משמעית מבחינת יעילות האלגוריתמים, הכל עניין של tradeoff. והיינו צריכות להחליט מה יותר חשוב לנו.

בתחילת דרכנו בפרויקט דברים נראו מעורפלים. מאיפה מתחילים? איך הכי נכון לעשות? מה השיטה הכי טובה?

ככל שנכנסנו לעומק הדברים נחשפנו להרבה מידע מעבר לציפיות הפרויקט. רעננו חומר ממחברות של קורסים ישנים כמו הסתברות וסטטיסטיקה ועקרונות. ביררנו על יותר מ2 שיטות שנדרשנו בפרויקט...

לדוגמא, לאחר שסיימנו להשוות בין שתי השיטות בפרויקט הנ"ל למדנו רבות גם על Random Forest שהוא מבוסס על עץ החלטות ואפילו יותר טוב. כל הרעיון בבינה מלאכותית היא להתחיל מלמצוא פתרון לבעיה פשוטה ומשם להתקדם. צריך להתחיל ממקום מסוים. וזו התובנה העיקרית שלנו מהפרויקט שנותן לנו כלים לעתיד ולפרויקט גמר שלנו בין היתר.

בנוסף, ברמה האישית גם במהלך התואר אבל בעיקר מקורס זה הוא ההתנסות העצמית בחקר מעמיק ואבן הדרך הראשונה שלנו בתחום הAI שהייתה חלק ניכר מתחושת הסיפוק.

ביבליוגרפיה:

https://it.unt.edu/sites/default/files/mlr_jds_aug2011.pdf

https://www.ibm.com/topics/logistic-regression

https://scikit-learn.org/stable/modules

https://en.wikipedia.org/wiki/ID3_algorithm

https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d

https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/

https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114

https://towardsdatascience.com/ml-from-scratch-multinomial-logistic-regression-6dda9cbacf9d

קורס מבוא לבינה מלאכותית ד"ר תמר שרוט , AI Lecture-07