**פרויקט מסכם – גירסה סופית**

בפרויקט זה (ניתן לבצע בקבוצות עד 3 סטודנטים), ננסה ללמוד מאפיינים המבחינים בין אנשים המרוויחים שכר שנתי גבוה (מעל כ 180 אלף ש"ח לשנה) לשכר נמוך (מתחת לאותו הסכום). סגנון הפרויקט יזכיר לכם את התרגילים וגם את התרגולים שביצענו בכיתה, אך כאן ההיקף קצת רחב יותר וגם הפעם תצטרכו במקרים רבים לשלב חלקי קוד/פקודות מתוך עבודות קודמות – כמו בתרגילים או ממקורות שתופנו אליהם. נשמע כיף....😊

מוכנים??? קדימה!!!!

1. אוקי... בדיוק כמו בתרגילים שניתנו במהלך הסמסטר, עליכם לפתוח את סביבת Jupyter (למי ששכח יש הנחיות בתרגילים הקודמים) וליצור קובץ מחברת. שם הקובץ יהא על שם מספרי ת.ז של מגישי הפרויקט. זה גם הקובץ שתגישו בתיבת ההגשה לכשתסיימו את הפרויקט.
2. נתונים ... נתונים... – את הנתונים של הפרויקט תוכלו להוריד מהמודל – תחת **קובץ נתונים עבור הפרויקט**. העתיקו את קובץ ה csv תחת אותה ספריה יחד עם המחברת שלכם
3. בנוסף, נצטרך קוד של ספריית עזר שתשמש אותנו לבניית גרפים, תוכלו להוריד מהמודל תחת **ספריית גרפים עבור הפרויקט**. העתיקו את קבצי הספריה תחת אותה ספריה יחד עם המחברת שלכם.
4. זהו.. יש לנו את כל הקבצים הנדרשים, אפשר להתחיל במלאכת הניתוח. כמו בכל ניתוח שביצענו עד כה, מתחילים במקבץ של ייבוא ספריות. שימו לב – הפעם במקום להעתיק ישירות קוד מתוך מסמך ההנחיות (מסמך זה) עליכם למצוא את הגדרות הייבוא של הספריות שאבקש לבד – הן כולן נמצאות בתרגילים שקיבלתם. נא לייבא את הספריות הבאותL
   1. ספריית numpy לחישובים נומריים. נא לקרוא למשתנה המייצג את הספרייה בשם np
   2. ספריית pandas לביצוע פעולות טבלאיות. נא לקרוא למשתנה המייצג את הספריה בשם pd
   3. ספריית time באמצעות הפקודה הבאה:

**from time import time**

* 1. ספריית הגרפים באמצעות הפקודה הבאה:

**import visuals as vs**

* 1. הוסיפו פקודה המורה לספריה matplotlib להציג את הגרפים בתוך המחברת (תזכורת... כמעט בכל השיעורים עשינו את זה וגם בתרגילים...)

1. יופי, עכשיו צריכים לקרוא את קובץ הנתונים (.csv) שלנו. בצעו זאת באמצעות הפקודה read\_csv של pandas –אל משתנה (Data Frame) בשם **data**
2. הדפיסו את **3** השורות הראשונות של הקובץ אל המחברת
3. הסבר על השדות של הקובץ תוכלו לקבל כאן: <https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.names>
4. כתבו במחברת הסבר משלכם בעברית לגבי המשמעות של כל שדה.
5. חשבו באמצעות פקודה את מספר הרשומות (השורות) בנתונים שלנו.
6. בכיתה למדנו על הפקודה describe על מנת להציג מדדים סטטיטיים על נתונים נומריים. השתמשו בפקודה וכתבו במחברת:
   1. מהו החציון של גיל האנשים בנתונים שלנו?
   2. מהו האחוזון ה 75 של מספר שנות הלימוד (education-num)
   3. האם השדה capital-gain מתפלג התפלגות נורמאלית? על פי מה קבעתם?
   4. האם השדה hours-per-week מתפלג הפלגות נורמאלית? על פי מה קבעתם?
7. בצע היסטוגרמה (כפי שלמדת בתרגיל) של השדה age בלבד. ההיסטוגרמה תכיל 70 תאים.
8. חשב מהו **אחוז** **האנשים** אשר מרוויחים מעל 50 אלף דולרים. ניתן לבצע זאת על ידי חלוקת מספר האנשים המרווחים מעל 50 אלף דולרים במספר השורות שבטבלה. כנוסחת עזר, על מנת לחשב את מספר האנשים המרווחים מעל 50 אלף דולרים תוכל לבצע באמצעות הפקודה:

**(data['income'] == '>50K').sum()**

1. חשב את מספר האנשים הגרושים. בעיקרון יכלנו למצוא את מספר השורות שבהן הערך של השדה marital-status יהיה Divorced אך שימו לב ש Divorced עשוי להכיל גם תווי רווח – ואז ההשוואה תיכשל. נסו בעצמכם:

**(data['marital-status'] == 'Divorced').sum()**

כיוון שזה לא עבד, בואו נראה לדוגמא שורה שבה רואים בבירור שהסטאטוס Divorced עשוי להכיל רווחים. למשל נציג את השורה השלישית:

**data['marital-status'][2]**

שימו לב לרווחים שם. כיוון שלא נרצה להמר על מספר הרווחים, פשוט נבקש את השורות אשר שדה הסטאטוס **מכיל** את המחרוזת "Divorced" במקום ממש להשוות. דוגמא לאיך לבקש שורות בתוך data frame המכילות ערך ספציפי בשדה מסוים תוכלו לראות כאן למשל:

<https://stackoverflow.com/questions/32616261/filtering-pandas-dataframe-rows-by-contains-str>

ברגע שתקבלו את השורות המתאימות ב dataframe, את מספר השורות תוכלו לקבל באמצעות הפונציה len. הראו באמצעות החישוב את מספר השורות המכילות את המחרוזת Divorced בשדה marital-status. *(אם ביצעתם זאת נכון, סכום הספרות של המספר שתקבלו אמור לצאת 24...)*

1. הציגו מפת חום של האחוזונים של השדות בטבלה עבור השורות עם האינדקסים הבאים: **206,8123,33112,45120,10000**. .
2. כתוצאה ממפת החום שהצגתם, כתבו במחברת:
   1. במה שורות 8123, 33112 דומים מבחינת האחוזונים?
   2. אותו הדבר לשורות 206, 45120
3. על מנת שנוכל לבצע חישובים עתידיים על נגזרות של הטבלה, ניצור טבלה המכילה רק את עמודת ה income (שאותה נרצה לחזור בהמשך) וטבלה המכילה את כל העמודות מלבד העמודה income. לטבלה המכילה רק את income נקרא בשם המשתנה income\_raw. לטבלה המכילה את יתר השדות נקרא בשם features\_raw.
4. כעת נסתכל על נתונים הרווח וההפסד ההוני (capital-gain, capital-loss) של האנשים בטבלה שלנו. למי שצריך הסבר על משמעות המושגים – ניתן למשל לקבל כאן <https://economictimes.indiatimes.com/definition/capital-gainloss> . ראשית – במחברת - הציגו היסטוגרמה של Capital Gain ושל Capital Loss
5. שימו לב עד כמה הנתונים של Capital Gain ושל Capital Loss מתפלגים עם Skew חזק. נשתמש במה שלמדנו על טרנסופרציות Box Cox – ונפעיל פונקציית Log על מנת לנרמל את הנתונים במידת האפשר. נתחיל בכך שנייצר רשימה של השדות שאנחנו מעוניינים לנרמל:

skewed = ['capital-gain', 'capital-loss']

כעת נייצר DataFrame עם עותק מהנתונים של המשתנה שיצרנו קודם - features\_raw באמצעות הפקודה:

features\_log\_transformed = pd.DataFrame(features\_raw)

כעת נבצע טרנספורמציית Log לשדות שביקשנו:

features\_log\_transformed[skewed] = features\_raw[skewed].apply(lambda x: np.log(x + 1))

עכשיו, השתמשו ב features\_log\_transformed על מנת להציג היסטוגרמה של Capital Gain ושל Capital Loss לאחר הנירמול. שימו לב כי בצד שמאל של ההיסטוגרמה יש ערך שמשתלט על הגובה , לכן על מנת להציג את אותו הגרף עם מגבלה בציר ה y אפשר למשל לכתוב כך לאחר פקודת התצוגה להיסטוגרמה:

plt.ylim(ymax=2000)

זה יגביל את ציר ה y לערך 2000.

אמנם התיקון שביצענו לא היה מושלם מבחינת הנירמול, אך כעת הנתונים מאפשרים ניתוח יותר מעמיק מהמצב הקודם.

1. אם נציץ רגע בתוצאת ה describe שביצענו בשלבים הקודמים, ניתן לראות שלכל פרמטר נומרי יש טווח ערכים שונה. למשל Capital-Loss מגיע עד לערך 99999 ואילו Capital-gain מגיע רק עד 4356. ההבדלים בטווח הערכים עלולים להשפיע על השיטות/אלגוריתמים שנשתמש בהם בהמשך.על מנת למזער את השפעת השוני בטווחי הערכים, נרצה שכל הערכים הנומריים ינורמלו בין 0 ל 1 (נקרא גם "סקיילינגScaling/ וגם טרנספורמציית minmax"– כך שלא תהיה לכל אחד מהם השפעה ניכרת באופן יחסי על השיטות המתמטיות השונות שנפעיל בהמשך. נתחיל ב 2 ספריות שצריך לייבא:

ספריה עבור הנירמול (מתבצע אוטומאטית בין 0 ל 1):

from sklearn.preprocessing import MinMaxScaler

וכן ספריה עבור תצוגת תוצאות (כפי שהצגנו כבר בתרגילים קודמים):

from IPython.display import clear\_output, Image, display

השלב הבא הוא להגדיר משתנה עבור מנגנון הסקיילינג בין 0 ל 1:

scaler = MinMaxScaler()

ולייצר טבלה (data frame) חדשה עבור הנתונים שעברו סקיילינג בשם features\_log\_minmax\_transform כך:

features\_log\_minmax\_transform = pd.DataFrame(data = features\_log\_transformed)

כעת נרצה לבצע את הסקיילינג, ראשית נגדיר מי הם השדות הרלוונטיים (עמודות נומריות):

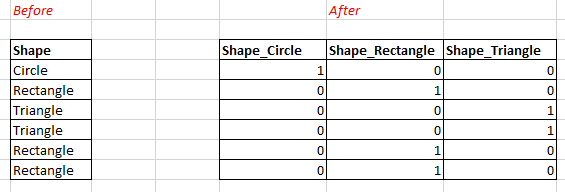
numerical = ['age', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week']

ונבצע את הסקיילינג עצמו על ערכי העמודות הנומריות:

features\_log\_minmax\_transform[numerical] = scaler.fit\_transform(features\_log\_transformed[numerical])

לסיום, הציגו את 7 השורות הראשונות של הנתונים שעברו סקיילינג (*אשר נמצאים ב features\_log\_minmax\_transform*)

1. כפי שהצהרנו בהתחלה, המטרה שלנו היא ללמוד (באמצעות אלגוריתם קלאסיפיקציה) מה מאפיין את מי שמרוויח הרבה לעומת מי שמרוויח מעט, או אם לנסח את זה אחרת – לחזות אם אדם מסויים מרוויח מעל או מתחת ל 50 אלף דולרים בהינתן הידע על יתר השדות. הבעיה היא שבעוד שאלגוריתמי הלמידה של הספריה שלנו – Scikit Learn יודעים להתמודד היטב עם נתונים נומריים (ערכים מספריים) – הם לא מתמודדים עם ערכים דיסקרטיים/בדידים/מחרוזות/שמות. על מנת לאפשר לשיטות שלמדנו לעבוד גם על נתונים שאינם נומריים, נצטרך להמיר אותם לנומריים. אחת הטכניקות לבצע זאת נקראת one-hot-encoding – והרעיון מאוד פשוט – נניך ששדה (עמודה בטבלה) מסוים יכול להכיל 3 ערכים שונים:לצורך הדוגמא: משולש, עיגול וריבוע. אז נקודד את הערכים האלה באמצעות 3 עמודות חדשות המייצגות כל ערך בהתאמה כך:



למזלנו, ב pandas יש פונקציה מוכנה שיודעת לבצע זאת על השדות הלא נומריים באופן אוטומאטי. היא נקראת get\_dummies, ומשתמשים בה כך:

הטבלה החדשה = pd.get\_dummies(הטבלה שעליה רוצים לבצע)

בצעו זאת על הטבלה שקיבלנו מהשלב האחרון של טרנספורמציית minmax. לטבלה החדשה קראו בשם features\_final

1. במחברת – הדפיסו את מספר העמודות שקיבלנו לאחר המרת one-hot-encoding
2. הדפיסו את שמות כל העמודות לאחר המרת one-hot-encoding
3. כעת נבצע גם המרה לשדה שרצינו לחזות (income\_raw). שם אין צורך להשתמש בכלי כבד כמו one-hot-encoding כי השדה הזה מכיל רק 2 ערכים, אז פשוט נמיר את הערך של מי שמרוויח הרבה (>50K) לערך 1 ואת הערך של מי שמרוויח מעט נמיר ל 0. נבצע זאת כך:

income = income\_raw.apply(lambda x: 1 if x == '>50K' else 0)

1. כעת, פנינו לכיוון הפעלת אלגוריתם הקלאסיפיקציה... אך רגע לפני... זיכרו שכל אלגוריתם מפעילים בד"כ על חלק אחד (רוב) של הנתונים ונותנים לו ללמוד מודל (נקרא גם training set) ולאחר מכן כדי לבחון עד כמה האלגוריתם טוב, בודקים אותו על החלק השני – מה שנותר מהנתונים (נקרא גם test set). בצעו חלוקה של שדות התכונות (מתוך features\_final) ושל שדה המטרה/פרדיקציה (income) על פי משקל של 80 אחוז לטובת ה training set באמצעות הפקודה train\_test\_split אל משתנים נפרדים כפי שלמדנו בשמות : X\_train, X\_test, y\_train, y\_test.
2. הדפיסו במחברת את מספר השורות של ה training set ושל ה test set – וודאו שהם מתחברים יחד למספר השורות בטבלה המקורית (שחישבתם כבר בשלב מוקדם יותר)
3. כעת נרצה לבצע את הפרדיקציה עצמה. הפעם, במקום להריץ שיטה אחת – נבחן מספר שיטות ונשווה את הביצועים שלהן. ראשית, נגדיר ספריות שמטרתן לבחון את ביצועי האלגוריתמים השונים:

from sklearn.metrics import fbeta\_score, accuracy\_score

תוכלו לקבל אינפורמציה לגבי המשמעות של accuracy ושל Fbeta score בלינק הבא:

<https://en.wikipedia.org/wiki/F1_score>

כעת, נגדיר פונקציה שמטרתה להריץ אלגוריתם ספציפי על מדגם בגודל מסוים מתוך הנתונים.

נתחיל בהגדרת תבנית הפונקציה:

def train\_predict(learner, sample\_size, X\_train, y\_train, X\_test, y\_test):

שימו לב שקוד הפונקציה נמצא תחת Tab אחד ימינה מההגדרה שלה (זה יקרה אוטומאטית אם פשוט תלחצו על מקש ה enter לאחר שורת ההגדרה).

בקוד הפונקציה נבצע את הפעולות הבאות:

* 1. הפונקציה צריכה להחזיר את תוצאות הביצועים של האלגוריתם הספציפי למבנה בשם results. ראשית נאתחל אותו להיות ריק כך:

results = {}

* 1. לאלגוריתמי למידה מסוג קלאסיפיקציה ישנם 2 שלבים כזכור מהשיעורים: אחד נקרא fit שבו אנו לומדים מודל למידה מנתונים ה training set והשני נקרא predict שבו אנו בודקים את הדיוק של המודל שלמדנו על ה test set. מכיוון שבעבודה הזאת כמות הנתונים גדולה, זמן ריצת האלגוריתם בשלביו השונים (fit, predict) עשוי להיות חלק מהשיקולים שלנו בבחירת האלגוריתם המתאים. לכן נרצה למדוד את זמני הריצה של fit ו predict. נתחיל ב fit כמובן – נאתחל משתנה בשם start שיזכור עבורנו את שעת התחלת ריצת האלגוריתם כך:

start = time()

כעת נריץ את שלב ה fit עבור לימוד המודל מנתוני ה training set כך:

learner = learner.fit(X\_train[:sample\_size], y\_train[:sample\_size])

שימו לב שבמקום לתת את כל הנתונים, אנו בוחרים מתוך ה training רק כמות מסויימת של שורות הנקראת sample\_size – שזה גם פרמטר שהעברנו לפונקציה שלנו. זאת על מנת שנוכל בהמשך לקרוא לאותה הפונקציה בכל פעם עם sample בגודל אחר (המשך יבוא... 😊). נחזור לעניינינו... אחרי שהרצנו את שלב ה fit נרצה למדוד כמה זמן זה לקח, אז ניקח גם את השעה של אחרי זמן הריצה כך:

end = time()

כל מה שנותר זה לחשב את הפרש הזמן בין start לבין end ולשמור בתוצאות את זמן הריצה, כך:

results['train\_time'] = end – start

* 1. כעת נבצע את מדידת הזמן גם לשלב ה predict. אנו נבצע את הפרדיקציה פעמיים, פעם אחת על הנתונים של ה test set (X\_test) ופעם שניה על 300 השורות הראשונות מתוך ה training set.

ראשית עלינו להפעיל את השעון להתחלת המדידה, כפי שביצענו לפני הקריאה של fit:

start = השם את הקוד הנדרש

נתחיל ב test set:

predictions\_test = learner.predict(X\_test)

ואז שוב נפעיל את predict על ה 300 שורות הראשונות של ה training set:

predictions\_train = **השלם את הקוד שחסר**

ושוב נעדכן את השעון לציון סיום הפעולה:

end = השלם את הקוד הנדרש

ונשמור את הפרש הזמנים למשתנה בתוצאות :

results['pred\_time'] = השלם את הקוד הנדרש

* 1. נרצה לשמור בתוצאות גם את הביצועים/מדדי הדיוק (accuracy, fbeta) של האלגוריתמים:

נתחיל במדד ה accuracy עבור ה training set שנעשה ב 300 השורות הראשונות על ידי השוואת שדה החיזוי של ה training set (y\_train) עם תוצאת הפרדיקציה של ה train:

results['acc\_train'] = accuracy\_score(y\_train[:300], predictions\_train)

נמשיך עם ה accuracy של ה test set , על ידי השוואת שדה החיזוי של ה test set עם תוצאות הפרדיקציה של ה test:

results['acc\_test'] = השלם את הקוד

* 1. כעת נבצע את שמירת הביצועים על פי שיטת FBetaScore עבור ה train וה test:

results['f\_train'] = fbeta\_score(y\_train[:300], predictions\_train, beta = 0.5)

results['f\_test'] = fbeta\_score(y\_test, predictions\_test, beta = 0.5)

* 1. לקראת סיום הפונקציה, נרצה להדפיס את תוצאות הריצה של האלגוריתם שנקרא עבור הפונקציה במקרה הספציפי כך:

print("{} trained on {} samples.".format(learner.\_\_class\_\_.\_\_name\_\_, sample\_size))

* 1. ולסיום הפונקציה צריכה להחזיר את התוצאות:

return results

1. הגענו לשלב המעניין... נרצה להריץ 3 אלגוריתמים שונים על הנתונים שלנו. האלגוריתמים הם:
   1. AdaBoostClassifier
   2. GradientBoostingClassifier
   3. SGDClassifier

בנוסף, עבור כל אחד מהאלגוריתמים, נריץ אותו למעשה 3 פעמים – עבור גדלים שונים של דגימות/sample (כזכור הפונקציה שבנינו יותר לקבל את גודל המדגם כפרמטר).

נתחיל בהצהרה על ספריות האלגוריתמים שבחרנו:

from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier

from sklearn.linear\_model import SGDClassifier

נאתחל את המודלים של האלגוריתמים שבחרנו:

clf\_A = AdaBoostClassifier(random\_state=0)

clf\_B = GradientBoostingClassifier(random\_state=0)

clf\_C = SGDClassifier(random\_state=0)

נקבע את הגודל של 3 הסוגים של המדגמים:

* אחד של 100 אחוז (ללא דגימה למעשה...)
* אחד של 10 אחוז מהנתונים
* אחד של אחוז אחד מהנתונים

נבצע זאת כך:

samples\_100 = int(len(y\_train))

samples\_10 = int(len(y\_train) \* 0.1)

samples\_1 = השלם את החסר

כעת נאתחל את התוצאות במשתנה מקומי, טרם הרצת האלגוריתמים:

results = {}

ואז עבור על אחד מהאלגוריתמים, נרוץ על 3 האפשרויות של המדגמים ונריץ את הפונקציה שבנינו שתאסוף עבורנו את תוצאות הריצה של האלגוריתמים השונים:

for clf in [clf\_A, clf\_B, clf\_C]:

clf\_name = clf.\_\_class\_\_.\_\_name\_\_

results[clf\_name] = {}

for i, samples in enumerate([samples\_1, samples\_10, samples\_100]):

results[clf\_name][i] = train\_predict(clf, samples, X\_train, y\_train, X\_test, y\_test)

לבסוף, נשתמש בספריה שצרפנו על מנת להציג את הגרפים המייצגים את תוצאות הניסויים שערכנו:

vs.evaluate(results)

1. בגרפים שקיבלנו: (ענו במחברת)
   1. באיזה מהאלגוריתמים שהרצנו תשתמשו על מנת לקבל תוצאות בזמן המהיר ביותר?
   2. אם **זמן** **לא היה** שיקול בבחירת האלגוריתמים, באיזה מבין 3 האלגוריתמים תבחרו ומדוע?
2. כעת נרצה לבחון אם רמת הדמיון בין אשכולות (קלאסטרים) שונים של אנשים מתוך הנתונים שלנו. בצעו Clustering באמצעות אלגוריתם GMM עם מספר קומפוננטות בין 2 ל 7 ומצאו מהו מספר הקופוננטות המספק את ה cluster-ים עם ה silhouette הגבוה ביותר. הנתונים עבור תהליך ה clustering יילקחו מתוך features\_final, 10000 שורות ראשונות.
3. בצעו את הקלאסטרינג גם עבור אלגוריתם KMeans עם K בין 2 ל 7.

**בהצלחה!!!**