אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

23/1/2022

פרויקט חלק ב'

קורס לימוד מכונה

מרצה: ד"ר ניר ניסים

עוזר הוראה: דור זזון

מגישות:

נוי כשר 314963810

ניצן אגם 313360489

קבוצה: 3

תוכן עניינים

Model Training

על מנת להכין את ה-dataset בצורה הטובה ביותר לפני אימון המודלים השונים עלינו לבצע שינויים לאחר קבלת המשוב מחלק א'. השינויים שבוצעו בשלב זה הינם:

1. לאחר ההתמודדות עם **missing values**, החלטנו למחוק רשומות שנותרו עם ערכים מסוימים חסרים.
2. השמת הערך הנפוץ ביותר עבור samples בעלי missing values בפיצ'רים כמו company_type, last_new_job, company_size.
3. **לא נשתמש ב-PCA** – מכיוון שזו שיטה לפיצ'רים רציפים, ורוב ה-data שלנו מורכב מפיצ'רים קטגוריאליים.
4. ביצוע **shuffle** לתצפיות – על מנת להגדיל את ה-exploration ולמנוע חוסר איזון בחלוקת הsamples בשיטת הוולידציה נשנה את סדר התצפיות.
5. חלוקת ה-data ל- K=10 כדי להשתמש בשיטת ה- **k-fold cross validation.**
6. **סטנדרטיזציה** – המודלים ANN, SVM ו-K-MEDOIDS Clustering מספקים תוצאות מוצלחות יותר כאשר המשתנים הרציפים בעלי סקאלות זהות. את הנרמול ביצענו באמצעות נוסחת התקנון $z=(x-\mu)/\sigma$, כך שכל פיצ'ר יהיה בעל תוחלת 0 וסטיית תקן 1.

evelopment_	ent_exper	lled_unive	experience
-3.10607	0	1	-1.21816
-1.67372	1	2	-0.0347675
0.65384	1	0	0.409006
0.442244	1	2	-0.922314
0.65384	1	0	-0.0347675
-0.208824	1	0	1.5924
0.775916	1	0	-0.626465
0.548042	1	0	1.5924
-1.67372	1	2	-0.626465

Decision Trees

Hyperparameter Tuning

תחילה נריץ את המודל עם הפרמטרים הדיפולטיביים לצורך השוואה. לאחר מכן, נבצע תהליך של Hyperparameter Tuning ע"י grid search CV (בהתאמה ל-K-folds שבחרנו), על מנת לבחור בקונפיגורציה המיטבית עבור משימת הלימוד.

<u>משמעות הפרמטרים אותם נבחן בתהליך:</u>

- criterion המדד לבחירת איכות הפיצול – gini\entropy.
- Splitter - האסטרטגיה לבחירת הפיצול – לפי בחירה מיטבית או רנדומלית.
- max_depth עומק עץ מקסימלי – טווח הערכים נע בין 1:35. נשאף שיהיה קטן ככל שניתן על מנת שהמודל יהיה מובן וקל להסברה.
- min_samples_split מספר דגימות מינימלי הנדרש לפיצול צומת פנימי בעץ – טווח ערכים נע בין 10:500 עם קפיצות של 10.
- ccp_alpha רמת מובהקות של קטימת הענפים – טווח הערכים נע בין 0:1 עם קפיצות של 0.05.

<u>עבור מודל הלמידה הטוב ביותר התקבלו הפרמטרים:</u> (<u>נספח טבלת קריטריונים</u>)

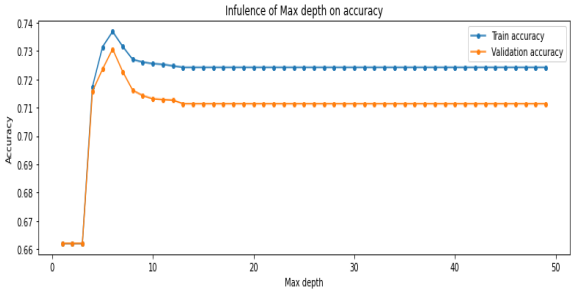
קריטריון – gini, **פיצול** – best, **max_depth** – 7, **min_samples_split** – 370, **ccp_alpha** – 0.

<u>אחוזי הדיוק במודל המדויק ביותר:</u> **סט האימון:** 93.8175%, **סט הוולידיציה:** 79.6018%

אחוזי הדיוק של סט הוולידציה נמוכים יותר מאשר על סט האימון כמצופה, זאת משום שהמודל נמדד על דאטה שאינו התאמן עליו. בנוסף, ממבט ראשוני ניתן לחשוב כי אנחנו נמצאים במצב של over fit, זאת כיוון שאחוזי הדיוק של סט האימון גבוהים מאוד. אולם, הקונפיגורציה המיטבית אשר נבחרה על ידי grid search CV הניבה את אחוזי הדיוק הטובים ביותר בסט הוולידציה ועל כן בחרנו בה.

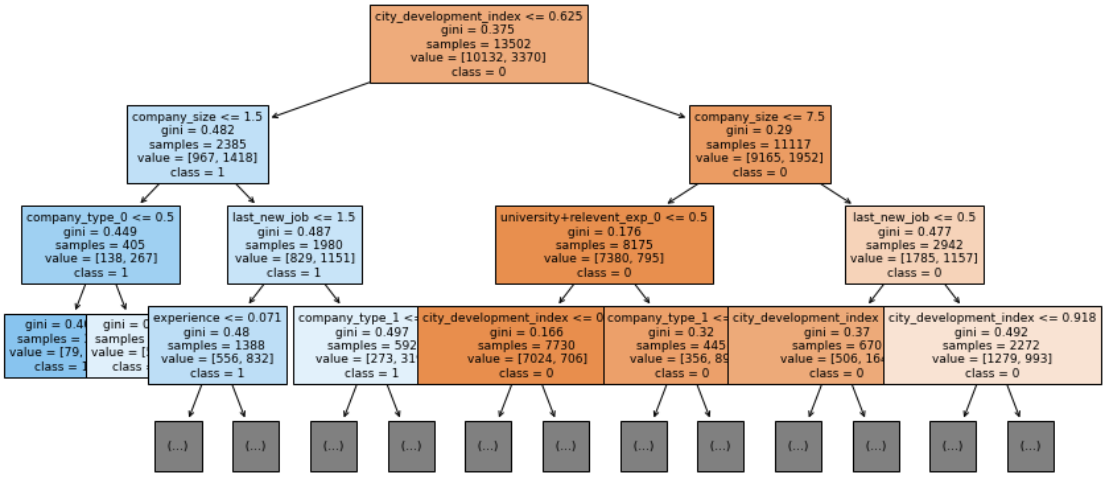
Interpretability

אחד היתרונות של מודל עץ ההחלטה הוא יכולת ההסברה של המודל, כלומר בעת הסתכלות על המודל ניתן להבין איזו החלטה התקבלה בכל שלב – איזה feature היה המשמעותי ביותר לחלוקה בכל פיצול (מפחית יותר רעש במודל) ומהו ה-threshold לחלוקה, וכך ניתן להבין מהו הסיווג אותו נקבל בהינתן sample חדש מהתבוננות בצורה הוויזואלית של העץ בלבד.

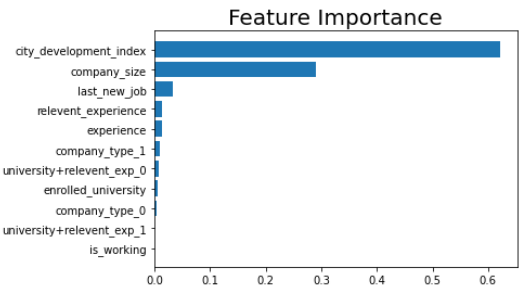


עם זאת, ככל שנאפשר לעומק העץ המקסימלי להיות גדול יותר כך גם יכולת ההסברה של המודל תרד. יכולת ההסברה של המודל מסייעת לנו להבין את משימת הלימוד בצורה טובה יותר, שכן נוכל לצפות ברמות הראשונות של העץ על מנת להבין מיהם הפיצ'רים בעלי ההשפעה המשמעותית ביותר במודל, וכך לבחון שינויים ושיפורים נוספים בהם.

Graphs



מהתבוננות בגרף, ניתן לראות כי ברמות הראשונות של העץ ה-feature המשמעותי ביותר הינו city_development_index , כיוון שהוא מצליח להבחין בצורה יחסית טובה בין class=0 לבין class=1.



כמו כן, הוא ממשיך לשמש אותנו בפיצולים מתקדמים יותר. בהקשר למשימת הלמידה נסיק כי אם העובד מגיע מעיר מפותחת אזי הסיכוי שיעזוב את מקום עבודתו קטן. Features משמעותיים נוספים הינם company_size כלומר גודל החברה של העובד, וכן last_new_job כלומר כמה שנים חלפו מאז עבודתו האחרונה.

בגרף ה-**feature importance** ניתן לראות כי שלושת ה-features המשמעותיים ביותר למשימת הלמידה הם כפי שצפינו בהתבוננות בגרף העץ. הם האינפורמטיביים ביותר להבנה האם העובד יעזוב את מקום עבודתו. עם זאת, נראה כי שאר ה-features בעלי חשיבות כמעט אפסית לסיווג.

Artifitial Neural Networks

Default Network

ברשת הנוירונים יש משמעות לסקאלת המשתנים הרציפים, ועל כן ננרמל אותם לפי standard-scaler ולא לפי min-max כפי שביצענו בשלב ה-model training.

<u>משמעות הקונפיגורציה בערכי ברירת המחדל</u>

א. מספר נוירונים בשכבת הכניסה – כמספר ה-features ב-data.

ב. מספר השכבות החבויות בברירת המחדל היא אחת.

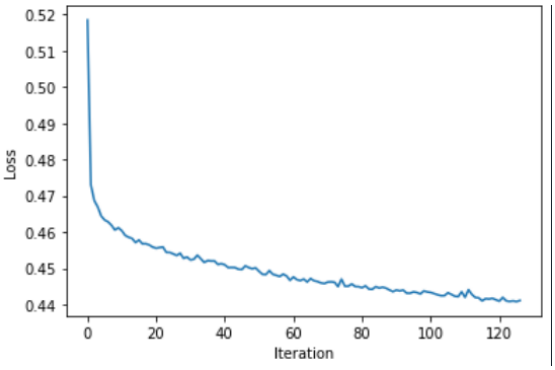
ג. מספר נויורונים חבויים בכל שכבה – 100 (מספר יחידות העיבוד עבור כל מאפיין).

ד. שיטת אקטיבציה – 'relu'. שיטת ברירת המחדל אינה מתאימה למשימת הלימוד שלנו בשל אופיה הלינארי, ולכן בשלב כיוונון הפרמטרים נחליפה ל-logistic.

ה. שיטת solver – 'adam', שיטה זו מתאימה ל-datadest גדולים כמו שלנו, מבחינת זמני אימון וגם אחוז דיוק של סט הולידציה.

ו. מספר נוירונים בשכבת היציאה – 2, כמספר מחלקות הסיווג של משימת הלימוד.

<u>אחוזי דיוק במודל הדיפולטיבי</u> (<u>נספח אחוזי דיוק</u>): **סט האימון:** 67.85%, **סט הולידציה:** 66.17%.



לכאורה אחוזי דיוק לא נמוכים מאוד, אך נרצה לשפר אותם ולהגיע לאחוזי דיוק גבוהים בעזרת כוונון היפר פרמטרים שיניבו תוצאות טובות יותר מאלו שקיבלנו במודל זה. ניתן לראות שערך ה-loss הולך וקטן ככל שמספר האיטרציות עולה. המחיר שנשלם על הטעות הוא יקר, ואם האלגוריתם "ידלג" על תצפיות המהוות רעש ב-data הסיכוי להיות במצב over fitting הולך ופוחת, ולהפך.

Hyperparameter Tuning

<u>בחירת פרמטרים, המוטיבציה לבחירתם והמשמעות על הרשת</u>

- **מספר השכבות החבויות –** בחרנו לכוון את משתנה זה על מנת לראות את ההשפעה של רשת עמוקה על ביצועי משימת הלימוד. ככל שנעמיק את מספר השכבות כך המודל יהיה מורכב יותר כתלות במספר הקומבינציות הלינאריות של הנוירונים בשכבה הקודמת. נבדוק 1-3 שכבות חבויות, על מנת לראות אם קיים שינוי באיכות הפתרון.
- **מספר נויורונים חבויים בכל שכבה –** ככל שמספר הנוירונים בשכבה חבויה גדל כך גדלה יכולת המיפוי של הרשת, כלומר כוח ההסברה של המודל גדל. מנגד, מספר נוירונים גדול מידי ימנע לימוד יעיל. לצורך בחירת הפרמטרים בחרנו בטווח ערכים של 8:32 בקפיצות של 8.

- **שיטת אקטיבציה –** פונקציית האקטיבציה עבור כל נוירון בשכבה החבויה. פונקציה זו מאפשרת לרשת להפוך input ל-output רצוי. בחרנו בשיטה logistic: $f(x) = 1/(1 + \exp(-x))$, כדי להתאים למשימת סיווג.
- **Learning rate initial –** קצב הלמידה שולט על גודל הצעד בעדכון המשקולות. קיים טרייד-אוף בין קצב הלמידה לזמן הלמידה של הרשת כולה: ככל שקצב הלמידה גדול יותר, עדכון המשקולות יהיה גדול יותר כתלות בשגיאה הנוצרה בתהליך ה-back propagation, כלומר נצטרך פחות epoks. החסרון הוא שאנו עלולים 'לדלג' על גודל המשקולות האופטימלי. לעומת זאת, קצב למידה נמוך יחסית יביא אותנו למשקל אופטימלי אך עלול לקחת זמן רב. טווח הערכים הנבחר : 0.001:0.05 בקפיצות של 0.01, כיוון שנרצה לבחון את ההשפעה הנ"ל על דיוק המודל ויכולתו להביא לתוצאות טובות ומהירות יותר.
- **Learning rate –** קצב למידת הרשת יהיה קבוע/משתנה. במשתנה, קצב הלמידה יישאר קבוע כל עוד פונקציית ה-training loss ממשיכה לרדת.

Index	params	mean train scor
36	{'activation': 'logistic', 'hidden_layer_sizes': (32, 32), 'learning_rate_init': 0.011, 'max_iter': 500}	0.845578
37	{'activation': 'logistic', 'hidden_layer_sizes': (32, 32), 'learning_rate_init': 0.020999999999999998, 'max_iter': 500}	0.842655
38	{'activation': 'logistic', 'hidden_layer_sizes': (32, 32), 'learning_rate_init': 0.030999999999999996, 'max_iter': 500}	0.830268
39	{'activation': 'logistic', 'hidden_layer_sizes': (32, 32), 'learning_rate_init': 0.040999999999999995, 'max_iter': 500}	0.820814
28	{'activation': 'logistic', 'hidden_layer_sizes': (16, 32), 'learning_rate_init': 0.030999999999999996, 'max_iter': 500}	0.813705
31	{'activation': 'logistic', 'hidden_layer_sizes': (32,), 'learning_rate_init': 0.011, 'max_iter': 500}	0.812911
21	{'activation': 'logistic', 'hidden_layer_sizes': (16, 16), 'learning_rate_init': 0.011, 'max_iter': 500}	0.812751
32	{'activation': 'logistic', 'hidden_layer_sizes': (32,), 'learning_rate_init': 0.020999999999999998, 'max_iter': 500}	0.811191
33	{'activation': 'logistic', 'hidden_layer_sizes': (32,), 'learning_rate_init': 0.030999999999999996, 'max_iter': 500}	0.810235
27	{'activation': 'logistic', 'hidden_layer_sizes': (16, 32), 'learning_rate_init': 0.020999999999999998, 'max_iter': 500}	0.81021
22	{'activation': 'logistic', 'hidden_layer_sizes': (16, 16), 'learning_rate_init': 0.020999999999999998, 'max_iter': 500}	0.810131

Index	params	mean test scor
26	{'activation': 'logistic', 'hidden_layer_sizes': (16, 32), 'learning_rate_init': 0.011, 'max_iter': 500}	0.789087
25	{'activation': 'logistic', 'hidden_layer_sizes': (16, 32), 'learning_rate_init': 0.001, 'max_iter': 500}	0.788688
6	{'activation': 'logistic', 'hidden_layer_sizes': (8, 16), 'learning_rate_init': 0.011, 'max_iter': 500}	0.788464
29	{'activation': 'logistic', 'hidden_layer_sizes': (16, 32), 'learning_rate_init': 0.040999999999999995, 'max_iter': 500}	0.788406
12	{'activation': 'logistic', 'hidden_layer_sizes': (8, 32), 'learning_rate_init': 0.020999999999999998, 'max_iter': 500}	0.78839
11	{'activation': 'logistic', 'hidden_layer_sizes': (8, 32), 'learning_rate_init': 0.011, 'max_iter': 500}	0.788349
30	{'activation': 'logistic', 'hidden_layer_sizes': (32,), 'learning_rate_init': 0.001, 'max_iter': 500}	0.788262
7	{'activation': 'logistic', 'hidden_layer_sizes': (8, 16), 'learning_rate_init': 0.020999999999999998, 'max_iter': 500}	0.788231
15	{'activation': 'logistic', 'hidden_layer_sizes': (16,), 'learning_rate_init': 0.001, 'max_iter': 500}	0.788201
35	{'activation': 'logistic', 'hidden_layer_sizes': (32, 32), 'learning_rate_init': 0.001, 'max_iter': 500}	0.787805

<u>ערכי ההיפר-פרמטרים כפונקציה של אחוזי הדיוק</u>

בסט האימון ניתן לראות שככל שמספר הנוירונים בשכבה חבויה עולה, כך עולה אחוז הדיוק של המודל. בנוסף, ככל שנאתחל את המודל עם קצב למידה קטן אחוזי הדיוק גדלים גם כן. מספר השכבות החבויות בשלושת המודלים הטובים ביותר עומד על 2, מה שמרמז על כך שהמודל יכול להתמודד עם dataset מורכב באופן יחסי.

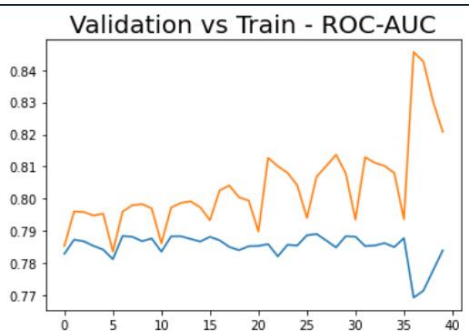
גם בסט הולידציה מספר השכבות החבויות הוא 2, ושלושת המודלים הטובים ביותר הם בשילוב מספר נוירונים גבוה בכל שכבה חבויה. בנוסף, נבחין שקצב הלמידה ההתחלתי אינו הנמוך היותר, אך הוא אכן ביון שלושת המודלים הטובים ביותר.

Best Model Configuration

הפרמטרים הטובים היותר עבור הdataset שלנו הם: שיטת אקטיבציה לוגיסטית, שתי שכבות חבויות עם (16,32) נוירונים, קצב הלמידה ההתחלתי הוא 0.011, מספר איטרציות מקסימלי של 500. (<u>נספח</u>)

<u>אחוזי דיוק במודל המדויק ביותר</u>

סט האימון: 80.69%, **סט הולידציה:** 78.90%.



לאורך הגרף אחוזי הדיוק של סט הולידציה נמוכים מאשר סט האימון, ובעזרת ה-grid search לא הגענו למצב של over fitting בגלל שבחירת המודל הטוב ביותר מתבסס לפי ה-validation score הגבוה ביותר.

<u>ההבדל מתוצאות הרשת הדיפולטיבית</u>

אחוזי הדיוק במודל הטוב ביותר גבוהים מהמודל הדיפולטיבי, שכן ביצענו כוונון פרמטרים שונים שמטיבים עם ה-dataset יותר מאשר הפרמטרים הדיפולטיביים, למשל – בסט האימון השתמשנו ב2 שכבות חבויות במקום אחת ומספר הנוירונים בכל שכבה 32. בנוסף, שיטת האקטיבציה הדיפולטיבית relu אינה מתאימה למשימת הלמידה שלנו ולכן כשהחלפנו לשיטת logistic תוצאות הדיוק עלו בהתאם.

<u>Confusion Matrix</u>

```
In [168]:
     ...: print(confusion_matrix(y_true=y_train,
y_pred=grid_search.predict(X_train_S)))
[[9828 1422]
 [1786 1966]]
```

SVM

Default Baseline

על מנת למצוא baseline, נאמן את ה-dataset על המודל הלינארי הדיפולטיבי, בו ערך הפרמטר C=1. פרמטר זה הוא חלק מה-loss function של SVM. ככל שערכו יהיה גבוה יותר, כך המודל יחמיר את העונש הניתן למרחקים ממישור ההפרדה של תצפיות שסווגו ל-class שגוי.

<u>אחוזי דיוק במודל הדיפולטיבי (נספח):</u> **סט האימון**: 50% , **סט הולידציה:** 50%.

Hyperparameter Tuning

בכוונון הפרמטרים בחרנו לכוונן את ערך הפרמטר C ובחנו ערכים שונים שלו: [0.01,0.1,0.5,1,3,10,100].

ערך הפרמטר המניב את התוצאות הטובות ביותר עבור המודל הוא: **C=100.**

<u>אחוזי דיוק במודל המדויק ביותר:</u> **סט האימון:** 76.7885%, **סט הולידיציה:** 76.6353%.

<u>משוואת המישור המפריד:</u> אלגוריתם ה-SVM מייצר מישור המפריד בצורה הטובה ביותר (כלומר מישור ההפרדה הגדול ביותר) בין שתי המחלקות השונות. כאשר נקבל sample חדש, נבדוק אם הוא נמצא מעל המישור או מתחתיו ולסווגו בהתאם. על מנת לעשות זאת, נשתמש בשיטה **one vs. rest (OVR)** המתאימה בין היתר לסיווג בינארי. נגדיר וקטור המשקולות W_i, i = {0,1} עבור שתי המחלקות הקיימות. בנוסף, נגדיר את β_{0i} עבור חותך המחלקה הi. משוואת המישור המפריד המתקבלת עבור המחלקה ה-i היא: $W_i * X + \beta_{0i} = 0$. ממשוואת המישור ניתן לראות כי משתני ה-CDI הוא בעל החשיבות הגבוהה ביותר למשימת הסיווג, זאת משום שהמקדם המתאים לו הוא המקסימלי בערך מוחלט. מסקנה זו מתיישבת היטב עם המסקנות מהסעיפים הקודמים.

```
In [47]: # printing the 'Betaot'

In [48]: print(model_bestsvm.coef_)
[[-0.05193542  0.00495139  0.00260843 -0.01478783 -0.00056607 -0.01026407
   0.00068354  0.00564568 -0.07518298  0.02418293 -0.00426312]]

In [49]: print('The intercept is:')
The intercept is:

In [50]: print(model_bestsvm.intercept_)
[-0.95249886]
```

בנוסף, המשתנים אשר חילצנו בשלב ה-Feature Extraction – is_working ו-university+relevant_experience גם הם התגלו כמשפיעים על הסיווג. הסימן השלילי של המקדמים של המשתנים שצוינו מרמז על קשר הפוך עם משתנה המטרה.

Unsupervised Learning - Clustering

<u>משימת הלימוד:</u>

מטרת משימת הלימוד משתנה ממשימת Supervised ל-Unsupevised. לעומת המודלים הקודמים, במשימה זו ה-dataset צריך להתקבל **ללא labels**, ומטרת המשימה אינה לסווג את ה-data (מכיוון שבשלב האימון אין ביכולתינו ללמוד על ה-labaling של ה-data), אלא לנסות ליצור אשכולי samples לקבוצות על בסיס מאפיינים דומים. על מנת להתאים את סט הנתונים למשימת הלימוד המתאימה, נסיר את ערכי ה-target מה-dataset.

<u>סט הפיצ'רים ומטריקת המרחק</u>

בקורס למדנו על מטריקות מרחק פופולריות כמו מרחק אוקלידי ומרחק מנהטן, אך הן לא רלוונטיות עבור ה-dataset שברשותנו, זאת מכיוון שהן מתאימות למשתנים רציפים בלבד. ה-dataset שלנו מורכב מפיצ'רים רציפים וקטגוריאליים כאחד, ולכן עלינו להשתמש ב-**Gower distance** על מנת להשתמש **בכל ה-features שלנו.** נוסף על כך, ההבדל בין האלגוריתם K-Means (1) לבין K-Medoids (2) הוא שהראשון מחשב את ה-centroid מתוך ה-cluster בעוד שהשני בוחר את ה-centroid מתוך ה-samples ב-cluster. כלומר, בכל איטרציה באלגוריתם

K-Medoids יבחר centroid בעל הדמיון הממוצע הגבוה ביותר לכל שאר ה-samples באותו ה-cluster. השימוש באלגוריתם זה מותאם גם הוא ל-data קטגוריאלי בשל תהליך בחירת ה-centroid שפירטנו מעלה.

<u>הרצת האלגוריתם</u>

נריץ את אלגוריתם K-Mediods עם K בטווח ערכים של 2:10. כפי שפירטנו לעיל, שיטה זו מאתחלת את ה-centriods בדומה לשיטת K-Means++ ועובדת בדומה לשיטת K-Means מלבד בחירת ה-centrids.

שיטה זו בוחרת בsample מתוך ה-cluster אשר ממזער את ה-loss function (Mi הוא נקודה ב-cluster):

$$M_1, M_2, \cdots, M_k = argmin \sum_{i=1}^{k} \sum_{x \epsilon S_i} ||x - M_i||^2$$

<u>אותם נצפה לראות Clusters</u>

נצפה לראות הפרדה יחסית טובה בין samples המגיעים מערים מפותחות יחסית ובעלי ניסיון תעסוקתי לעומת samples ללא ניסיון המגיעים מערים לא מפותחות.

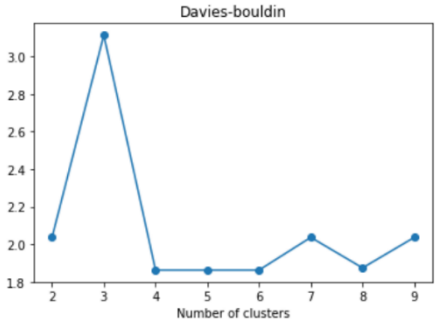
<u>הקריטריונים להשוואה</u>

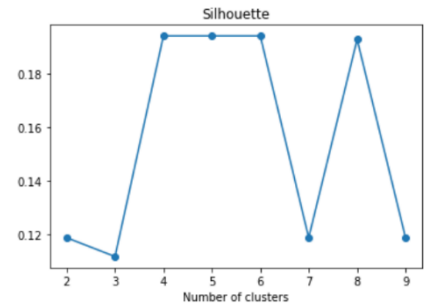
על מנת להשוות בין ה-K השונים נבחן את שני המדדים הבאים:

- **Davies-Bouldin** (מדד מינימום) - מדד זה משלב בין ההומוגניות וההפרדה הקיימת ב-dataset.
- **Silhouette** (מדד מקסימום) – בדומה לקודם, מדד זה משלב הומוגניות והפרדה. טווח הערכים יכול להתקבל בין 1:1-, בשאיפה להגיע ל-1 (הערך המקסימלי). כאשר:
  1: ניתן להבדיל בין ה-clusters בצורה טובה.

0: המרחק בין ה-clusters לא משמעותי וקשה להבחין ביניהם.

-1: ה-clusters מחולקים בצורה שגויה.





מספר המחלקות לחלוקה (נספח מדדי clustering)

במדד **David Bouldin**, ערך המדד הנמוך ביותר מתקבל ב-K=4 (בדומה לK=5,6) עם ערך מדד 1.8623. תופעת ה"ברך" לא מתקיימת עבור המודל, ועל כן התרומה השולית של כל cluster אינה פוחתת ככל שמתקדמים בערכי ה-K, להפך.

במדד **Silhouette** , ערך המדד הגבוה ביותר מתקבל ב-K=4 (בדומה ל-K=5,6,8) עם ערך מדד 0.194.

לאחר שבחנו את שני המדדים יחד, נסיק כי הבחירה האולטימטיבית היא חלוקת ה-data ל-**K=4** אשכולים.

השערה למשמעות ה-Clusters המתקבלים במודל המדויק ביותר:

הטבלה הבאה דוגמת שני samples מכל ארבעת ה-clusters:

Cluster	university+relevent_exp_0	is_working	last_new_job	company_type_0	company_size	experience	enrolled_university	relevent_experience	city_development_index	index
0	0	1	0	0	3	-0.626465386	2	1	-1.673724915	8
0	0	1	1	0	2	-0.774389863	0	1	-2.275962093	9
1	1	0	1	0	8	-0.626465386	2	0	0.735223799	14925
1	0	0	0	0	8	-1.070238816	0	0	0.784053841	14930
2	0	1	5	0	4	1.592401768	0	0	0.735223799	29
2	0	1	5	0	7	-0.034767478	0	0	0.735223799	39
3	0	1	3	0	3	-0.182691955	0	0	-0.509942259	64
3	0	1	4	0	3	-0.182691955	0	1	0.735223799	66

ניתן לראות שערכי ה-city_development_index המנורמלים קרובים מאוד עד זהים בכל אחד מהקלאסטרים, וזה מתכתב עם הבנתנו שפיצ'ר זה תורם משמעותית להבנת המודל. ב-cluster 3 קיים פער בין שני העובדים, אך הדימיון ביניהם נובע בשל שנות הניסיון שלהם, גודל החברה וסוגה.

Evaluation

נבצע השוואה לשלושת המודלים במשימת הלימוד ה-Supervised. המטריקה לפיה תתקבל ההחלטה היא המטריקה איתה אנו עובדים - ROC-AUC. בנוסף, רצינו להעשיר את ידיעותינו ולהשתמש במדדים נוספים שמתאימים למשימת הלימוד ול-Dataset שאינו מאוזן:

$T_i - Total\ true\ calssifications\ of\ class\ i$

$F_i - Total\ false\ calssifications\ of\ class\ i$

$N_i - Total\ samples\ in\ dataset\ of\ class\ i$

מדד Precision – מחשב את הדיוק עבור ה-Class הקטן. $percision = \frac{T_i}{T_i+F_i}$

מדד Recall – מספק אינדיקציה לתחזיות מה-Class הקטן שהוחמצו. $recall = \frac{T_i}{N_i}$

מקסום המדד הראשון ימזער את ה-FP, בעוד שמקסום המדד השני ימזער את ה-FN. האתגר הוא שיש בניהם Trade-Off. ולכן: **מדד F –** מדד שמשלב בין השניים, מתאים ל-Dataset שאינו מאוזן. $F = \frac{2precision*recall}{precision+recall}$.

המודלים DT ו-ANN מניבים תוצאות יחסית זהות במדדים שנבחנו. אף על פי כן, **נבחר במודל עץ ההחלטה**; ראשית, משום שהתוצאות שלו גבוהות יותר, דבר הנותן ביטחון במודל ובנכונותו. שנית, אף על פי שהפער בין תוצאות המדדים בין שני המודלים הוא זעיר, יכולת ההסברה בדיעבד של מודל עץ ההחלטה הינו יתרון משמעותי ושיקול מרכזי בבחירתו. נספח

מודל	ROC-AUC	F
DT	0.7960	0.5841
ANN	0.7862	0.5314
SVM	0.7663	0.3854

Improvments

1. **שיפור במודל:** המודל הנבחר הוא Decision Tree, עם זאת עץ יחיד עלול לא למצות את ה-Dataset שלנו. על כן, נציע שימוש באלגוריתם **Random Forest** אשר משתמש במספר גדול של עצי החלטה על מנת לבצע סיווג מדויק יותר. כל עץ מאומן רק על חלק מה-Samples (מקטין את ההטיה במודל) בעזרת תת קבוצה של ה-Features. לאחר שכלל העצים אומנו, נזריק את ה-Samples אל תוך כל אחד מהעצים הנ"ל, כאשר הסיווג הסופי יקבע עפ"י החלטת רוב העצים. על מנת לכוונן את המודל לקונפיגורציה המיטבית, נשתמש גם כאן ב- Grid Search CV ונעשה זאת באיטרציות (נכוונן את העומק המקסימלי של העץ, לאחר מכן את קריטריון ההחלטה ואת האלפא לקיטום, ולבסוף את מספר העצים ביער). אנו סבורות שאלגוריתם זה ישפר את אחוזי הדיוק שלנו (כמובן על פי מטריקת ROC-AUC).

   <u>עבור מודל הלמידה הטוב ביותר התקבלו הפרמטרים: (נספח טבלת קריטריונים)</u>

   עומק מקסימלי – 8, **קריטריון החלטה –** entropy, **אלפא –** 0, **מספר עצים ביער –** 60.

   <u>אחוזי הדיוק במודל המדויק ביותר:</u> **סט האימון:** 83.8938%, **סט הווליידציה:** 79.9613%.

2. **שיפור בנתונים:** ה-Dataset שברשותנו איננו מאוזן, דבר היוצר אתגר בסיווג Samples מה-Class המינורי. זאת משום שיש מעט מידי דוגמאות מ-Class שהמודל יוכל ללמוד מהן בצורה יעילה. על כן, ביצענו **Oversample** של ה-Class הקטן על ידי סנתוז של Samples חדשים בשיטת **SMOTE**. השיטה בוחרת Sample באופן רנדומלי, מוצאת את K ה-Samples השכנים הדומים לו ביותר (**KNN**), מייצרת מישור המחבר בניהם ויוצרת Sample חדש שנמצא על מישור זה. לבחירת ה-K האופטימלי לפרוצדורה, בחנו ערכים בטווח (1,7) והשוונו את מדדי ה-ROC-AUC של מודל Random Forest על כל אחד מה-Datasets המאוזנים שיצרנו. לבסוף ה-K האופטימלי היה **K=5** עם מדד ROC-AUC=0.758. את ה-Oversample ביצענו כך שה-Class המינורי יהיה בגודל של **40%** מה-Class המז'ורי (4500 דוגמאות), וכן ביצענו **Downsample** רנדומלי ל-Class המז'ורי על מנת להקטין אותו להיות **50%** יותר מה-Class המינורי.

   לבסוף, ביצענו תהליך של הייפטר פרמטרים למציאת הקונפיגורציה האופטימלית במודל Random Forest עבור ה-Dataset המאוזן. לצערנו, השיפור היה זניח ועל כן נשארנו עם המודל מסעיף 1. נספח

נספחים

עץ החלטה – כוונון פרמטרים

סט הולידציה:

Index	params	an_test_sc
3992	{'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 370, 'splitter': 'best'}	0.796018
4006	{'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 440, 'splitter': 'best'}	0.796014
4002	{'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 420, 'splitter': 'best'}	0.796009
4004	{'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 430, 'splitter': 'best'}	0.796008
3994	{'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 380, 'splitter': 'best'}	0.795979
4008	{'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 450, 'splitter': 'best'}	0.795974
660	{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 370, 'splitter': 'best'}	0.795935
670	{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 420, 'splitter': 'best'}	0.795895
674	{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 440, 'splitter': 'best'}	0.795885
672	{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 430, 'splitter': 'best'}	0.79587
968	{'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 440, 'splitter': 'best'}	0.795863
662	{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 380, 'splitter': 'best'}	0.795851
664	{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 390, 'splitter': 'best'}	0.795851
3996	{'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 390, 'splitter': 'best'}	0.795846
3990	{'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 360, 'splitter': 'best'}	0.795817
676	{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 450, 'splitter': 'best'}	0.7958
966	{'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 430, 'splitter': 'best'}	0.795797
668	{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 410, 'splitter': 'best'}	0.795789
666	{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 400, 'splitter': 'best'}	0.795766
658	{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 360, 'splitter': 'best'}	0.795762
970	{'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 450, 'splitter': 'best'}	0.795752

סט האימון:

Index	params	an_train_sc
3136	{'criterion': 'entropy', 'max_depth': 33, 'min_samples_split': 10, 'splitter': 'best'}	0.938175
3234	{'criterion': 'entropy', 'max_depth': 34, 'min_samples_split': 10, 'splitter': 'best'}	0.938169
3038	{'criterion': 'entropy', 'max_depth': 32, 'min_samples_split': 10, 'splitter': 'best'}	0.938139
2842	{'criterion': 'entropy', 'max_depth': 30, 'min_samples_split': 10, 'splitter': 'best'}	0.938121
2744	{'criterion': 'entropy', 'max_depth': 29, 'min_samples_split': 10, 'splitter': 'best'}	0.938116
2940	{'criterion': 'entropy', 'max_depth': 31, 'min_samples_split': 10, 'splitter': 'best'}	0.938113
2646	{'criterion': 'entropy', 'max_depth': 28, 'min_samples_split': 10, 'splitter': 'best'}	0.937986
2548	{'criterion': 'entropy', 'max_depth': 27, 'min_samples_split': 10, 'splitter': 'best'}	0.937756
2450	{'criterion': 'entropy', 'max_depth': 26, 'min_samples_split': 10, 'splitter': 'best'}	0.937462
2352	{'criterion': 'entropy', 'max_depth': 25, 'min_samples_split': 10, 'splitter': 'best'}	0.937087
2254	{'criterion': 'entropy', 'max_depth': 24, 'min_samples_split': 10, 'splitter': 'best'}	0.936334
2156	{'criterion': 'entropy', 'max_depth': 23, 'min_samples_split': 10, 'splitter': 'best'}	0.935236
6272	{'criterion': 'gini', 'max_depth': 31, 'min_samples_split': 10, 'splitter': 'best'}	0.933833
6468	{'criterion': 'gini', 'max_depth': 33, 'min_samples_split': 10, 'splitter': 'best'}	0.933832
6566	{'criterion': 'gini', 'max_depth': 34, 'min_samples_split': 10, 'splitter': 'best'}	0.933831
6370	{'criterion': 'gini', 'max_depth': 32, 'min_samples_split': 10, 'splitter': 'best'}	0.933828
6174	{'criterion': 'gini', 'max_depth': 30, 'min_samples_split': 10, 'splitter': 'best'}	0.933808
6076	{'criterion': 'gini', 'max_depth': 29, 'min_samples_split': 10, 'splitter': 'best'}	0.933786
5978	{'criterion': 'gini', 'max_depth': 28, 'min_samples_split': 10, 'splitter': 'best'}	0.933728
5880	{'criterion': 'gini', 'max_depth': 27, 'min_samples_split': 10, 'splitter': 'best'}	0.933688
5782	{'criterion': 'gini', 'max_depth': 26, 'min_samples_split': 10, 'splitter': 'best'}	0.933629
2058	{'criterion': 'entropy', 'max_depth': 22, 'min_samples_split': 10, 'splitter': 'best'}	0.93355
5684	{'criterion': 'gini', 'max_depth': 25, 'min_samples_split': 10, 'splitter': 'best'}	0.933386
	{'criterion': 'gini', 'max_depth': 24,	

ANN

מודל דיפולטיבי – אחוזי דיוק

```
In [148]: print(ann_avg_train)
Train Best ROC-AUC    0.678557
dtype: float64

In [149]: print(ann_avg_valid)
Validation Best ROC-AUC    0.661767
dtype: float64
```

ANN - כוונון פרמטרים

```
In [166]: print('The best parameters are:', grid_search.best_params_)
The best parameters are: {'activation': 'logistic', 'hidden_layer_sizes':
(16, 32), 'learning_rate_init': 0.011, 'max_iter': 500}
```

SVM – מודל דיפולטיבי – אחוזי דיוק

```
In [223]: print(avg_roc_auc_train_svm)
Train ROC-AUC    0.5
dtype: float64

In [224]: print(avg_roc_auc_val_svm)
Validation ROC-AUC    0.5
dtype: float64

In [225]: print('The coef is:')
The coef is:

In [226]: print(model_svm.coef_)
[[-3.37762550e-04  1.49592368e-04 -3.81153832e-05 -8.69672372e-05
   1.64437024e-05 -1.31431428e-04  8.24037402e-05  1.49431835e-05
  -5.76042024e-04  1.33261417e-04 -1.47829887e-05]]

In [227]: print('The intercept is:')
The intercept is:

In [228]: print(model_svm.intercept_)
[-0.99983328]
```

SVM – כוונון פרמטרים

```
In [236]: print('The best parameters are:' ,grid_search.best_params_)
The best parameters are: {'C': 100, 'decision_function_shape': 'ovr',
'gamma': 'scale', 'kernel': 'linear'}
```

סט הולידציה:

Index	params	mean test scor
12	{'C': 100, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.766353
13	{'C': 100, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.766353
8	{'C': 3, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.766252
9	{'C': 3, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.766252
10	{'C': 10, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.766217
11	{'C': 10, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.766217
6	{'C': 1, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.764555
7	{'C': 1, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.764555
4	{'C': 0.5, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.680915
5	{'C': 0.5, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.680915
2	{'C': 0.1, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.556024
3	{'C': 0.1, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.556024
0	{'C': 0.01, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.543186
1	{'C': 0.01, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.543186

סט האימון:

Index	params	mean train scor
12	{'C': 100, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.767885
13	{'C': 100, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.767885
10	{'C': 10, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.767684
11	{'C': 10, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.767684
8	{'C': 3, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.767657
9	{'C': 3, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.767657
6	{'C': 1, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.765862
7	{'C': 1, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.765862
4	{'C': 0.5, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.681309
5	{'C': 0.5, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.681309
2	{'C': 0.1, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.55335
3	{'C': 0.1, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.55335
0	{'C': 0.01, 'decision_function_shape': 'ovr', 'gamma': 'scale', 'kernel': 'linear'}	0.54273
1	{'C': 0.01, 'decision_function_shape': 'ovr', 'gamma': 'auto', 'kernel': 'linear'}	0.54273

K-Medoids - מדד David Bouldin

```
For k= 2 , Davies-Bouldin score is: 2.0362995673492295
For k= 3 , Davies-Bouldin score is: 3.1128772227693298
For k= 4 , Davies-Bouldin score is: 1.8623000146258661
For k= 5 , Davies-Bouldin score is: 1.8623000146258661
For k= 6 , Davies-Bouldin score is: 1.8623000146258661
For k= 7 , Davies-Bouldin score is: 2.0362995673492295
For k= 8 , Davies-Bouldin score is: 1.8741177864539353
For k= 9 , Davies-Bouldin score is: 2.0362995673492295
```

Silhouette מדד - K-Medoids

```
For k= 2 , Silhouette score is: 0.11892797934350076
For k= 3 , Silhouette score is: 0.11191642435340309
For k= 4 , Silhouette score is: 0.19405004179536905
For k= 5 , Silhouette score is: 0.19405004179536905
For k= 6 , Silhouette score is: 0.19405004179536905
For k= 7 , Silhouette score is: 0.11892797934350076
For k= 8 , Silhouette score is: 0.1927457598503673
For k= 9 , Silhouette score is: 0.11892797934350076
```

:DT – Evaluation

```
In [10]: evaluate_dt= cross_val_score(best_dt, X_train,
y_train, cv=k_folds, scoring='roc_auc')

In [11]: print(evaluate_dt.mean())
0.7960175649069359

In [12]: F1 = cross_val_score(best_dt, X_train, y_train,
cv=k_folds, scoring='f1')

In [13]: print (F1.mean())
0.5841338871770941
```

:ANN

```
roc auc mean 0.7862794247044025
f1 mean 0.5314160832297494
```

:SVM

```
In [6]: print(roc_auc_svm.mean())
0.7663664670246196
```

```
In [19]: sum = 0
    ...: for i in range (1,10):
    ...:     sum += (2*precision[i]*recall[i])/
(precision[i]+recall[i])
    ...: mean_f = sum/10

In [20]: print(mean_f)
0.38545037445142927
```

Random Forest - כוונון פרמטרים – עומק מקסימלי:

```
In [174]: print('The best parameters are:', grid_search.best_params_)
The best parameters are: {'max_depth': 8}
```

סט הולידציה:

Index	params	mean test scor
7	{'max_depth': 8}	0.798741
8	{'max_depth': 9}	0.7976
6	{'max_depth': 7}	0.79714
9	{'max_depth': 10}	0.796552
5	{'max_depth': 6}	0.795791
4	{'max_depth': 5}	0.794349
10	{'max_depth': 11}	0.793602
11	{'max_depth': 12}	0.790739
3	{'max_depth': 4}	0.789748
12	{'max_depth': 13}	0.785647
2	{'max_depth': 3}	0.784092
13	{'max_depth': 14}	0.780296
14	{'max_depth': 15}	0.775784
1	{'max_depth': 2}	0.772177
15	{'max_depth': 16}	0.770759

סט האימון:

Index	params	mean train scor
28	{'max_depth': 29}	0.976039
27	{'max_depth': 28}	0.976019
26	{'max_depth': 27}	0.97598
25	{'max_depth': 26}	0.975902
24	{'max_depth': 25}	0.975735
23	{'max_depth': 24}	0.975484
22	{'max_depth': 23}	0.975062
21	{'max_depth': 22}	0.974228
20	{'max_depth': 21}	0.973132
19	{'max_depth': 20}	0.971376
18	{'max_depth': 19}	0.968717
17	{'max_depth': 18}	0.965114
16	{'max_depth': 17}	0.960049
15	{'max_depth': 16}	0.953365
14	{'max_depth': 15}	0.944743
13	{'max_depth': 14}	0.933687

Random Forest - כוונון פרמטרים – קריטריון ואלפא:

```
In [192]: print('The best parameters are:' ,grid_search.best_params_)
The best parameters are: {'ccp_alpha': 0.0, 'criterion': 'entropy',
'max_depth': 8}
```

סט הולידציה:

Index	params	mean test scor
0	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8}	0.79916
1	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 8}	0.798741
2	{'ccp_alpha': 0.05, 'criterion': 'entropy', 'max_depth': 8}	0.765773
3	{'ccp_alpha': 0.05, 'criterion': 'gini', 'max_depth': 8}	0.661861
23	{'ccp_alpha': 0.55, 'criterion': 'gini', 'max_depth': 8}	0.5
24	{'ccp_alpha': 0.6000000000000001, 'criterion': 'entropy', 'max_depth': 8}	0.5
25	{'ccp_alpha': 0.6000000000000001, 'criterion': 'gini', 'max_depth': 8}	0.5
26	{'ccp_alpha': 0.65, 'criterion': 'entropy', 'max_depth': 8}	0.5

סט האימון:

Index	params	mean train scor
1	{'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 8}	0.839021
0	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8}	0.838618
2	{'ccp_alpha': 0.05, 'criterion': 'entropy', 'max_depth': 8}	0.765542
3	{'ccp_alpha': 0.05, 'criterion': 'gini', 'max_depth': 8}	0.661974
30	{'ccp_alpha': 0.75, 'criterion': 'entropy', 'max_depth': 8}	0.5
23	{'ccp_alpha': 0.55, 'criterion': 'gini', 'max_depth': 8}	0.5
24	{'ccp_alpha': 0.6000000000000001, 'criterion': 'entropy', 'max_depth': 8}	0.5
25	{'ccp_alpha': 0.6000000000000001, 'criterion': 'gini', 'max_depth': 8}	0.5

Random Forest - כוונון פרמטרים – מספר עצים ביער:

```
In [199]: print('The best parameters are:' ,grid_search.best_params_)
The best parameters are: {'ccp_alpha': 0.0, 'criterion': 'entropy',
'max_depth': 8, 'n_estimators': 60}
```

סט הולידציה:

Index	params	mean test scor
25	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 60}	0.799613
24	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 58}	0.799529
26	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 62}	0.799487
29	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 68}	0.799415
27	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 64}	0.799399
41	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 92}	0.799347
23	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 56}	0.799345
42	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 94}	0.799305
28	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 66}	0.799298
43	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 96}	0.799286
40	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 90}	0.799285
35	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 80}	0.799215
68	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 146}	0.799212
33	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 76}	0.79921
44	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 98}	0.799204
69	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 148}	0.799197
63	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 136}	0.799196
31	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 72}	0.799189
39	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 88}	0.799187

סט האימון:

Index	params	mean train scor
69	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 148}	0.838938
68	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 146}	0.838934
67	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 144}	0.838908
66	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 142}	0.838872
58	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 126}	0.838839
65	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 140}	0.838825
64	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 138}	0.838801
59	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 128}	0.838801
63	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 136}	0.838795
60	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 130}	0.838792
57	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 124}	0.83879
61	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 132}	0.838788
62	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 134}	0.838772
56	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 122}	0.838749
53	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 116}	0.838714
55	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 120}	0.838709
54	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 118}	0.838701
51	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 112}	0.838699
52	{'ccp_alpha': 0.0, 'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 114}	0.838698

<u>טיפול בדאטה לא מאוזן:</u>

יחס בין הקלאסים – על מנת לדעת מה היחס של האיזון צריך להיות:

```
In [33]: print(Counter(y_train_smote))
Counter({0: 11250, 1: 3752})
```

כוונון K – לבחירת K השכנים הדומים ביותר ולסנתוז דוגמאות חדשות בשיטת SMOTE:

```
> k=1, Mean ROC AUC: 0.756
> k=2, Mean ROC AUC: 0.755
> k=3, Mean ROC AUC: 0.752
> k=4, Mean ROC AUC: 0.755
> k=5, Mean ROC AUC: 0.758
> k=6, Mean ROC AUC: 0.754
> k=7, Mean ROC AUC: 0.754
```

Random Forest - לאחר כוונון פרמטרים במודל מאוזן:

סט הולידציה:

Index	params	an_test_sc
65	{'model__n_estimators': 140}	0.799353
60	{'model__n_estimators': 130}	0.799284
41	{'model__n_estimators': 92}	0.798907
45	{'model__n_estimators': 100}	0.798884
54	{'model__n_estimators': 118}	0.798849
64	{'model__n_estimators': 138}	0.798846
57	{'model__n_estimators': 124}	0.798804
12	{'model__n_estimators': 34}	0.798789
66	{'model__n_estimators': 142}	0.798782
22	{'model__n_estimators': 54}	0.798735
28	{'model__n_estimators': 66}	0.798727
15	{'model__n_estimators': 40}	0.798706
44	{'model__n_estimators': 98}	0.798596
55	{'model__n_estimators': 120}	0.798585
63	{'model__n_estimators': 136}	0.798564
30	{'model__n_estimators': 70}	0.798477
53	{'model__n_estimators': 116}	0.79842

סט האימון:

68	{'model__n_estimators': 146}	0.835388
60	{'model__n_estimators': 130}	0.835226
69	{'model__n_estimators': 148}	0.835219
66	{'model__n_estimators': 142}	0.835176
47	{'model__n_estimators': 104}	0.83509
64	{'model__n_estimators': 138}	0.835082
54	{'model__n_estimators': 118}	0.835066
65	{'model__n_estimators': 140}	0.835014
63	{'model__n_estimators': 136}	0.834938
58	{'model__n_estimators': 126}	0.834934
61	{'model__n_estimators': 132}	0.834933
43	{'model__n_estimators': 96}	0.834913
32	{'model__n_estimators': 74}	0.83491
53	{'model__n_estimators': 116}	0.834882
36	{'model__n_estimators': 82}	0.834864
62	{'model__n_estimators': 134}	0.83486
34	{'model__n_estimators': 78}	0.83484
40	{'model__n_estimators': 90}	0.834836