תרגיל בלימוד מכונה

חלק ב'

קבוצה 10

מגישים:

יונתן ניסים - 305719221

יואב גלבנד – 204097737

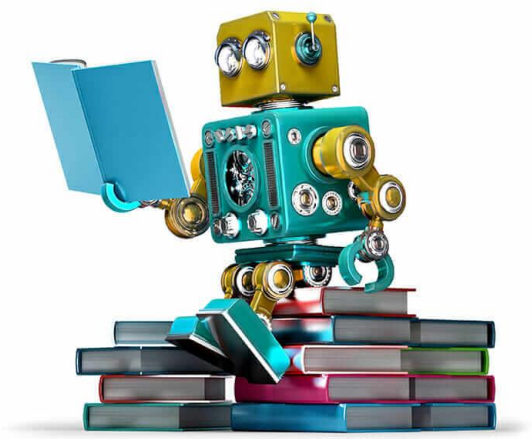
ברק לובנשטיין - 203625355

מתרגל הקורס:

בן חדד

תאריך הגשה:

23/06/2019



קבוצה 10

תוכן עניינים

0.הקדמה

חלק זה של הפרויקט מגיע לאחר חלק א' בו בוצעו הניתוחים וההבנות על הנתונים הגולמיים של הבעיה. חלק א' התמקד בהבנת הנתונים והבעיה הנחקרת, טיפול בנתונים חסרים ו-Feature engineering, תוך שימוש במטודולוגיית CRISP-DM. את חלק ב' נתחיל בהתאמת השינויים לסט הבחינה הסופי. בחלק זה נבצע ניתוחים שונים וננסה מספר מודלים מתחום לימוד המכונה על מנת לנסות ולמקסם את דיוק הניבוי בסט הבחינה הסופי. עבור כל מודל ניתן ערך דיוק מקורב ולבסוף נבחר במודל הטוב ביותר. לבסוף, נריץ מספר אלגוריתמים שונים שאינם כלולים בחומר הנלמד ונבחן גם אותם.

טיפול במאפיינים:

<u>קטגוריאלים</u>:עבור המשתנים הקטגוראליים נבצע ניסוי וטעייה. בתחילה , ננסה להריץ את המודל עם המשתנים איך שהם .לאחר מכן ננסה להפוך את המשתנים הקטגוריאלים למשתני דמה ונראה כיצד המודל מגיב לכך. עבור רשתות ועצים התקבלו מסקנות שונות.

<u>רציפים</u>: עבור כל המשתנים הרציפים ביצענו Min_Max_Scaling כאשר עבור כל מאפיין t נבצע :

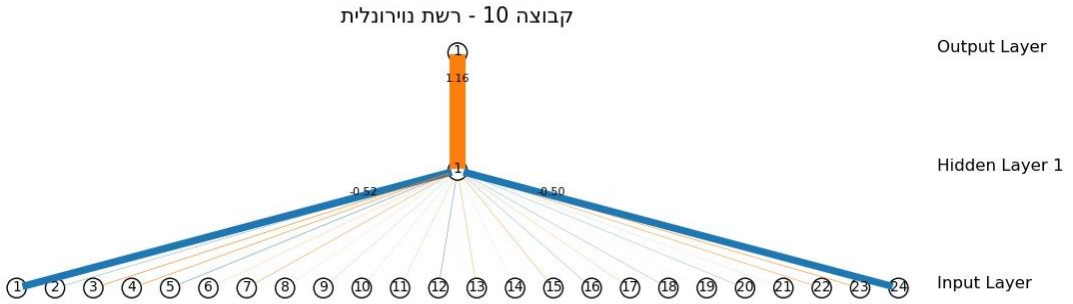
$$t_i = \frac{t_i - \bar{t}}{t_{Max} - t_{Min}}$$

<u>בינאריים:</u> עבור משתנים בינאריים ביצענו חלוקה של 0 ו-1 (גם כאן ניסינו מספר שיטות – לדוגמא 1,-1. לבסוף לקחנו את זו שהניבה דיוקים טובים יותר)

Neural Networks

1. הרצת הרשת בערכי ברירת מחדל

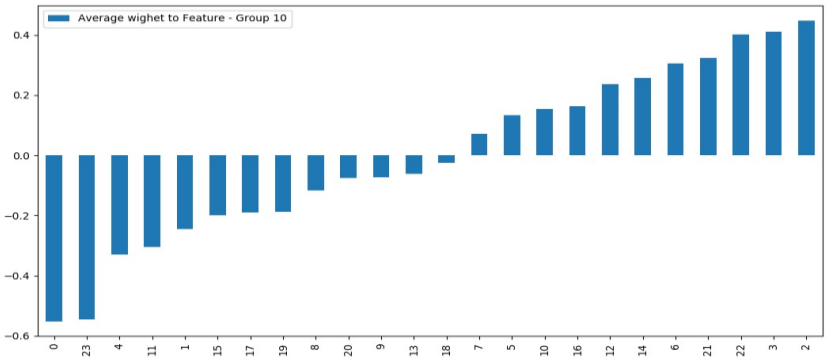
לפני ביצוע התמרה של משתנים קטגוראלים ביצענו הרצה של רשת פשוטה בעלת נוירון אחד בשכבה החבויה (בדומה לערכי ברירת המחדל בR) על מנת לראות את מידת השפעת נוירוני הכניסה. שאר הפרמטרים בוצעו על פי ברירת המחדל של Python.



ניתן לראות כי למשתנים type ו-PhotoAmt (1,24) יש השפעה חזקה יותר מלשאר נוירוני הכניסה בארכיטקטורה זו. נסייג ונאמר כי רשת זו היא פשוטה מאוד ואחוזי הדיוק בה הם:

the module train fit is: **0.5228**
the module test fit is: **0.5263**

קבוצה 10



כדי להבין את הנתונים טוב יותר נסתכל על שקלול המשקלים של כל אחד מהנוירונים בכניסה ברשת הפשוטה לנוירון בשכבה החבויה. כך לכל נוירון כניסה, משוקללת השפעתו על השכבה החבויה ובכך על הפלט.

- בתרשים זה ניתן לראות כי למשתנה 2, log(age) ,ישנה השפעה חיובית על סיווג מהירות האימוץ. בחלק א' של העבודה הוחלט לבצע התמרה לוגריתמית למשתנה age על סמך מבחנים שבוצעו. נראה כי ההתמרה אכן משפרת את איכות החיזוי של המשתנה.
- ניתן לראות כי למשתנה (0) , type ישנה השפעה שלילית חזקה על מהירות החיזוי. בחלק א' הוצג כי קיים הבדל גדול בין ההסתברות לאימוץ באותו היום בהינתן חתול=type לבין אימוץ באותו יום בהינתן כלב=type, כך שלחתול ישנה הסתברות גבוהה בהרבה להיות מאומץ באותו היום. אכן הגיוני כי ישנה השפעה שלילית בהתאם לממצא זה.
- בדומה לממצאים בהרצה עם ערכי ברירת המחדל בR ניתן לראות כי לPhotoAmt, משתנה 23, ישנה השפעה שלילית חזקה על מהירות האימוץ.

בשלב זה נבצע נרמול למשתנים ומעבר למשתנים קטגוראליים. נריץ את הרשת עם ערכי ברירת המחדל של Python. בהתאם ישנם 209 נוירונים בשכבת הכניסה, בשכבה החבויה ישנם 100 נוירונים ונוירון פלט אחד. כמו כן ישנו קצב למידה של 0.001 הנשאר קבוע (בניגוד לשיטות המשנות את קצב הלמידה לאורך האלגוריתם), כשמספר האיטרציות המקסימלי הוא 200.

פונקציית האקטיבציה במצב ברירת המחדל היא פונקציית $RELU$ הנראית כך:

$$f(x) = \mathrm{Max}(0, x)$$

בהתאם לכל הפרמטרים בברירת המחדל התקבלו אחוזי הדיוק הבאים:

```
the module train fit is: 0.868511081362
the module test fit is: 0.604578563996
```

קיים שיפור מרשת ברירת המחדל בR, אך עדיין קיים הבדל גדול בין הדיוק בסט האימון לבין סט הבחינה, מה שמצביע על סבירות לכך שיש Overfitting.

2. הרצת הרשת לאחר כוונון פרמפטרים

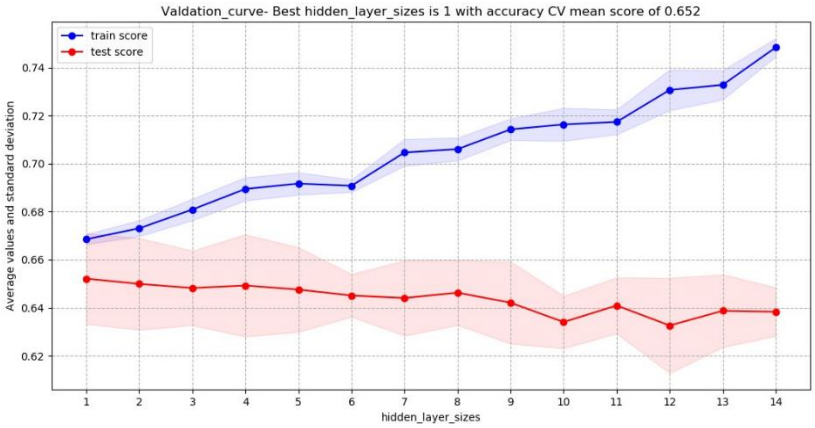
בוצע מבחן K folds cross validation על מספר הנוירונים בשכבה החבויה בכדי למצוא מה מספר הנוירונים המביא לדיוק מרבי. לאחר הרצה של רשתות בין 5 ל-50 שכבות חבויות עם קפיצות של 5 מצאנו כי הK המדויק ביותר הוא 5.

קבוצה 10

```
('Max accuracy for list', 0.6452311397994661)
('bestK:', 5)
```

כלומר הוספת נוירונים לשכבה החבויה ככל הנראה לא תסייע לנו כאשר מדובר במעל ל-5 נוירונים.

לכן החלטנו לבדוק רזולוציה גבוהה יותר ולהתמקד במבחן זה בקפיצות של 1 החל מנוירון יחיד בשכבה חבויה ועד 15 נוירונים בשכבה חבויה.



בגרף זה ניתן לראות בבירור כי בשכבה בעלת נוירון יחיד ישנו דיוק מקסימלי לסט המבחן. וגם כי ככל שנגדיל את מספר הנוירונים בשכבה כך תגדל בעיית ה Over fitting כתוצאה מירידה בדיוק סט הבחינה לעומת עלייה בדיוק סט האימון. לכן נבחר שמשתנה hidden_layer_size יהיה שווה ל-1.

<u>שינוי פונקציית אקטיבציה:</u> הפונקציה בברירת המחדל הינה פונקציית "relu", מהצורה $f(x) = \max(0, x)$.

הפונקציה הינה פונקציה פשוטה יחסית. בעבודה זו בוצע מיקוד בפונקציה שנלמדה בכיתה שהיא פונקציית סיגמואיד מהצורה:

$$f(x) = 1/(1 + \exp(-x))$$

לפונקציה זו פלט בין 0 ל-1 כך שהיא מדמה הסתברות בצורה טובה. בפונקציה זו כאשר אנו רחוקים מהפתרון הלמידה מהירה, ככל שנמצאים קרוב יותר לפתרון הלמידה נהיית איטית יותר. לאחר הרצה נוספת של 2 רשתות בשינוי פונקציית אקטיבציה מצורת $Relu$ לצורת סיגמואיד חל שיפור בדיוק

```
the module train for relu fit is: 0.664584262978
the module test  for relu fit is: 0.646201873049
the module train for sigmoid fit is: 0.668600327235
the module test fit sigmoid is: 0.65452653486
```

לכן נכוונן את משתנה פונקציית האקטיבציה לצורת סיגמואיד.

3. הרצת 2 רשתות נוספות עם קונפיגורציה שונה

<u>שינוי קצב למידה:</u> בוצעה הרצה של מספר רשתות בשינוי של משתנה Learning rate בלבד על מנת לבודד את השפעתו על איכות הדיוק. במידה וישנם מספר קצבי למידה אשר

מביאים לאותה רמה של דיוק, ייבחר קצב הלמידה המהיר מבין אלו המביאים לאותה תוצאה על מנת להגיע לדיוק הזהה בזמן מהיר יותר.

לאחר הרצה של 250 רשתות עם קצבי למידה שונים החל מ0.001 ועד 0.25 בקפיצות של 0.001 לקצב הלמידה נמצא כי קצב הלמידה המהיר ביותר אשר מביא לדיוק המקסימלי הוא 0.036 אשר מביא לדיוק של 0.6711

```
('the learning rate is:', 0.036000000000000004)
the module train fit is: 0.678566116317
the module test fit is: 0.671175858481
```

<u>שינוי מספר איטרציות:</u> בחלק זה בוצע שינוי של מספר איטרציות מקסימלי, בחלק זה ייבדק מספר האיטרציות אשר מביא למקסום הדיוק. יש ליצור איזון כך שהמודל לא יהיה מותאם מדי לסט אימון ולא למציאות. בחלק זה הורצו מספר איטרציות מקסימלי בין 100 ל1000 בקפיצות של 100.

נמצא כי הדיוק המקסימלי מתקבל לאחר העלאה ל 300 איטרציות, לאור תוצאה זו הורצו רשתות נוספות כאשר מספר איטרציות נע בין 200 ל400 בקפיצות של 10. כאן נמצא כי מספר האיטרציות המביא לדיוק המקסימלי הוא 350. קצב הלמידה בו נעשה שימוש בהרצות אלו הוא הקצב שנמצא כממקסם בבדיקת קצב הלמידה.

```
('the MaxIterations is:', 350)
the module train fit is: 0.67678119887
the module test fit is: 0.670135275754
```

<u>קונפיגורציה 1:</u> הורצה רשת עם 2 שכבות. שכבה ראשונה בעלת נוירון יחיד בשכבה החבויה ושכבה שניה בעלת שני נוירונים בשכבה החבויה. לא נצפה שיפור בדיוק סט הבחינה.

```
the module train for  fit is: 0.6697902722
the module test fit is: 0.660770031217
```

<u>קונפיגורציה 2:</u> הרצה של שתי שכבות חבויות כאשר ישנם 2 נוירונים ו-3 נוירונים בשכבה החבויה בהתאמה. ניתן לראטות כי אין שיפור בתוצאות סט הבחינה נסיק מכך שהוסת שכבות חבויות אינה תורמת לדיוק המודל.

```
the module train for  fit is: 0.684515841142
the module test fit is: 0.655567117586
```

Decision Trees

1.בניית עץ החלטה

ראשית נפנה לטיפול במאפיינים לטובת בניית העץ:

עבור המשתנים הקטגוריאליים ביצענו ניסוי וטעייה על מנת להבין כיצד לטפל במאפיינים. כאשר טיפלנו במשתנים הקטגוריאליים ראינו שהוספת Breed1 כמשתנה כקטגוריאלי מוריד את דיוק העצים בכ1-2(%), לכן החלטנו להשאיר משתנה זה כפי שהוא.

כאשר בונים עץ החלטה באמצעות חבילת rpart מתבצעת וולידציה של הנתונים באופן אוטומטי, כלומר האלגוריתם משתמש בCross-Validation ומחלק את סט הנתונים למספר

סטים ומתבצע אימות בין תתי הסט השונים. מאחר והאלגוריתם מבצע קרוס וולידציה , נרצה להשתמש בכל הנתונים כדי לאמן את המודל בצורה הטובה ביותר.

במקרה שלנו (שימוש בpython) לא קיימת חבילת הrpart ולכן נבצע שימוש ב-Cross Validation.

מהרצת עץ ברירת המחדל של פייטון אנו איננו מגבילים אף פרמטר ובפרט לא את העומק, נקבל את התוצאות הבאות עבור הממוצע של הCross-Validation:

Train score: 0.99633
Test score: 0.570146

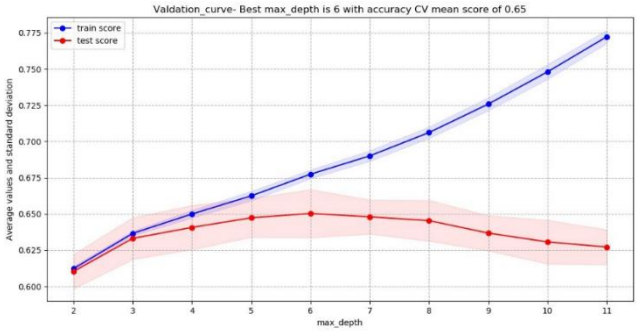
מצביע על התאמת יתר לנתוני האימון.

2.הרצת פקודת פרמטר הסיבוכיות

מבחינת מציאת פרמטר הסיבוכיות (cp) ופקודת ()printcp, פקודה זו לא קיימת בפייטון. פרמטר זה נועד על מנת למנוע מצב של התאמת יתר של המודל לסט האימון וקבלת תוצאות דיוק נמוכות בסט הבחינה. פקודת ה ()printcp יוצרת טבלה עם הערכים האפשריים השונים של cp ,גודל העץ שמקבל ודיוק המודל. מכיוון שמדד ה cp אינו מיושם בפייטון, אנו נתיחס לפרמטר Max_depth כאל פרמטר מקביל. פרמפטר העומק מצביע גם הוא במידה מסויימת על סיבוכיות המודל, ויכול להעיד על התאמת-יתר או חסר לנתוני האימון.כיוון שהעץ נבנה על בסיס סט האימון, ככל שהעץ מסועף ועמוק יותר כך ההתאמה לסט האימון גדולה יותר.

למציאת עומק העץ המיטבי בחרנו להשתמש בלולאה על עומק העץ כאשר בכל איטרציה נבצע cross validation ונקבל וקטור של תוצאות הדיוק על סט הבחינה, כאשר עומק העץ הנבחר הוא זה עם ממוצע הדיוק הטוב ביותר. גודל החלוקה יהיה 12.5% בחינה ו87.5% אימון.

לבחירת הפרמטרים הנוספים של עץ ההחלטה בחרנו בברירת המחדל של תוכנת הפייטון וחבילת sklearn. בבחירת פיצול העץ פייטון נותן כברירת מחדל את מדד ג'יני (פירוט על המדד בהמשך) ולא את מדד האנתרופיה כפי שנלמד בכיתה. בהמשך ננסה גם את מדד האנתרופיה.



קיבלנו כי עבור גובה עץ K=6 נקבל את הדיוק המירבי עבור סט הבחינה, דיוק של 0.650701. (ראה נספח).

ניתן לראות כי ככל שעומק העץ גדל, כך גם מידת התאמתו לסט האימון גדלה.

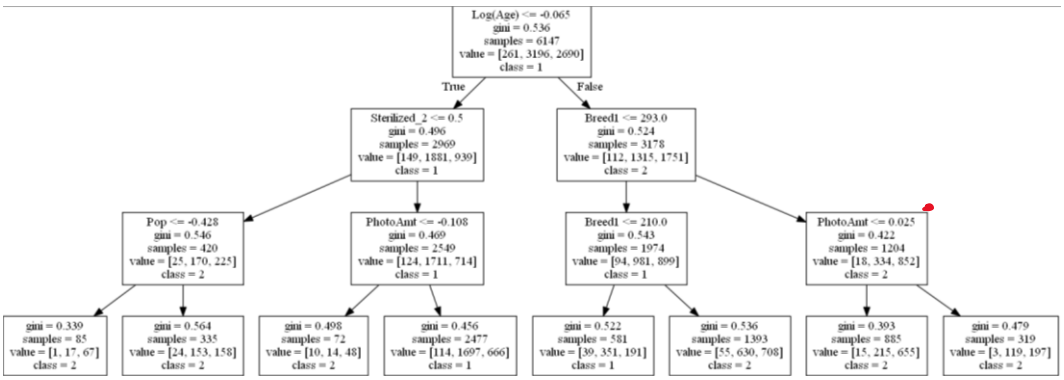
ככל שהעץ עמוק יותר האלגוריתם "משנן" יותר את סט האימון ,מדייק עליו טוב יותר, ולכן אנחנו נכנסים לOver-Fitting. על מנת לקבל הכללה ומודל פשוט יותר צריך לקטום את העץ. אך עם זאת, מודל פשוט מדי עשוי להביא דיוקים נמוכים. לכן, יש למצוא את עמק השווה. במקרה שלנו , המדד הוא דיוק על סט **הולידציה** ולכן נבחר בעומק עץ של Max_depth=6 כפי שניתן לראות בכותרת.

קבוצה 10

עבור סביבת עץ בעומק 6 נקבל את הנתונים הבאים:

Mean Score Test	Mean Score Train	Max-depth
0.640679	0.649940	**4**
0.647317	0.662471	**5**
0.650701	0.677400	**6**
0.648489	0.690042	**7**

כעת, נקטום את העץ עד אשר מתקבלת תמונה ברורה ונסביר על פיצול אחד לדוגמא,



העץ נבנה עבור 80% מסט הנתונים (אימון). עבור הפיצול לדוגמא שמסומן באדום קיימות 1204 רשומות העומדות בתנאים של ההסתעפויות הקודמות. כאשר הפילוג של ערך המטרה הוא [18,334,852] בהתאמה. מדד ג'יני: מחושב ע"י $I_G(p)=1-\sum_j p_j^2$. זהו מדד המראה על עד כמה טהורים הנתונים בהתפצלות הנתונה, כלומר עד כמה יש רוב למחלקה מסויימת. p_j^2 הוא ההסתברות לקחת רשומה ממחלקה j באקראי וגם לשייך אותה נכון.

עבור הרשומה הבאה נבצע את האלגוריתם דרך העץ. ניקח רק את המאפיינים הרלוונטים:

Pop	Sterilized_2	PhotoAmt	Breed1	Log(Age)	Y
0.307684	0	-0.025	307	**-0.288**	2

- Log (Age) <= -0.065 → True
- Sterilized_2 <= 0.5 → True
- Pop<=-0.428 → False

Class --------> **2**

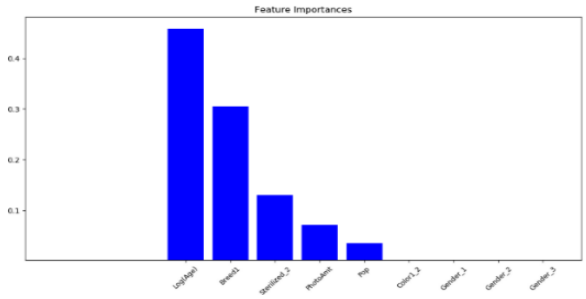
ניתן לראות כי העץ סיווג כראוי.

חשיבות המאפיינים:

מבחינת חשיבות המאפיינים , ככל שעולים בעץ חשיבות המאפיינים עולה. השאיפה היא בכל פיצול לפצל את הנתונים כך שנמזער את מדד ג'יני בפיצולים הבאים. **גיל** הכלב הוא המאפיין המשפיע ביותר (ראינו זאת גם בחלק א') . אחריו ניתן לראות כי **זן הכלב והאם מחוסן** בעלי השפעה רבה גם הם. בנוסף אליהם , **כמות האוכולוסיה במחוז** ו**כמות התמונות** משפיעים

קבוצה 10

גם הם אך במידה קטנה יותר.



Feature Importance

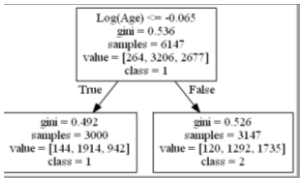
כיוון ופונקציית ה Feature Importance מבוססת על מדד ג'יני, המאפיינים היחידים שמקבלים ערך כלשהו במודל זה, הם אלו המופיעים בעץ (כל שאר המאפיינים הלא מופיעים קיבלו את ערך 0). הגרף מראה את מה שראינו בעץ – ככל שהמאפיין גבוה יותר בעץ, חשיבותו במודל גדולה יותר.

עבור **השוואה בין 2 מאפיינים** ניקח לדוגמא את *Pop* ואת *Log(Age)*. הגיל הוא מאפיין משמעותי ודבר זה מתיישב עם ההגיון. סביר שלאנשים תהיה נטייה לאמץ כלבים צעירים יותר\גורים. כלבים מבוגרים לרוב מגיעים עם אופי ולבעלים חדשים קשה להתחבר אליהם. עבור כמות האוכלוסיה במחוז(*Pop*) – משתנה זה נותן משמעות אורדינלית למשתנה המחוז. בדרך כלל האינטואיצה אומרת כי אם אוכלוסיית המחוז גדולה אז סביר שנראה משם יותר תצפיות אבל לא בהכרח אומר כי מהירות האמוץ גדולה יותר. עם זאת, עם כמות אוכלוסייה גדולה באה גם תחרות ולחץ מצד אנשים, ולכן ייתכן כי יש השפעה חיובית על מהירות האימוץ אך אנו משערים כי אינה חזקה.

ניתן לראות כיצד ההסתברויות המותנות משתנות בפיצול הראשון בגרף. (נספח4)

בחירת 3 מודלי עץ החלטה נוספים.

1. עבור cp=0 או במקרה שלנו Max_Depth מקסימלי , ניתן לראות כי בעומקי עץ של 36-37 הדיוק על הבחינה והאימון מתכנסים ונסיק מכך שזהו העומק המקביל ל cp=0. זהו גם הערך שהביא לשגיאה הגבוהה ביותר על סט הולידציה



2.כיוון שעץ עם cp=0 הביא לשגיאה הגבוהה על סט הולידציה נבחר עץ שונה. העץ שנבחר יהיה Max_Depth=1 כלומר עץ עם פיצול אחד. ראוי לציין כי עץ זה אינו יכול לסווג ל3 מחלקות אלא רק ל2 בכל מקרה. גרף של העץ המתקבל :

3.את העץ השלישי נבחר קצת שונה. כפי שנלמד בכיתה לעיתים מודלים פשוטים יותר מציגים טוב יותר את הבעיה. לכן נבחר עץ בעומק 4. בנוסף, המדד הנבחר יהיה מדד האנתרופיה בדומה לנלמד בכיתה. מטרת המדד למקסם את ה"Information Gain" בכל פיצול. בנוסף, עקב סט לא מאוזן, וכך שהאלגוריתם במחלק גדול מהמקרים לא משייך למחלקה 0, נכניס לעץ מילון של משקלים עבור כל מחלקה. המשקלים יהיו :{5,1,1} אל המחלקות בהתאמה.(ראוי לציין כי המשקלים אינם בהכרח מתאימים עבור הבעיה אלא רק כדי לראות כיצד המודל מתמודד עם שינוי משקלים, נפרט על כך בהמשך)

3. השוואת ארבעת המודלים

כעת נשווה בין המודלים, אחוזי הדיוק חושבו באמצעות Cross-Validation, מטריצת המבוכה חושבה על ולידציה בודדת.

קבוצה 10

מספר	עומק	מדד	דיוק על האימון	דיוק על הבחינה
1	37	Gini	0.996	0.573
2	1	Gini	0.595	0.595
3	4	Entropy	0.637	0.6258
4	6	Gini	0.677400	0.650701

הבעיה, כפי שהוצגה באתר Kaggle הינה לחזות עבור חיות המגיעות לאימוץ את מהירות האימוץ שלהם. לפיכך נבחר את המודל שיביא לדיוק הגבוה ביותר על סט הבחינה , כלומר את עץ 4.בהקשר לבעיה העסקית , ייתכן ובעלי האתר ירצו לעיתים "להחמיר" או "להקל" עם הסיווגים השונים ואז יהיה ניתן לתת משקלים שונים למחלקות השונות. הבעיה כפי שהוצגה, הייתה לתת את החיזויים הטובים ביותר,לכן אנו נבחר למקסם את דיוק הניבואים בלבד.

K-Means

1.מטרת הוצאת משתנה המטרה

במעבר לאלגוריתם K-Means אנו עוברים לשיטת למידה לא מונחית. הסיבה להוצאת משתנה המטרה היא כיוון שבלמידה לא מונחית אנו לא יודעים את ערכי פונקציית המטרה, והמטרה הכללית היא לאשכל את הנתונים, על מנת לחלק אותם לקבוצות דומות.

2.הרצת המודל עם ערכי ברירת המחדל

בבעיית הסיווג הנתונה קיימת חלוקה ל-3 מחלקות , אלגוריתם K-Means מקבל כקלט את מספר האשכולות ולכן נבחר בתור התחלה n_cluster=3.

ראשית נבחן את המודל לאחר שהפכנו את משתני הקטגוריה למשתני דמה בעלי ערכים 1,0 ונבצע למודל נרמול לנתונים, כלומר שכל ערכי הנתונים יהיו בטווח [-1,1]. בנוסף נבצע ניסוי וטעייה של המודל שאינה כוללת בתוכה את חלק מהמשתנים הקטגוריאליים.את השוואת המודלים נבצע באמצעות מספר מדדי דמיון:

Adjusted Rand Index - בהנתן וקטור החיזויים ווקטור התצפיות האמיתי מדד זה מבטא דמיון בין 2 הסטים בהתעלם מפרמוטציות האינדקסים. נע בין [-1,1]. מדד זה מתאים לאישכול 3 אשכולות בלבד , כיוון שהוא בודק את הדמיון לוקטור האמיתי המכיל 3 ערכים.
Inertia – בהנתן מרכזי האשכולות לכל אשכול, מדד זה סוכם את המחרק הריבועי הכולל של כל התצפיות אל המרכז המתאים להן. כיוון שמדד זה הוא סכום מרחקים ריבועים , נשאף למזער אותו.
Silhouette Score – בהנתן סט הנתונים וקבוצת התיוגים, מדד זה מאפשר להבין עד כמה דומות התצפיות בכל קבוצה. נע בין [-1,1].
אנו נמנע מלהגיע כלל לערכים שליליים במדדים אלו. ערכים שליליים מראים על בנייה לקויה של מודל ואי התאמה כלל לאשכול.

3. דיון בתוצאות

עבור הרצת המודל הפשוט עם K=3 נוציא את המדדים בהתאמה:

K	Adj_Rand_idx	Inertia	Silhouette Score	$\frac{Inertia}{silhouette}$
3	0.0065	47126.101933	0.093920	501771.154326

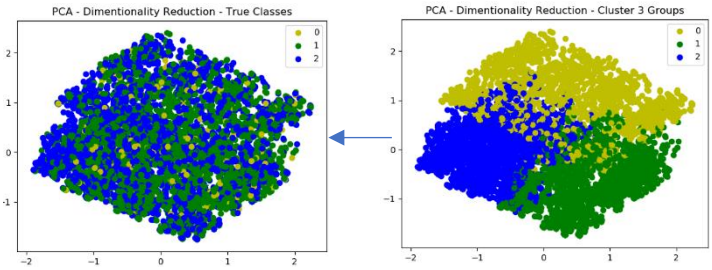
בבעיה אשכול השאיפה היא למזער את מדד ה Inertia ובמקביל למקסם את מדד הSilhouette. ולכן מדד מתבקש יהיה למזער את המנה Inertia/Silhouette.כעת נבדוק האם קיימת תאימות כלשהי לוקטור משתנה המטרה.

התוצאה האמיתית/מספר האשכול	0	1	2	סה"כ
0	90	121	117	**328**
1	1032	1533	1458	**4023**
2	972	948	1413	**3333**

על פי התפלגות המלחקות שהמאשכל הבסיסי סיווג, ניתן לראות כי אין הרבה קשר בין המחלקות והאשכולות גם לא בשינוי פרמוטציות. **תאורטית** במקרה זה , על מנת למצוא מינימום טעויות ,התיוג הכי טוב הוא אם כל התצפיות יתוייגו למחלקה 1 (מחלקת הרוב).

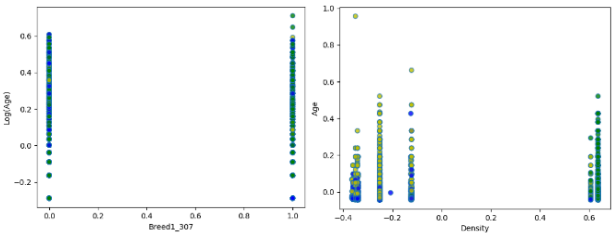
במקרה זה נקבל דיוק: $\frac{4023}{328+4023+3333}=\frac{4023}{7684}=0.5235$.

בנוסף, כדי לראות בצורה גרפית האם יש דמיון בין המחלקות לאשכולות ביצענו הורדת מימד באמצעות אלגוריתם PCA והטלנו את התצפיות על 2 מימדים. נראה את האשכולות אל מול המחלקות האמיתיות.



ניתן לראות כי אין דמיון רב בין השיוכים גם בהתעלם מהצבעים, וכן כי הבעיה קשה לאישכול.

בנקודה זו יש לציין כי האלגוריתם מאשכל את הנתונים לקבוצות עם מאפיינים דומים אך **לא בהכרח** לפי ערך פונ' המטרה.לכן חוללנו מעט גרפים כדי למצוא "Feature Similarity".בנוסף, בוצע ניתוח של מרחקי המרכזים של כל של כל אשכול והתרומה של כל Feature.



ניתן לראות כי האלגוריתם הפריד את בצורה די יפה את המחלקות באמצעות משתנה הדמה Breed307

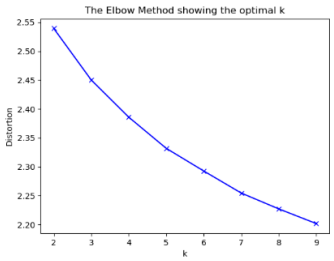
כפי שהיה ניתן לראות בחלק א' משתנה זה די מאוזן בסט הנתונים ולכן יש בכך הגיון. בנוסף ניתן לראות כי האלגוריתם הפריד בצורה חד משמעית את התצפיות כתלות בDensity של המחוז. דבר זה מתיישב עם ההגיון שכן משתנה זה גם מקבל ערכים קיצוניים. מאפיינים אלו, כפי שראינו בסעיף הקודם, לא משפיעים רבות על משתנה המטרה.

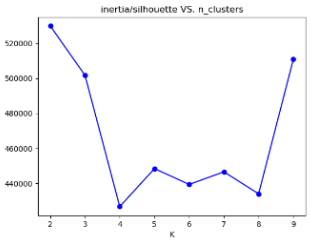
4. אימון שמונה מודלים שונים

לבחירת הK המיטבי נריץ מודלים שונים בטווח k של [2,9]. נשתמש ב- Elbow Method ובמדד Inertia\Silhoette. **שיטת המרפק** היא שיטה היוריסטית-גרפית. מטרתה למצוא את הנקודה(ות) בה הוספת של אשכול נוסף לא תוסיף רבות להורדת פונקציית המטרה.

קבוצה 10

פונ' המטרה במקרה זה תהיה וריאציה של סכום המרחקים הריבועיים – מדד Distortion. השיטה השנייה תהיה מדד **Inertia\Silhoette**, עליו פירטנו מקודם.





על פי גרף המרפק(הגרף השמאלי) קשה לראות נקודת שבירה משמעותית. לכן היוריסטיקה זו בעייתית כאן. בגרף הימני ניתן לראות את מדד המטרה אל מול הK נחפש את המקומות בהם פונקציית המטרה נמוכה. בהתאם , ניתן לראות כי K=4 הינו הנקודה ההנמוכה ביותר. נסיק מכך שהיא האופטימלית. (תוצאות מלאות בנספח)

Inertia\Silhoette = 426689.61

השוואה בין מודלים

כיוון שהלמידה הלא מונחית שונה מהלימדה המונחית בדרכה, עלינו למצוא דרך להשוות בין המודלים. את הלמידה המונחית בדקנו באמצעות דיוק על סט הבחינה. בחלק זה , נסביר את הדברך בה בחרנו כדי למצוא את הדיוק מהמודל הלא מונחה.

בסעיף הקודם ראינו כיK=4 הוא המודל הטוב ביותר עבור אישכול סט הנתונים. בשלב זה, נחלק את הסט לאימון ובחינה (20%-80%), נאשכל את הנתונים על סט האימון כפי שעשינו קודם.אחר כך נבחר לתייג את האשכול למחלקת הרוב כדי למעזר את הטעות של השיטה. מיד לאחריו נאשכל את סט הבחינה ונבדוק את הדיוק המתקבל מהתיוגים שבחרנו.

בתוצאות האימון ניתן לראות כי קבוצת הרוב בכל האשכולות היא 1. לפיכך על פי מה שהצגנו לעיל , יש לבצע אישכול לסט הבחינה ולסווג את כל התצפיות ל1. כדי להבין מהו הדיוק עלינו פשוט להבין מה פילוג ה-y בסט הבחינה:

אימון					
התוצאה האמיתית/מס' אשכול	0	1	2	3	סה"כ
0	68	67	36	96	**208**
1	759	924	709	822	**2547**
2	726	832	628	480	**2150**

כיוון שכל התצפיות שבאות מ0 ו2 יסווגו לטעות,

הדיוק יהיה: $\frac{809}{809+667+61}=0.5263$

בחינה			
0	1	2	סה"כ
61	809	667	**208**

לסיכום, כיוון שהמטרה הסופית היא למקסם את דיוק הניבואים, נמליץ על מודל הרשת הנוירונלית – המתמודד עם בעיית סיווג מונחית כפי שהוצגה, ומביא דיוקים גבוהים יותר.

מודל	ביצועים	מורכבות	יתרונות	חסרונות
רשתות נוירונים	בעל אחוזי דיוק גבוהים.	דורש מספר איטרציות גבוה, בעל סיבוכיות ריצה גבוהה.	-אחוזי דיוק גבוהים. -מתמודד עם נתונים חסרי התפלגות -מתמודד עם הפרדה לא לינארית של הנתונים. -אלגוריתם דינאמי. -מתאים לבעיות עם נתונים חסרי התפלגות.	מודל מורכב. -דורש ידע מקדים ליישום. -קשה לדעת מה קורה בשכבות החבויות. זמן ריצה גבוה ביחס לשאר האלגוריתמים.

קבוצה 10

עצי החלטה	בעל אחוזי דיוק בינוניים.	מודל פשוט בצורה יחסית.	-פשוט ולא דורש ידע מקדים נרחב. -מתמודד בצורה טובה עם משתנים פאקטוריאליים. - מסווג את המשתנים השונים לפי סוג ההשפעה.(ככל המשתנה יותר גבוה בעץ כך הוא יותר משפיע) -לא מושפע מקשרים לא ליניאריים בין משתנים	-למידה מונחת בלבד -חוסר בזיהוי של חריגים -מעודד התאמת יתר
K-means	בעל אחוזי דיוק נמוכים יחסית.	-מצריך עיבוד נתונים. -פשוט וקל להבנה.	-מאפשר הורדת מימד על מנת להציג את הבעיה בצורה גרפית. -מודל פשוט וקל להבנה.	-מאשכל לא בהכרח לפי משתנה המטרה. -קושי במציאת הK האופטימלי. -אחוזי דיוק נמוכים. -לא מובטח אישכול אופטימלי, תלוי בתנאי התחלה.

להלן סיכום התוצאות עבור המודלים אותם הרצנו:

אלגוריתם	קונפיגורציה	דיוק
רשת נוירונלית	שכבה חבויה אחת, נוירון אחד בשכבה החבויה, פונקציית אקטיבציה Sigmoid , Learning_Rate=0.036,Max_iter=350	0.6701%
עץ החלטה	עומק עץ 6 , פונקציית מטרה ג'יני	0.6507%
K-Means	3 אשכולות , מרחקים אוקלידים	0.5263%

ניתן להבחין בהבדל הרב בין הלמידה המונחית ללמידה הלא מונחית. באופן כללי כאשר ניתן לעשות זאת, נעדיף תמיד למידה מונחית על למידה לא מונחית

המודל הנבחר

המודל הנבחר בעינינו הוא זה המביא לדיוקים הגבוהים ביותר. מודל זה הוא הרשת הנוירונלית עם שכבה חבויה אחת עם נוירון אחד, פונקציית האקטיבציה היא Sigmoid ערך הלמידה 0.036 ומס' איטרציות מקסימלי של 350. מאפיינים אלו נבחרו על מנת למצוא מודל הלומד טוב יותר את הנתונים ומביא לדיוקים הטובים ביותר.

נבחן את תקינות התוצאות באמצעות מטריצת מבוכה, תצפיות אלו : $\begin{bmatrix} 0 & 34 & 10 \\ 0 & 374 & 127 \\ 0 & 149 & 267 \end{bmatrix}$. יש לציין כי תוצאה זו היא על מבחן בודד , להבדיל מבניית המודלים שבה השתמשנו ב K-folds Cross Validation. ניתן לראות כי הרשת סיווגה את התצפיות באופן מוטה כלומר לא סיווגה כלל למחלקה 0. דבר זה מתיישב עם העובדה שסט הנתונים מאופיין בחוזר איזון , עם מעט מאוד תצפיות ממחלקה 0.

סיכום תוצאות האלגוריתם על סט הבחינה :

Class	סך תצפיות על סט הבחינה	סווגו נכונה	דיוק
0	46	0	0.0
1	501	374	0.746
2	416	267	0.642

את הדיוק הכולל נחשב ע"י סכום האלכסון חלקי סכום המטריצה.התוצאה שתתקבל: 0.6656. בנוסף לאלגוריתמים המבוצעים כאן, ביצענו מספר אלגוריתמי סיווג על מנת לנסות ולקבל דיוקים טובים יותר . בינהם **KNN ,Random-Forest , Gradient-Boosting , XGBoost**. את הפירוט אודות התוצאות ניתן לראות ב(נספח).

תוצאות החיזויים מוגשות בקובץ CSV נפרד.

נספחים

נספח 1: תיעוד קוד – רשתות נוירונים

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
import matplotlib
import matplotlib.pyplot as plt
#Extraction the Data and seperate it to X and Y
EditedData = pd.read_csv('/Users/yoavgelband/Downloads/EditedDataa.csv')##Add the data here


# ##Scaling The DATA for All Orlinaly and Continueus Features
def ScalingMinMaxFeat(data,ListOfFeatures):
    for Featu in ListOfFeatures:
        data[Featu] = (data[Featu] - np.mean(data[Featu])) / (
                np.max(data[Featu]) - np.min(data[Featu]))

ListOfFeaturesToScale =
['Age','Log(Age)','Fee','Log(Fee)','Quantity','PhotoAmt','Pop','Density','GDPPP']
ScalingMinMaxFeat(EditedData,ListOfFeaturesToScale)

X = EditedData[EditedData.columns[:-1]]
Y = EditedData['y']



##Handeling Binary Features Putting 0,1
def EncodeBinaryFeatures(data,ListOfFeatures):
    lb_make = LabelEncoder()
    for Featu in ListOfFeatures:
        EditedData[Featu] = lb_make.fit_transform(EditedData[Featu])

ListOfFeaturesToBinary = ['Type']
EncodeBinaryFeatures(X,ListOfFeaturesToBinary)

X = pd.get_dummies(X,
columns=['Breed1','Gender','Color1','Color2','MaturitySize','FurLength','Vaccinated','Dewormed','Sterilize
d','Health','State'])

print(X.head())
```

קבוצה 10

```
EditedData['PhotoAmt'] = pd.cut(EditedData['PhotoAmt'], ( -1, 1, 5, 10 ,100), right=True,
                    labels=['0', '1', '2','3'])

EditedData.head()


from sklearn.model_selection import train_test_split
#
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.125, random_state=27)

mlp1score=0
learning_rate1=0.001
bestScore=0
bestRate=0
iters=0
BestMaxIter=0
iterNumArray=[]
iterIterArray=pd.DataFrame(columns=["max_iteration","score"])


##iterations check
for i in np.arange(200,400,100):
      mlp1= MLPClassifier(max_iter=i,activation="logistic",hidden_layer_sizes=1,
learning_rate_init=0.036)
      mlp1.fit(x_train, y_train)
      iters+=1
      if bestScore< mlp1.score(x_test, y_test):
        BestMaxIter=i
        bestScore=mlp1.score(x_test, y_test)
        iterIterArray = iterIterArray.append(i,mlp1.score(x_test, y_test))

      print("the MaxIterations is:",i)
      print ("the module train fit is: {}".format(mlp1.score(x_train, y_train)))
      print ("the module test fit is: {}".format(mlp1.score(x_test, y_test)))
      print("the best iteration number so far is:",BestMaxIter )
      print("the best score so far is:", bestScore)
      print("the iteration number is:", iters)



mlp1score=0
learning_rate1=0.001
bestScore=0
bestRate=0
iters=0
BestMaxIter=0
LearningRArray=[]
LearningIterArray=[]

##learning rate check
for i in np.arange(0.001,0.25,0.001):
      mlp1= MLPClassifier(learning_rate_init=i,activation="logistic",hidden_layer_sizes=1)
      mlp1.fit(x_train,y_train)
```

קבוצה 10

```
    iters+=1
    if bestScore<= mlp1.score(x_test, y_test):
      bestScore= mlp1.score(x_test, y_test)
      bestRate=i

    print("the learning rate is:",i)
    print ("the module train fit is: {}".format(mlp1.score(x_train, y_train)))
    print ("the module test fit is: {}".format(mlp1.score(x_test, y_test)))
    print("the best rate so far is:", bestRate)
    print("the best score so far is:", bestScore)
    print("the iteration number is:",iters)
```

נספח 2: קרוס ולדציה של עצים

Train Score	Test Score	Max-depth	
0.595393	0.595393	1.0	0
0.612256	0.610360	2.0	1
0.636592	0.633131	3.0	2
0.649940	0.640679	4.0	3
0.662471	0.647317	5.0	4
0.677400	0.650701	6.0	5
0.690042	0.648489	7.0	6
0.706068	0.644452	8.0	7
0.725850	0.638208	9.0	8
0.748029	0.631702	10.0	9
0.772291	0.625979	11.0	10
0.798505	0.615825	12.0	11
0.822860	0.614390	13.0	12
0.847791	0.603457	14.0	13
0.870343	0.599292	15.0	14
0.892969	0.594481	16.0	15
0.914367	0.585891	17.0	16
0.930969	0.581076	18.0	17
0.945917	0.582509	19.0	18
0.958503	0.581597	20.0	19
0.968171	0.575091	21.0	20
0.975385	0.574830	22.0	21
0.980720	0.570928	23.0	22
0.985536	0.571445	24.0	23
0.988529	0.574833	25.0	24
0.991299	0.571575	26.0	25
0.992954	0.575090	27.0	26
0.994274	0.571055	28.0	27
0.995110	0.572354	29.0	28
0.995687	0.576909	30.0	29

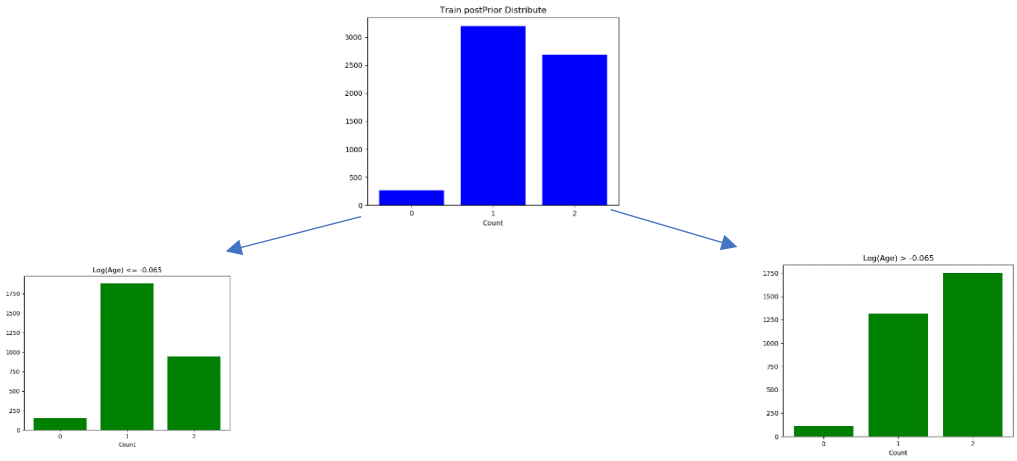
קבוצה 10

0.995873	0.573916	31.0	30
0.996021	0.574568	32.0	31
0.996170	0.573527	33.0	32
0.996245	0.574438	34.0	33
0.996319	0.573396	35.0	34
0.996337	0.573396	36.0	35
0.996337	0.573396	37.0	36
0.996337	0.573396	38.0	37
0.996337	0.573396	39.0	38
0.996337	0.573396	40.0	39
0.996337	0.573396	41.0	40
0.996337	0.573396	42.0	41
0.996337	0.573396	43.0	42
0.996337	0.573396	44.0	43
0.996337	0.573396	45.0	44
0.996337	0.573396	46.0	45
0.996337	0.573396	47.0	46
0.996337	0.573396	48.0	47
0.996337	0.573396	49.0	48

נספח 3: תוספת של מטריצת מבוכה

מספר	עומק	מדד	דיוק על האימון	דיוק על הבחינה	מטריצת מבוכה על בחינה בודדת
1	37	Gini	0.573	0.996	$\begin{pmatrix} 10 & 35 & 9 \\ 35 & 520 & 249 \\ 16 & 275 & 388 \end{pmatrix}$
2	1	Gini	0.595	0.595	$\begin{pmatrix} 0 & 35 & 22 \\ 0 & 479 & 328 \\ 0 & 223 & 450 \end{pmatrix}$
3	4	Entropy	0.637	0.6258	$\begin{pmatrix} 3 & 38 & 17 \\ 14 & 539 & 255 \\ 9 & 242 & 420 \end{pmatrix}$
4	6	Gini	0.677400	0.650701	$\begin{pmatrix} 2 & 44 & 15 \\ 0 & 618 & 185 \\ 0 & 297 & 376 \end{pmatrix}$

נספח 4: הסתברויות אפריוריות בעצים



נספח 5: הרחבה על K-means

	n_cluters	silhouette_avg	inertia	inertia/silhouette
0	**2.0**	**0.095584**	**50656.262752**	**529964.185804**
1	**3.0**	**0.093920**	**47126.101933**	**501771.154326**
2	**4.0**	**0.105070**	**44832.235346**	**426689.912294**
3	**5.0**	**0.095584**	**42859.997914**	**448399.946489**
4	**6.0**	**0.094138**	**41361.668755**	**439372.390687**

קבוצה 10

446562.078808	**40125.150920**	**0.089853**	**7.0**	**5**
433988.943280	**39313.697060**	**0.090587**	**8.0**	**6**
510917.258487	**38692.433493**	**0.075731**	**9.0**	**7**

נספח 6: מודלים מורחבים

Random-Forest

Mean_Score_Train	Mean_Score_Test	depth	n_est
0.664442	0.646667	5	**40**
0.681527	0.651349	6	**40**
0.699561	0.657076	7	**40**
0.722596	0.657597	8	**40**
0.74604	0.658897	9	**40**
0.771808	0.662936	10	**40**
0.801052	0.662806	11	**40**
0.829962	0.663064	12	**40**
0.858333	**0.669703**	13	**40**
0.883747	0.662676	14	**40**
0.906634	0.66801	15	**40**
0.923831	0.667877	16	**40**
0.940656	0.665798	17	**40**
0.955176	0.662287	18	**40**
0.965847	0.665017	19	**40**

Gradient-Boosting

Mean_Score_Train	Mean_Score_Test	depth	n_est
0.663661	0.658636	2	**40**
0.678293	0.662802	3	**40**
0.698297	0.665665	4	**40**
0.725552	0.666057	5	**40**
0.763888	0.66749	6	**40**
0.805868	0.662936	7	**40**
0.849725	0.664364	8	**40**
0.89044	0.664889	9	**40**

XGBoost

Mean_Score_Train	Mean_Score_Test	depth	n_est
0.675522	0.664627	2	**100**
0.688629	0.666447	3	**100**
0.707258	0.663977	4	**100**
0.730163	0.668012	5	**100**

קבוצה 10

0.758106	0.666579	6	**100**
0.790827	0.666711	7	**100**
0.824329	0.664627	8	**100**
0.858593	0.661636	9	**100**

KNN

Mean_Score_Train	Mean_Score_Test	n_neighbors
0.665575812	0.592788158	**10**
0.65179957	0.600595646	**15**
0.640179796	0.599815752	**20**
0.636945042	0.604633194	**25**
0.631293241	0.604112225	**30**
0.626478022	0.603460368	**35**
0.622164734	0.601770235	**40**
0.622443622	0.603201714	**45**