תרגיל בלימוד מכונה

חלק ב'

קבוצה 10

**מגישים:**

יונתן ניסים - 305719221

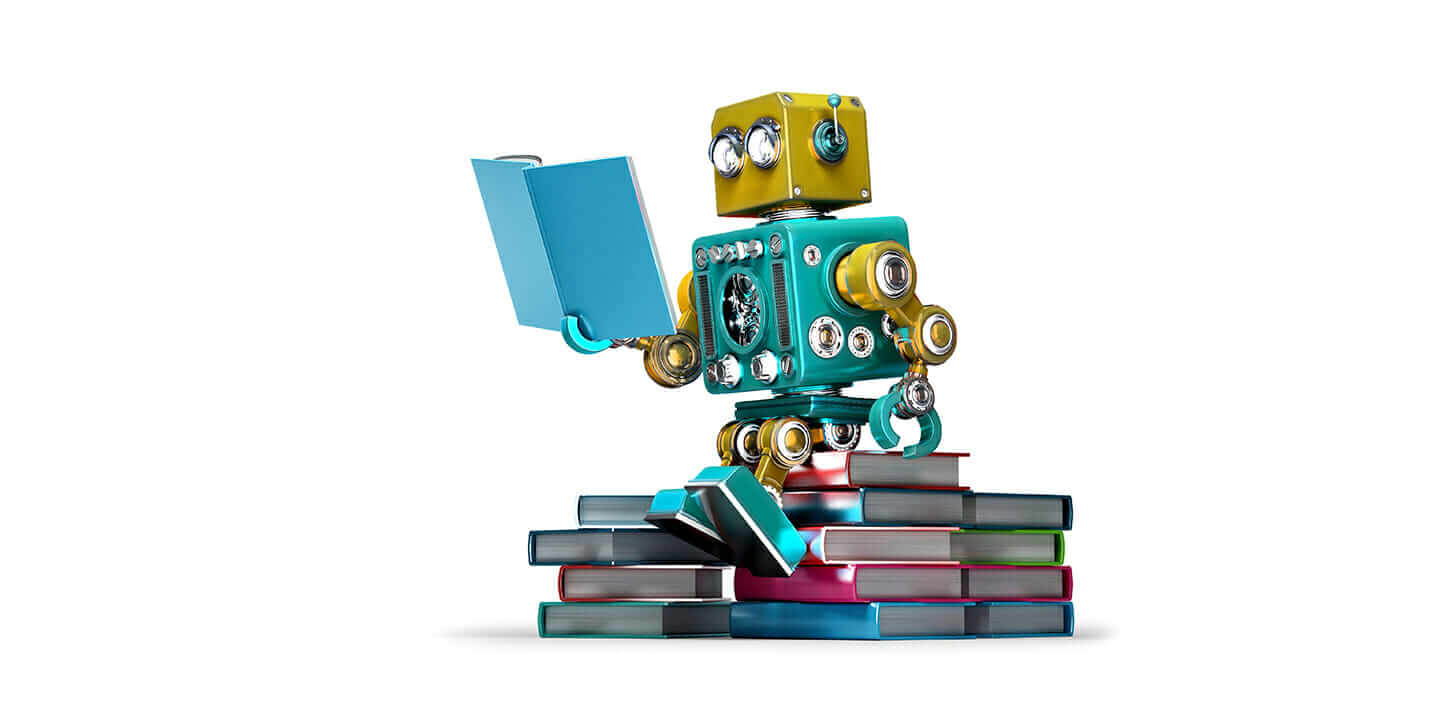
יואב גלבנד – 204097737

ברק לובנשטיין - 203625355

**מתרגל הקורס:**

בן חדד

**תאריך הגשה:**

23/06/2019

תוכן עניינים

[Neural Networks 3](#_Toc9408107)

[1. הרצת הרשת בערכי ברירת מחדל 3](#_Toc9408108)

[2. הרצת הרשת לאחר כוונון פרמפטרים 3](#_Toc9408109)

[3. הרצת 2 רשתות נוספות עם קונפיגורציה שונה 3](#_Toc9408110)

[Decision Trees 3](#_Toc9408111)

[1.בניית עץ החלטה 3](#_Toc9408112)

[2.הרצת פקודת פרמטר הסיבוכיות 3](#_Toc9408113)

[3. השוואת ארבעת המודלים 3](#_Toc9408114)

[K-Means 3](#_Toc9408115)

[1.אשכול למשתנה המטרה 3](#_Toc9408116)

[2.הרצת המודל עם ערכי ברירת המחדל 3](#_Toc9408117)

[3. דיון בתוצאות 3](#_Toc9408118)

[4. אימון שמונה מודלים שונים 3](#_Toc9408119)

[השוואה בין מודלים 3](#_Toc9408120)

# 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 engeneering , תוך שימוש במטודולוגיית CRISP-DM .את חלק ב' נתחיל בהתאמת השינויים לסט הבחינה שניתן לנו בהתאמה לשינויים מחלק א'. 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:**

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

רציפים: עבור כל המשתנים הרציפים ביצענו Min\_Max\_Scaling כאשר עבור כל מאפיין t נבצע :

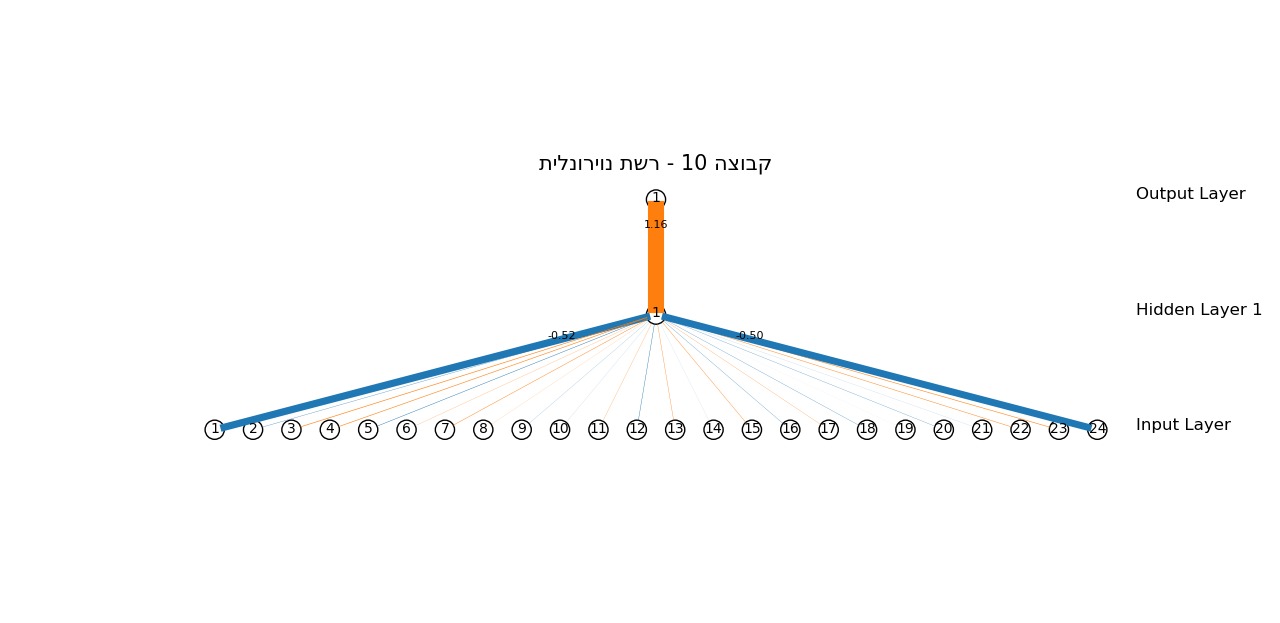
בינאריים: עבור משתנים בינאריים ביצענו חלוקה של 0 ו-1 (גם כאן ניסינו מספר שיטות – לדוגמא 1,-1)/

# Neural Networks

## הרצת הרשת בערכי ברירת מחדל

בעבודה זו בוצע שימוש בשפת תכנות Python ובאלגוריתם MLPCllassifier המשמש ליצירת רשת נוירונלית.

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.

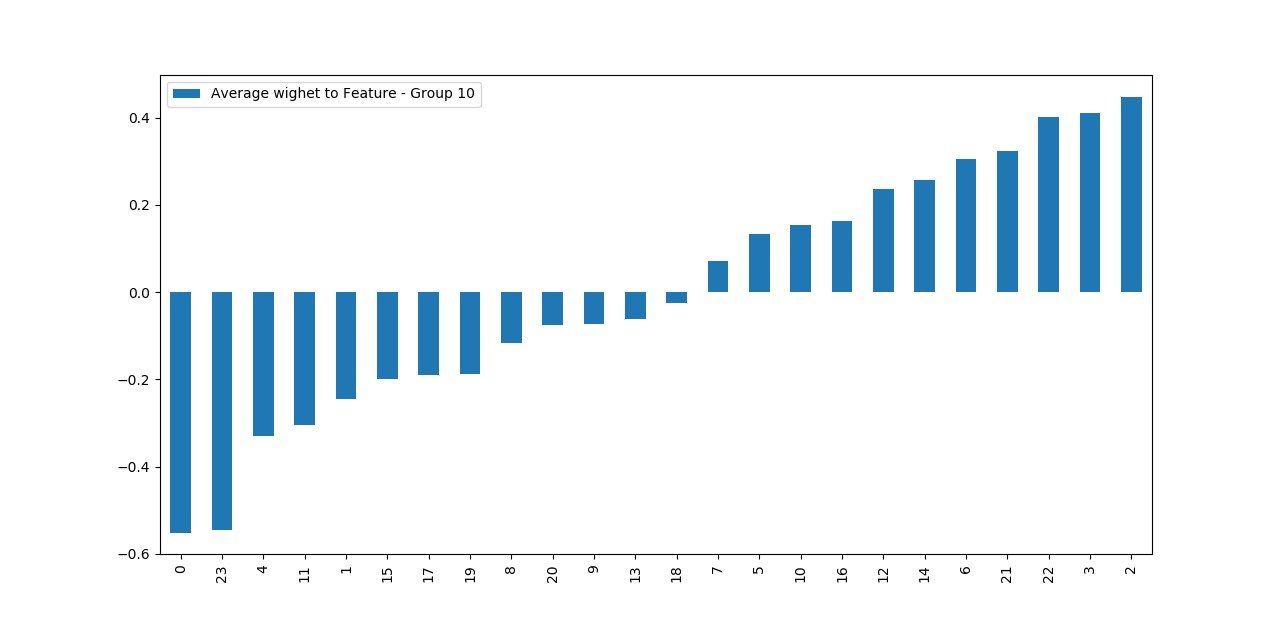


ניתן לראות כי למשתנים type ו-PhotoAmt (1,24) יש השפעה חזקה יותר מלשאר נוירוני הכניסה בארכיטקטורה זו. אמנם נסייג ונאמר כי רשת זו היא רשת פשוטה מאוד ואחוזי הדיוק בה הם:

the module train fit is: 0.5228566780543354

the module test fit is: 0.5263500325309044

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 חתולtype= לבין אימוץ באותו יום בהינתן כלבtype=, 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 ל,PhotoAmt משתנה ,23 ישנה השפעה שלילית חזקה על מהירות האימוץ.

בנוסף נבדקה רשת לפי ערכי ברירת המחדל בPython. בשלב זה בוצע נרמול למשתנים ומעבר למשתנים קטגוראליים. 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:

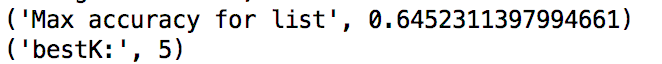
בהתאם לכל הפרמטרים בברירת המחדל התקבלו אחוזי הדיוק הבאים:

../Desktop/Screen%20Shot%202019-06-02%20at%2012.53.38.png

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.

## 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.



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.

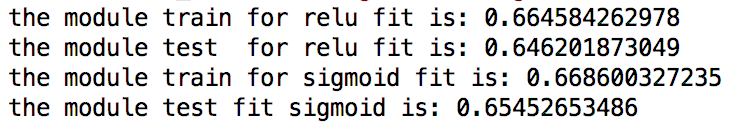


בגרף זה ניתן לראות בבירור כי בשכבה בעלת נוירון יחיד ישנו דיוק מקסימלי לסט המבחן. וגם כי ככל שנגדיל את מספר הנוירונים בשכבה כך תגדל בעיית הOver fitting כתוצאה מירידה בדיוק סט הבחינה לעומת עלייה בדיוק סט האימון. לכן נבחר שמשתנה hidden\_layer\_size יהיה שווה ל-1.

שינוי פונקציית אקטיבציה: הפונקציה בברירת המחדל הינה פונקציית "relu", מהצורה .

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

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

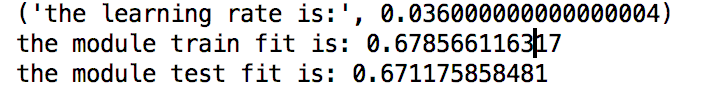


לכן נכוונן את משתנה פונקציית האקטיבציה לצורת סיגמואיד.

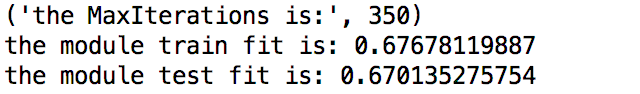
## 3. הרצת 2 רשתות נוספות עם קונפיגורציה שונה

שינוי קצב למידה: 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.

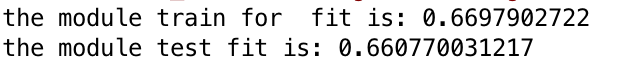
לאחר הרצה של 250 רשתות עם קצבי למידה שונים החל מ0.001 ועד 0.25 בקפיצות של 0.001 לקצב הלמידה נמצא כי קצב הלמידה המהיר ביותר אשר מביא לדיוק המקסימלי הוא 0.036 אשר מביא לדיוק של 0.6711

שינוי מספר איטרציות: 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.

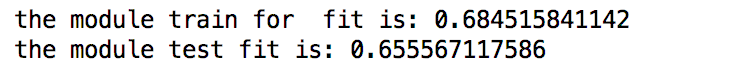
נמצא כי הדיוק המקסימלי מתקבל לאחר העלאה ל 300 איטרציות, לאור תוצאה זו הורצו רשתות נוספות כאשר מספר איטרציות נע בין 200 ל400 בקפיצות של 10. כאן נמצא כי מספר האיטרציות המביא לדיוק המקסימלי הוא 350. קצב הלמידה בו נעשה שימוש בהרצות אלו הוא הקצב שנמצא כממקסם בבדיקת קצב הלמידה.



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



קונפיגורציה 2: גם לאחר הרצה של שתי שכבות חבויות כאשר ישנם 2 נוירונים ו-3 נוירונים בשכבה החבויה בהתאמה אין שיפור בתוצאות סט הבחינה.



# Decision Trees

## 1.בניית עץ החלטה

ראשית נפנה לטיפול במאפניים 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 עד למציאת העץ הטוב ביותר. מאחר והאלגוריתם מבצע וולידציה , נרצה להשתמש בכל הנתונים כדי לאמן את המודל בצורה הטובה ביותר.

במקרה שלנו (שימוש בpython) לא קיימת חבילת ה rpartולכן נבצע שימוש בCross-Validation

מהרצת עץ ברירת 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

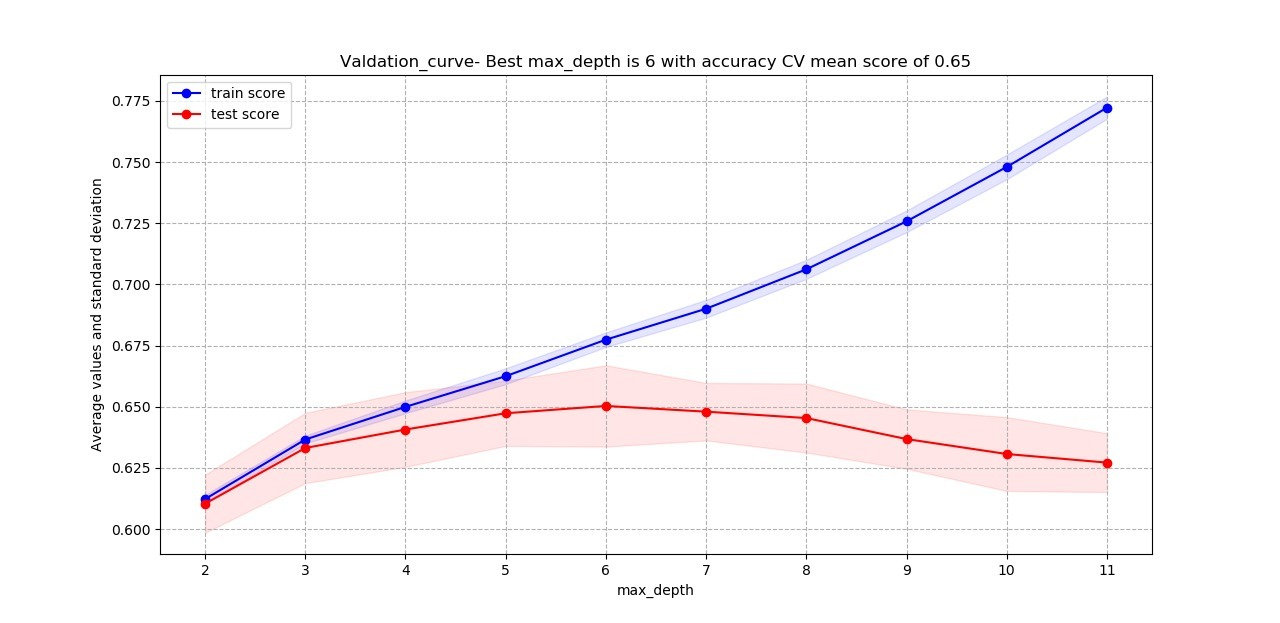
מצביע על התאמת יתר לנתוני האימון.

## 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 הcp אינו מיושם בפייטון, אנו נסתייח לפרמטר Max\_depth כאל פרמטר מקביל. פרמפטר העומק מצביא 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 האימון, 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).

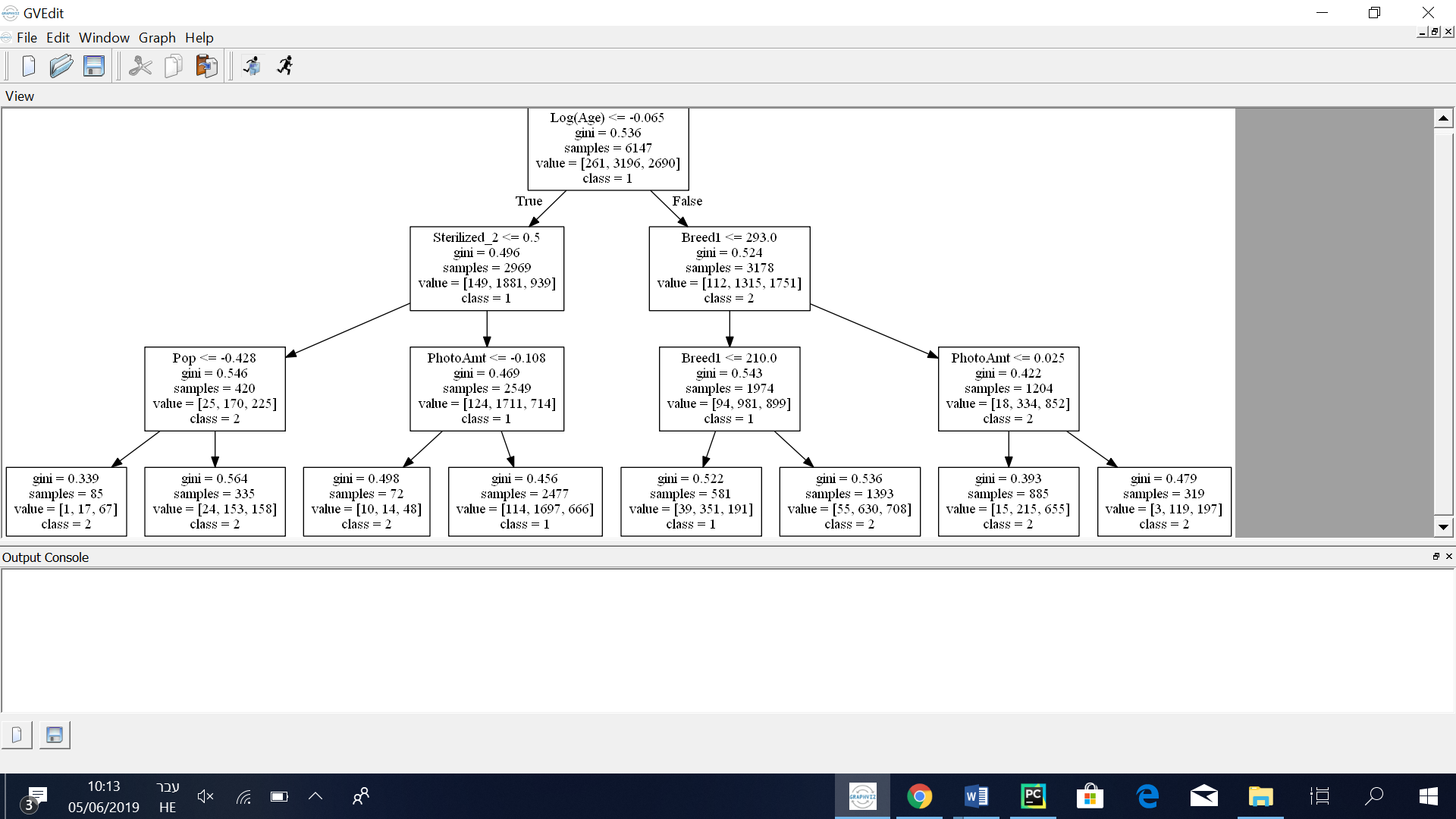
ניתן לראות כי ככל שעומק העץ גדל, כך גם מידת התאמתו לסט האימון גדלה.

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.

עבור סביבת עץ בעומק 6 נקבל את הנתונים הבאים:

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

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  . זהו מדד המראה על עד כמה טהורים הנתונים בהתפצלות הנתונה, כלומר עד כמה יש רוב למחלקה מסויימת. הוא ההסתברות לקחת רשומה ממחלקה *j* באקראי וגם לשייך אותה נכון.

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Y | Log(Age) | Breed1 | PhotoAmt | Sterilized\_2 | Pop |
| 2 | **-0.288** | 307 | 0.025- | 0 | 0.307684 |

* Log (Age) <= -0.065 🡪 True
* Sterilized\_2 <= 0.5 🡪 True
* Pop<=-0.428 🡪 False
* Class --------> 2

ניתן לראות כי העץ סיווג כראוי. את נתיב העץ ניתן לראות ב(נספח).

### חשיבות המאפיינים:

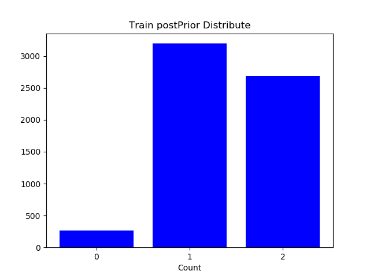
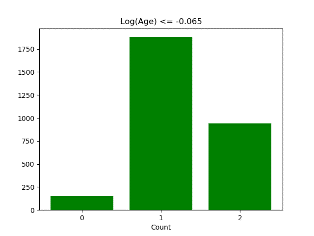
מבחינת חשיבות המאפיינים , ככל שעולים בעץ חשיבות המאפיינים עולה. השאיפה היא בכל פיצול לפצל את הנתונים כך שנמזער את מדד ג'יני בפיצולים הבאים. **גיל** הכלב הוא המאפיין המשפיע ביותר (ראינו זאת גם בחלק א') . אחריו ניתן לראות כי **זן הכלב** **והאם מחוסן** בעלי השפעה רבה גם הם. בנוסף אליהם , **כמות האוכולוסיה**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

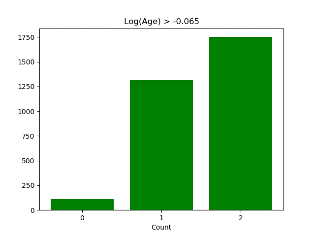
### Feature Importance

כיוון ופונקציית הFeature 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 וקשה ו לבעלים חדשים קשה להתחבר אליהם. עבור כמות האוכולסיה 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.

נראה לדוגמא כיצד נראות ההסתברויות האפריוריות המותנות משתנות בפיצול הראשון.





### 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.

## 3. השוואת ארבעת המודלים

# K-Means

## 1.אשכול למשתנה המטרה

במעבר לאלגוריתם K-Means אנו עוברים לשיטת למידה לא מונחית (Unsuoervised). בלמידה לא מונחית אנו לא יודעים את ערכי פונקציית המטרה, והמטרה הכללית היא לאשכל את הנתונים, על מנת לחלק אותם לקבוצות דומות.

הוצאת ערך פונקציית המטרה תגרום לנתונים לא להיות תלויים בערכים אלו והאשכול יתבצע על ידי הקבצת הרשומות בצורה הטובה ביותר ולאו דווקא הקבצה לפני מהירות אימוץ.

בבעיית הסיווג הנתונה קיימת חלוקה ל-3 מחלקות , מחלקה אחת כאשר בעל החיים אומץ ביום הרישום. נשתמש במודל ה K-means למציאת אישכול הקבוצות בצורה הטובה ביותר. נבחן את המודל תוך כדי טיפול בנתונים.

ראשית נבחן את המודל לאחר שהפכנו את משתני הקטגוריה למשתנה דמה בעלי ערכים 1,0, בהמשך נבחן את המודל לאחר ביצוע נרמול לנתונים, כלומר שכל ערכי הנתונים יהיו בטווח בין מינוס אחד לאחד. לבסוף נבחן תצורה של המודל שאינה כוללת בתוכה את המשתנים הקטגוריאליים.

את השוואת המודלים נבצע באמצעות מספר מדדי דמיון כפי שיפורט בהמשך.

## 2.הרצת המודל עם ערכי ברירת המחדל

בתחילת נריץ את מודל האשכול עם נתונים לא מנורמלים ונבחן מדדים רלוונטים.

Adjusted Rand Index - בהנתן וקטור החיזויים של המודל ווקטור התצפיות האמיתי מדד זה מבטא דמיון בין 2 הסטים בהתעלם מפרמוטציות האינדקסים. הערכים הינם בין [-1,1]

Adj\_Rand\_idx : 0.0104

Homogeneity- בהנתן וקטור החיזויים של המודל ווקטור התצפיות האמיתי מדד זה מבטא את ההומוגניות בתוך קבוצת השאכול. כלומר מדד לעד כמה כל אשכול מכיל קבוצה יחידה.

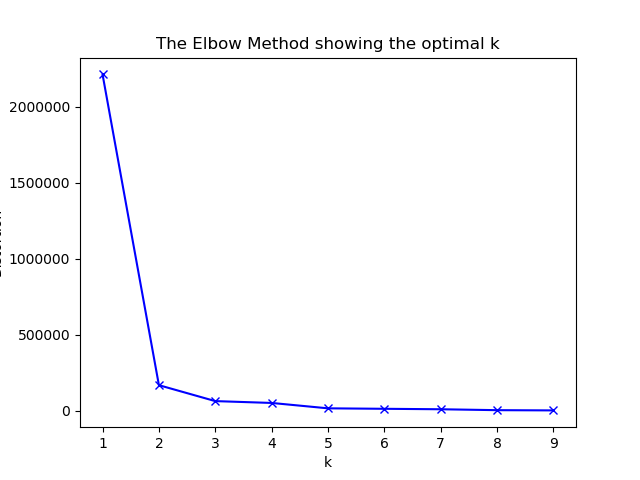
Homogenity : 0.0057

**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.

נריץ את אלגוריתם האשכול תחילה עם ערכי ברירת המחדל ונבדקוק האם קיימת תאימות כלשהי לוקטור משתנה המטרה.

KMNS = KMeans(n\_clusters=3,random\_state=123)  
KMNS.fit(X)

|  |  |  |  |
| --- | --- | --- | --- |
| התוצאה האמיתית/לאן סווג | 0 | 1 | 2 |
| **0** | 36 | 318 | 121 |
| **1** | 177 | 2646 | 2428 |
| **2** | 115 | 1059 | 784 |



## 3. דיון בתוצאות

## 4. אימון שמונה מודלים שונים

# השוואה בין מודלים

# נספחים

תיעוד קוד:

**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'**,**'Sterilized'**,**'Health'**,**'State'**])  
  
 **print**(X.head())  
  
 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)  
 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)

|  |
| --- |
| Train Score Test Score Max-depth |
| 0 1.0 0.595393 0.595393 |
| 1 2.0 0.610360 0.612256 |
| 2 3.0 0.633131 0.636592 |
| 3 4.0 0.640679 0.649940 |
| 4 5.0 0.647317 0.662471 |
| 5 6.0 0.650701 0.677400 |
| 6 7.0 0.648489 0.690042 |
| 7 8.0 0.644452 0.706068 |
| 8 9.0 0.638208 0.725850 |
| 9 10.0 0.631702 0.748029 |
| 10 11.0 0.625979 0.772291 |
| 11 12.0 0.615825 0.798505 |
| 12 13.0 0.614390 0.822860 |
| 13 14.0 0.603457 0.847791 |
| 14 15.0 0.599292 0.870343 |
| 15 16.0 0.594481 0.892969 |
| 16 17.0 0.585891 0.914367 |
| 17 18.0 0.581076 0.930969 |
| 18 19.0 0.582509 0.945917 |
| 19 20.0 0.581597 0.958503 |
| 20 21.0 0.575091 0.968171 |
| 21 22.0 0.574830 0.975385 |
| 22 23.0 0.570928 0.980720 |
| 23 24.0 0.571445 0.985536 |
| 24 25.0 0.574833 0.988529 |
| 25 26.0 0.571575 0.991299 |
| 26 27.0 0.575090 0.992954 |
| 27 28.0 0.571055 0.994274 |
| 28 29.0 0.572354 0.995110 |
| 29 30.0 0.576909 0.995687 |
| 30 31.0 0.573916 0.995873 |
| 31 32.0 0.574568 0.996021 |
| 32 33.0 0.573527 0.996170 |
| 33 34.0 0.574438 0.996245 |
| 34 35.0 0.573396 0.996319 |
| 35 36.0 0.573396 0.996337 |
| 36 37.0 0.573396 0.996337 |
| 37 38.0 0.573396 0.996337 |
| 38 39.0 0.573396 0.996337 |
| 39 40.0 0.573396 0.996337 |
| 40 41.0 0.573396 0.996337 |
| 41 42.0 0.573396 0.996337 |
| 42 43.0 0.573396 0.996337 |
| 43 44.0 0.573396 0.996337 |
| 44 45.0 0.573396 0.996337 |
| 45 46.0 0.573396 0.996337 |
| 46 47.0 0.573396 0.996337 |
| 47 48.0 0.573396 0.996337 |
| 48 49.0 0.573396 0.996337 |