Names: Dan

Alex

Introduction:

הפרוייקט שלנו מבוסס על CLI, אשר אותו אנו מפעילים דרך חלון CMD בסביבת Windows, אך הוא אינו מוגבל לסביבה זו בלבד, כאשר אנו מריצים את קובץ ההפעלה שלנו בחלון הCMD אנו קולטים את הארגומנטים אשר נשלחים בנוסף, ובכך יודעים להפעיל את הפעולות הרלוונטיות, הוספנו תמיכה לפקודת help אשר נותנת מידע למשתמש, כיצד לתפעל את הקוד שלנו.

ראשית כל, לאחר הפעלה, הקוד בודק האם נשלחו ארגומנטים נוספים, במילה וכן עובר לפונקציית Menu ובא הוא פותח קבצים במידה וניתן, אחרת זורק חריגה, לאחר מכן עושה עיבוד מקדים לקבצים, ולאחר מכן הוא שולח את הקבצים לסיווג תלוי לפי הפקודה, עבור כל אחד מהסיווגים חוץ משימוש במודל חיצוני, אנו נשלחים לפונקציה modelAndClassify אשר בונה את המודל, מריצה את הסיווג ומחזירה את המודל לשמירה במידה ונדרש. עבור מודל חיצוני אנו נשלחים לפונקציה outerModelClassify אשר יודעת לקחת את המודל החיצוני ולהריץ סיווג על הקבצים, ניתן להוסיף כמות לא מוגבלת של מודלים.

לאחר שהצלחנו לסווג ישנן 2 אופציות נוספות אשר ניתנות לבחירה: שמירת המודל, יצירת plot השוואה. שמירת מודל אפשרית אך ורק עבור יצירת מודל חדש, ולא עבור טעינת מודל חיצוני, יצירת plot ניתנת עבור כל האופציות, כאשר נבנה מודל חדש, הפלוט יהיה השוואה בין סיווג קובץ הtest לבין סיווג קובץ הtrain, עבור מודל חיצוני, נעשית השוואה של סיווג הקובץ עבור כלל המודלים שנטענו.

ולבסוף מודפס לנו הConfusion Matrix עבור הסיווג אותו ביצענו.

לטובת הפרוייקט אנו עושים שימוש ב4 מסווגים מבית Sklearn ו2 מסווגים במימוש שלנו אשר יצרנו בעבודות בית קודמות, ושיכללנו ותיקנו לטובת פרוייקט זה, כלל שאר הפונקציות אשר אנו מריצים נכתבו על ידינו בלבד.

Files and Functions:

להלן הסבר על הפונקציות אשר אנו מריצים בקוד:

The project contains 2 files: "CLI.py", "Funcs.py. "

CLI.py contains: CLI implementation, menu, model saving .

main function - imports args from cmd, checks if there're any, if so sends to menu .

menu function - the user CLI menu with instructions and inputs .

helpFault function - prints error if user writes after "Help. "

noCommandFault function - prints suggestion to use "help. "

openFiles function - opens input files and catches exceptions .

preprocessing function - preprocesses testFile and if needed trainFile .

model_saving function - saves model to a ".sav" file using dill .

results argument - dictionary containing algorithem as key with (accuracy,precision,recall,Fmeasure) results on testFile .

Tresults argument - dictionary containing algorithem as key with (accuracy,precision,recall,Fmeasure) results on trainFile . Func.py contains: algorithems and functions install_and_import function - import package or installs and imports it if necessary .

InputToDict function - creates a dictionary of attributes from structure file .

numericBinning function - discretizies numeric values to equal-width bins .

probTable function - creates a table with all attributes and all class values with dictionaries .

probsCalc function - creates a dictionary with probabilities (yes and no) for each value of an attribute .

dictHelper function - creates a dictionary with all possible values as keys and counts the number of each value appearances . bestIGattr function - provides the attribute with the best info gain .

splitInfo function - provide entropy of an attributes value in order to calculate gain ratio .

ID3_tree function - creates a decision tree recursivly with dictionaries .

our_preprocessing function - preprocessess data in order to use our functions (remove rows with null 'class' values, fill null numeric values with means of the column, fill null categorial values with most common value) .

sklearn_preprocessing function - preprocessess data in order to use sklearn functions (remove rows with null 'class' values, fill null numeric values with means of the column, fill null categorial values with most common value and normalize numeric values with min-max method) .

Naive_bayes_classify function - our implementation of the naive bayes algorithem .

sklearn_NB_classify function - classification using sklearn naive bayes .

ID3_classify function - our implementation of the id3 algorithem .

sklearn_ID3_classify function - classification using sklearn id3 .

KNN_classify function - classification using sklearn knn .

KMeans_classify function - classification using sklearn kmeans .

resultsDict function - adds algorithems accuracy, precision, recall and FMeasure to results dictionary .

comparing_plotter function - creates a vbar plot comparing running testFile vs train File on the same model .

ready functions used - entrpoy, info gain, equal-width binning

our implementation - min-max normalization ,

EDA:

להלן הDB אשר איתו אנו נעבוד:

class	poutcome	previous	campaign	duration	month	day	contact	loan	housing	balance	default	education	marital	job	age	
no	unknown	0	1	261.0	may	5	unknown	no	yes	2143.0	no	tertiary	married	management	58	**0**
no	unknown	0	1	151.0	may	5	unknown	no	yes	29.0	no	secondary	single	technician	44	**1**
no	unknown	0	1	76.0	may	5	unknown	yes	yes	2.0	no	secondary	married	entrepreneur	33	**2**
no	unknown	0	1	92.0	may	5	unknown	no	yes	1506.0	no	unknown	married	blue-collar	47	3
no	unknown	0	1	198.0	may	5	unknown	no	no	1.0	no	unknown	single	unknown	33	**4**
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	click to e
yes	failure	2	1	98.0	nov	9	cellular	no	no	10861.0	no	tertiary	married	self-employed	60	**42175**
yes	unknown	0	1	201.0	nov	9	telephone	no	no	2331.0	no	tertiary	married	entrepreneur	64	**42176**
yes	failure	2	1	161.0	nov	9	cellular	no	yes	2374.0	no	secondary	married	.admin	34	**42177**
yes	failure	12	2	696.0	nov	9	cellular	no	no	1228.0	no	tertiary	single	management	35	**42178**
no	failure	4	2	371.0	nov	9	cellular	no	yes	1356.0	no	tertiary	married	management	53	**42179**

כעת נרצה לנתח וללמוד את מערך הנתונים אשר איתו נעבוד, ז"א נרצה לדעת אילו פעולות נצטרך לבצע על מנת שנביא את הDB שלנו למצב שבו נוכל לעבוד איתו, נתחיל בניתוח פשוט של אילו שדות חסרים לנו, על מנת שנוכל להשלים אותם במידת הצורך.

perc_missing	total_missing	total_count	
0.000000	0	42180	age
0.000000	0	42180	job
0.000000	0	42180	marital
0.004742	2	42178	education
0.000000	0	42180	default
0.011854	5	42175	balance
0.007112	3	42177	housing
0.002371	1	42179	loan
0.016596	7	42173	contact
0.000000	0	42180	day
0.000000	0	42180	month
0.026079	11	42169	duration
0.000000	0	42180	campaign
0.000000	0	42180	previous
0.000000	0	42180	poutcome
0.000000	0	42180	class

כפי שניתן לראות בטבלה 10 מתוך 16 עמודות שלנו הינם ללא ערכים חסרים בכלל, לעומת זו ישנן 6 עמודות אשר ישנן מספר ערכים חסרים, שנדרש לטפל בהם, אך האחוזים הינם נמוכים מאוד. בכדי לדעת כיצד נוכל לטפל באותם ערכים חסרים נצטרך לגשת לקובץ הStruct שלנו כדי להבין עם איזה סוג נתונים אנו מתמודדים איתו.

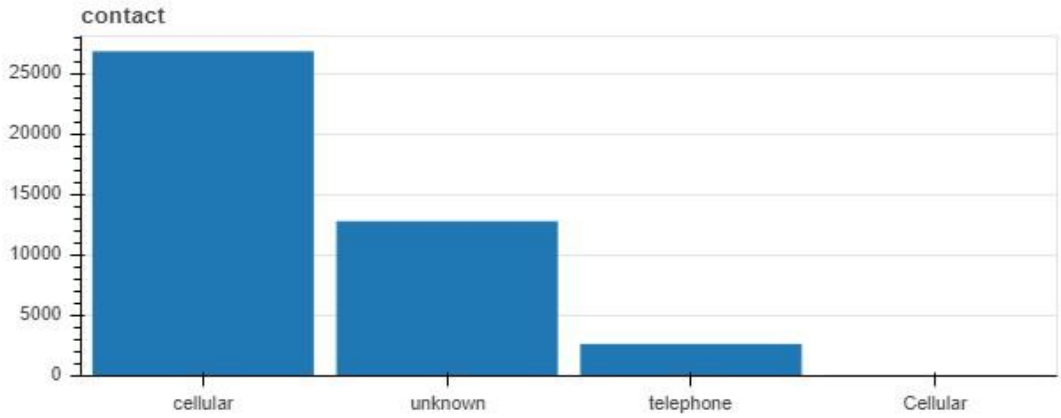
```
@ATTRIBUTE age NUMERIC
@ATTRIBUTE job {admin.,unknown,unemployed,management,housemaid,entrepreneur,student,blue-collar,self-employed,retired,technician,services}
@ATTRIBUTE marital {married,divorced,single,widowed}
@ATTRIBUTE education {unknown,secondary,primary,tertiary}
@ATTRIBUTE default {yes,no}
@ATTRIBUTE balance NUMERIC
@ATTRIBUTE housing {yes,no}
@ATTRIBUTE loan {yes,no}
@ATTRIBUTE contact {unknown,telephone,cellular}
@ATTRIBUTE day NUMERIC
@ATTRIBUTE month {jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec}
@ATTRIBUTE duration NUMERIC
@ATTRIBUTE campaign NUMERIC
@ATTRIBUTE previous NUMERIC
```

ניתן לראות כי עמודות education,housing,loan,contact הינם עמודות קטגוריאליות, בכדי לנסות להשפיע ככל הפחות על הDB נמלא עמודות אלו בערך הכי נפוץ של אותן עמודות.

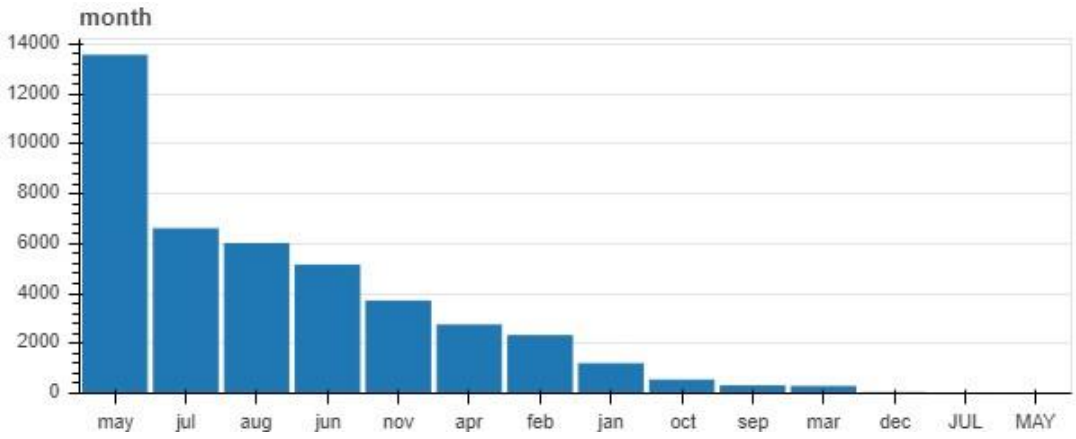
לעומתן, עמודות balance,duration הינם עמודות נומריות, ועמודות אלו נמלא בעזרת ממוצע העמודה.

במידה והיו לנו שורות עם ערכי class חסרים, היינו נדרשים למחוק אותן, מאחר ולא נוכל ללמוד משורות אלו אף מידע, מאחר ולא נדע לאיזה סיווג לשייך מידע זה.

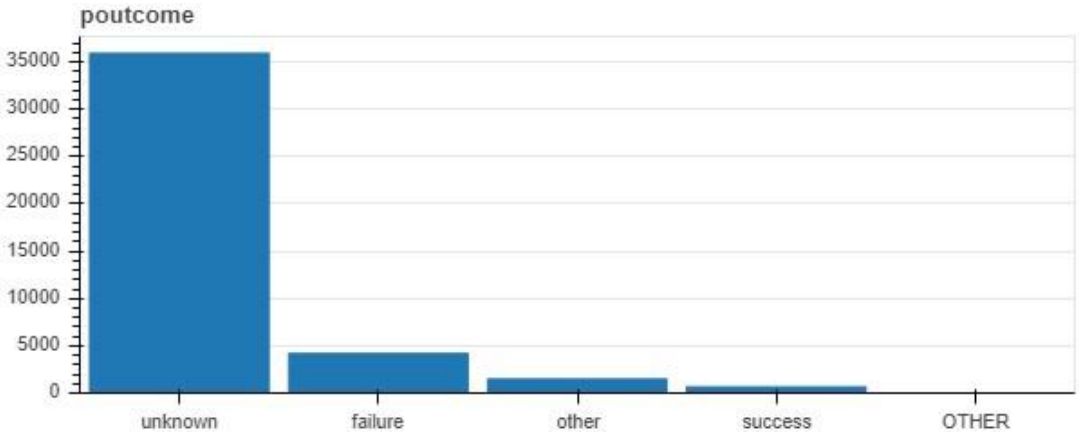
כעת נראה לראות האם יש לנו התנהגות חריגה או ערכים חריגים של עמודות מסויימות:



ניתן לראות כי ישנו ערך חריג בעמודה contact, כך שעבור ער ך Cellular ישנם כמות מסויימת של ערכים אותם נצטרך להמיר לcellular אחרת יוכל להשפיע על הסיוו ג.



כך גם בעמודה month ישנם 2 ערכים חריגים JUL/MAY אשר נצטרך להחליפם.



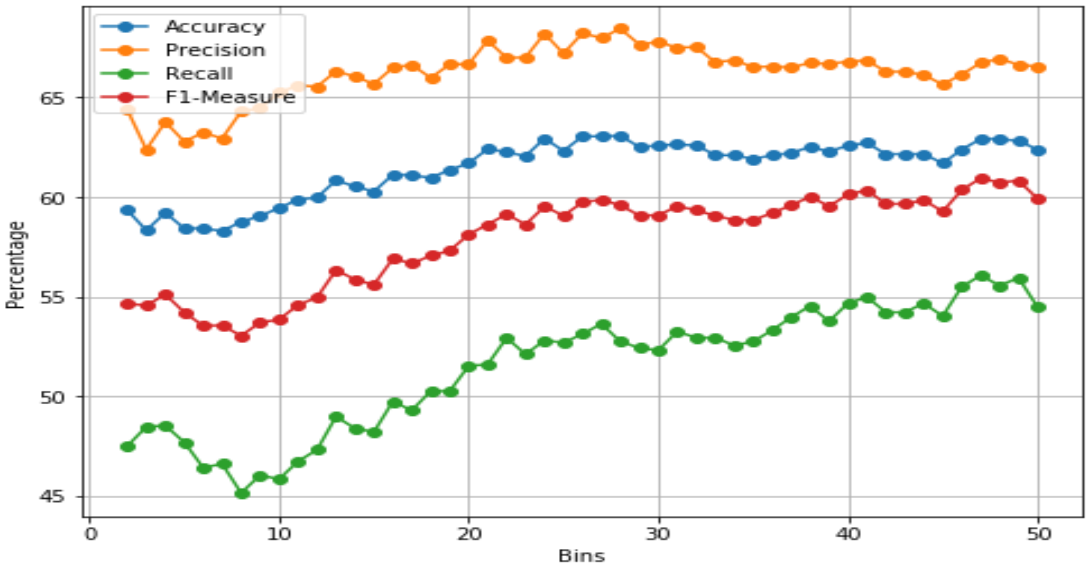
ובעמודה poutcome ישנו ערך OTHER שנרצה להחליפו.

לסיכום, נדרש להשלים מספר ערכים במספר עמודות, וישנם ערכים חריגים אשר נדרש לטפל בהם, אך מעבר לכך הDB הנתון הינו יחסית איכותי, ללא הרבה ערכים חסרים או חריגים.

Model settings:

כעת נרצה לבחון הגדרות עבור המודל שלנו, נציג הרצת מדדי הערכה לפי כמות הבינים בדיסקרטיזציה של עמודות נומריות במודלים שלנו:

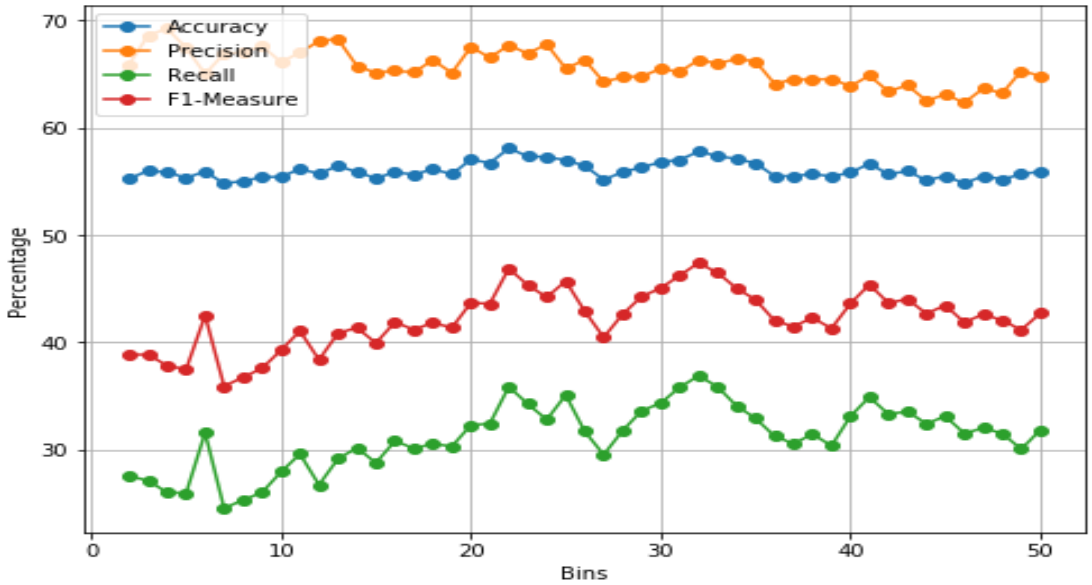
מודל Naïve Bayes מימוש שלנו:



כמות הבינים הטובה ביותר עבור:

Accuracy: 28 Precision: 28 Recall: 47 Fmeasure: 47

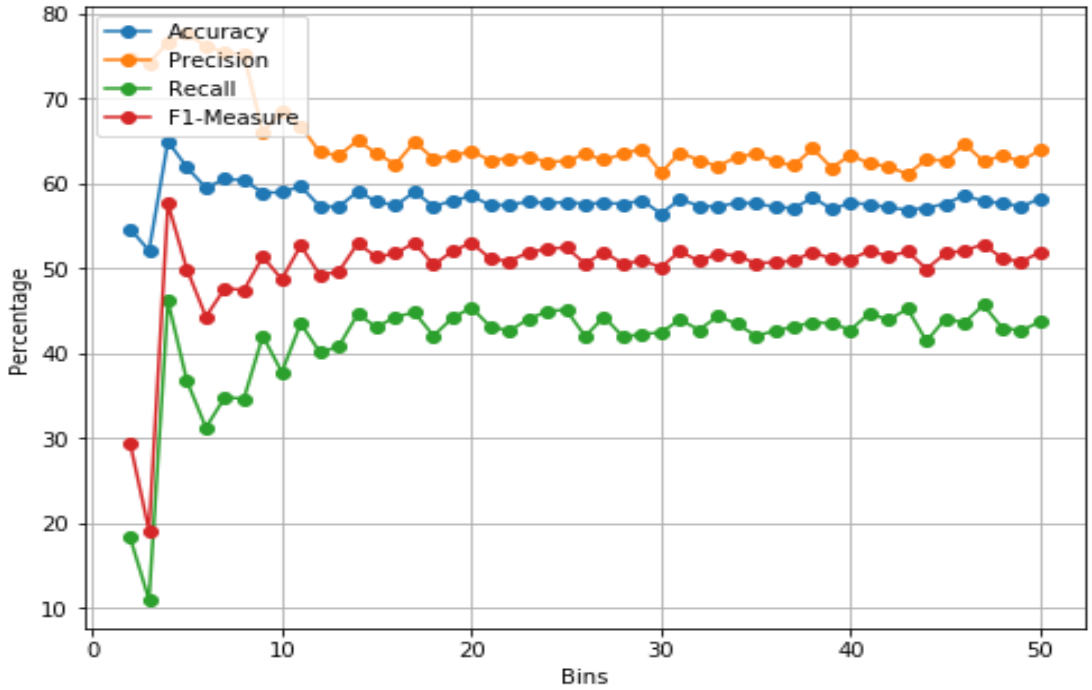
מודל ID3 מימוש שלנו:



כמות הבינים הטובה ביותר עבור:

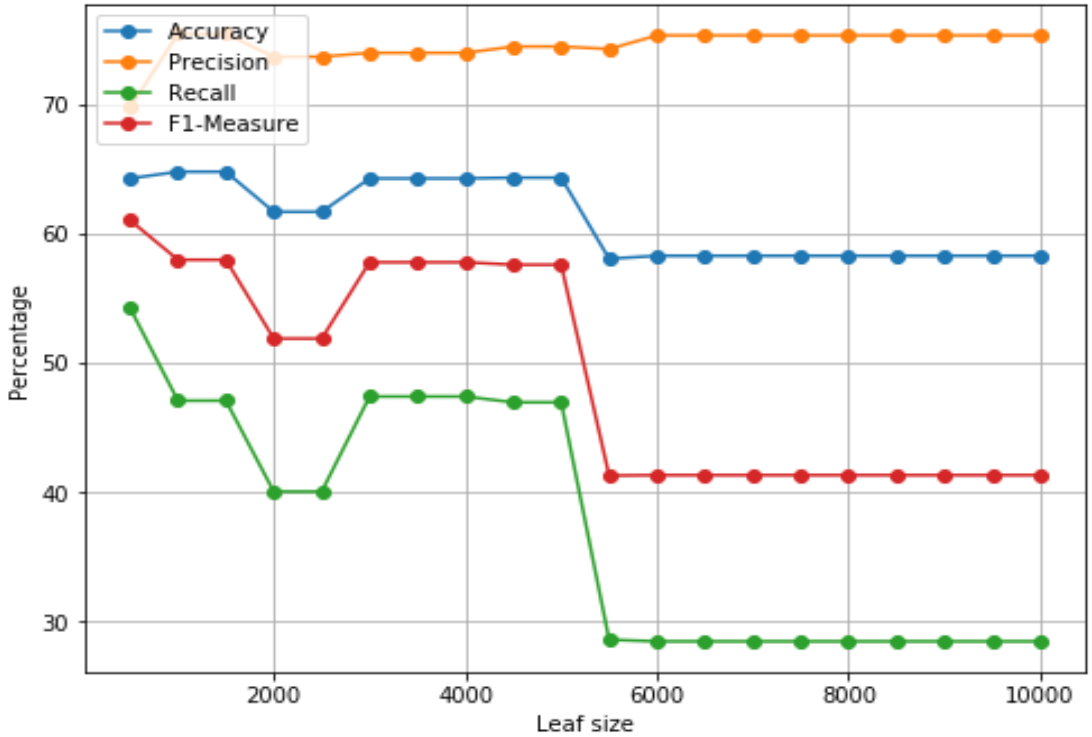
Accuracy: 22 Precision: 4 Recall: 32 Fmeasure: 32

מודל ID3 מימוש sklearn:



עומק המומלץ ביותר עבור:

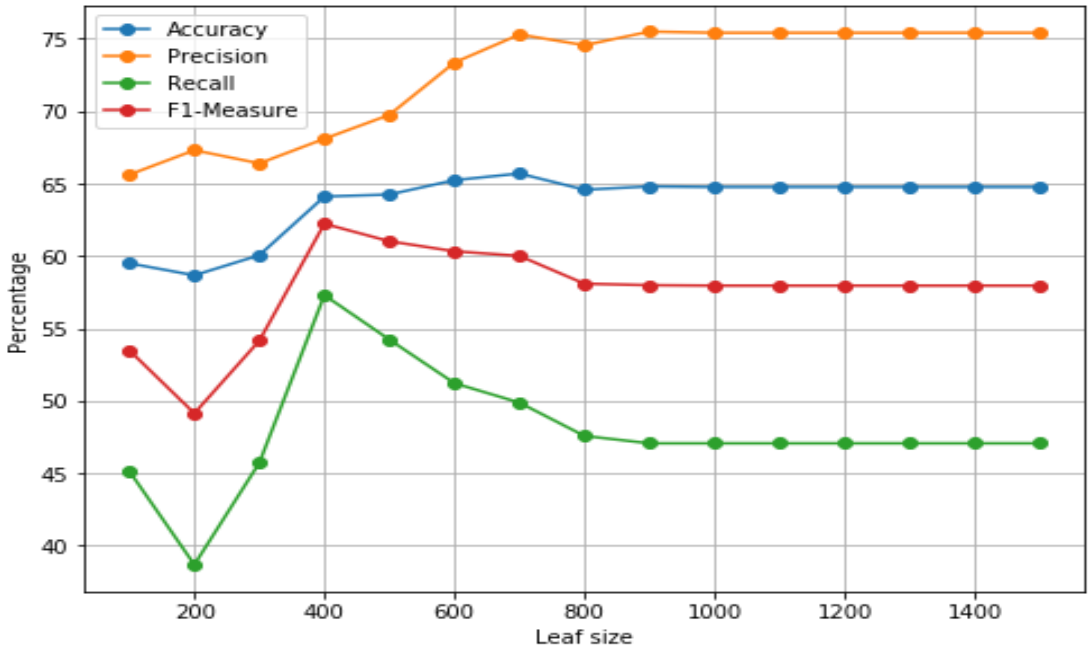
Accuracy: 4 Precision: 5 Recall: 4 Fmeasure: 4



Recall: 500 Fmeasure: 500

כמות העלים בה מומלץ לחתוך את העץ:

Precision: 1000 1000
Fmeasure: 400 Accuracy:
Recall: 400

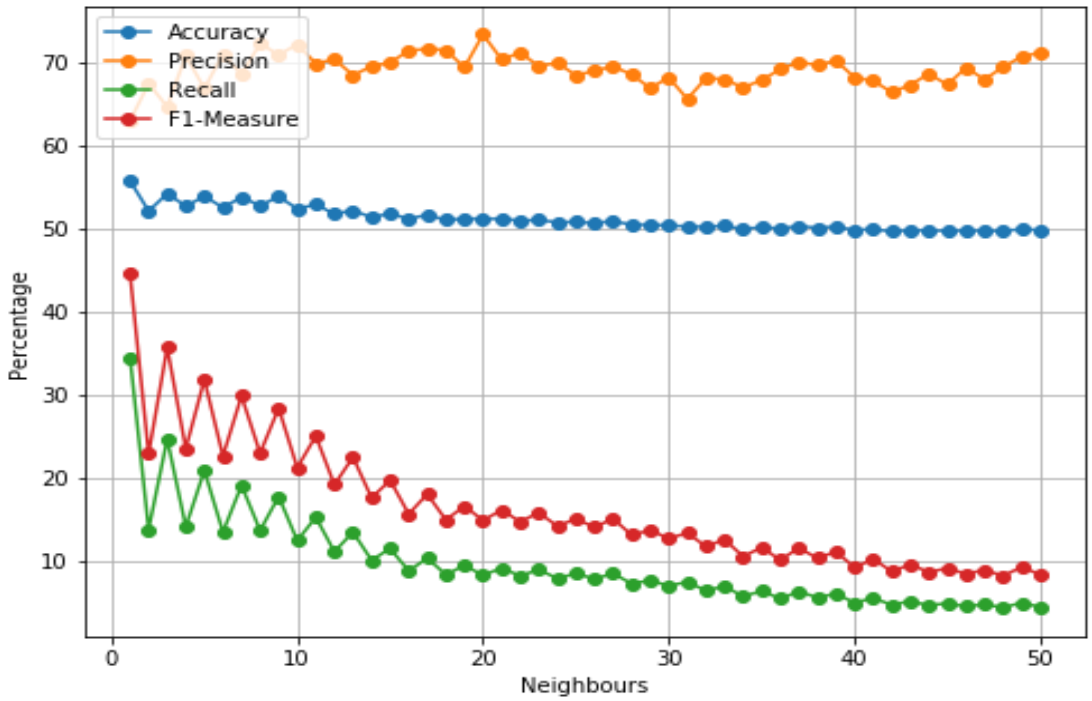


וכאשר נביט מקרוב יותר)קפיצות של 100(
:

כמות העלים בה מומלץ לחתוך את העץ:

Accuracy: 700 Precision: 900

מודל KNN מימוש sklearn:

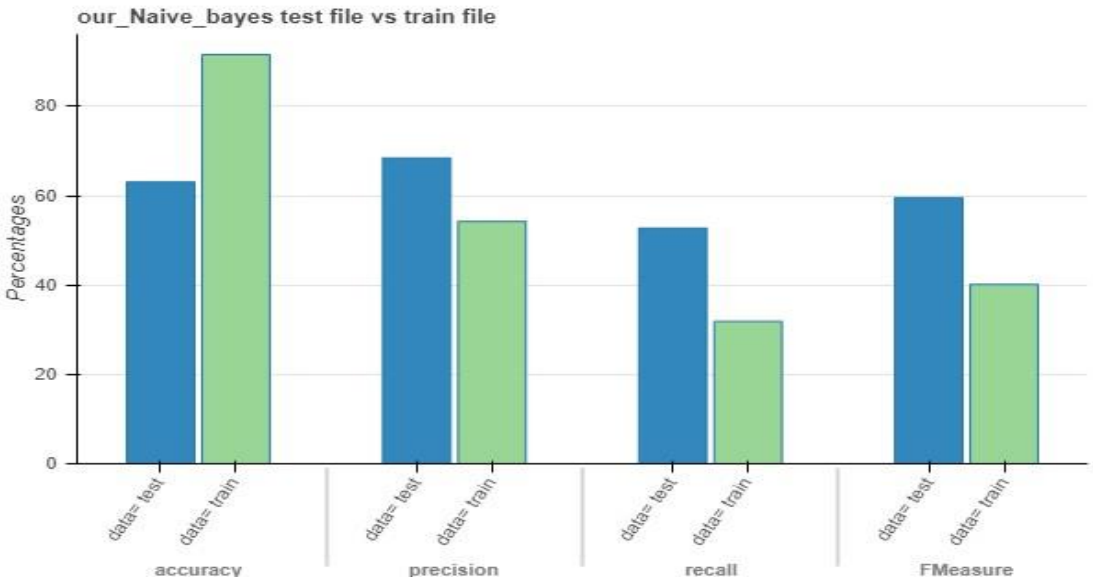


כמות השכנים הטובה ביותר עבור:

Accuracy: 1 Precision: 20 Recall: 1 Fmeasure: 1

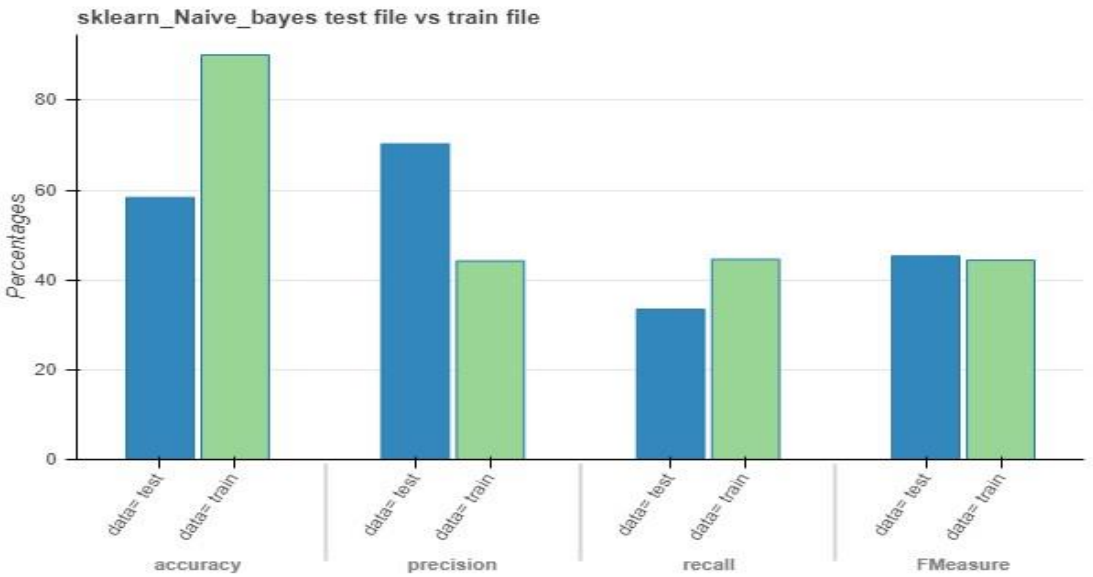
Classifying:

הרצת Naïve Bayes מימוש שלנו:



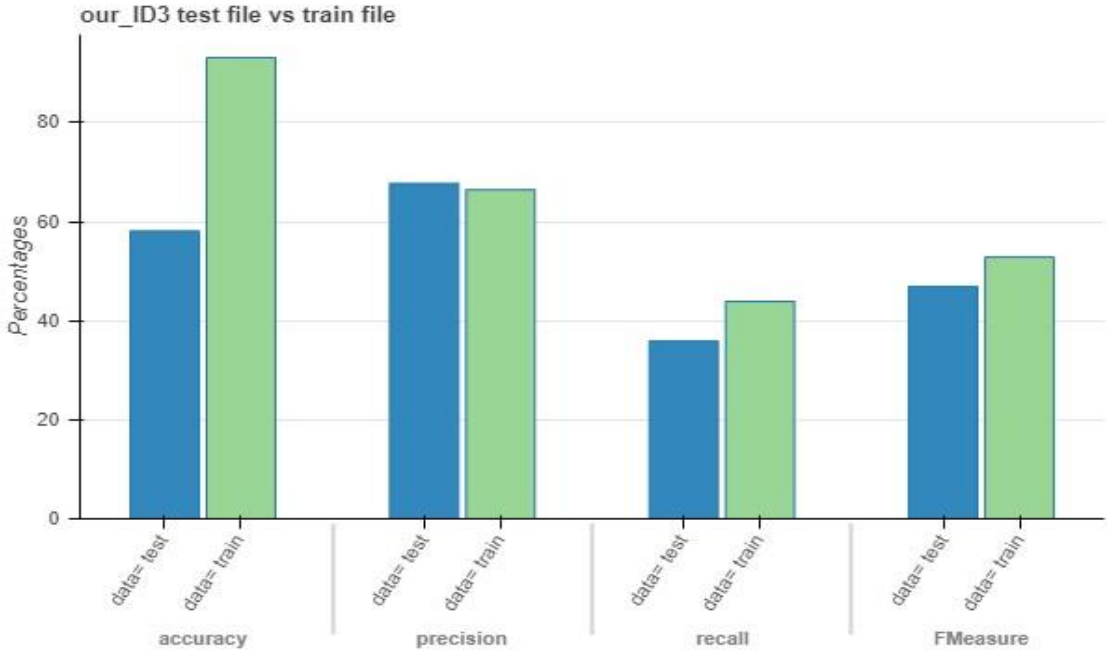

```
Algorithem test:  our_Naive_bayes  :
{'accuracy': 63.08, 'precision': 68.46, 'recall': 52.75, 'FMeasure': 59.59}
Algorithem train:  our_Naive_bayes  :
{'accuracy': 91.61, 'precision': 54.25, 'recall': 31.87, 'FMeasure': 40.15}
```

הרצת Naïve Bayes מימוש sklearn:



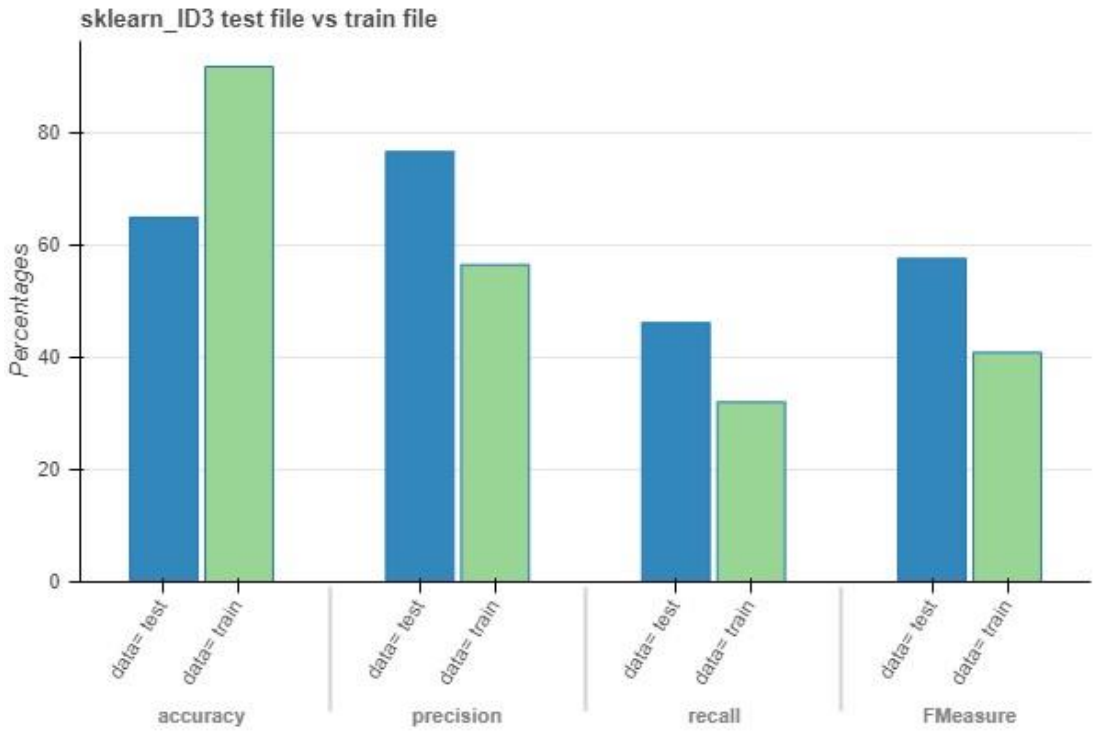

```
Algorithem test:  sklearn_Naive_bayes  :
{'accuracy': 58.36, 'precision': 70.3, 'recall': 33.44, 'FMeasure': 45.32}
Algorithem train:  sklearn_Naive_bayes  :
{'accuracy': 90.14, 'precision': 44.23, 'recall': 44.59, 'FMeasure': 44.41}
```

הרצת ID3 מימוש שלנו:



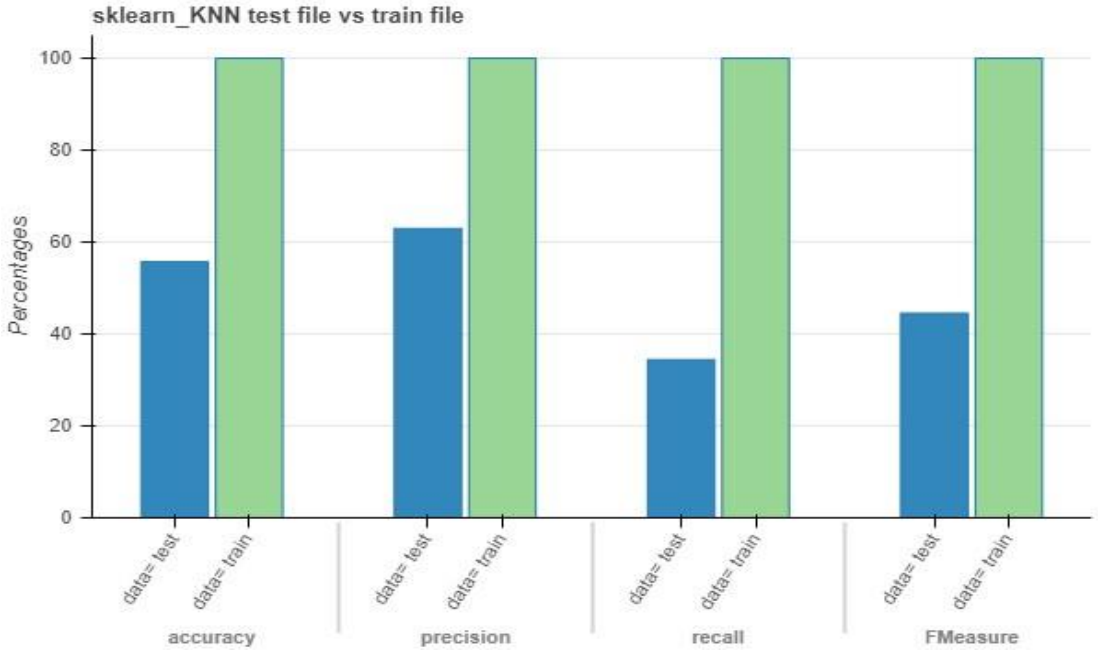

```
{'accuracy': 58.07, 'precision': 67.67, 'recall': 35.87, 'FMeasure': 46.89}
Algorithem train:  our_ID3  :
{'accuracy': 93.08, 'precision': 66.4, 'recall': 43.87, 'FMeasure': 52.83}
```

הרצת ID3 מימוש sklearn:



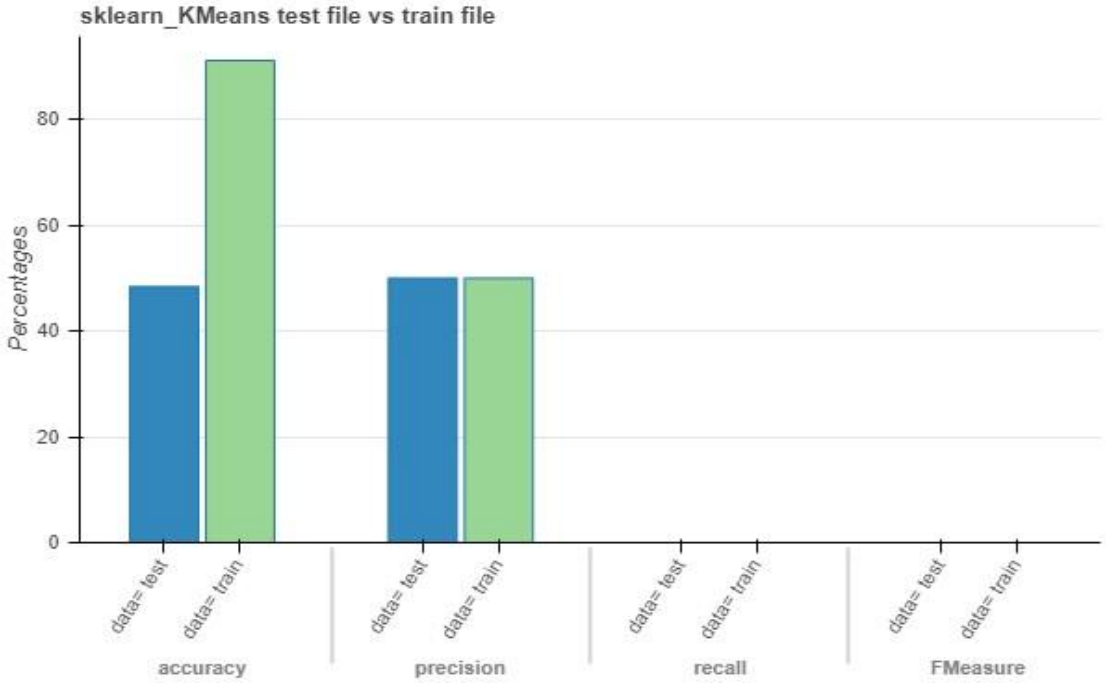

```
Algorithem test:  sklearn_ID3  :
{'accuracy': 64.96, 'precision': 76.65, 'recall': 46.16, 'FMeasure': 57.62}
Algorithem train:  sklearn_ID3  :
{'accuracy': 91.82, 'precision': 56.51, 'recall': 32.03, 'FMeasure': 40.88}
```

הרצת KNN מימוש sklearn:




```
Algorithem test:  sklearn_KNN  :
{'accuracy': 55.69, 'precision': 62.92, 'recall': 34.4, 'FMeasure': 44.48}
Algorithem train:  sklearn_KNN  :
{'accuracy': 100.0, 'precision': 100.0, 'recall': 100.0, 'FMeasure': 100.0}
```

הרצת סיווג Kmeans מימוש sklearn לאחר אישכול:




```
Algorithem test:  sklearn_KMeans  :
{'accuracy': 48.4, 'precision': 50.0, 'recall': 0.0, 'FMeasure': 0.0}
Algorithem train:  sklearn_KMeans  :
{'accuracy': 91.17, 'precision': 50.0, 'recall': 0.0, 'FMeasure': 0.0}
```

ולסיכום:

המדדים הטובים ביותר פר אלגוריתם פר מימוש:

מדדי הערכה Naïve bayes מימוש שלנו:
Accuracy: 63.08 Precision: 68.46 Recall: 56.07 Fmeasure: 60.95
מדדי הערכה ID3 מימוש שלנו:

Accuracy:58.07 Precision: 69.22 Recall: 36.89 Fmeasure: 47.41
מדדי הערכה ID3 מימוש sklearn עבור עומק הכי טוב:

Accuracy:64.96 Precision: 77.6 Recall: 46.16 Fmeasure: 57.62

מדדי הערכה ID3 מימוש sklearn עבור כמות Nodes בעלה הכי טובה:

Accuracy:65.69 Precision: 75.49 Recall: 57.29 Fmeasure: 62.22

מדדי הערכה KNN מימוש sklearn עבור כמות שכנים הטובה ביותר:

Accuracy:55.69 Precision: 73.3 Recall: 34.4 Fmeasure: 44.48

לכן במידה והיינו מעוניינים לסווג לפי:

Accuracy: ID3 של sklearn עם עומק 4

Precision: ID3 של sklearn עם עומק 5

Recall: ID3 של sklearn עם כמות Nodes של 400

Fmeasure: ID3 של sklearn עם כמות Nodes של 400

בסופו של דבר בחירת ההגדרות הנכונות ביותר והמודל הנכון ביותר הינו עניין של שיקול דעת, ז"א איזה מדד הכי רלוונטי בשבילך, כאשר יש משקל שונה עבורך לאחד המדדים TP,FP,TN,FN נדרש לבחור את האלגוריתם הרלוונטי.

אם בצורה הכי פשוטה היה אכפת לנו כמה המודל שלנו יודע לסווג נכון היינו בוחרים לפי Accuracy. במידה וישנו משקל גבוה לסיווג של FP, נרצה לבחור לפי Precision, במידה והמשקל ג בוה לסיווג של FN נבחר לפי Recall, כאשר נחפש מדד מאוזן יחסית בין FP לבין FN כך שהמשקל שלהם קרוב אחד לשני מבחינתנו נבחר בFmeasure.