**מיני פרויקט בשיטות לכריית מידע שומרות פרטיות**

**תיאור הבעיה והסבר האלגוריתם**

הבעיה עמה נדרשנו להתמודד ולממש הינה:

בהינתן מספר מקורות מידע שונים, בעלי Databases שאפיונם דומה, נרצה לכרות מידע מאיחוד ה-Databases, זאת מבלי שכל מקור יחשוף את המידע שברשותו.

בפרויקט זה מומש אלגוריתם לבניית עץ החלטה הנבנה על סמך איחוד מספר מקורות מידע שונים בצורה משמרת פרטיות. עץ ההחלטה נבנה בעזרת האלגוריתםID3 , אשר מסתמך על ה-Database המאוחד המתקבל מהרצת פרוטוקול שימור הפרטיות על כל אחד ממקורות המידע – Shamir's privacy preserving summation of secrets .

**תיאור של אלגוריתם כריית המידע**

אלגוריתם כריית המידע אותו החלטנו לממש הינו האלגוריתם ID3 לבניית עץ החלטה.

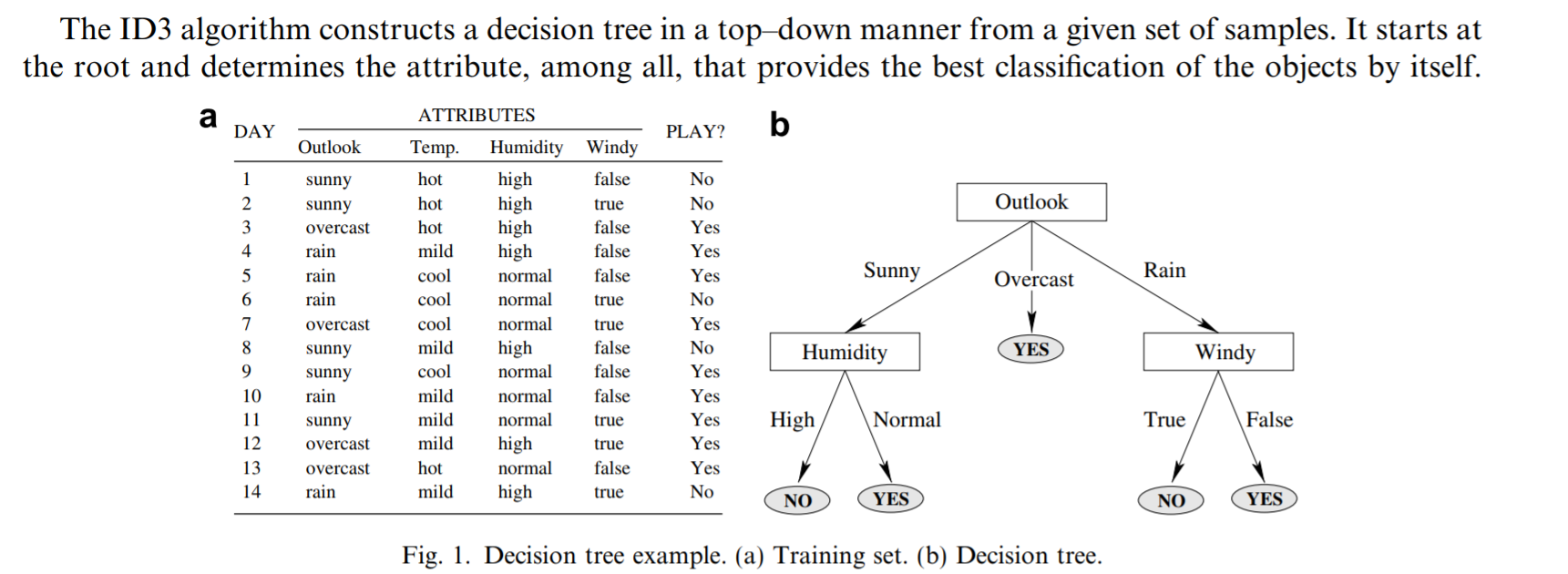
עץ ההחלטה הינו מודל חיזוי, שבהנתן ערכי מאפיינים שונים של מופע, יוכל העץ לחזות בצורה טובה את הקטגוריה אליו המופע שייך.

האלגוריתם מבצע ניתוח של המידע הקיים, "Training set" – מאגר מידע בעזרתו אנו "מאמנים" את המודל, על מנת לבנות את עץ ההחלטה – עבור מידע חדש שעתיד להיבחן.

הכוונה בחיזוי בצורה טובה הינו חיזוי באחוז וודאות גבוה ככל הניתן, בהתבסס על ניתוח הנתונים המתקבלים ב-Training set.

דוגמה פשוטה למודל חיזוי תהיה מערכות לחיזוי מזג אוויר – מערכות אלו נשענות על מדידות עבר של מאפיינים שונים, כגון טמפרטורה וכמות משקעים בתקופות שונות בשנה במקומות שונים, ובעזרת כך, לדוגמא, ניתן לבצע שאילתא כמו "מהי הטמפרטורה הצפויה לסוף חודש ינואר בצפון הארץ".

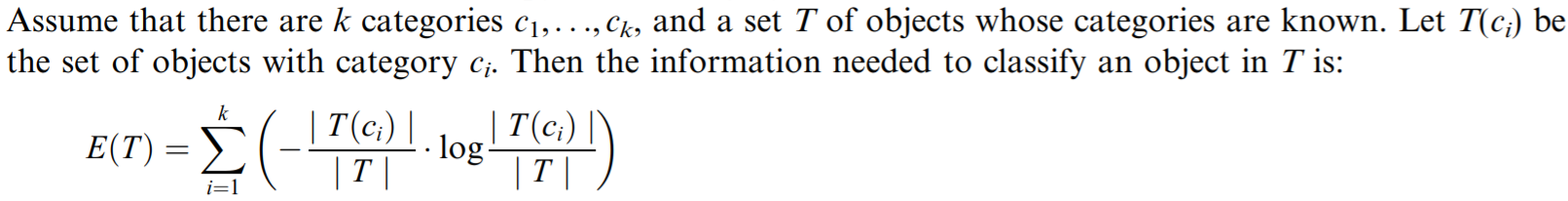
האלגוריתם בונה את עץ ההחלטה, אשר ניתן להגדירו כגרף פשוט מכוון, או כעץ (שאינו בהכרח בינארי), שקודקוד ההתחלה שלו, או השורש, הינו המאפיין אשר מסווג את המופעים הנתונים במאגר המידע בצורה ה'טובה' ביותר, ובניו של השורש יהיו ערכי מאפיין זה, כך שתתי העצים שהם שורשיהם, ייבנו בצורה רקורסיבית עבור המופעים במאגר המידע שהם בעלי ערכים אלו. עלי עץ ההחלטה יהיו הקטגוריות השונות – התשובות לשאילתא – שאת תשובתה אנו נרצה לדעת לחזות.



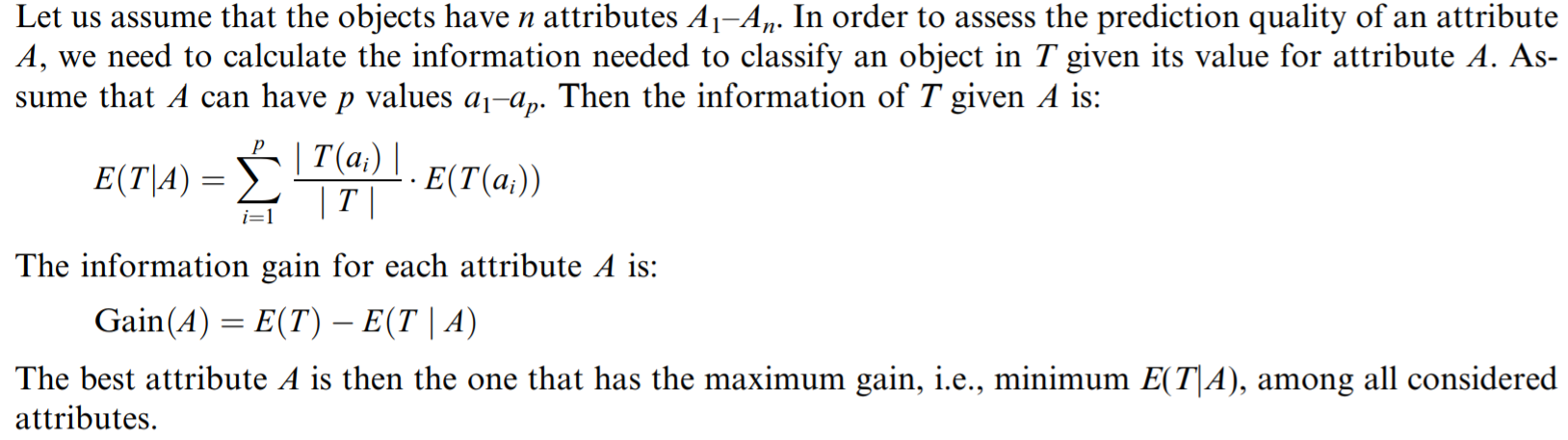
בדוגמא זו, הלקוחה מן המאמר, ניתן לראות כי המאפיין Outlook הינו המסווג האיכותי ביותר – לכל מופעי הערך Overcast של מאפיין זה, הקטגוריה אליה משתייכים מופעים אלו היא "Yes", עבור השאילתא עליה עונה העץ - "האם מומלץ לשחק?"

המאפיין שנותן את החיזוי הטוב ביותר נקבע על ידי שימוש בתאוריית המידע (Information Theory).

מאפיין החיזוי הטוב ביותר בכל שלב בבניית העץ יהיה המאפיין שערך האנטרופיה שלו הוא המינימלי בהינתן אינפורמציית הקטגוריות השונות.

חישוב אינפורמציית הקטגוריות השונות מוגדר כך: 

וכך מחושבים ערכי האנטרופיה של כל אחד מהמאפיינים הנתונים לנו ב-Database:

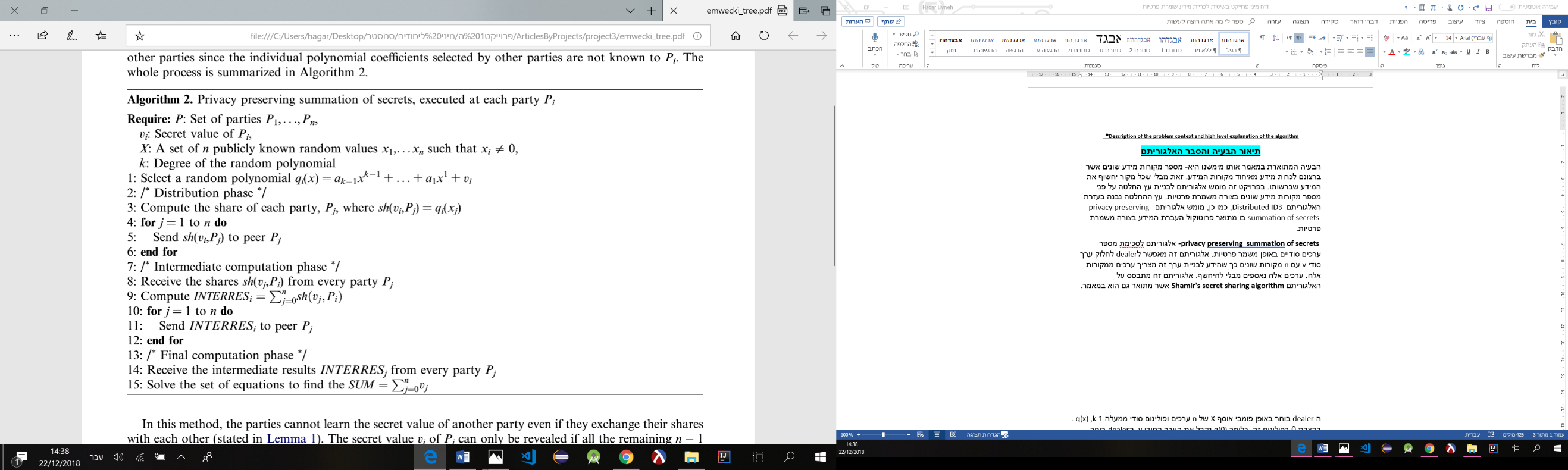


מתוך כלל חישובי ה-Gain, נבחר בכל שלב בבניית העץ את המאפיין בעל ה-Gain המקסימלי, שכן E(T) הינו ערך קבוע לכלל המאפיינים בשלב זה, ו מייצג את ערך האנטרופיה של מאפיין A תחת קבוצת האובייקטים T, קבוצת האובייקטים הרלוונטים שנותרו – לדוגמא, עבור תת העץ שניתן בדוגמא, המושרש תחת הערך Sunny של מאפיין Outlook, נחפש מבין קבוצת האובייקטים שערך המאפיין Outlook שלהם הוא Sunny, את המאפיין הדומיננטי הבא.

הבעיה איתה אנו נתקלים במימוש בניית העץ באופן משמר פרטיות, היא שמאגר המידע עלינו נרצה לבסס את בניית העץ, מפוצל בין מספר משתתפים, אשר מעוניינים לשמור על פרטיות המאגרים שלהם. לכן, לפני שנפנה לבנות את העץ, נידרש לאגד את מאגרי המידע יחד במיקום יחיד, ונעשה זאת תוך שמירה על פרטיות המאגרים. נבצע זאת בעזרת פרוטוקול המתואר הבא:

**Shamir's privacy preserving summation of secrets-** פרוטוקול זה מתאר אלגוריתם לסכימת מספר ערכים סודיים, כך שאף משתתף אינו חשוף לערכי המשתתפים האחרים, אך כולם חשופים לסכום ערכים אלו בסיום הרצת האלגוריתם.

האלגוריתם משתמש ב- (server) dealer, צד חיצוני כלשהו אשר 'מנהל' את הרצת האלגוריתם, על מנת לחשב סכום של n ערכים סודיים המגיעים מ-n מקורות שונים ונותרים מוסווים (גם עבור הדילר). אלגוריתם זה מתואר במאמר עליו הסתמכנו בעבודתנו, כך:



במימוש שלנו, הserver מגריל n ערכים רנדומליים – , כמספר המשתתפים(clients=), ושולח את מערך הערכים האלו לכל n המשתתפים. כל משתתף מגריל פולינום רנדומלי  שדרגתו לכל היותר n-1, כך שהאיבר החופשי ב- שהגריל יהיה ה-secret value שלו. בפולינום זה הוא מציב את הערכים המוגרלים מן הסרבר – , ואת תוצאת כל הצבה, נסמנה , מכניס למערך לתא ה-j ושולח בחזרה לserver-.

הserver- כאמור מקבל n מערכים בעלי n תאים כל אחד, כך שבכל מערך i , בתא ה-j מופיעה תוצאת הצבת הערך  בפולינום . ה-server סוכם לכדי מערך חדש, נסמנו X, בגודל n, לתא ה-i, את כל תוצאות הצבת  ב-n הפולינומים השונים – ז"א תא i בכל אחד מהמערכים שהתקבלו מן המשתתפים. לאחר סכימת כל הערכים השונים, מתקבל המערך X בתצורתו הסופית, כך שבתא ה-i במערך זה, יישב הערך . כעת, מחזיק ה-server במערך של n ערכים, , כך שכל ערך כזה מהווה את תמונת הצבת הערך  ב-n הפולינומים שבכל אחד מהם, הערך החופשי הינו הערך הסודי של המשתתף המחזיק פולינום זה. במילים אחרות – הסרבר מחזיק ב-n נקודות מהצורה  וזהו מידע מספיק על מנת לבצע אינטרפולציית לגראנז' על n הנקודות האלו, ולקבל את  הפולינום אשר מורכב מחיבור n הפולינומים השונים (שכן דרגת פולינום זה היא לכל היותר כדרגת הפולינום הגבוהה ביותר בסכום – n-1). לאחר קבלת P, ה-server יציב את הערך 0 ב-P, ויקבל את האיבר החופשי של P, שמהווה סכום האיברים החופשיים של הפולינומים המוגרלים – וזהו הערך אותו חיפשנו. חישוב זה ונכונותו מפורטים ב חלק 4 במאמר.

תהליך זה המתואר לעיל, של סכימת ערכים סודיים, מתבצע עבור כל אחד מהתאים ב-dictionary שבמחלקה processed\_db.

dictionary זה מהווה מבנה נתונים, אשר לו מופע אצל כל אחד מה-clients וכן ה-server. עיבוד המידע והסבר על מחלקה זו מופיעים בסעיף על **תיאור מאגר הנתונים**.

בסיום תהליך זה, כאשר סכימת הערכים הסודיים מתבצעת עבור כל אחד מהתאים במבני הנתונים של ה-clients, מתקבל dictionary במחלקת processed\_db שמשוייך ל-server , כך שבכל תא ב-dictionary זה נמצא סכום ערכי תא זה של כל ה-clients.

**בסיום התהליך, נקבל מאגר מידע מוכלל המוחזק ב-server, המורכב מחיבור כל מאגרי המידע של ה-clients המשתתפים בתהליך.**

כעת, נוכל לעבור לבניית עץ החלטה הנבנה על סמך מאגר המידע המוכלל, כפי שמפורט בסעיף **תיאור של אלגוריתם כריית המידע**. לאחר שהעץ נבנה על סמך הנתונים המוכללים, ניתן בעזרת הפונקציה predict להזין לו אובייקט המאופיין באופן דומה לאובייקטים שעליהם נבנה כל database, ולקבל 'ניבוי' מושכל של קטגוריה כלשהי אליה משתייך האובייקט. סוגי האובייקטים וקטגוריות אפשריות מפורטות בסעיף **תיאור מאגר הנתונים**.

**תיאור התוכנה (מחלקות ראשיות)**

התוכנה מורכבת מהמחלקות הבאות:

* **client.py** מחלקות המייצגות 5 מקורות מידע שונים. כל מחלקה כזו מאופיינת ע"י השדות הבאים:
* client\_index - ערך מספרי המייצג את האינדקס של client זה
* num\_of\_clients – ערך מספרי המייצג את מס' המשתתפים
* מופע של processed\_db-מחלקה המעבדת את הdatabase של הclient הספציפי.
* secret\_vals\_table – מבנה נתונים מסוג dictionary אשר מחזיק ערכים של המידע המעובד.
* polynom – מערך של ערכים מספרים, כך שכל תא מייצג את המקדם של  בפולינום של המשתתף.

מחלקות אלו מחזיקות בראשית התהליך את מאגרי המידע הגולמיים, אותם נרצה לאחד למאגר מידע מאוחד תוך שמירה על פרטיות המאגרים. הקליינטים(=משתתפים) אחראיים לחשב עבור ה-server(=שרת) חישובים הנדרשים על מנת שיוכל לסכום את הערכים הסודיים שאותם הם מחזיקים בנפרד.

כל קליינט(=משתתף) יבצע את חלקו בפרוטוקול האבטחה אל מול השרת עבור כל ערך של כל חיתוך אפשרי של מאפייני מאגר המידע.

אחריות המשתתפים בפרוטוקול האבטחה כוללת הגרלת פולינום בו לוקח חלק הערך הסודי של המשתתף, וחישוב הערכים הרנדומליים המתקבלים מהשרת בפולינום זה.

פונקציות עיקריות של המחלקה :

* cal\_val\_in\_poly – פונק' זו מקבלת את מערך הערכים הרנדומליים מהסרבר, מציבה כל ערך בנפרד בפולינום האישי של המשתתף, ומעבירה את התוצאות חזרה ל-server.
* generate\_poly – פונק' זו תקבל את כמות המשתתפים ואת הערך הסודי של הקליינט, ותגריל מקדמים לאיברי פולינום שדרגתו תהיה כמות המשתתפים פחות 1. הערך הסודי יהיה האיבר החופשי בפולינום. שאר המקדמים יוגרלו כמס' שלמים בטווח [-50,50].
* **Server.py** מחלקה זו מאופיינת על ידי השדות הבאים:
* num\_of\_parties- ערך מספרי המייצג את מס' המשתתפים
* deg – ערך מספרי המייצג את דרגת הפולינום המקס' שכל אחד מהמשתתפים יידרש להגריל
* summedXis – מערך אשר יחזיק את סכום ערכי הצבת הערכים הרנדומליים בכל הפולינומים המשתתפים.
* rand\_vals – מערך אשר יחזיק את n הערכים הרנדומליים המוגרלים ע"י ה-server
* clientnum – ערך מספרי המונה את כמות המשתתפים
* מופע של processed\_db – כמתואר לעיל, מופע של המחלקה אשר יחזיק בתום ריצת האלגוריתם את סכומי הsecret vals של כל הclients.

מחלקה זו אחראית לתפעל את איחוד מאגרי המידע באופן משמר פרטיות, ובעזרת המאגר המוכלל לעזור לבנות את עץ ההחלטה המוכלל.

איחוד מאגרי המידע כולל בתוכו הרצת פרוטוקול אבטחה המתחלק בין השרת למשתתפים.

אחריות השרת בפרוטוקול כוללת הגרלת מספרים רנדומליים שלמים בטווח [1,50], כך שעל כל משתתף יש מס' רנדומלי, שליחתם למשתתפים השונים על מנת שיוכלו לבצע את חישוביהם, וקבלה בחזרה ערכים מכל משתתף. בעזרת ערכים אלו והערכים המוגרלים, מבצע השרת אינטרפולציית לגראנז' דרכה הוא מוצא את סכום הערכים הסודיים ומאחסן את סכום זה במאגר המידע המאוחד אותו הוא בונה.

פונקציות עיקריות של המחלקה:

* find\_out\_sum\_value – פונק' זו מבצעת את אינטרפולציית לגראנז' בעזרת הנקודות המורכבות מהערכים המוגרלים והערכים המתקבלים מכל משתתף.
* set\_rand\_vals – פונק' זו מגרילה n ערכים רנדומליים שלמים שונים זה מזה ומאפס בטווח [1,50], כאשר n מייצג את מס' המשתתפים בפרוטוקול האבטחה.
* בלולאת פור יסודית העוברת על כל ערכי החיתוכים האפשריים, מריץ השרת את פרוטוקול האבטחה, לכל ערך בנפרד, בהשתתפות כל המשתתפים, על מנת לחשב את סכום ערכי החיתוכים מכל משתתף.
* **Processed\_db**  מחלקה זו מאופיינת על ידי השדות הבאים:
  + attributes – dictionary המחזיק את בתוכו את המאפיינים השונים של ה-database, ותחת כל מאפיין ישנם מונים המונים את מס' המופעים לכל ערך של מאפיין.
  + couples – dictionary המחזיק בתוכו את חיתוכי כל זוגות ערכי המאפיינים השונים של ה-database, ותחת כל חיתוך של זוג מאפיינים, ישנם מונים המונים את מס' המופעים לכל חיתוך שני ערכים של 2 המאפיינים.
  + trios – dictionary המחזיק בתוכו את חיתוכי כל שלישיות ערכי המאפיינים השונים של ה-database, ותחת כל חיתוך של שלישיית מאפיינים, ישנם מונים המונים את מס' המופעים לכל חיתוך שלושה ערכים של 3 המאפיינים.
  + quads – dictionary המחזיק בתוכו את חיתוכי כל רביעיות ערכי המאפיינים השונים של ה-database, ותחת כל חיתוך של רביעיית מאפיינים, ישנם מונים המונים את מס' המופעים לכל חיתוך רביעיית ערכים של 4 המאפיינים.
  + pentas – dictionary המחזיק בתוכו את חיתוכי כל חמישיות ערכי המאפיינים השונים של ה-database, ותחת כל חיתוך של חמישיית מאפיינים, ישנם מונים המונים את מס' המופעים לכל חיתוך חמישיית ערכים של 5 המאפיינים.
  + db\_name – משתנה מסוג string אשר מחזיק את שם הקובץ בו מוחזק database של המשתתף

מחלקה זו מייצגת מבנה נתונים מסוג dictionary המכיל את כל החיתוכים האפשריים בין המאפיינים של ה-database.

פונקציות עיקריות של המחלקה:

* countValues- פונק' זו מקבלת את שם ה-database אותו היא עתידה לעבד, מבצעת parsing וסורקת את האובייקטים שב-database, על מנת למנות את כמויות המופעים של כל ערך של כל מאפיין וכל חיתוך אפשרי בין ערכים של מאפיינים שונים.
* **Dec\_tree** מחלקה זו מאופיינת על ידי השדות הבאים:
* processed\_db – מופע של אותו שדה של ה-server
* root – שורש עץ ההחלטה אשר ייבנה על ידי מחלקה זו
* tree\_dict – עץ ההחלטה. מיוצג על ידי מבנה נתונים מסוג dictionary

מחלקה זו אחראית על בניית עץ ההחלטה על סמך הנתונים המאוגדים ב-processed\_db של ה-server, לאחר שה-server איגד את כלל הנתונים מכלל המשתתפים בעזרת פרוטוקול שמירת המידע. בעלת פונקציה predict אשר בהינתן אובייקט חוזה את הcategory אליו הוא משויך.

המחלקה נעזרת במחלקה נוספת, Node, אשר מגדירה צמתים בעץ ההחלטה, ותפקידה העיקרי הוא לחשב בכל שלב בבניית העץ את האנטרופיה של המאפיינים הנותרים, על מנת לבנות את העץ בצורה 'חכמה'.

פונקציות עיקריות של המחלקה:

* build\_tree – פונק' זו מקבלת פרמטר depth המייצג את עומק העץ הנוכחי ו-node אשר מייצג את שורש תת העץ כרגע. פונק' זו מחשבת בכל שלב, בהינתן המאפיינים שנותר לבחון, ועל סמך המידע הרלוונטי ב-processed\_db המוכלל, מהו המאפיין הדומיננטי ביותר אותו כדאי לבחון בשלב זה בעץ, ומפתחת את המשך בניית העץ בהתאם. פונק' זו בונה את העץ באופן רקורסיבי.
* predict – פונק' זו מקבלת inst – קיצור ל-instance, מופע של אובייקט מתוך מאגר המידע, יחד עם עץ ההחלטה, דגל parsed בוליאני, ומשתנה בשם string. הפונק', במידה והדגל הינו false, מבצעת parsing למופע המתקבל, ומייצגת אותו כמחרוזת הנשמרת במשתנה string, וכ-dictionary הנשמר במשתנה instance. לאחר מכן, פונה הפונק' למלאכת החיזוי – הפונק' עוברת על הערכים המגדירים את האובייקט, ובאופן רקורסיבי, במידה והעץ עדיין אינו יודע 'לנחש' מהי קטגוריית האובייקט, קוראת הפונק' לעצמה עם תת-העץ הרלוונטי לערכים המגדירים את האובייקט וממשיכה בתהליך זה. הפונק' תחזיר כפלט את הקטגוריה החזויה כאשר תגיע לעלה בעץ המייצג סוף מסלול בעץ הרלוונטי לערכי האובייקט עליו נרצה לבצע את החיזוי.
* **Node** תת מחלקה זו של Dec\_tree מאופיינת תחת השדות הבאים:
  + tree - אובייקט מסוג Dec\_tree
  + processed\_db - מופע של אותו שדה של ה-server
  + father – אובייקט מסוג Node, שהוא האב של צומת זה בעץ
  + depth – ערך מספרי המייצג את עומק הצומת בעץ
  + atts – מערך של המאפיינים המוגדרים בצומת זו בעץ (לדוגמא, עבור צומת מעומק 2, שהאב שלה בעץ הינו ערך של מאפיין age, ובצומת זו המאפיין הדומיננטי הינו education, המערך יהיה ["age", "education"]
  + vals – מערך של ערכי המאפיינים המוגדרים בצומת זו בעץ (כנ"ל, רק שבמערך יופיעו ערכי המאפיינים האלה הרלוונטים לתת עץ שצומת זה הוא שורשו)
  + stringed\_atts – מופע של atts כמחרוזת, כך שאיברי המערך מופרדים ע"י " & " ביניהם.
  + stringed\_vals – מופע של vals כמחרוזת, כך שאיברי המערך מופרדים ע"י "\_" ביניהם.
  + tuple – תת ה-dictionary של processed\_db, שהמפתח stringed\_atts הוא אחד מהמפתחות הראשיים שלו (לדוגמא, עבור stringed\_atts = "age & education", ה-tuple יוגדר להיות ה-dictionary שמוחזק במשתנה couples במופע של processed\_db).
  + left\_atts – מערך של המאפיינים שלא נבדקו עוד עד צומת זה בעץ. לדוגמא, עבור צומת שערך ה-stringed\_atts שלה הוא "age & education", המערך יהיה ["kids", "religious"].
  + is\_leaf – דגל בוליאני המסמל האם צומת זה הוא עלה בעץ
  + amount\_of\_object – ערך מספרי המייצג את כמות המופעים מתוך מאגר המידע המאוחר אשר חולקים vals דומים.
  + dominant – מחרוזת המייצגת את המאפיין הדומיננטי מתוך המאפיינים שנותרו ב-left\_atts עבור תת העץ שצומת זה מייצג את שורשו. ייתכן כי dominant ייצג קטגוריה, במידה וצומת זה הוא עלה.

תת מחלקה זו אחראית לעזור ל-Dec\_tree בתהליך בניית העץ. אחריות המחלקה היא לחשב את מידת האנטרופיה של המאפיינים הרלוונטים לכל שלב בבניית העץ, ולייצר צמתים בעץ כך שכל צומת שמיוצר מייצג את המאפיין הדומיננטי ביותר בתת העץ המושרש בצומת זה.

פונקציות עיקריות של המחלקה:

* calculate\_gain– כמוסבר ב"תיאור של אלגוריתם כריית המידע", פונק' זו מחשבת את הערך E(T)-E(T|A) לכל מאפיין A שנותר לחשבו בתת העץ שצומת זה הוא שורשו, ומחזירה את המאפיין הזה. במידה וצומת זה אמור לייצג עלה בעץ (בין אם כל האובייקטים הרלוונטים לצומת זה חולקים את אותה קטגוריה, ובין אם עומק הצומת הוא ככמות המאפיינים וכל שנותר הוא לחלק את האובייקטים הרלוונטים לצומת זה לקטגוריה חזויה אחת), תחזיר הפונק' צומת עלה המייצג את הקטגוריה הדומיננטית ביותר – ז"א הקטגוריה בעלת כמות המופעים המקס' מתוך כלל האובייקטים הרלוונטים לצומת זה.

את החישובים הנעשים בפונק' זו, מבצעת הפונק' בעזרת 2 פונק' עזר – calculate\_ET, calculate\_ETA, פונק' המחשבות את הערכים E(T), ו-E(T|A) כמוסבר ב"תיאור של אלגוריתם כריית המידע".

* **GUI** מחלקה זו מאופיינת על ידי השדות הבאים:
  + tree\_dec- אובייקט מסוג Dec\_tree
  + container – אובייקט מטיפוס Frame.
  + frames-רשימה של אובייקטים מטיפוס Frame

מחלקה זו אחראית לממשק הgui הנפתח בתום ריצת האלגוריתם. רשימת הframes מאותחלת להיות StartPage, Page\_statistics, Page\_user\_prediction, Page\_tree\_image.

StartPage- דף הפתיחה אותו רואים. ממנו ניתן לבחור לעבור לכל אחד מהדפים האחרים.

Page\_statistics- דף המריץ את פונקציית הpredict מהמחלקה dec\_tree על כל אחד מהאובייקטים של הdatabase test\_db.txt כפי שמתואר בתוצאות הבדיקה. הוא מציג את תוצאות אלה בדף זה.

Page\_user\_prediction- בדף זה על המשתמש להזין את גילו בשדה הage, לבחור את רמת השכלתו בeducation, לבחור כמה ילדים יש לו וכן לבחור בשדה הreligious באם מגדיר עצמו דתי או לא. לאחר מכן עליו ללחוץ על כפתור הpredict ויופיע לו חיזוי העץ ורמת הודאות בה על גבי הדף. ביכולתו גם ללחוץ על כפתור הclear לניקוי השדות או על back לחזרה לדף הראשי.

Page\_tree\_image- דף המציג את העץ אשר נוצר בתום ריצת האלגוריתם.

**תיאור מאגר הנתונים- איזה עיבוד מקדים נעשה בו עבור טכניקת כריית המידע הנבחרת**

[ [ftp://ftp.ics.uci.edu/pub/machine-learning-databases/cmc](ftp://ftp.ics.uci.edu/pub/machine-learning-databases/cmc%20) ] .Contraceptive Method Choiceמאגר הנתונים שבחרנו הוא

את המאגר, שהינו בעל 1473 מופעים, חילקנו ל-6 מאגרים, על מנת לדמות פיזור של המאגר בין 5 משתתפים שונים, והחלק השישי הינו החלק שבעזרתו אנחנו בוחנים את רמת החיזוי של העץ שקיבלנו.

המאגר שלנו מייצג נשים באינדונזיה המאופיינות, בין היתר, על ידי המאפיינים הבאים:

גיל, רמת החינוך, מס' הילדים של האישה, והאם היא דתיה או לא.

כמו כן, במאגר מחולקות הנשים ל-3 קטגוריות שונות – none, long, short – כאשר כל קטגוריה מייצגת את סוג אמצעי המניעה בה האישה משתמשת.

**העיבוד המקדים שבוצע על כל חלק מהמאגר, נועד לשרת 2 מטרות:**

* המטרה הראשונה היתה שמירה על פרטיות המאגרים והאובייקטים הבונים אותם:

לצורך כך, בחרנו מס' מאפיינים דומיננטיים מתוך המאפיינים הנתונים לנו, ועיבדנו את המידע הנתון לנו על סמך מאפיינים אלו בלבד.

בנוסף, שאלנו שימוש בטכניקת שמירת פרטיות נוספת, כאשר הגדרנו את ערכי המאפיינים הנבחרים מחדש, כך שצמצמנו את מספר הערכים האפשריים לכל מאפיין, והשתמשנו בערכי טווחים על פני ערכים מספריים מדויקים –

לדוג', עבור מאפיין הגיל, שבמאגר המידע המקורי קיבל ערכים מדויקים של מספרים, צמצמנו כך שהערכים האפשריים המייצגים את המאפיין הם U30, 30-40 ו-40+, כאשר U30 מייצג אובייקט שגילו מתחת ל30, 30-40 מייצג אובייקט שגילו בין 30 ל-40 ו40+ מייצג אובייקט שגילו 40 ומעלה.

* המטרה השניה היתה לעבד את המידע כך שנוכל לאגד את המאגרים בצורה יעילה ולבנות עץ החלטה על סמך המידע המוכלל המעובד.

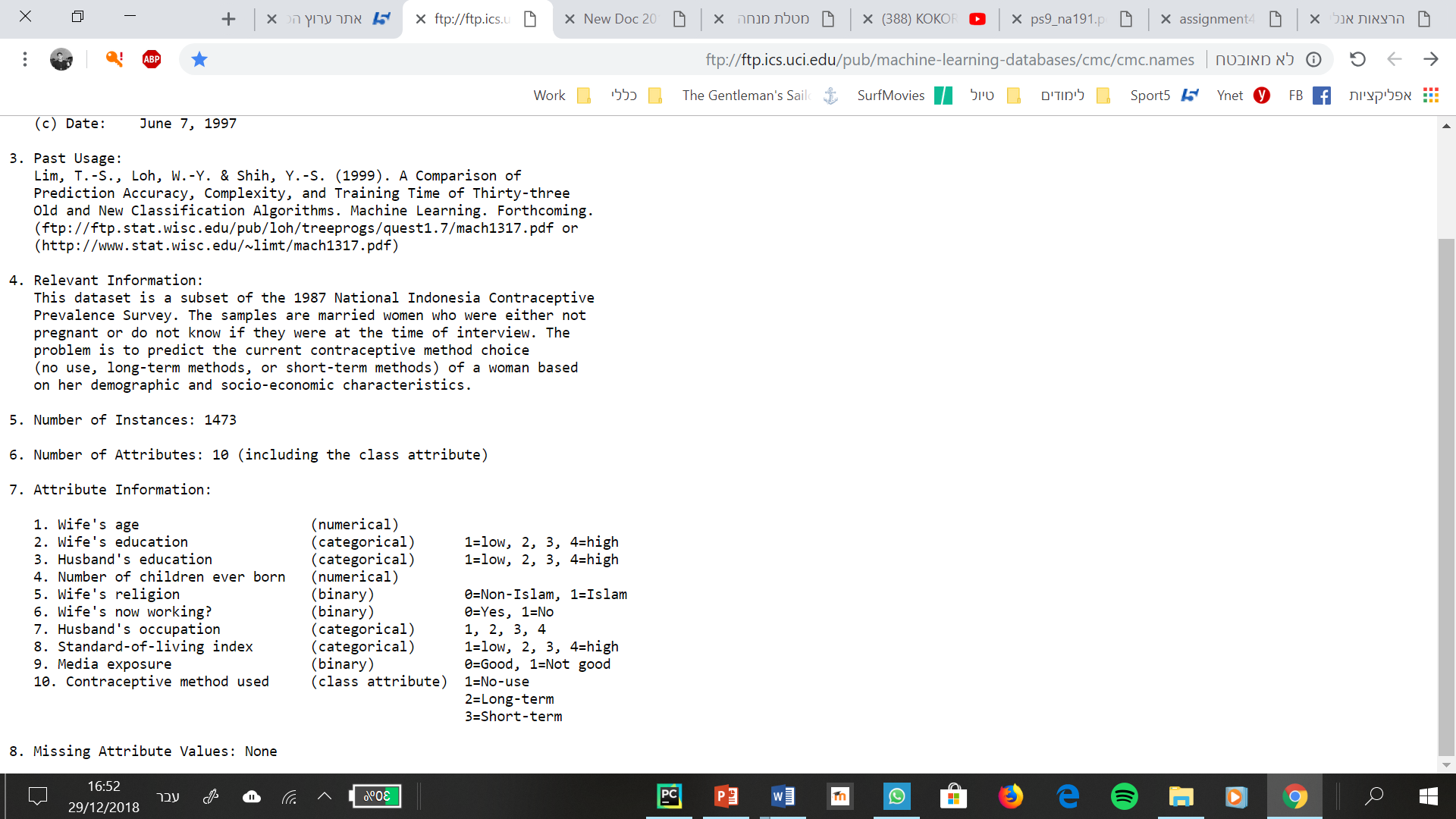
לצורך כך, הגדרנו עבור כל משתתף מבנה נתונים בו הגדרנו מונה לכל חיתוך אפשרי של ערכי המאפיינים, וביצענו עיבוד למאגר המידע הנתון ב-2 שלבים:

ראשית, ביצענו Parsing למאגר המידע על מנת שנוכל לשלוף ממנו מידע בצורה נוחה.

לאחר מכן סרקנו את מאגר המידע, ולכל אובייקט במאגר, סיווגנו את ערכיו מחדש על פי הערכים החדשים שהגדרנו, ועדכנו את המונים הרלוונטים אשר מייצגים את כמות המופעים של חיתוכי ערכיו השונים.

שיטת עיבוד זו, שבסופה מתקבל לנו מבנה נתונים המייצג כמויות של מופעים של כל חיתוך, מהווה 'אמצעי הגנה' נוסף על שמירת פרטיות האובייקטים, שכן מבנה הנתונים המוכלל שלנו מתקבל על סמך איחוד תתי-מבני הנתונים אלו שכל משתתף מחזיק, וכך ה-server שלנו לא נחשף לאף מופע של אובייקט, אלא רק לכמויות המופעים השונים.

מאגר הנתונים המקורי שקיבלנו, אופיין כך:



לאחר עיבוד המידע כמפורט לעיל, מאגר הנתונים שלנו אופיין כך:







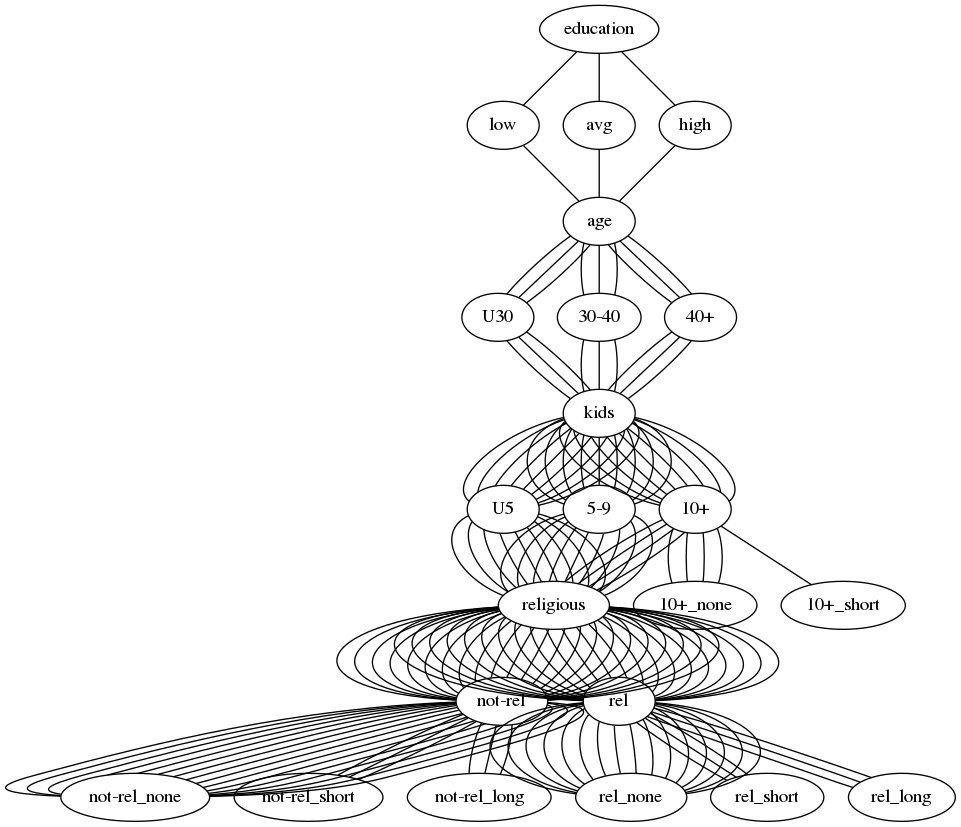


**הדפסות של ממשק המשתמש, והצגת תוצאות הבדיקה**

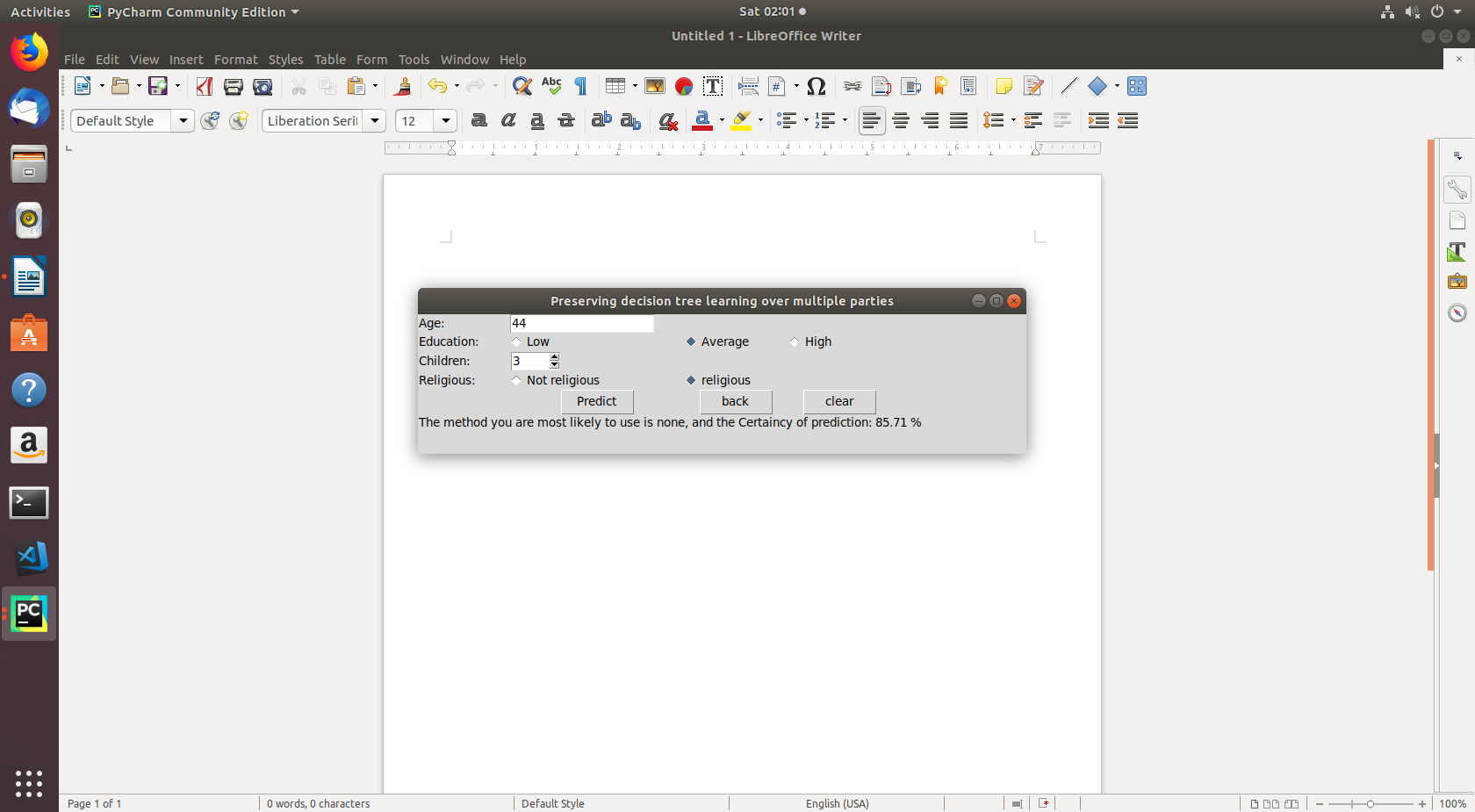
**תוצאות הבדיקה:**

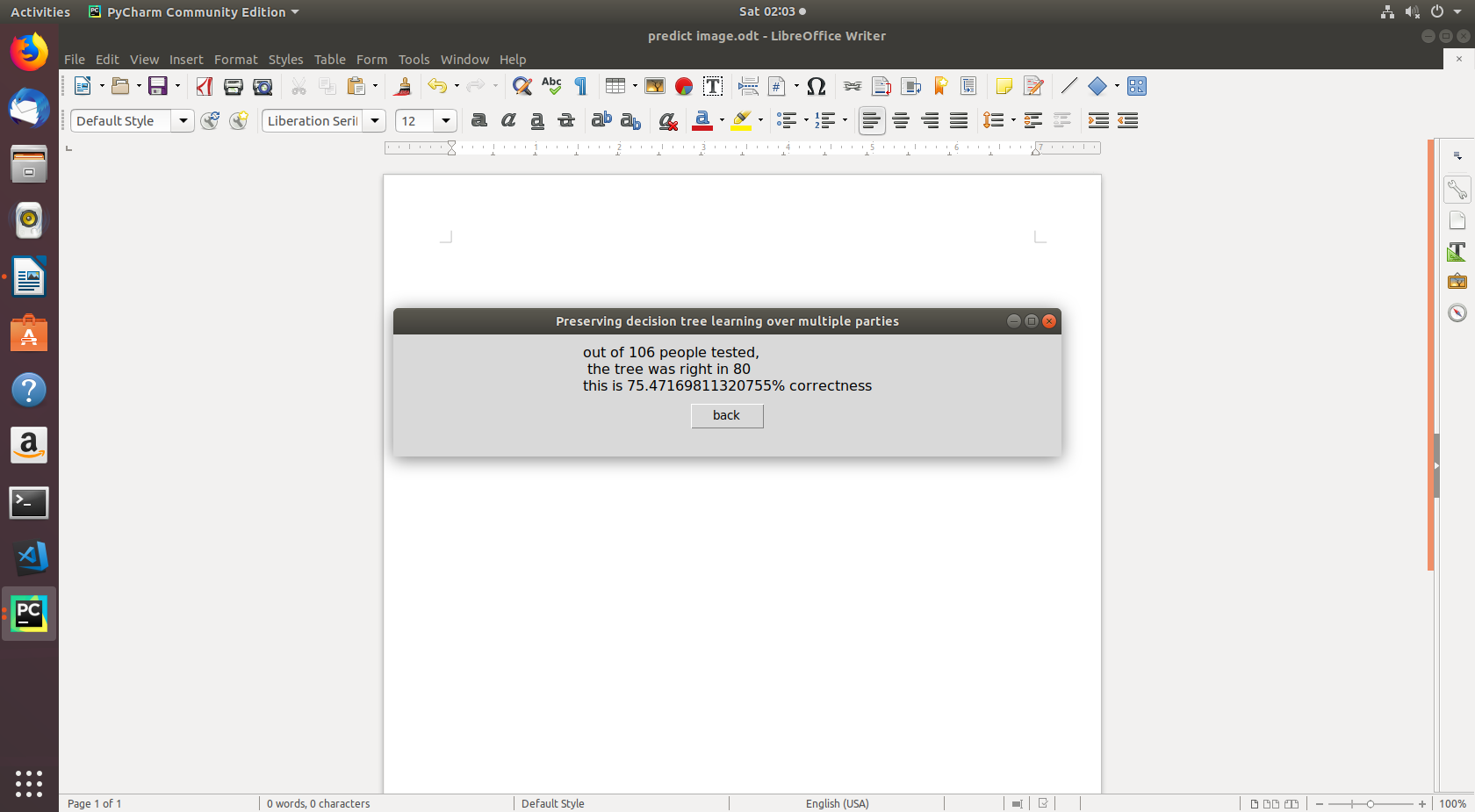
העץ אותו בנינו נבנה על סמך 1242 אובייקטים מתוך מאגר הנתונים המלא איתו עבדנו. כמו כן, מתוך המאגר הכולל, לא עשינו שימוש ב106 אובייקטים. אלו הם אובייקטים שהעץ לא למד מהם והשתמשנו בהם על מנת לבדוק את רמת ההתאמה בין החיזוי שמנפיק  לבין הקטגוריה בפועל של אובייקטים אלו. עבור כל אחד מהם ביצענו את הפונקציה שחוזה את אמצעי המניעה של האובייקט והשווינו מול האמצעי שבו משתמש האובייקט לפי מאגר הנתונים. הגענו להתאמה של 75.47% בין חיזוי העץ לתוצאות האמיתיות.

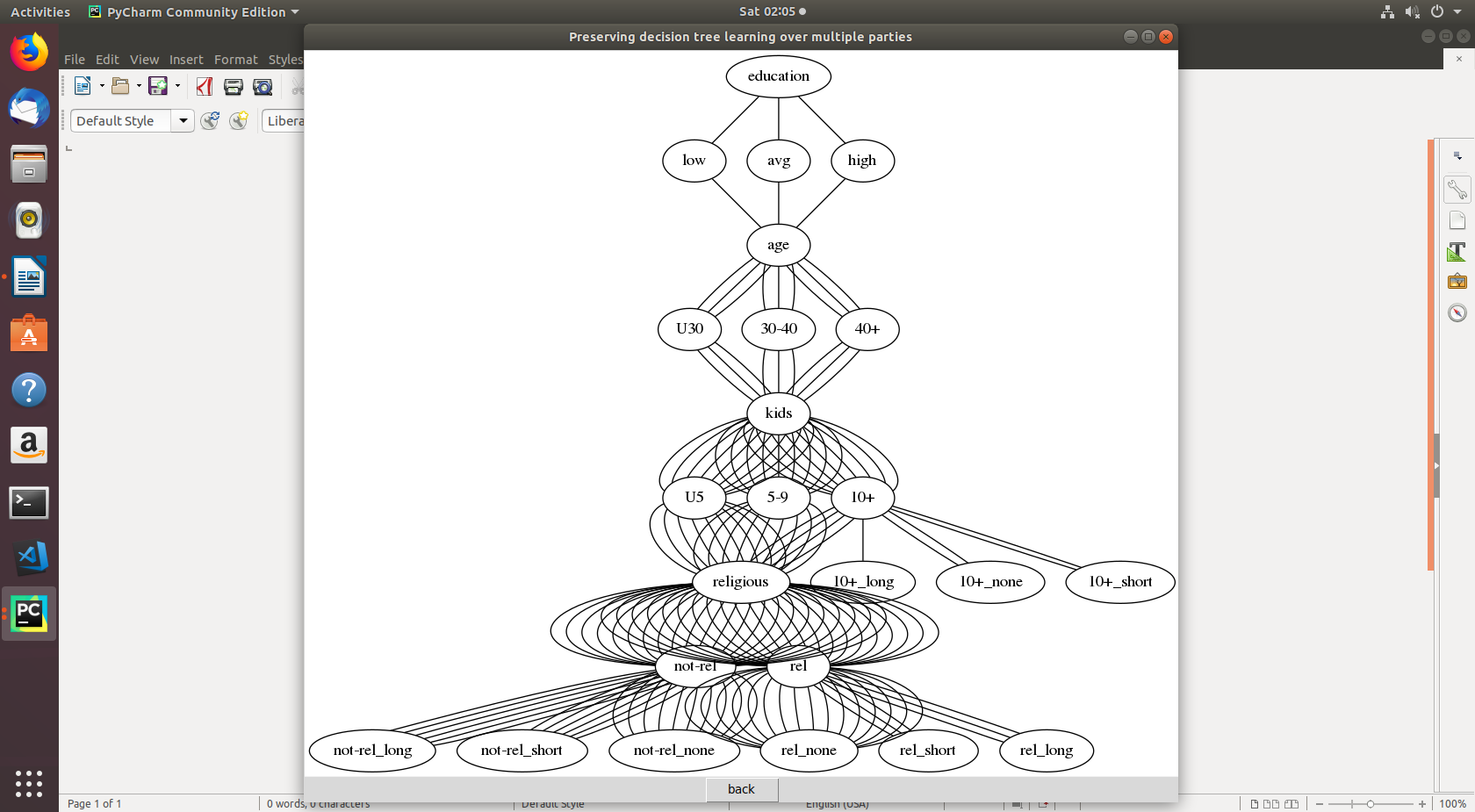
**העץ שהתקבל בתום הריצה:**

****

**הדפסות של ממשק המשתמש**







**ניתוח זמן ריצה הנדרש ושימוש בזכרון על מנת לעבד את המידע של כל Database:**

**זמן הריצה:** עבור Database בגודל n (ז"א – קיימים n אובייקטים ב-Database), נבצע לכל היותר 162 בדיקות לכל אובייקט.

**הסבר:** במקרה הגרוע, בפונקציה countValues, האובייקט תמיד יחזיק בערך האחרון של כל מאפיין, לדוגמה, אובייקט המיוצג כך:

40+\_high\_10+\_rel\_short

בשלב עיבוד המידע, אובייקט זה יעבור 3 בדיקות למציאת ערך הגיל הנכון, לאחר מכן בקינון הפנימי יעבור 3 בדיקות למציאת ערך רמת החינוך, לאחר מכן 3 בדיקות למציאת ערך כמות הילדים, לאחר מכן 2 בדיקות למציאות ערך הדת ולבסוף 3 בדיקות לבדיקת ערך הקטגוריה (שיטת המניעה)

סה"כ, עבור Pre-process זמן הריצה הינו 

**שימוש בזיכרון**: על מנת לכמת את השימוש בזיכרון, נתבונן בכמות הזיכרון הנדרש עבור ה-server, שכן ה-server יאגד תחתיו database שגודלו כגודל כל ה-databases יחד. ה-dictionary שיחזיק את כל המידע הנדרש בסיום תהלי עיבוד המידע ואיגודו, יהיה בנוי מהמפורט:

בתת ה-dictionary, "attributes", יהיו מונים לכל ערך, ישנם 5 יחידונים כך שכל יחידון מחזיק לכל היותר 3 ערכים מוצלבים שונים – סה"כ 15.

בתת ה-dictionary, "couples", יהיו מונים לכל ערך, ישנם 10 זוגות כך שכל זוג מחזיק לכל היותר 9 ערכים מוצלבים שונים – סה"כ 90.

בתת ה-dictionary, "trios", יהיו מונים לכל ערך, ישנן 10 שלישיות כך שכל שלישייה מחזיקה לכל היותר 27 ערכים מוצלבים שונים – סה"כ 270.

בתת ה-dictionary, "quads", יהיו מונים לכל ערך, ישנן 5 רביעיות כך שכל רביעייה מחזיקה לכל היותר 81 ערכים מוצלבים שונים – סה"כ 405.

בתת ה-dictionary, "pentas", יהיו מונים לכל ערך, ישנה חמישייה יחידה, המחזיקה 162 ערכים מוצלבים שונים – סה"כ 162.

בסה"כ 942=90+270+405+162+15 משתנים שונים, שמורים בתוך Dictioneries. את כמות הזיכרון הזו נכפיל ב-6, עבור ה-server וחמשת הקליינטים המשתתפים. נקבל כי עבור מאגר המידע שלנו, אשר בעל 1242 אובייקטים, נחזיק בזיכרון של 942\*5=4710 מונים שונים.