מגישים:

דן חזן 203527452

כפיר רימברג 311374185

דו"ח פרויקט – נושאים באנליזה של מידע:

שאלה 1:

בגרסה השנייה של מימוש האלגוריתם השתמשנו בקבוצת תנאים גנריים, הניתנים לכיול על ידי קונפיגורציה.
תחילה יצרנו את המנגנון המאפשר לנו לייצר את אותם התנאים, ולבסוף בחרנו את הקונפיגורציה שהניבה את השיפור הטוב ביותר בתוצאות החיזוי של האלגוריתם.

המנגנון עליו מתבססים התנאים ממומש באופן הבא:

תחילה כל תמונה מחולקת ליחידות (Units) נפרדות לפי 3 קטגוריות:

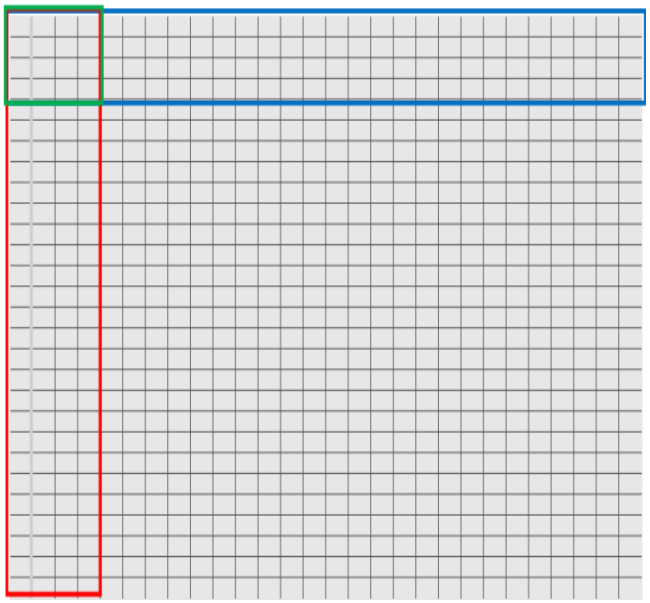
* שורות – חלוקת מטריצת הפיקסלים ל"שכבות" לרוחב, כשכל שכבה מכילה N שורות פיקסלים.

* עמודות – חלוקת מטריצת הפיקסלים ל"שכבות" לאורך, כשכל שכבה מכילה N עמודות פיקסלים.

* בלוקים – חלוקה לבלוקים בתוך המטריצה, כאשר כל בלוק הוא ריבוע בגודל NxN פיקסלים.

N יכול לנוע בין 1 ל-28, אבל חייב להיות מחלק של 28 (כלומר 2, 4, 7, 14 ו-28) כדי שהחלוקות לשורות/עמודות/בלוקים יהיו שלמות (אין חפיפה בין היחידות – זה היה מוסיף משמעותית יותר שאלות ומגביל את הרזולוציה).

למשל – עבור N=4, נקבל 7 שכבות של שורות/עמודות המכילות 4 שורות/עמודות פיקסלים כל אחת, ו- $7^2 = 49$ בלוקים המכילים $4x4 = 16$ פיקסלים כל אחד.



לאחר החלוקה, השאלות נוצרות כפונקציה של היחידות כפול "רמות השחור":

לכל קטגוריה מוגדר משתנה המייצג את מספר "רמות השחור" L.

לכל יחידה, מחושב הסכום המקסימאלי M בהינתן וכלל הפיקסלים בתוכה בעלי ערך 255 (שחור לגמרי).

כעת לכל יחידה X נוצרות L שאלות בצורה:

האם סכום ערכי הפיקסלים שבה קטן או שווה לV, כאשר V הוא ערך בין 0 לM, בקפיצות של M/L.

כלומר:

$$Q_{x_i} = does \left\{Sum(X) \leq \frac{M}{L} \cdot i\right\}, i = 0,1, \dots, L-1$$

כאשר Q_x היא קבוצת השאלות הנוצרות עבור היחידה X.

לכן נוצרות בסך הכל $Q_{category} = L \cdot Units$ שאלות עבור כל קטגוריה, כיוון ש L יכול להיות שונה בכל אחד, וקבוצת השאלות הסופית היא שרשור של שלוש קבוצות השאלות הללו $C = Q_1 + Q_2 + Q_3$.

הבחירה במנגנון חלוקת התנאים המפורט לעיל נעשתה משיקולי:

כלליות – החלוקה לשורות/עמודות/בלוקים היא רלוונטית לכל סוגי הספרות, ובפרט גם לאלו המערביות, וגם לאלו שמופיעות ב- *Kannada-MNIST*.

כיול – ניתן בשינוי קונפיגורציה פשוט לבצע שינוי משמעותי בקבוצת התנאים, מה שמאפשר הרצת ניסויים והגעה לפרמטרים שמספקים את התוצאות הטובות ביותר בדטה סט מסוים.

ומתוך סברה שהרחבת היחידות הבסיסיות מפיקסלים (של גרסה 1) ורמת שחור בודדת (<128) תספק קבוצת תנאים המשפרת את זו שבגרסה 1.

בחירת התנאים עצמם נעשתה מתוך הרצת מגוון רחב של ניסויים על קונפיגורציות שונות, ובחירת זו האופטימלית.

כאן נעשה ניסיון לשפר את התוצאות של האלגוריתם על הדטה סט של *MNIST*, כאשר שיפור זה הוא מדד לעליונות הקונפיגורציה. עוד יורחב בשאלה 2, רק נציין שהקונפיגורציה הסופית שנבחרה כללה:

שורות: $L = 15$ $N = 14$

עמודות: $L = 15$ $N = 14$

בלוקים: $L = 1$ $N = 28$

L = מספר "רמות השחור".

N = לכמה שורות/עמודות/בלוקים לחלק את מטריצת הפיקסלים.

כפי שניתן לראות, משיקולים פרקטיים נבדקו *N* ו-*L* זהים עבור שורות ועמודות בכל הרצה.

בנוסף דבר מעניין שהתגלה הוא שמספק הבלוקים האופטימלי שהתקבל הוא **28**, כלומר **כל בלוק הוא פיקסל!**

בנוסף **הפרמטר *L* האופטימלי עבור הבלוקים הוא 1!** כלומר בדיקה של האם הפיקסל בעל הערך 0 או גדול מ-0. זאת לא רק משיקולי זמן ריצה, אלא גם מאופטימליות החיזוי. בכך קיבלנו כי גרסה 2 היא מעין הרחבה של גרסה 1, למרות שהמנגנון בו נבחרו התנאים לא כלל במפורש תנאים המתייחסים לפיקסלים בנפרד.

<u>שאלה 2:</u>

טרם העמקה ב-2 הגרסאות, אתגר משמעותי ראשון שנתקלנו בו היה שפת התכנות בה נממש את הפרויקט. תחילה עבדנו על הפרויקט ב-*python* ולאחר ניסיונות רבים בהם ניסינו לשפר את זמני הריצה של התוכנית כדי שיעמדו בתנאים הדרושים (כולל שימוש מאסיבי ב-*numpy* ועבודה עם המידע כ-*numpy arrays*), הגענו למבוי סתום וזמן הריצה עדיין לא עמד בתנאים.

לכן שכתבנו את הפרויקט ב-*Java*, תוך שמירה על האופטימיזציה המטריצה שנעשתה קודם לכן (ומפורטת מטה).

אופן מימוש הגרסאות:

בשני האלגוריתמים, דבר ראשון נטען הטסה סט לתוך מערך דו מימדי מסוג *int*.

לאחר מכן, נעשית המרה של המערך הנ"ל למעין "מטריצת תשובות", כלומר כל עמודה במטריצה מייצגת שאלה ממאגר השאלות (מאגר התנאים), ובכל משבצת נמצאת התשובה הבינארית (0 או 1) לשאלה על הדוגמה שבאותה השורה. זאת למעט עמודה 0, שבה נשאר ה-*label* של אותה שדוגמה.

label	q1	q2	q3	...	q362	q363	q364	q365
4	0	0	0		0	1	1	0
6	0	0	0		0	0	0	0
8	0	0	0		1	1	0	1
2	0	0	0		0	0	0	0
0	0	0	0		0	0	1	0
4	0	0	0		0	1	1	1
1	0	0	0		0	0	0	0
7	0	0	0		0	0	1	1
6	0	0	0		0	0	0	0
4	0	0	0		1	1	0	0
9	0	0	0		1	1	1	1
5	0	0	0		0	1	0	0
2	0	0	0		0	0	1	0

בנוסף המערך הוא מערך סטטי של *java*, וכך נשמרת יעילות מקסימלית בגישה אליו. כלומר בכל גישה לתשובה על שאלה על דוגמה מסוימת, הסיבוכיות היא *O(1)* בלבד.

אופטימיזציה זו נעשה כצעד מקדים לשיפור זמני הריצה ככל שניתן.

ההמרה לפי גרסאות:

גרסה 1 – ההמרה נעשית *in place*, כאשר כל עמודה שייצגה פיקס ספציפי, הופכת להיות עמודה המייצגת את התשובה על הפיקסל הזה. זאת כיוון שבגרסה 1 יש בדיוק 784 שאלות – שאלה לכל פיקסל בתמונה.

$$answers[i][j+1] = \begin{cases} 1\,, Image[i][j] > 128 \\ 0\,, Image\ [i][j] \leq 128 \end{cases}$$

גרסה 2 – כפי שפורט בשאלה 1, בגרסה זו מנגנון יצירת השאלות הוא סבוך יותר ומבוסס קונפיגורציה.

לכן בעת יצירת מטריצת התשובות, נעשה מעבר על שורות מטריצת הקלט, המכילות את הדוגמאות עם ערכי הפיקסלים שלה, ולכל דוגמה:

הפיקסלים נשמרים במטריצה דו מימדית לשם נוחות החלוקה ליחידות.

נבנה מערך מאוחד של התשובות על החלוקה השורות + עמודות + בלוקים, והוא נשמר במקום השורה המקורית במערך הקלט. התשובות מוחזרות מפונקציות מופרדות, המקבלות גם את הפרמטרים בקונפיגורציה ומשתמשים במטריצת הפיקסלים שנוצרה.

בסוף התהליך המערך המקורי מכיל שורות כמספר הדוגמאות, אך מספר העמודות הוא כמספר השאלות שיוצרו – שתלוי בקונפיגורציה שהועברה.

אלגוריתם הלמידה – לאחר יצירת מטריצת התשובות למדגם האימון, אלגוריתם בניית העץ רץ באופן זהה כתלות באותה המטריצה, כלומר ללא הבחנה בין הגרסאות.

נעשה שימוש במבנה נתונים של עץ בינארי, שמומש על ידנו ונקרא *DecisionTree*, ובו נשמר גודלו (T), גרסת השאלות שבו, מטריצת השאלות הכוללת ושורש העץ.

הוא מכיל 2 סוגי חוליות:

InternalNode- צומת פנימי, המכיל מזהה שאלה, בן ימני ובן שמאלי.

LeafNode- עלה, המכיל ספרה חזויה/הספרה הנפוצה ביותר.

שתי המחלקות יורשות ממחלקה אבסטרקטית *Node*, ששומרת את המידע הבא:

- אובייקט העץ המקורי, בעיקר לשם גישה למטריצת התשובות.
- חוליית ההורה.
- רשימת האינדקסים של הדוגמאות שהגיעו לחוליה הנוכחית. כלומר האינדקסים במטריצת התשובות. כך מועבר מידע מינימאלי בכל איטרציה של האלגוריתם לגבי איזה דוגמאות הגיעו עד הלום, וניתן לקבל את המידע שלהן מתוך המטריצה שנשמרה באובייקט העץ.

בנוסף, לשם עבודה נוחה עם מציאת *Information Gain* מקסימלי, יצרנו מחלקת עזר בשם *IGStruct*, המכילה עלה מסוים, יחד עם ערך *Information Gain* שנמצא עבור הצבת שאלה מסוימת באותו עלה, ומזהה השאלה.

ריצת האלגוריתם:

- מטריצת התשובות, שכאמור מכילה מבעוד מועד את הדוגמאות עם התשובות לשאלות שבגרסה הרלוונטית, מחולקת למדגם ולידציה ומדגם אימון באופן רנדומלי (כמערכי אינדקסים לשם יעילות).
- מאותחל עץ עם מטריצת התשובות המלאה, אך השורש שלו מקבל את האינדקסים של מדגם האימון בלבד.
- נעשה שימוש ברשימה השומרת את העלים שבעץ בכל שלב בתור אובייקט *IGStruct* , כלומר כבר מחושב עבורם ה *Information Gain* המקסימלי והשאלה האופטימלית.
- בכל איטרציה נבחר זה המקסימלי, העלה מוחלף בצומת פנימית, והעלים החדשים מתווספים לרשימה.
- הריצה מסתיימת לאחר 2^L איטרציות, כאשר בכל פעם שגודל העץ הוא חזקה של 2, נשמר עותק של העץ, תוך הקפדה להשיל ממנו את כל המידע ה"כבד" שדרוש אך ורק לתהליך הבנייה.
- בסוף הריצה נבדקים כל עותקי העץ ע"י מדגם הולידציה, ונבחר זה האופטימלי.
- נבנה מחדש עץ בגודל הנבחר על כלל הדטה סט.

אלגוריתם החיזוי – גם הוא מקבל מטריצת תשובות, הפעם למדגם הבדיקה (לפי הגרסה שנשמרה בעץ). הוא מבצע הרצה של כל אחת מהדוגמאות בעץ, ושומר את החיזוי במערך.

לבסוף האלגוריתם מדפיס את אותו המערך.

אופן בחירת הקונפיגורציה האידאלית לגרסה 2:

אתגר מרכזי שנתקלנו בו הוא בחירת הקונפיגורציה האידיאלית לשאלות מגרסה 2, הרי בנינו את המנגנון כך שנוכל לבצע ניסויים.

לשם כך יצרנו סקריפט המריץ את האלגוריתם על ערכים קונפיגורציה במרווחים שונים.

כיוון שהחלוקה לבלוקים מניסה מספר גדול בהרבה של שאלות מאשר אלו של השורות/עמודות, וכמו כן ישנו מספר קטן אפשרי של חלוקה לבלוקים, ביצענו את הניסויים כאשר קבענו את קונפיגורציית הבלוקים כקבועה לכל אורך הריצה. כך בודדנו את מציאת הקונפיגורציה של הבלוקים, והקטנו את זמן הריצה של הניסויים, שהיה גבוה גם כך.

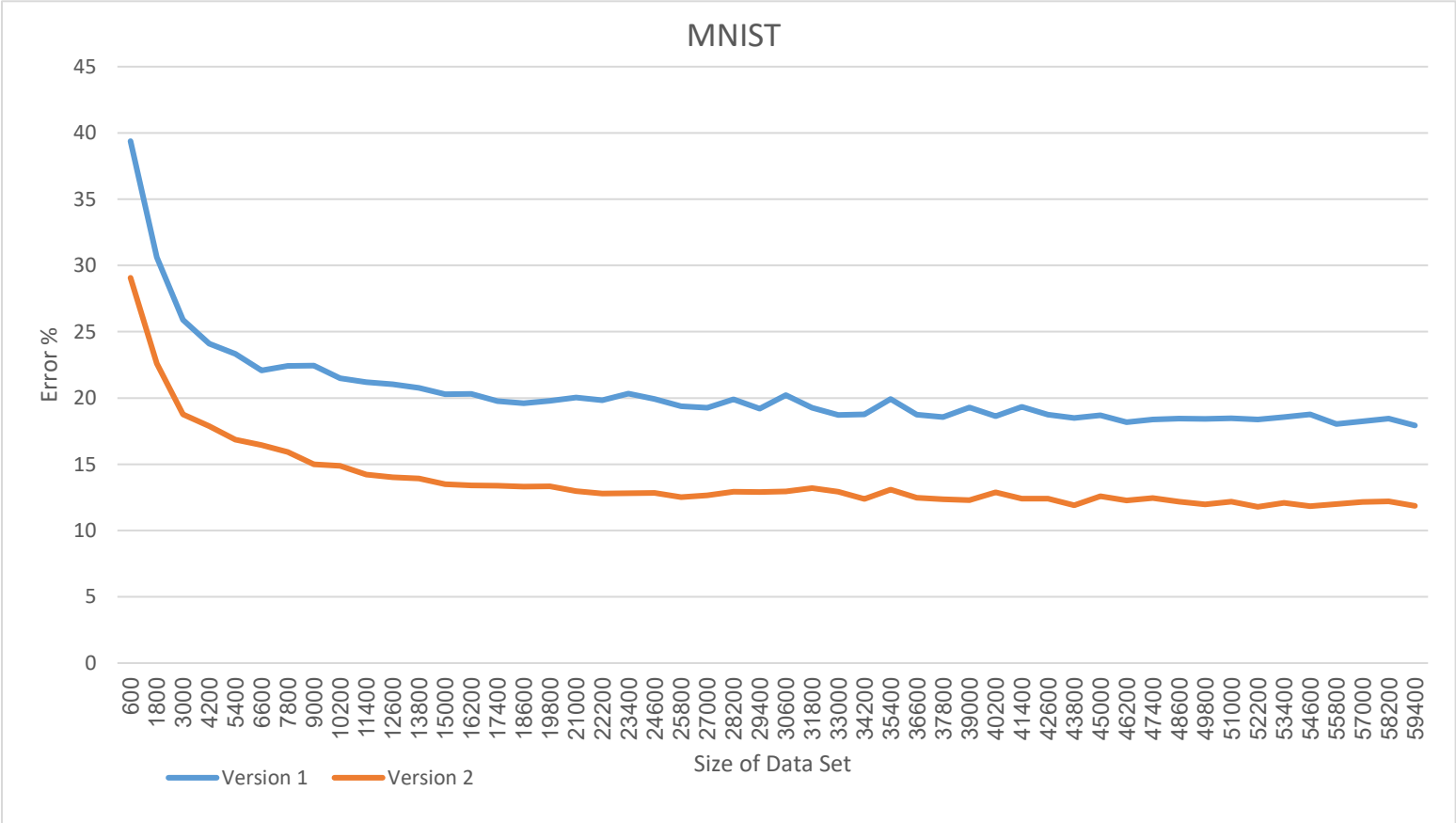
בניסויים אלו הבחנו כי הקטנת גדלי הבלוקים הובילה לשיפור משמעותי של ביצועי החיזוי. עד כדי כך שהבלוק האידיאלי היה בגודל $1X1$ (כלומר פיקסל). ניתן לראות בתמונה הבאה חלק מהרצת ניסוי, בה נבדקים חלוקה למספר בלוקים שונה (משמאל) יחד עם מספר רמות השחור (באמצע) ואחוזי ההצלה בחיזוי (מימין).

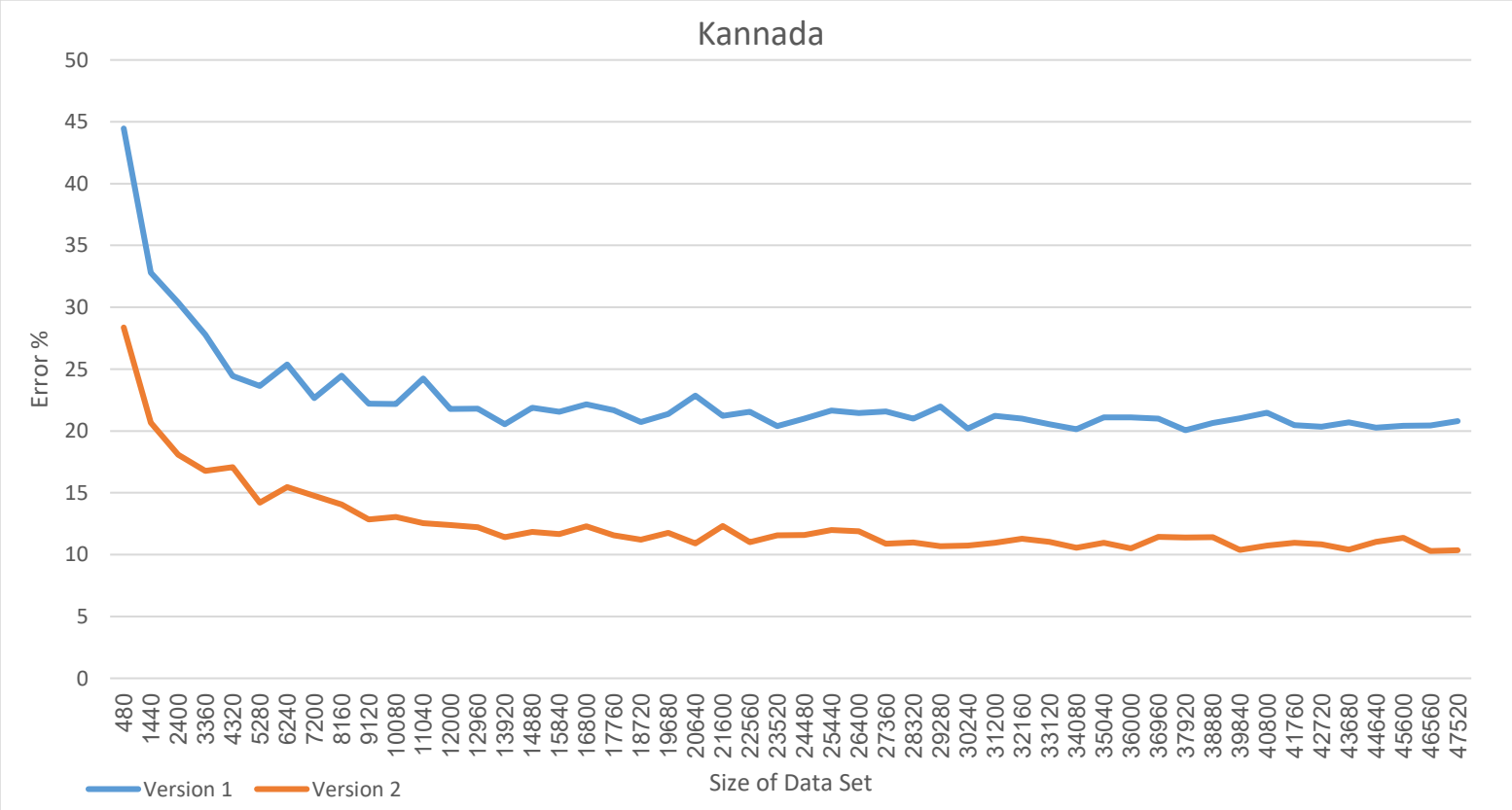
```
[14.0, 7.0, 0.8759]
[14.0, 10.0, 0.8667]
[14.0, 13.0, 0.8726]
[14.0, 16.0, 0.8743]
[14.0, 19.0, 0.8748]
[14.0, 22.0, 0.8767]
[14.0, 25.0, 0.8725]
[14.0, 28.0, 0.8752]
[28.0, 1.0, 0.8713]
[28.0, 4.0, 0.8693]
[28.0, 7.0, 0.8742]
[28.0, 10.0, 0.8732]
[28.0, 13.0, 0.878]
[28.0, 16.0, 0.8763]
[28.0, 19.0, 0.8798]
[28.0, 22.0, 0.8742]
[28.0, 25.0, 0.8797]
[28.0, 28.0, 0.8759]
Best:
[28.0, 19.0, 0.8798]
```

כמו כן ניתן לראות כי הגדלת מספר רמות השחור לא גדלה באופן משמעותי במקרה של בלוקים, ומשיקולי זמן ריצה היה מתבקש לבחור את החלוקה הגדולה ליותר (חלוקה לפיקסלים) יחד עם של שחור (כלומר שאלה של האם הערך > 0).

שאלה 3 – A:

מתחת מוצגים שני גרפים, המתארים את אחוז השגיאה המתקבל מהרצת האלגוריתם כתלות בגודל המדגם, כאשר הדוגמאות נבחרו באופן אקראי. (הפרמטרים נבחרו – גרסה 2, L=8, P=20)





כאשר ביצענו את הניסויים, הרצנו סה"כ 3 פעמים כל גודל מדגם (מחולק באופן אקראי) ובכלל התוצאות קיבלנו כמעט במדויק (סטייה של *~0.01%*) את אותם אחוזים, כלומר התוצאות לא השתנו באופן משמעותי מניסוי לניסוי על אותו גודל מדגם.

<u>שאלה 3 – B:</u>

<u>עבור דטה סט *Mnist*:</u>

נשים לב כי ב-2 הגרסאות נעשה שיפור חד באחוזי השגיאה בהרצות הראשונות (עד כ-7% מגודל מדגם האימון) עד התייצבות יחסית של אחוזי השגיאה. לאחר חציית סף 7% לערך, אחוזי השגיאה אמנם ממשיכים לרדת אך מדובר בירידה יחסית קבועה ומזערית ככל שגודל מדגם האימון עולה.

בנוסף, עולה מהגרף כי יחס אחוזי השגיאות בין גרסה 1 לגרסה 2 נשמר וההבדל המרכזי הוא באחוזי השגיאה בהתחלה (39.38% בגרסה 1 אל מול 29.07% בגרסה 2) ובסוף (17.93% אל מול 11.86%).

אך התנהגות הגרפים זהה וב-2 המקרים יש ירידה חדה באחוזים עד להתייצבות כאשר ניתן לחלק את הקטעים לבדיקות גודל מדגם שהוא עד 7% מגודל המדגם ומ-7% ועד לכולו.

<u>עבור דטה סט *Kannada*:</u>

כמו בדטה סט *Mnist* ניתן לשים לב כי יש שיפור חד באחוזי השגיאה בהרצות הראשונות (עד כ-9% מגודל מדגם האימון) ועד להתייצבות יחסית של אחוזי השגיאה.

יחסי השגיאות בין הגרסאות נשמר גם במקרה זה אך בשונה מ-*Mnist* ניתן להבחין כי הירידה באחוזי השגיאה בגרסה 1 של *Kannada* חדה משמעותית מהירידה באחוזי השגיאה בגרסה 2 של הדטה סט כאשר גודל מדגם האימון הוא עד 9% מגודלו המלא.

בנוסף, כמו בדטה סט *Mnist*, גם ב-*Kannada* בא לידי ביטוי ההפרש בין אחוזי השגיאה בהתחלה (44.46% אל מול 28.37%) ובסוף (20.79% אל מול 10.35%).

ההתנהגות באופן כללי אכן דומה ב-2 הדטה סטים, אך ההבדל המשמעותי הוא בירידה החדה יותר בין גרסה 1 ל-2 ב-*Kannada* כאשר גודל מדגם האימון הוא עד 9%.

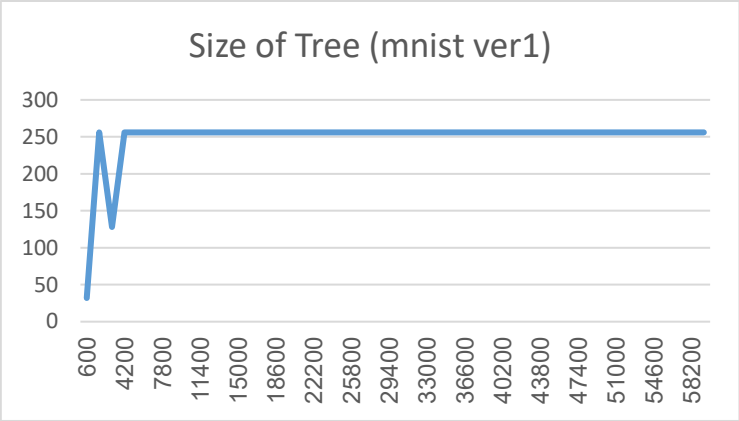
בנוסף, נשים לב כי השיפור בשתי הגרסאות, בשני הדטה סטים יחסית זהה, אך בגרסה 1, ההצלחה על *Mnist* היא גדולה יותר לכל אורך הדרך לעומת *Kannada*.

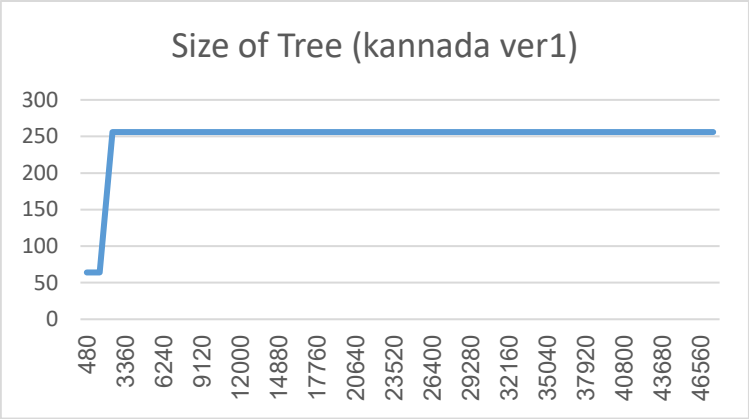
<u>שאלה 3 – C:</u>

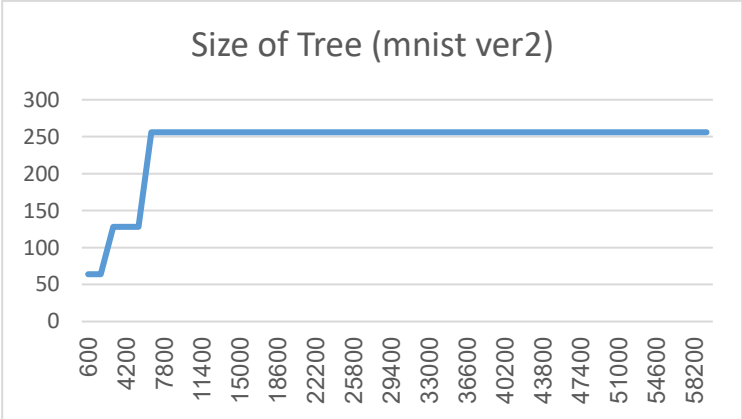
ההבדלים באחוזי השגיאה הן תוצר ישיר של מאגר השאלות שנבחנו במהלך האלגוריתם.

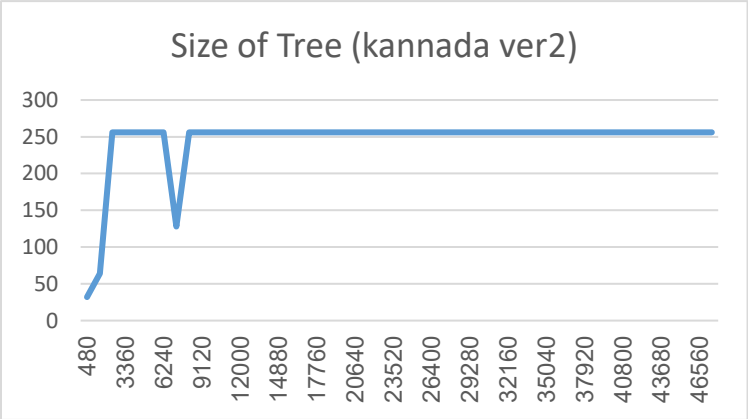
מכיוון שבגרסה 2 בחרנו להרחיב את מאגר השאלות של גרסה 1 באופן שמשפר את אחוזי ההצלחה על הדטה סט *Mnist*, דבר שנעשה תוך שימוש בשאלות כלליות ולא ספציפיות מידי באופיין, אותו שיפור נראה גם בדטה סט *Kannada* וההתנהגות באחוזי השגיאה הייתה דומה לגדלי מדגם האימון השונים.

שאלה 4 – A:









שאלה 4 – B:

קודם כל ניתן להבחין בהבדל היחידי שקיים בין הדטה סטים – ב-2 הגרסאות אכן גודל העץ מתקבע על T=256, בmnist כאשר גודל המדגם חוצה את סף 11.5%~ ואילו בkannada כאשר גודל המדגם חוצה את סף 6.5%~.

בנוסף, נשים לב כי mnist גרסה 2 וkannada גרסה 1 מציגים אותה מגמה, כלומר מונוטונית עולה עד להגעה לT=256 והמשך קבוע, ואילו בmnist גרסה 1 וkannada גרסה 2 גם הם מציגים אותה מגמה, אך לא מונוטונית עולה שכן לאחר הגעה לT=256, 2 הפונקציות יורדות חזרה ל-128 ולאחר מכן חזרה ל-256 והמשך קבוע.

שאלה 4 – C:

ככל שמדגם האימון קטן יותר, בניית עץ גדול מגדילה את ספציפיות העץ לדוגמאות שבמדגם. החל מסף מסוים העץ לא ספציפי מידי ולכן משפר את החיזוי.

ניתן לראות שב-Mnist הסף הזה מופיע בשלב מאוחר יותר, וזאת ככל הנראה כיוון שספרות בכתב מערבי הן פשוטות יותר (צורות שלמות) ולכן עבור מדגם קטן יותר, נגיע מהר יותר לשלב של Overfitting כאשר העץ גדול.

בספרות של Kannada לעומת זאת, שהן יותר מסובכות, ויש שוני מהותי יותר בין דוגמאות שונות של אותה הספרה, גם מדגמים קטנים יחסית יאפשרו לעצים גדולים להניב חיזוי טוב ולא ספציפי אליהם.

שאלה 5 – A:

(1) סיווג ספרות kannada כmnist: היינו מצפים לתוצאות נמוכות משמעותית מהתוצאות שהוצגו בגרפים לעיל, כאשר מדגם האימון ומדגם הבדיקה משתייכים לאותו דטה סט. חלוקה ל-3 קטגוריות (סיווג גבוה, סיווג נמוך ואקראי):

<u>ספרות אשר יסווגו נכון באחוז גבוה:</u>

0 – סיכוי גבוה לסיווג נכון שכן צורת הספרה 0 בכתיב מערבי וכתב Kannada זהה.

<u>ספרות אשר יסווגו כספרה אחרת:</u>

3 – תסווג כספרה 2 באחוז גבוה.

4 – תסווג כספרה 8 באחוז גבוה.

7 – תסווג כספרה 2 באחוז גבוה.

<u>ספרות אשר יסווגו באופן אקראי:</u>

1, 2, 5, 6, 8, 9.

(2) סיווג ספרות mnist כkannada: כמו סעיף (1), היינו מצפים לתוצאות נמוכות מהתוצאות שהוצגו בגרפים לעיל, אם כי במקרה ההפוך מסעיף (1) התוצאות עשויות להיות מעט גבוהות יותר מהתוצאות בסעיף (1). חלוקה ל-3 קטגוריות (סיווג גבוה, סיווג נמוך ואקראי):

<u>ספרות אשר יסווגו נכון באחוז גבוה:</u>

0 – סיכוי גבוה לסיווג נכון שכן צורת הספרה 0 בכתיב מערבי וכתב Kannada זהה.

<u>ספרות אשר יסווגו כספרה אחרת:</u>

2 – תסווג כספרה 3 או 7 באחוז גבוה.

8 – תסווג כספרה 4 באחוז גבוה.

<u>ספרות אשר יסווגו באופן אקראי:</u>

1, 2, 5, 6, 8, 9.

שאלה 5 – B:

מדגם אימון – mnist, מדגם בדיקה – kannada, גרסה 2.

כאשר השורות הן ספרות kannada שנבדקו, והעמודות הן ספרות mnist שנחזו:

	0	1	2	3	4	5	6	7	8	9
0	0.635	0.048	0.035	0.04	0.037	0.11	0.015	0.057	0.006	0.021
1	0.078	0.065	0.026	0.006	0.068	0.117	0	0.419	0.005	0.22
2	0.018	0.17	0.409	0.158	0.046	0.075	0.033	0.085	0.005	0.006
3	0.014	0.327	0.424	0.043	0.052	0.03	0.06	0.035	0.013	0.006
4	0.001	0.21	0.042	0.042	0.422	0.112	0.001	0.082	0.05	0.042
5	0.012	0.108	0.219	0.041	0.381	0.03	0.012	0.102	0.093	0.006
6	0.006	0.157	0.037	0.07	0.004	0.165	0.557	0.002	0.005	0
7	0.009	0.371	0.28	0.069	0.005	0.074	0.167	0.01	0.018	0
8	0.326	0.013	0.035	0.015	0.11	0.195	0.005	0.295	0.005	0.006
9	0.039	0.09	0.014	0.021	0.014	0.436	0.025	0.239	0.075	0.051

מדגם אימון – kannada, מדגם בדיקה – mnist, גרסה 2

כאשר השורות הן ספרות mnist שנבדקו, והעמודות הן ספרות kannada שנחזו:

	0	1	2	3	4	5	6	7	8	9
0	0.394	0.011	0.191	0.02	0	0.065	0.009	0.003	0.312	0
1	0.008	0.003	0.008	0.34	0.267	0.23	0.002	0.11	0.002	0.036
2	0.04	0.008	0.273	0.179	0.017	0.186	0.026	0.124	0.147	0.006
3	0.318	0.037	0.298	0.105	0.093	0.109	0.006	0.006	0.024	0.007
4	0.019	0.015	0.024	0.061	0.309	0.455	0	0.01	0.099	0.014
5	0.173	0.02	0.101	0.06	0.053	0.048	0.016	0.07	0.23	0.234
6	0.05	0.022	0.037	0.415	0.017	0.251	0.026	0.075	0.098	0.015
7	0.024	0.078	0.019	0.047	0.265	0.491	0	0.005	0.056	0.019
8	0.098	0	0.026	0.095	0.163	0.3	0.003	0.007	0.257	0.056
9	0.058	0.033	0.015	0.152	0.377	0.322	0	0.001	0.032	0.013

מסקנות:

- הספרה 0 אכן סווגה נכון באחוזים גבוהים יחסית בבדיקת דוגמאות מ kannada על עץ של mnist, אך במקרה ההפוך החיזוי היה נמוך משמעותית, וקרוב לסיווג כ 8.
  הדבר הגיוני כיוון שבספרות מערביות אין ספרה נוספת מלבד 0 שנראית כמו עיגול, אך בספרות kannada, 8 קרובה מספיק, ואף בבדיקת שורות (של גרסה 2), צורה 8 תתאים לשאלות המסווגות 0 בשל "כמות השחור" שדומה בחלק העליון של הספרה, על אף שהעיגול אינו נסגר.

עבור kannada:

- הספרה 3 של kannada אכן סווגה לרוב כ 2, אך גם סווגה כ-1 אחוז לא מבוטל מהפעמים.
- הספרה 4 של kannada לא סווגה כ-8 כמעט כלל, אלא סווגה כ-4 רוב הפעמים!
- הספרה 7 של kannada אכן סווגה כ2 באחוזים משמעותיים, אך סווגה יותר כ-1. מסתמן כי 1 מזוהה כ-2 לא מעט בעץ שנוצר. אולי בגלל קל לסווג אותה לפי הקו הדומיננטי באמצע הצורה, שלפעמים גם מוטה באלכסון.
- בנוסף נראה כי הספרה 6 של kannada סווגה נכון באחוז גבוה, מה שהגיוני כיוון שיש לה גם כן עיגול פנימי וקו מוטה ממעל.

כל שאר ספרות kannada סווגו בצורה יחסית אקראית.

עבור ספרות מערביות:

- 2 המערבית לא סווגה כ-3 או 7, או יחסית אקראית. שוב, 2 היא צורה לא צפויה כל כך.
- 8 המערבית לא סווגה כ-4, מה שתואם את מה שקרה במקרה ההפוך.

ובאופן כללי ספרות מערביות סווגו באופן אקראי למדי, מה שכול לנבוע מהעובדה שספרות kannada מורכבות יותר, ולכן כל דמיון אפשרי הוא פחות ניכר עבור העץ שלנו, ויש הרבה יותר וריאציות ושינויים בין דוגמאות של kannada, לעומת האחידות היחסית בספרות מערביות שנובע מפשטותן.