מבוא לבינה מלאכותית - 236501

תרגיל בית 3

מגישים:
יקיר חלץ 305028441
גל פלייסיג 302912985

חלק א׳ – מבוא והנחיות

חלק ב׳ – פיתוח KNN

1) מימוש בקוד.

2) מימוש בקוד.
3)
   a. מימוש בקוד.

   b. הקבצים מצורפים להגשה. הסבר מדוע חשוב לשמור על עקביות בחלוקה ל-folds (כלומר יש לקרוא לפונ׳ פעם אחת בלבד ולהשתמש באותה חלוקה): ראשית נבין מהי החלוקה הנ״ל ומדוע אנחנו מבצעים אותה. הסיבה שלשמה אנחנו מבצעים את החלוקה (כאן זוהי 2-fold cross validation) היא כדי לשערך ביצועים עתידיים של המסווג. אנחנו מחלקים את קבוצת הדוגמאות המסומנות שלנו בחלוקה אקראית ל-k תתי-קבוצות, כאשר ראשית אנחנו מאמנים את המסווג בכל פעם על k-1 קבוצות ובודקים את ביצועים על הקבוצה הנותרת, ומבצעים את התהליך הנ״ל לכל חלוקה של k הקבוצות שלנו ל-k-1 ול-1. לבסוף ממצעים את הדיוק.

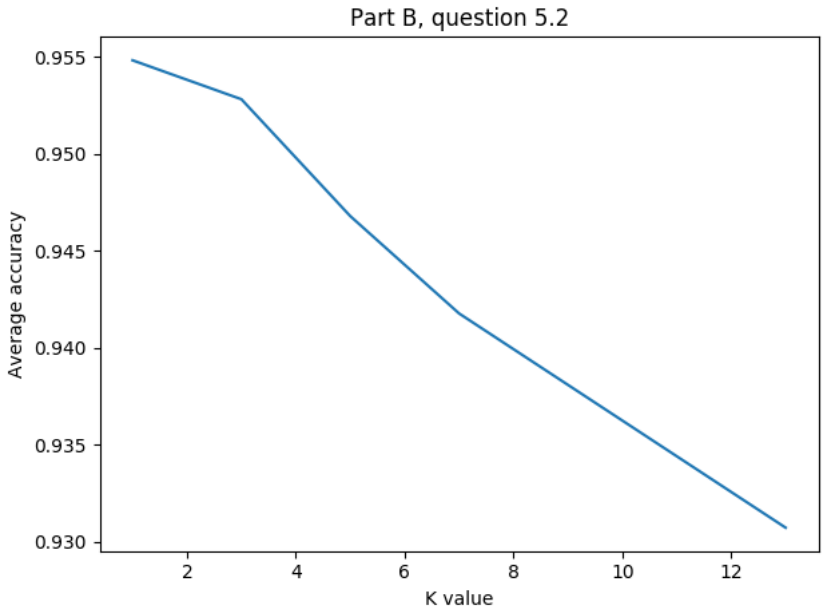
   במקרה שלנו אנחנו משתמשים ב-k=2, כלומר התהליך מורכב משני חלקים – ראשית מאמנים את המסווג על קבוצה מספר 1 (הדוגמאות מ-ecg_fold_1.data) ובוחנים אותו על קבוצה מספר 2 (הדוגמאות מ-ecg_fold_2.data) ואז מאמנים את המסווג על קבוצה מספר 2 ובוחנים אותו על קבוצה מספר 1 (לבסוף כמובן ממצעים את הדיוק).

   אם לא נשמור על אותה חלוקה כאשר אנחנו מאמנים את המסווג, ישנה סבירות גבוהה שבפעם השנייה שבה נאמן את המסווג על הקבוצה השנייה, קבוצה זו תכיל דוגמאות מסומנות שכבר ״התאמנו״ עליהן קודם לכן. זה יצור מצב שיהיה מספר לא מבוטל של דוגמאות שלא נתאמן עליהן כלל, וזה מהווה בזבוז של המשאבים שלנו, שכן אם אנחנו רוצים לשערך את דיוקו של מסווג בצורה הטובה ביותר אנחנו רוצים לבחון אותו אל מול מספר גדול ככל האפשר של דוגמאות מסומנות שיש ברשותנו, מה שישפר את השערוך.

4) מימוש בקוד.

5) תשובה:
   a. הקובץ experiments6.csv מצורף להגשה.

   b. להלן הגרף שהתקבל:



c. הערך של k עבורו התקבלו הביצועים הטובים ביותר, כלומר הדיוק הטוב ביותר, הוא k=1.

d. הסבר וניתוח הממצאים: הגרף מציג מגמה ברור: ככל שערך ה-k גדל, כך קטנה רמת הדיוק שהושגה באלגוריתם knn. ערך המקסימום של הגרף, כלומר רמת הדיוק המקסימלית (כאן: עבור k=1) הוא 0.954, וערך המינימום של הגרף, כלומר רמת הדיוק המינימלית (כאן: עבור k=13) הוא 0.930.
בכיתה נלמד כי בחירת ה-k היא משמעותית כאשר: מצד אחד יש לבחור k שלא יהיה קטן מדי, שכן במקרה זה עלול להיווצר מצב של overfitting שבו הקטנת שגיאת האימון תגרום לעליה בשגיאת המבחן (נובעת בד"כ מדוגמאות רועשות).
עבור k קטן מדי לא "נתשאל" מספיק שכנים כדי לקבל תשובה אמינה, ומצד שני יש לבחור k שלא יהיה גדול מדי, שכן אז במקרה זה אנחנו עלולים להגדיל את שגיאת המבחן בגלל שאנחנו מתייעצים עם שכנים יותר רחוקים. במקרה שלנו קיבלנו שהדיוק המקסימלי הוא עבור k=1. ראשית כל נבחין כי ההפרש בדיוק בין k=1 לבין k=13 קטן יחסית: 0.02%. אולם, בכל זאת ניתן להסביר את התופעה הנ"ל בכך שיתכן כי דוגמאות האימון אינן מחולקות בצורה חדה, דהיינו ליד דוגמאות המתויגות כ-True יש דוגמאות המתויגות כ-False ולהיפך. וזה יוצר מצב שבו אם נשאל שכנים רבים מדי, נגיע לתיוג ההפוך מהתיוג האמיתי שצריך להתקבל ולהורדת הדיוק.
הסבר נוסף שיתכן הוא שאם נסתכל על דוגמאות האימון המסופקות לנו נבחין כי קרוב ל-90% מהן מתויגות כ-True. כלומר אם דוגמה מסוימת היא True ומקבלת את התיוג True אז המסווג צדק, אבל במקרים אחרים שבהם הסיווג האמיתי הוא False, הסבירות שהסיווג שינתן לאובייקט הוא True בהינתן שהוא שואל שכנים רבים יחסית וגם רוב דוגמאות האימון הן True – גבוהה יחסית. זה יצור מצבים של חוסר התאמה באחוזים קטנים בין תשובת המסווג לתשובה האמיתית, וזה מה שאנחנו חוזים בו בתוצאות שהתקבלו.

7) מימוש בקוד.

a. מבין כל הניסויים כולל עבור כלל ערכי ה-k שבדקנו, הניסוי שבו התקבלו התוצאות הטובות ביותר הוא: knn עם ערך k=1 (עבור ID3 ועבור Perceptron התקבל דיוק שנע בין 88 ל-89 אחוזים – מצוי בקובץ המצורף).

חלק ג׳ – תחרות ובונוס

מהות חלק זה הינה לבנות מסווג, כך שבסופו של דבר הוא יתן את הדיוק הגבוה ביותר עבור 300 הדוגמאות הלא-מסומנות המסופקות לנו. נחלק את הבנייה הזאת להקדמה (מידע על התכונות) ולשני חלקים נוספים אשר השילוב שלהם ירכיב את המסווג האופטימלי.

החלק של ההקדמה (מידע על התכונות) יתמקד בניסיון להשיג יותר מידע על וקטור התכונות, אילו תכונות משמעותיות יותר לתהליך הסיווג ואילו פחות, ועוד. החלק הראשון (חלק מספר 1) יהיה בחירת המסווג (עבור אלגוריתם למידה מסוים), והחלק השני (חלק מספר 2) יהיה בחירת תת-קבוצה של התכונות עליה יתאמן המסווג שנבחר (הסבר בהמשך). השילוב של שני אלה יניב לנו מסווג אופטימלי.

דרך העבודה שלנו תהיה הבאה: ראשית נמנה ונממש מספר אלגוריתמי למידה מתוך האפשרויות הרבות הקיימות. לאחר מכן נבחן אפשרויות שונות לבחירת תת-קבוצה של תכונות. ולסיום נבצע הצלבה, כלומר נבחן כל אחד מאלגוריתמי הלמידה ביחד עם כל אחת מהאפשרויות שמנינו לתתי-קבוצות. נמצא היכן מתקבל אחוז הדיוק המקסימלי **ונקבע כמסווג הסופי את המסווג שהתקבל תוך שהוא מאומן על 1000 הדוגמאות שסופקו לנו כאשר נבחרה תת-הקבוצה האופטימלית של התכונות עבורן**. נרצה להשתמש בכל 1000 הדוגמאות מכיוון שאחרי שבחרנו מיהו המסווג האופטימלי ועבור איזו קבוצת תכונות, נרצה לספק למסווג שלנו מספר מקסימלי של דוגמאות כדי שהוא יוכל ללמוד מהן (כמובן נספק לו את התכונות המצומצמות בהתאם לערך ה-KBest האופטימלי שנמצא).

* כאן המקום לציין כי מספר האפשרויות לבחירה הוא עצום, הן מכיוון שמספר אלגוריתמי הלמידה שלמדנו ושמצויים בספריות כמו sklearn הוא גדול, הן מכיוון שטווח הפרמטרים שניתן למסווגים של אלגוריתמים אלה הוא גדול, והן מכיוון שמספר התכונות שאפשר לבחור מתוך סך התכונות בדוגמה אחת הוא גדול גם כן (שלל אפשרויות לתתי-קבוצות מתוך קבוצה גדולה, בת 187 איברים במקרה שלנו). אלמנט נוסף שאותו עושה אלגוריתם Random Forest עליו נדבר בהמשך הוא בחירת תת-קבוצה של הדוגמאות מתוך ניסיון לנקות דוגמאות רועשות, וגם לכך אפשרויות רבות.

הקבצים בהגשה של חלק ג׳:

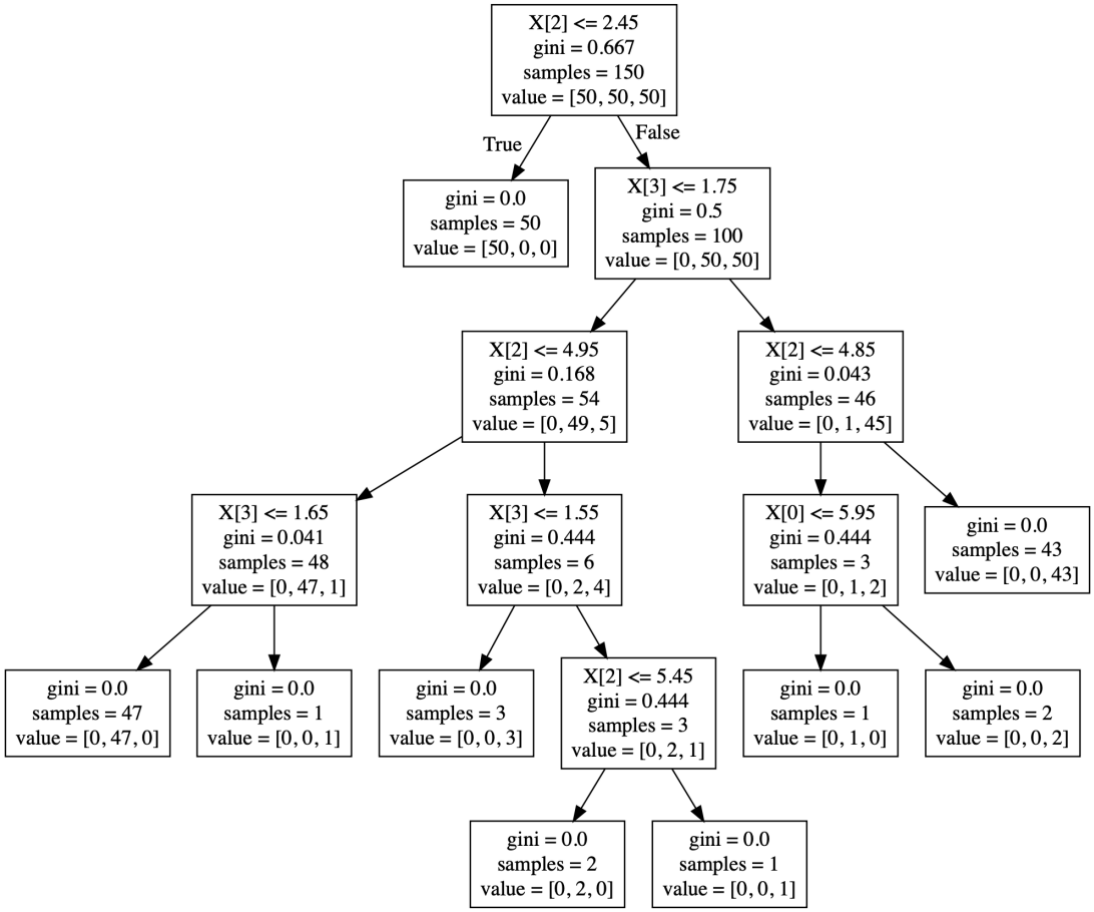
- **part_c_classifiers.py** – מכיל את המימושים של המסווגים השונים כאשר ישנו מימוש של שתי מחלקות עבור כל מסווג, מחלקה עם השם factory שמממשת את המתודה train, ומחלקה עם השם classifier שמממשת את המתודה classify. זאת על-מנת שנוכל להזין את ה-factory לפונ׳ evaluate שמימשנו בחלק ב׳

- **part_c_tests.py** – מכיל מימוש של הקוד המריץ את כל המסווגים ולכל מסווג עם כל האפשרויות שנבחרו עבור ערכי ה-KBest. קוד זה מוציא פלט בצורה של גרפים (אותם נצרף לדוח – בחלק הסיכום) שיעזור להבין מהו המסווג האופטימלי ועם איזו קבוצת תכונות (מהו ערך ה-K עבור KBest). כאן אנחנו משערכים את כלל האלגוריתמים עם 10-fold cross validation

- **main.py** – מכיל בנוסף למימוש הקוד של חלק ב׳ גם קוד המאמן את המסווג הנבחר על כל 1000 הדוגמאות המסומנות (יתכן עם הקטנה של מספר התכונות) ואז מסווג את הדוגמאות הלא-מסומנות המסופקות, ומייצר את הקובץ results.data עם התוצאות

מידע על התכונות

ראשית, ננסה לקבל מידע נוסף אודות התכונות של הדוגמאות שהתקבלו. נבחין כי ישנן 187 תכונות במספר. נוכל באמצעות הקוד שכתבנו (המחלקה factory של ID3) לקבל מסווג ID3 בצורת עץ החלטה. לאחר מכן נמיר אותו לקובץ dot. ומשם לגרף בפורמט תמונה אותו נוכל לראות להלן:



ראשית ניתן להבחין כי הגרף קטן בהתחשב בכך שישנן 187 תכונות בוקטור, וזו גדולתו של אלגוריתם ID3. בגרף אנו יכולים לראות כי מופיעות התכונה ה-0, התכונה ה-2 והתכונה ה-3 בלבד (מופיעים X[0], X[2], X[3]) בצמתי החלטה פנימיים בעץ ההחלטה. מכאן אנחנו יכולים להסיק שניתן להפיק עץ החלטה עקבי רק מתכונות אלה. מצד אחד יתכן שישנן תכונות רועשות רבות ו-ID3 "מתגבר" עליהן, והיינו רוצים להתחשב בעץ זה בבואנו לסווג דוגמה. מצד שני, ראינו כי אלגוריתם ID3 דווקא לא הפיק את התוצאות הטובות ביותר בבדיקות שעשינו בחלק ב', ולכן יש לבדוק את כלל דיוקי המסווגים בצורה מסודרת (נבדק בהמשך).

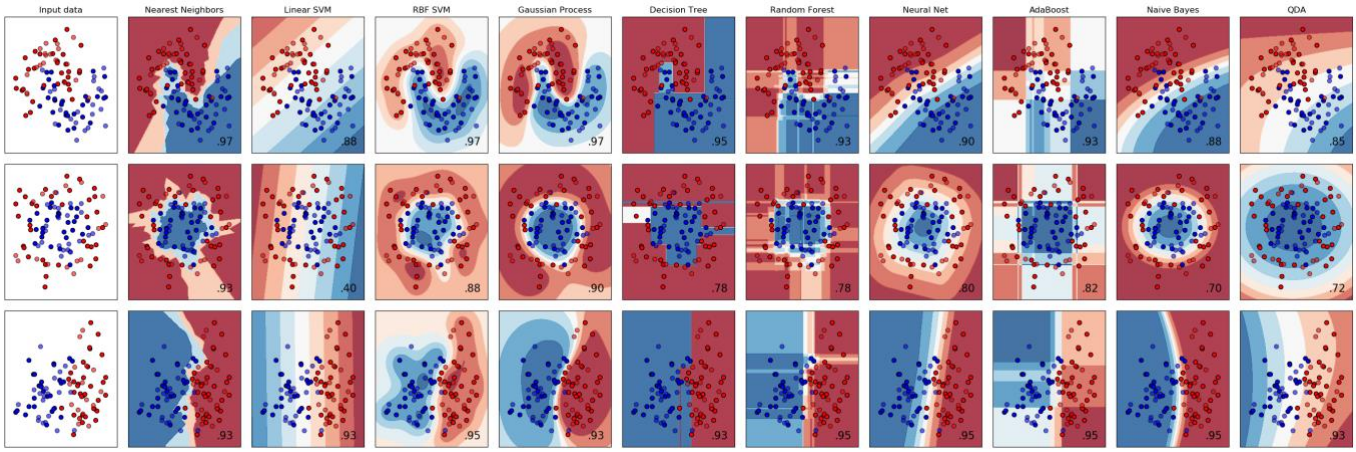
* כזכור מההרצאות אלגוריתם ID3 הוא אלגוריתם חמדני ללמידת עצים עקביים מדוגמאות, ומבוסס על המסגרת האלגוריתמית של TDIDT, כאשר הוא בוחר את התכונה עם תוספת האינפורמציה הגדולה ביותר.

בהמשך נרצה להיות מסוגלים לעשות שני דברים:
הדבר הראשון הוא לבחור תת קבוצה של תכונות שהיא K התכונות הטובות ביותר מתוך 187 התכונות הקיימות באובייקט. את זאת אנו מבצעים בחלק מספר 2 – פירוט בחלק זה.
הדבר השני הוא לבחור תת קבוצה של הדוגמאות, כלומר נרצה "לנקות" דוגמאות רועשות מן הקלט שלנו. **את זאת מבצע אלגוריתם Random Forest בחלק מספר 1 בהמשך.**

חלק מספר 1

כאמור, בחלק זה נציג מסווגים שונים אופציונליים לבעיה, ונסביר עליהם. לכל אחד מן המסווגים שנציג כאן מימשנו מחלקת factory המכילה את המתודה train וכן מחלקת classifier המכילה את המתודה classify, זאת על-מנת שנוכל להעביר אותן בהמשך לפונ׳ evaluate שבנינו בחלק ב׳ לקבלת אחוז הדיוק. בהמשך בקטע הקוד בקובץ **part_c_tests.py** גם נבחן אותם עם אפשרויות שונות מחלק מספר 2.

נשים לב כי בדוקומנטציה של ספרית scikit-learn (באתר scikit-learn.org) מוצגים לנו בתמונה סוגים שונים של מסווגים ואחוזי הדיוק שלהם עבור קלטים שונים. להלן התמונה:



בהתאם לתמונה ובהתאם למסווגים שונים שלמדנו ושהשתמשנו בהם בתרגיל זה, נבחר במספר סוגים של מסווגים:

Nearest Neighbors – נבחר באלגוריתם 1-NN הן מכיוון שערך k=1 שלו הפיק לנו את התוצאות הטובות ביותר בחלק ב׳ מבין כלל האלגוריתמים (גם אלה שאינם KNN) והן מכיוון שעל פי התמונה אלגוריתמי nearest neighbours משיגים דיוקים גבוהים מאד. אלגוריתם KNN נלמד בהרצאות ובתרגולים. אנו נבחר בגרסה שלו עם k=1 מכיוון שכאמור בתוצאות בחלק ב׳ באימון על כלל הדוגמאות גילינו עם 2-fold-cross-validation כי עם k=1 השיג האלגוריתם את התוצאות האופטימליות.

Linear SVM – נבחר באלגוריתם Perceptron (אשר נלמד בתרגול 13) שהוא מסווג לינארי והשתמשנו בו בחלק ב׳ של תרגיל זה. נבחר בו כדי לבחון כיצד מתמודד מסווג לינארי עם הקלט הנתון, בדגש על ערכי KBest שונים (ראינו שעבור לקיחת כלל התכונות הוא אינו מיטבי ביחס לשאר האלגוריתמים שנבדקו בחלק ב׳).

Gaussian Process – נבחר באלגוריתם מכיוון שאנחנו רואים שהוא משיג דיוקים גבוהים מאד עבור קלטים שונים לפי התמונה המצורפת. אלגוריתם זה הוא תהליך סטוכסטי (אוסף של משתנים רנדומליים) כך שכל קומבינציה לינארית של המשתנים הללו מתפלגת באופן אחיד. נשתמש בו עם הערך **הדיפולטיבי** שהוא 1.0 * RBF(1.0). פרמטר זה הוא אובייקט kernel המתאר את ה-covariance, שהוא פרמטר פנימי של Gaussian Process לפיו פועל האלגוריתם.

Decision Tree – נבחר באלגוריתם ID3 (כפי שנלמד בהרצאות והשתמשנו בו בחלק ב׳ של תרגיל זה) שהוא כאמור אלגוריתם חמדני מסוג עץ החלטה ללמידת עצים עקביים מדוגמאות, מבוסס TDIDT. נבחר בו כדי לבחון כיצד מתמודד מסווג מהצורה של עץ החלטה עם הקלט הנתון, שוב, בדגש על ערכי KBest שונים (גם עבורו ראינו שעבור לקיחת כלל התכונות הוא אינו מיטבי ביחס לשאר האלגוריתמים שנבדקו בחלק ב׳).

Random Forest – נבחר באלגוריתם זה כפי שנלמד בהרצאות. אלגוריתם זה הוא אלגוריתם ועדה המשלב בחירת תת-קבוצות של דוגמאות ובחירת תת קבוצות של תכונות. נבחר בו שכן הוא נחשב כיום בין אלגוריתמי הלמידה החזקים כפי שנאמר בהרצאות והוא משיג תוצאות דיוק גבוהות. הוא מקבל את הפרמטר n_estimators שהוא מספר העצים ביער, כלומר כמה "מעריכים" יכללו באלגוריתם. נשתמש ב-Random Forest עם הפרמטר 10 שהוא הפרמטר הדיפולטיבי בגרסה 0.20 (בגרסה 0.22 ערכו הדיפולטיבי של n_estimators עלה ל-100).

Combined classifier – זהו אלגוריתם אשר מסתכל על התוצאות של 1-NN, של Random Forest ושל Gaussian Process עבור האובייקט features ומחזיר תשובה תוך שהוא מתייחס אליהם כאל ועדה – התשובה (True או False) שחזרה ב-2 מתוך 3 מקרים לפחות היא זו שתוחזר על-ידו. **זהו רעיון לנסות ולמטב את הדיוק** על ידי יצירת אלגוריתם שיקבל מהדיוק המרבי של כל אחד משלושת האלגוריתמים עליהם הוא מסתמך.
בחרנו דווקא בשלושת אלגוריתמים אלה מכיוון שבניסויים שביצענו (מפורטים בסיכום) הם הפיקו את התוצאות הטובות ביותר.

חלק מספר 2

בחלק זה כאמור בהקדמה אנחנו רוצים לבחור תת-קבוצה של 187 התכונות. אנו נשתמש בפונ' SelectKBest של sklearn אשר בוחנת את כלל התכונות ונותנת להן דירוג לפי הפרמטר הראשון שמועבר אליה, ואז משתמשת בפרמטר K שהוא הפרמטר השני שהועבר אליה כדי לבחור מתוך (במקרה שלנו) 187 התכונות את K התכונות הטובות ביותר לפי הדירוג שהיא ביצעה. היא מחזירה רשימה של הדוגמאות עם התכונות המעודכנות (הטובות ביותר) בלבד.

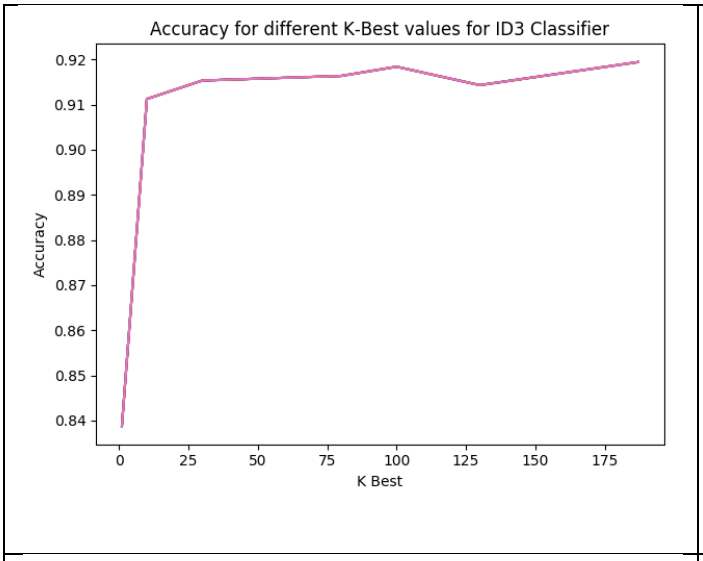
הפונ' SelectKBest מקבלת שני פרמטרים כאמור, האחד הוא score_func והוא פרמטר הקובע לפי איזו שיטה תדרג הפונ' את התכונות, והשני הוא ה-K כמוסבר לעיל.

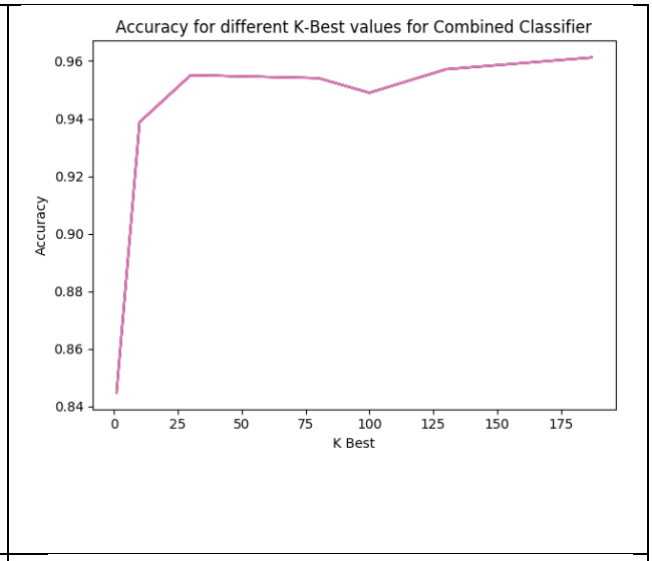
עבור הפרמטר score_func, נבחר את הערך הדיפולטיבי f_classif, אשר מסתמך על מבחן ה-F. מבחן זה הוא מבחן סטטיסטי אשר בו סטטיסטי המבחן הוא בעל התפלגות F (התפלגות רציפה) תחת השערת האפס.

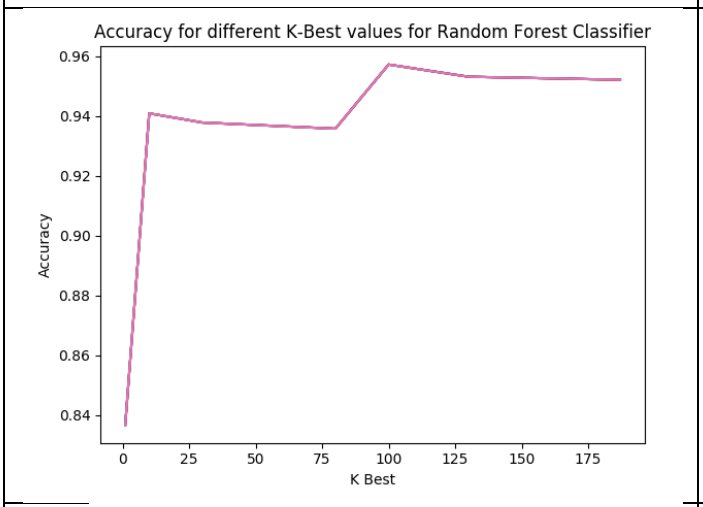
עבור הפרמטר K נבחר בכל הרצה ערך אחר:
במהלך הבדיקות בקובץ **part_c_tests.py** אנחנו נבדוק כל אחד מהמסווגים בחלק מספר 1 על כל אחד מהערכים הבאים של K: [1, 10, 30, 80, 100, 130, 187] עם כל אחת מ-1000 הדוגמאות (כמובן נלקחות K התכונות הטובות ביותר מהדוגמאות בכל איטרציה על K).
בחרנו דווקא וקטור ערכים זה כדי לקבל דגימות המתפרסות על כלל טווח המספרים שבין בחירת תכונה אחת בלבד (K=1) לבין בחירת כלל התכונות (K=187). נראה בחלק הסיכום כיצד משפיעים הערכים השונים על התוצאות.

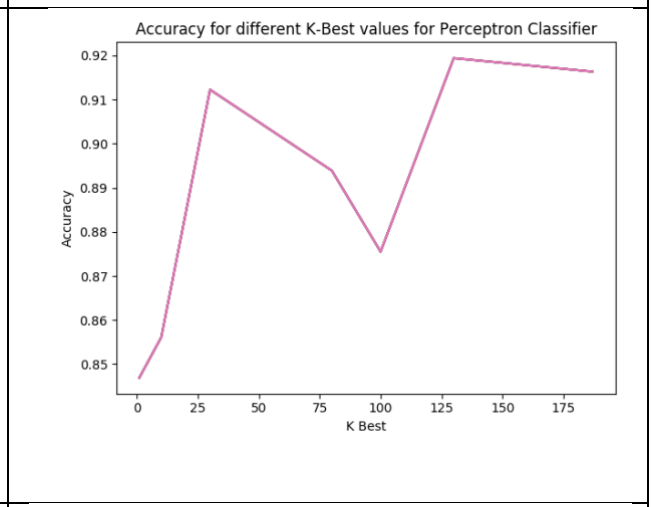
סיכום

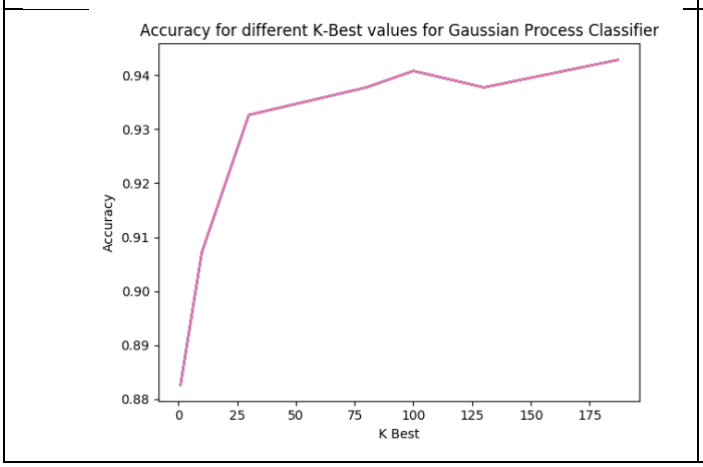
להלן גרפים המסכמים את התוצאות של ההצלבה (גרף אחד עבור כל אחד מששת המסווגים הנבדקים, כאשר הכותרת מציינת מיהו המסווג):

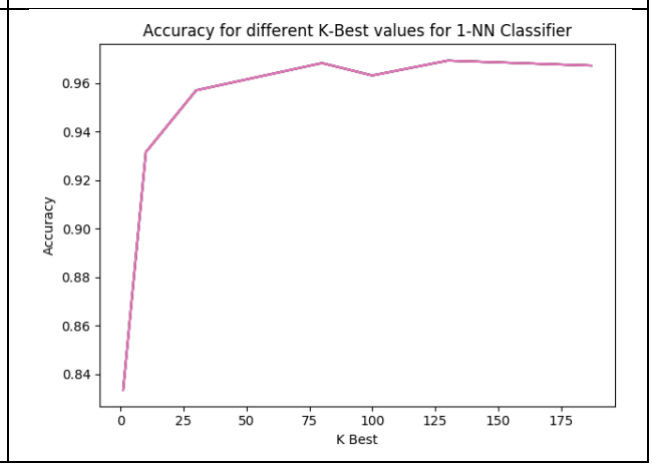




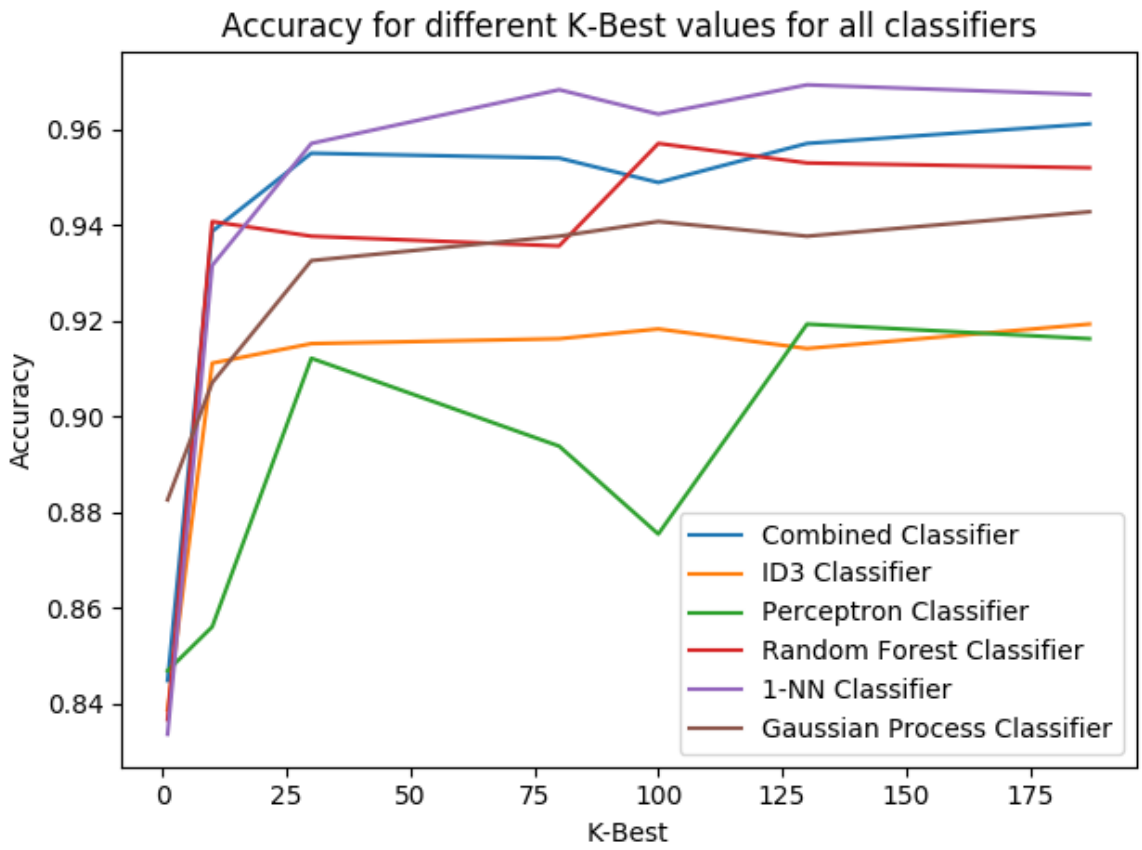








להלן גרף המאגד את כלל הגרפים הנ"ל:



<u>ניתוח תוצאות הגרפים:</u>
ניתן להבחין בשני דפוסים עיקריים שעולים מהגרפים:

ראשית כל, ניתן באופן כללי לתת דירוג למסווגים (יחס סדר) מהטוב לפחות טוב ללא קשר לערך ה-KBest. למעט טעויות סטטיסטיות, המסווג הטוב ביותר הוא 1-NN, השני הטוב ביותר הוא Combined, השלישי הטוב ביותר הוא Random Forest, הרביעי הוא Gaussian Process, החמישי הוא ID3 ולבסוף המסווג הכי פחות טוב הוא המסווג הלינארי Perceptron.
שנית, ניתן לראות כי באופן כללי ככל שמגדילים את KBest, כך עולה אחוז הדיוק של כל מסווג. אולם, הקפיצה המשמעותית היא בין K=1 לבין K=30. לאחר מכן עבור ערכי K גדולים מ-30 הדיוק באופן כללי משתפר באלגוריתמים השונים אך במעט (יכולות להיות לכך השלכות זמן ריצה, למשל עבור 1-NN אין צורך להשוות את כל התכונות כדי לקבל דיוק גבוה יחסית).

אנו נרצה לבחור בשילוב של אלגוריתם עם K הנותן את הדיוק המקסימלי.

אנחנו רואים כי הדיוק הטוב ביותר התקבל עבור אלגוריתם הלמידה 1-NN (כלומר KNN עם ערך k=1) והשילוב הטוב ביותר של התכונות הוא בחירת 130 התכונות הטובות ביותר. ולכן בסיכומו של דבר נבחר במסווג המורכב מ-1-NN עם KBest = 130. התוצאות שלו מצויות בקובץ results.data, אחוז הדיוק שלו הוא כ-97 אחוזים והמימוש של שתי המחלקות עבורו (factory & classifier) מצויות בקובץ **part_c_classifiers.py** כאמור.

* כזכור מההרצאות והתרגולים, המסווג 1-NN שבחרנו עובד בצורה הבאה: הוא בוחן את כלל הדוגמאות שברשותו ומסדר אותן לפי המרחק (האוקלידי במקרה זה) מהאובייקט אותו הוא מעוניין לסווג, ואז בודק מהו הסיווג – True או False – של רוב הדוגמאות מתוך ה-k. במקרה שלנו k = 1 ולכן הוא יחזיר את הסיווג של הדוגמה הקרובה אליו ביותר.

נספח: להלן תוצאות אחוזי הדיוק של המסווגים השונים עבור ערכי ה-KBest השונים (לפי הוקטור לעיל, הערך הראשון הוא KBest = 1 והאחרון הוא KBest = 187 לכל אחד מהמסווגים):

Combined
[0.8448979591836734, 0.9387755102040817, 0.9551020408163264, 0.9540816326530613, 0.9489795918367347, 0.9571428571428571, 0.9612244897959183]

ID3
[0.8387755102040815, 0.9112244897959183, 0.9153061224489797, 0.9163265306122449, 0.9183673469387756, 0.9142857142857143, 0.9193877551020408]

Perceptron
[0.846938775510204, 0.8561224489795919, 0.9122448979591835, 0.8938775510204081, 0.8755102040816327, 0.9193877551020411, 0.9163265306122451]

Random Forest
[0.8367346938775511, 0.9408163265306122, 0.9377551020408162, 0.9357142857142857, 0.957142857142857, 0.9530612244897959, 0.9520408163265307]

1-NN
[0.8336734693877552, 0.9316326530612246, 0.9571428571428571, 0.9683673469387756, 0.963265306122449, 0.9693877551020409, 0.9673469387755101]

Gaussian Process
[0.8826530612244898, 0.9071428571428571, 0.9326530612244899, 0.9377551020408162, 0.9408163265306122, 0.9377551020408162, 0.9428571428571427]