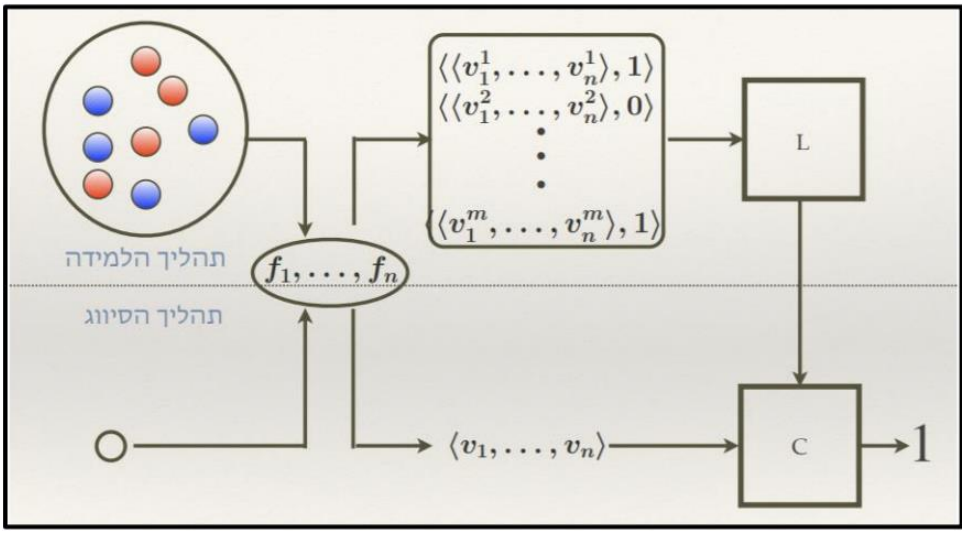
אלגוריתם לסיווג דוגמאות תחת הגבלת תקציב לרכישת ערכי תכונות

פרויקט בבינה מלאכותית

236502

הפקולטה למדעי המחשב, טכניון



מגישים:

סיון יוסבשוילי	318981586	sivan.y@campus.technion.ac.il
רז לוי	316579275	razle@campus.technion.ac.il

תוכן עניינים

מבוא

הרעיון שלנו לפרויקט עלה מתוך הצורך לענות על בעיה רפואית מציאותית מחיי היום-יום.

נציג כעת דוגמא שתלווה אותנו לכל אורך הפרויקט ותדגים את הצורך הכללי שאנו רוצים להציע לו פתרון - ישראל ישראלי, רוצה לגשת לרופא המשפחה שלו, ולבקש ממנו שיאמר לו מה הסיכויים שבעתיד יחלה במחלה כלשהי.

הרופא יכול לשלוח את ישראל לסדרת בדיקות ארוכות (ויקרות) שבסופן יוכל לענות לישראל על השאלה עם וודאות די גבוהה.

פעמים רבות, בדיקות מיוחדות אינן נמצאות בסל הבריאות, ולכן ישראל יאלץ לשלם עליהן.

ישראל אינו יכול להרשות לעצמו להוציא סכומי עתק, ולכן הוא מאתגר את הרופא –

"בהינתן שאני מוכן לשלם X שקלים על בדיקות רפואיות, מה הן הבדיקות שעליי לעבור כדי שתוכל לומר לי בוודאות גבוהה מה הסיכויים שבעתיד אחלה במחלה כלשהי"?

באופן כללי, פעמים רבות כשאנו נתקלים בבעיית סיווג, הדוגמא אותה אנו מנסים לסווג לא מכילה את כל המידע (יש בה חלקים חסרים).

נרצה תחת אילוץ התקציב, לרכוש את ערכי התכונות החסרות על מנת שבסופו של דבר, יהיה לנו מספיק מידע כדי לסווג את הדוגמה כהלכה.

ניסוח הבעיה באופן פורמלי-

בהינתן:

סט $\{X, Y\}$, אשר מורכב מקבוצת דוגמאות $X = \{\bar{x}_1, \dots \bar{x}_m\}$, $\bar{x}_i \in R^D$, כאשר D הוא מספר התכונות וקבוצת תיוגים בינאריים $Y = \{y_1, .., y_m\}$, כאשר y_i הוא התיוג לדוגמה $\bar{x}_i$.

$C = \{c_1, \dots, c_D\}$ קבוצת המחירים עבור כל תכונה, כאשר c_i היא עלות התכונה ה-i (המחיר הוא אי שלילי).

נרצה לייצר מודל, כך שבהינתן דוגמא חדשה $\hat{x} \in R^k$, כאשר $k < D$ (בה חסרים $(D - k)$ תכונות) ותקציב מקסימלי $\hat{C}$, מחזיר בכל שלב, איזו תכונה הוא מעוניין לרכוש על מנת לסווג את הדוגמה.

בסיום התהליך, יפלוט המודל את הסיווג של הדוגמה.

לאורך הדו"ח נציג מספר גישות לפתרון הבעיה.

באופן כללי, כל האלגוריתמים שנציע יהיו בעלי מבנה של אימון וסיווג כאשר מטרת האימון היא לאסוף מידע על הדוגמאות ולחקור את הקשרים בין התכונות.

מטרת הסיווג היא להציע סיווג לדוגמה, כאשר במהלך הסיווג, האלגוריתם יבחר לרכוש תכונות (תחת מגבלת התקציב), כאשר הבחירה לרכוש תכונה בכל שלב תתבסס על המידע שצבר האלגוריתם עד לאותו שלב.

האלגוריתמים שנציע יתמודדו עם מספר בעיות שיוצגו ויפורטו במהלך הדו"ח, ביניהן העובדה שבזמן האימון איננו מודעים לתקציב הניתן לרכישת התכונות וכן לתכונות ההתחלתיות הנתונות לדוגמה אותה נרצה לסווג. כל הנתונים הללו יינתנו רק בשלב הסיווג ולכן יהיה עלינו להתמודד עם חוסר המידע הנ"ל קודם לכן- בשלב האימון.

<u>פסאודו קוד לאלגוריתם הכללי (ממומש בקובץ abstract_algorithm.py)</u>

<u>אימון:</u>

- <u>קלט:</u>

(1) קבוצה $\{X, Y\}$ כאשר $X = \{\bar{x}_1, \dots, \bar{x}_m | \ \bar{x}_i \in \mathbb{R}^D\}$ קבוצת דוגמאות (D מספר התכונות) ו-$Y =$ $\{\bar{y}_1, \dots, \bar{y}_m | \bar{y}_i \in \{0,1\}$.

(2) קבוצת מחירים לתכונות $C = \{\bar{c}_1, \dots, \bar{c}_D \mid \bar{c}_i \in \mathbb{Z}\}$ בהתאמה.

...

<u>סיווג:</u>

- <u>קלט:</u>

(1) דוגמה $\hat{x} \in \mathbb{R}^k$ כאשר $k < D$ ו- D מספר התכונות האפשרי לכל דוגמה.

(2) תקציב מקסימלי $\tilde{C}$.

(3) קבוצת אינדקסים K של התכונות הנתונות.

- <u>פלט:</u>

(1) סיווג הדוגמה.

...

האלגוריתמים "הנאיביים"

תחילה, חשבנו על האלגוריתם "הנאיבי" שיאפשר את פתרון הבעיה. החשיבה על אלגוריתם "נאיבי" תעזור לנו למפות את הבעיות שעלולות לעלות במהלך הפתרון, למצוא את "צוואר הבקבוק" של הבעיה ובכך להתקדם אל עבר מציאת אלגוריתם טוב יותר. במהלך החשיבה, עלו שתי גישות אפשריות לאלגוריתם "נאיבי" שכזה.

גישה א'- האלגוריתם הריק

כאשר מטופל מגיע לרופא בבקשה לקבלת אבחנה רפואית ובהינתן שידועים לרופא $k < D$ תוצאות בדיקות על המטופל (D המספר המקסימלי לתוצאות הבדיקות), ייתכן כי הרופא יבחר לנסות ולאבחן את המטופל בעזרת k התכונות הללו בלבד.

נציג כעת את האלגוריתם בצורה פורמלית. לצורך כך נבחר אלגוריתם למידה כלשהו (כדוגמת KNN או ID3).

אימון:

- קלט: לפי האלגוריתם הכללי המפורט במבוא.

1. שמור את קבוצת הדוגמאות והתיוגים $\{X, Y\}$.

סיווג:

- קלט ופלט: לפי האלגוריתם הכללי המפורט במבוא.

1. הרץ את שלב האימון של אלגוריתם הלמידה שבחרת עבור קבוצת הדוגמאות והתיוגים ששמרת בשלב האימון לפי התכונות הנתונות על הדוגמה שברצוננו לסווג בלבד.
2. הרץ את שלב הסיווג של אלגוריתם הלמידה שבחרת.
3. החזר את הפלט שהתקבל.

מימשנו את "האלגוריתם הריק" בקובץ naive_algorithm.py.

ניתן להבחין שהבעיה המרכזית באלגוריתם היא הוויתור על האופציה לקבלת מידע נוסף. על אף שברשותנו תקציב לרכוש תכונות נוספות ובכך להקטין את ה-loss (שגיאת הסיווג) ולהגיע לאחוזי דיוק (accuracy) טובים יותר- אנו "מסתפקים" בתכונות הנתונות לנו ולכן ה-loss גבוה וה-accuracy נמוך.

נשים לב שבשלב הסיווג של האלגוריתם הנ"ל אנו מבצעים שלב סיווג של אלגוריתם למידה מוכר. על פניו, נראה ששלב זה הינו איטי, אם כי באלגוריתם הנ"ל לא הוספנו פעולות נוספות מלבד הרצת שלב הסיווג של האלגוריתם המוכר. לכן, כדאי לציין שהאלגוריתם הנ"ל הוא המהיר ביותר שנציע, שכן בהמשך נרצה להציע אלגוריתמים חכמים יותר שיבצעו פעולות נוספות בשלב הסיווג- על פניו, נראה שאנו עתידים לקבל אלגוריתמים

בעלי זמן ריצה ארוך בהשוואה לאלגוריתמי סיווג מוכרים, אלא שהבעיה שעלינו לפתור קשה ומורכבת יותר.

<u>גישה ב'-</u> האלגוריתם הרנדומלי

כאשר מטופל מגיע לרופא בבקשה לקבלת אבחנה רפואית ובהינתן שידועים לרופא $k < D$ תוצאות בדיקות על המטופל (D המספר המקסימלי לתוצאות הבדיקות), ייתכן כי הרופא יבקש מהמטופל לבצע בדיקות רנדומליות כל עוד ברשות המטופל תקציב מספק.

נציג את האלגוריתם בצורה פורמלית. לצורך כך נבחר אלגוריתם למידה כלשהו (כדוגמת KNN או ID3).

<u>אימון:</u>

- <u>קלט:</u> לפי האלגוריתם הכללי המפורט במבוא.

1. שמור את קבוצת הדוגמאות והתיוגים $\{X, Y\}$.
2. שמור את קבוצת המחירים לתכונות C.

<u>סיווג:</u>

- <u>קלט ופלט:</u> לפי האלגוריתם הכללי המפורט במבוא.

1. חשב את קבוצת התכונות שטרם נבחרו.
2. כל עוד יש עדיין תכונה שלא נבחרה ותקציב הרכישה גדול מאפס:
   - 2.1. בחר תכונה רנדומלית משלב אימון.
   - 2.2. הסר אותה מקבוצת התכונות שטרם נבחרו.
   - 2.3. אם מחיר התכונה שבחרת קטן או שווה לתקציב רכישת התכונות:
     - 2.3.1. החסר את מחיר התכונה מתקציב רכישת התכונות.
     - 2.3.2. הוסף את התכונה לקבוצת התכונות שנבחרו ("רכוש את התכונה").
3. לאחר שחישבת את קבוצת התכונות הנבחרות, הרץ את שלב האימון של אלגוריתם הלמידה שבחרת (כאשר התכונות הנתונות לכל דוגמה הן התכונות בקבוצת התכונות הנתונות שחישבנו בשלב 2) עבור קבוצת הדוגמאות והתיוגים ששמרת בשלב האימון.
4. הרץ את שלב הסיווג של אלגוריתם הלמידה שבחרת.
5. החזר את הפלט שהתקבל.

מימשנו את "האלגוריתם הרנדומלי" בקובץ naive_algorithm.py.

בהשוואה "לאלגוריתם הריק", כעת אנו אכן מנצלים את התקציב שברשותנו לקבלת מידע נוסף, אלא שאנו עושים זאת בצורה רנדומלית ולכן אנו למעשה עלולים לרכוש תכונה שלא בהכרח מוסיפה מידע, או שמוסיפה מעט מידע ביחס לתכונות נוספות שיכולנו לרכוש.

בנוסף, ניתן להבחין שאחת הבעיות באלגוריתם היא חוסר האופטימליות- ייתכן שנבחר קבוצת תכונות קטנה שגודלה קטן מגודל הקבוצה האופטימלית (למשל אם קיימת תכונה שמחירה כל התקציב ומספר תכונות שסכום מחיריהן הוא כל התקציב- אם באופן רנדומלי נבחרה התכונה היקרה, לא נוכל להמשיך לרכוש תכונות בעוד שאם היינו בוחרים רנדומלית בכל פעם את התכונות הזולות, היינו יכולים לרכוש מספר רב יותר של תכונות). כלומר, באלגוריתם הנ"ל ייתכן שנוסיף מעט מידע. יש לציין שהמצב הזה לא בהכרח יפגע בסיכויינו לסווג כראוי, שכן ייתכן שתכונה אחת תהיה משמעותית יותר ממספר תכונות.

- נוכל באופן דומה לחשוב על אלגוריתם חמדן שבשלב האימון יבחר בכל פעם את התכונה הזולה ביותר ובכך נקבל קבוצת תכונות בגודל אופטימלי.
  האלגוריתם החמדן הנ"ל ממומש גם הוא בקובץ naive_algorithm.py בשם "OptimalAlgorithm".
  יתרונות וחסרונות האלגוריתם יהיו זהים "לאלגוריתם הרנדומלי" לרבות הדיון על היחס בין הגודל האופטימלי לתרומת התכונות.

מהדיון על האלגוריתם "הנאיבי" אנו מסיקים שעלינו לנצל מידע נוסף (כלומר לרכוש תכונות).

כמו כן, נסיק מספר נקודות חשובות שעלינו לחקור במטרה להגיע לאלגוריתם מספק:

1. מציאת דרך לבחור את התכונה שתוסיף לנו את המידע הרב ביותר. על מנת לעשות זאת, עלינו להגדיר "פונקציית מחיר" לכל תכונה שתיתן ביטוי לתרומת התכונה לסיווג נכון.

2. עם זאת, ראינו את החיסרון של השיטה החמדנית שבוחרת בכל פעם את התכונה הזולה ביותר באותו הרגע- זהו קונפליקט בין אופטימליות מספר התכונות לתרומתן. עלינו למצוא דרך להתמודד עם הקונפליקט הנ"ל, שכן מצד אחד, נשאף לרכוש כמה שיותר תכונות על מנת להוסיף את כמות המידע המרבית. מצד שני, אם התכונה שמוסיפה את המידע הרב ביותר היא תכונה A, ייתכן שנעדיף לרכוש דווקא אותה גם במחיר כזה שבאמצעותו שנוכל לרכוש שתי תכונות זולות יותר.

3. עלינו למצוא דרך להתמודד עם בעיות פרקטיות כגון חוסר הידע של התקציב הנתון, שכן בזמן האימון איננו מודעים לתקציב שיינתן רק בשלב הסיווג. לכן עלינו למצוא דרך להתמודד עם כל גודל תקציב אפשרי שיינתן בשלב הסיווג, כלומר לשמור נתונים "אינפורמטיביים" ורלוונטיים ולעבדם כבר בשלב האימון מבלי לדעת את התקציב שיינתן בהמשך.

PCA

במהלך תהליך החשיבה שלנו על דרכי כיוון לפתרון לבעיה שלנו עלה רעיון להשתמש ב-PCA.
נסביר בקצרה מה הקשר של הנושא לבעיה שאנו דנים בה ומדוע הוא אינו כיוון פתרון נכון.

ראשית, נחדד שתי הגדרות חשובות-
בבואנו להשתמש ב-$k < d$ פיצ'רים, קיימות שתי גישות:

- Feature selection – בחירת תת קבוצה $F' \subset F$ של פיצ'רים מתוך הפיצ'רים הקיימים.
- Dimensionality Reduction – יצירת פיצ'רים חדשים, המשתמשים בפיצ'רים הקיימים $F' = \{\phi_1(F), .., \phi_k(F)\}$

שיטת ה-PCA (Principal Components Analysis, בעברית : ניתוח גורמים ראשוניים) הינה שיטה שמטרתה הינה הורדת מימד של דאטה.
שיטה זו מתמירה באופן לינארי את הדאטה למערכת קואורדינטות חדשה, בה המידע בקואורדינטות החדשות הוא אורתוגונלי עם שונות הולכת וקטנה.
שיטה זו נפוצה בעיקר להורדת מימד בסטטיסטיקה ולמידת מכונה (הומצאה ב-1901 על ידי קרל פיטרסון).

נבחין ש-PCA לא מבצע בחירת פיצ'רים (feature selection) כפי שאנו מנסים לעשות בבעיה שלנו, אלא יוצר פיצ'רים חדשים מתוך הפיצ'רים הקיימים (שהינם קומבינציה של הפיצ'רים הקיימים).
לכן בחרנו לא להיעזר ב-PCA.

נקודה זו גרמה לנו לפתח אלגוריתם נוסף בו נתעדף תכונות לפי שונות. החלטה זו נובעת מתוך ההבנה כי אנו לא יכולים לייצר ביטויים לינאריים של התכונות, ולכן הפעולה הכי דומה ל-PCA שנוכל לבצע היא לבחור תכונות על פי שונות מקסימלית.

אלגוריתם ביניים

בדרך אל פיתוח אלגוריתמים מתוחכמים ואופטימליים במובנים מסוימים, חשבנו על אלגוריתם מתוחכם באופן פשוט שיכול להוות פתרון ביניים להשוואה בהמשך.

אלגוריתם שונות מקסימלית

כאשר מטופל מגיע לרופא בבקשה לקבלת אבחנה רפואית ובהינתן שידועים לרופא $k < D$ תוצאות בדיקות על המטופל יכול הרופא להציע למטופל לעבור את הבדיקות ה"אקספרסיביות" ביותר. לדוגמה אם יודעים שבדרך כלל כשעוברים את בדיקה x מקבלים המון מידע, ימליץ הרופא למטופל לעבור את בדיקה x.

נציג כעת את האלגוריתם בצורה פורמלית. לשם כך נבחר אלגוריתם למידה.

אימון:

- קלט: לפי האלגוריתם הכללי המפורט במבוא.

1. חשב לכל תכונה את השונות שלה. שמור רשימה של התכונות ממוינות לפי השונות מהגבוהה לנמוכה.

סיווג:

- קלט ופלט: לפי האלגוריתם הכללי המפורט במבוא.

1. כל עוד יש עדיין תכונה שלא נבחרה ותקציב הרכישה גדול מאפס:
   1.1. מצא את התכונה שאינה נבחרה עם השונות המקסימלית.
   1.2. הסר אותה מקבוצת התכונות שטרם נבחרו.
   1.3. אם מחיר התכונה שבחרת קטן או שווה לתקציב רכישת התכונות:
      1.3.1. החסר את מחיר התכונה מתקציב רכישת התכונות.
      1.3.2. הוסף את התכונה לקבוצת התכונות שנבחרו.
2. לאחר שחישבת את קבוצת התכונות הנבחרות, הרץ את שלב האימון של אלגוריתם הלמידה שבחרת עבור קבוצת הדוגמאות והתיוגים ששמרת בשלב האימון.
3. הרץ את שלב הסיווג של אלגוריתם הלמידה שבחרת.
4. החזר את הפלט שהתקבל.

מימשנו את "אלגוריתם השונות המקסימלית" בקובץ mid_algorithm.py.

נשים לב כי אלגוריתם זה מתוחכם יותר מאלגוריתמים קודמים, שכן הוא מבצע בחירה של תכונות, והיא נעשית תוך לוגיקה בסיסית שאמורה לקדם אותנו אל עבר המטרה.

אלגוריתם זה הינו אלגוריתם מהיר, והלוגיקה שהוספנו לבחירת התכונות "זולה" ביחס לתועלת שתביא בבחירה חכמה של תכונות (שיפור במיקום על עקומת accuracy effort tradeoff).

חיסרון של גישה זו היא להסתכל על תכונות טוב אך ורק מנקודת מבט אחת (תכונה טובה היא תכונה שהשונות שלה גבוהה). אמירה זו אולי נכונה, אך כמובן שאפשר לפתח אותה באופן מעמיק ומתוחכם יותר, ונצפה שפיתוחים אלו יביאו לשיפור בתוצאות אל מול אלגוריתם בסיס זה.

לכן, נרצה להמשיך ולחקור מה היא תכונה "טובה" בבעיה שלנו.

נדגיש בשלב זה חסרון של האלגוריתם. במידה וקיימת לנו תכונה A כלשהי, ותכונה B שהיא כפולה ב-20 של תכונה A, אזי $Var(B) = 20^2 Var(A)$ ולכן נעדיף את תכונה B למרות שהיא "מספרת" לנו בדיוק את מה שתכונה A מספרת.

לכן אנו מבינים כי קיימת בעייתיות בגישה זו.

פונקציית Score

נגדיר פונקציה שתעזור לנו להעריך כמה טוב יהיה להוסיף תכונה ספציפית כאשר יש לנו סט תכונות מסוים.

לצורך הפרויקט, נגדיר שתי פונקציות Score ונבחן את התנהגויות האלגוריתמים עם שתי הפונקציות האלו.
בשתי הפונקציות כמובן, ניקח בחשבון את המחיר של התכונה הנוספת לסט התכונות הנתון.

- פונקציה A:

מוטיבציה – נרצה לראות כי הוספת תכונה תעזור לנו בחיזוי הסיווג. לשם כך נרצה לראות כי היא אכן קורולטיבית לתווית המטרה (לדוגמה, אם אנו מתעסקים בבעיית חיזוי מצטייני סמסטר, כנראה שלא נרצה לקנות תכונה כמו "מספר עוקבים ברשתות החברתיות").
נרצה להבחין כי תכונה מוסיפה לנו מידע חדש שלא קיים לנו בסט התכונות הקיימות (לדוגמה, אם יש לנו תכונה שהיא גיל, לא נרצה לקנות תכונה שהוא פונקציה לינארית של הגיל).
כמובן שכיאה לבעיה כמו שלנו, ככל שהתכונה יקרה יותר, כך בהיבט כולל פחות משתלם לנו לקנות אותה.
לסיכום, נסתכל על שלושה פרמטרים בהינתן תכונה f' ותגית מטרה $target$-

a) תכונה "טובה" היא תכונה שקורולטיבית לתגית. נסמן $corr(f', target)$.
b) תכונה "טובה" היא תכונה שאינה קורולטיבית לתכונות אחרות. באופן פורמלי, עבור תכונה f' נרצה להביא למינימום את $\sum_{f \in features, f \neq f'} corr(f, f')$. נסמן $score\_corr(f', features)$.
c) תכונה "טובה" היא תכונה זולה. נסמן $cost(f')$.

לפיכך, נגדיר את הנוסחה הבאה:

$$\text{ScoreFunctionA} = \frac{corr(f', target) + \gamma \cdot score\_corr(f', features)}{cost(f')}$$

בהמשך נבצע ולדיציה בה נכוונן את ההיפרפרמטר γ.

- <u>פונקציה B:</u>

<u>מוטיבציה –</u> לבעיה כמו שלנו, ככל שהתכונה יקרה יותר, כך בהיבט כולל פחות משתלם לנו לקנות אותה. נרצה להיות בטוחים ככל הניתן כי הסיווג שלנו נכון. לשם כך נגדיר את המושג הבא:

<u>ודאות הסיווג</u>

באלגוריתמים שנציע אנו נשתמש באלגוריתם למידה בסיסי בקבלת ההחלטות בכל שלב של האלגוריתם בזמן הסיווג.

הסיווג הסופי שלנו מתבצע על תת קבוצה $F' \subset F$ ולכן בכל שלב בדרך נוכל להשתמש בקבוצת התכונות $\bar{F}$ הקיימת לנו בשלב זה.

נסמן - $yes\_prob, no\_prob = learning\_algo.predict\_proba(sample)$.

בהנתן תכונה שאנו f ותת קבוצה $F' \subset F$, נריץ את אלגוריתם הסיווג הבסיסי שלנו על סט התכונות החדש $\bar{F} = \{f\} \cup F'$. באמצעות פונקציית $predictProba$ של אלגוריתם הסיווג נקבל את ה- yes_{prob}, no_{prob} ונגדיר את וודאות הסיווג כ-

1 - entropy(yes_prob, no_prob)

כאשר המטרה שלנו היא למקסם את פלט הפונקציה.

נסתכל על שני פרמטרים בהינתן תכונה f' –

c) תכונה "טובה" היא תכונה זולה. נסמן $cost(f')$.

d) תכונה "טובה" היא תכונה המשפרת את ודאות הסיווג. נסמן $certainty(f', features)$.

נגדיר את הפונקציה הבאה :

$$ScoreFunctionB = \frac{certainty(f', features)}{cost(f')}$$

בשלב הניסויים, נבחן איזו פונקציה מטיבה עם האלגוריתמים שפיתחנו.

- בהמשך, נציע אלגוריתם חיפוש מקומי שיקבל בכל איטרציה קבוצת תכונות נתונה ויחשב עבורה ערך Score. האלגוריתם "ימקסם" את פונקציית ה-Score במטרה למצוא את קבוצת התכונות "הטובה" ביותר. לצורך כך, נרצה להכליל את הגדרת ה-Score עבור קלט הכולל קבוצת תכונות בלבד (ולא להסתפק בהגדרה עבור קלט הכולל קבוצת תכונות וַתכונה חדשה).

לצורך כך נסמן ב-features את קבוצת התכונות הנתונה וכן ב-S פונקציית Score כלשהי (פונקציה A או B) של קבוצת תכונות ותכונה חדשה כפי שהגדרנו קודם לכן. לפיכך, נגדיר: $\sum_{f\in features} S(features\backslash\{f\}, f)$.

במילים אחרות, על מנת להעריך את ערך ה-Score של קבוצת תכונות בלבד, נסכום את ערכי ה-Score על פני כל התכונות בקבוצה- כל תכונה f, נחשב את ערך ה-Score לפי פונקציה A או B כפי שהגדרנו קודם (כאמור, הקלט לפונקציה כנ"ל הוא קבוצת תכונות וַתכונה חדשה) כאשר הקלט לפונקציה הוא קבוצת התכונות ללא התכונה f והתכונה החדשה היא f.

אלגוריתם חיפוש מקומי

הקדמה

נרצה להציע אלגוריתם מתוחכם שפותר את הבעיה שהצגנו. לשם כך, ניעזר באלגוריתמי החיפוש המקומי שלמדנו בקורס.

נרצה להסתכל על הבעיה המוצגת כבעיית אופטימיזציה אותה נפתור לפי חיפוש מקומי. לשם כך, נגדיר גרף מכוון שצומת בו הוא קבוצת תכונות אפשרית. קשת תחבר בין צומת A ל-B אם קבוצת התכונות שצומת A מייצג שונה מקבוצת התכונות שצומת B מייצג בתכונה אחת בלבד, כך שדרגת היציאה של כל צומת כנ"ל שווה למספר התכונות שלא חלק מקבוצת התכונות שהצומת מייצג. ההיוריסטיקה על כל צומת היא לפי פונקציית ה-Score שהגדרנו קודם.

הגדרה פורמלית של מרחב החיפוש

ראשית, לצורך פשטות, נקבע סימונים:

- K- קבוצת התכונות הנתונות לאלגוריתם (ניתן כקלט לאלגוריתם בשלב ה-predict).
- D- מספר התכונות האפשריות לכל דוגמה (מימד המרחב הוקטורי שכל הדוגמאות נלקחות ממנו).
- C- וקטור בגודל D אשר בכל כניסה $0 \leq i < D$ שלו מחיר התכונה ה-i. נסמן את מחיר התכונה ה-i כ-$C[i]$.

* $\mathcal{P}$- סימון לקבוצת חזקה, למשל $\mathcal{P}(A) = \{B \mid B \subseteq A\}$, כלומר $\mathcal{P}(A)$ היא קבוצת כל תתי הקבוצות של A.

נגדיר כעת את מרחב החיפוש $\mathcal{S} = (S, O, I)$, כאשר-

- **קבוצת המצבים:**

קבוצות המצבים היא קבוצת כל קבוצות התכונות האפשריות שניתן להוסיף לקבוצת התכונות הנתונות.

$$S = \{\mathrm{P} \cup \mathrm{K} \mid P \in \mathcal{P}([D] \backslash K)\}$$

- **קבוצת האופרטורים:**

ניתן לעבור ממצב אחד לעוקבו בתנאי שקבוצת התכונות המיוצגת ע"י המצב הראשון וקבוצת התכונות המיוצגת ע"י המצב השני נבדלים זה מזה בתכונה אחת בלבד.

$$O = \{o^{s_1,s_2} \mid s_1, s_2 \in S \wedge |s_2 \setminus s_1| = 1\}$$

- **המצב ההתחלתי:**

המצב הכולל את קבוצת התכונות הנתונות בלבד.

$$I = K$$

- **היוריסטיקה:**

לכל מצב $s \in S$ נגדיר ערך היוריסטי:

$$H(s) = \begin{cases} Score(s), & if \sum_{f \in s} C[f] \leq \tilde{C} \\ -\infty, & else \end{cases}$$

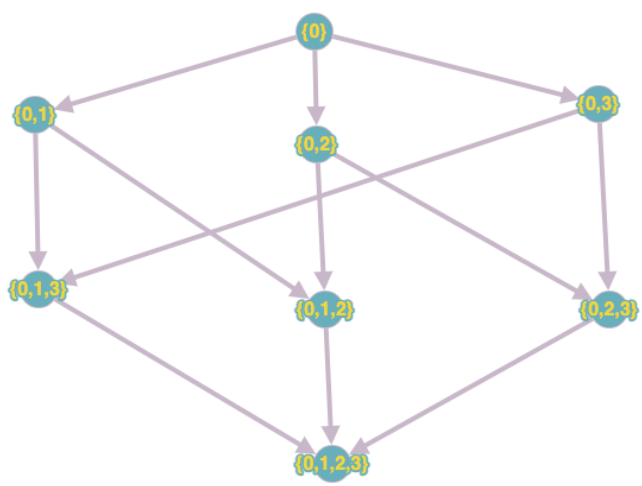
כלומר, הערך ההיוריסטי של כל מצב $s \in S$ יוגדר להיות ערך אחת מפונקציות ה-Score שהגדרנו בפרקים הקודמים אם המחיר הכולל של כל התכונות במצב s הוא לכל היותר התקציב המקסימלי המתקבל כקלט בשלב ה-Predict, אחרת $-\infty$.

<u>דוגמה</u>

נניח שקבוצת התכונות האפשריות היא $D = 4\ , [D] = \{0, 1, 2, 3\}$.

כמו כן, נניח שקבוצת התכונות הנתונות בתחילת האלגוריתם היא $K = \{0\}$.

הגרף המתקבל:



קיבלנו מצומת $\{0\}$ שלוש קשתות לצמתים $\{0,1\}, \{0,2\}, \{0,3\}$ וזאת מכיוון שכל אחד מהצמתים הללו נבדל מהצומת $\{0\}$ בתכונה אחת בלבד.

נשים לב שערך ההיוריסטיקה הוא מדד לטיב קבוצת התכונות. לפיכך, אם נרצה למצוא את קבוצת התכונות "הטובה ביותר" (ממנה נרצה לרכוש את התכונות החסרות לנו) נוכל באופן שקול למצוא את המצב בעל הערך ההיוריסטי הגדול ביותר.

נשתמש בסימונים שהגדרנו על מנת להגדיר בצורה פורמלית את בעיית האופטימיזציה אותה נרצה לפתור:

$$\max_{s \in S} H(s)$$

במילים אחרות, נחפש את המצב $s \in S$ עבורו הערך ההיוריסטי **מקסימלי**.

נפתור את בעיית האופטימיזציה ע"י הרצת אלגוריתם לחיפוש היוריסטי מקומי, כפי שלמדנו בקורס (כדוגמת Steepest Ascent hill-climbing, Stochastic hill-climbing, beam search וכ').

נציג כעת את האלגוריתם בצורה פורמלית. לצורך כך נבחר אלגוריתם למידה כלשהו (כדוגמת KNN או $ID3$) ואלגוריתם לחיפוש מקומי.

אימון:

- קלט: לפי האלגוריתם הכללי המפורט במבוא.

1. שמור את קבוצת הדוגמאות והתיוגים $\{X, Y\}$.
2. שמור את קבוצת המחירים לתכונות C.

סיווג:

- קלט ופלט: לפי האלגוריתם הכללי המפורט במבוא.

1. הרץ את האלגוריתם לחיפוש מקומי שבחרת החל מהמצב ההתחלתי K ומצא את המצב s^* עבורו הערך היוריסטי $H(s^*)$ מקסימלי.
2. עבור על התכונות ב-s^*. הוסף כל תכונה ב-s^* שאיננה בקבוצת התכונות ההתחלתית K לקבוצת התכונות ההתחלתית ("רכוש את התכונה"): $K \cup s^* \to K$.
3. הרץ את שלב האימון של אלגוריתם הלמידה שבחרת (כאשר התכונות הנתונות לכל דוגמה הן התכונות בקבוצת התכונות הנתונות שחישבנו בשלב 2) עבור קבוצת הדוגמאות והתיוגים ששמרת בשלב האימון.
4. הרץ את שלב הסיווג של אלגוריתם הלמידה שבחרת.
5. החזר את הפלט שהתקבל.

מימשנו את האלגוריתם בקובץ local_search_algorithm.py.

היתרונות של האלגוריתם הנ"ל הוא שאנו מנצלים את כל התקציב שברשותנו ומוסיפים כמה שיותר מידע (תכונות), שכן מאופן הגדרת ההיוריסטיקה, האלגוריתם הנ"ל תמיד ימצא מצב עם מספר אופטימלי של תכונות.

כמו כן, בניגוד לאלגוריתמים הנאיביים שהצענו, באלגוריתם זה אנו לא תלויים כלל באקראיות ואנו אף משתמשים בכלים מתוחכמים אותם למדנו בקורס על מנת לחפש את המצב "הטוב ביותר" מבין כלל המצבים- אנו לא משתמשים באלגוריתם חמדני שמסתכל רק צעד אחד קדימה בכל שלב. כעת אנו מצליחים להתחשב במספר פרמטרים שונים (לפיהם מוגדרת פונקציית ה-Score) ולא רק בפרמטר יחיד (כפי שאלגוריתם חמדני פועל) ובכך מתמודדים עם קונפליקט אופטימליות מספר התכונות כנגד תרומתן.

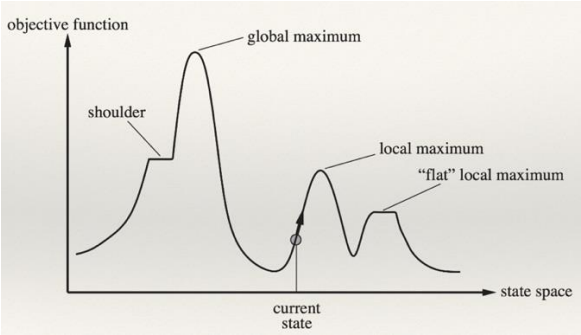
יתרה מכך, מתוקף הגדרת ההיוריסטיקה, אנו מבטיחים שתמיד נקבל קבוצת תכונות חוקית, כלומר כזו שלא עוברת את התקציב הנתון לנו. מכיוון שאנו מריצים את אלגוריתם החיפוש המקומי בשלב הסיווג (כאשר כבר ניתן לנו התקציב), אנו למעשה מצליחים להגדיר אלגוריתם שמסוגל לעבוד עבור כל תקציב אפשרי.

עם זאת, לאלגוריתם הנ"ל מספר חסרונות. ראשית, סיבוכיות הזמן והמקום. כעת, מלבד הרצת אלגוריתם למידה כלשהו (שעלול לדרוש הקצאה גדולה של זמן ומקום), אנו מפתחים גרף ומריצים עליו חיפוש מקומי (דבר שמגדיל את הזמן והמקום הנדרשים). עם זאת, נשים לב שאנו לא מפתחים את כל הגרף, אלא לכל היותר מצב אחד וכל שכניו ובכך חוסכים בזיכרון.

על מנת להתמודד עם החסרונות הללו, נבצע ניסויים במטרה להחליט על אלגוריתם החיפוש וסוג ההיוריסטיקה (פונקציית ה-Score) שיביא אותנו לתוצאות הטובות ביותר בזמן סביר.

למעשה, נוכל בעתיד אף לבצע שיפורים באלגוריתם- כמו למשל הגדרת היוריסטיקה טובה יותר וכ'.

כמו כן, פונקציית ה-Score הקובעת את ההיוריסטיקה איננה בהכרח קעורה- כלומר, האלגוריתם לחיפוש מקומי לא בהכרח ימצא מקסימום (או לחילופין, עלול להיתקע במקסימום מקומי ולא גלובלי)- לכן ביצועי האלגוריתם עלולים להיפגע (כמובן שנוכל להשתמש באלגוריתמי חיפוש מקומי מתוחכמים יותר, אלא שבעיית הקעירות תישאר קיימת).



אלגוריתם חיפוש בהשראת אלגוריתמים גנטיים

הקדמה

נרצה להציע אלגוריתם נוסף שבבסיסו עומד אלגוריתם גנטי. לשם כך נרצה להרחיב קצת בנושא של אלגוריתמים גנטיים.

אלגוריתמים גנטיים מבצעים תהליכי ברירה מלאכותית המבוססים על עקרונות ביולוגים בהשראת האבולוציה.

רגע של ביולוגיה – צ'ארלס דרווין התחיל את פיתוח התורה האבולוציונית בבסיסה עומדים עקרונות מנגנוני ההורשה והגנטיקה. העקרון המנחה הוא תהליך הברירה הטבעית.
תהליך הברירה הטבעית אומר כי תכונות תורשתיות של יצורים נעשות נפוצות יותר ויותר מדור לדור ככל שהן תורמות יותר לרביית הפרט ויכולת הישרדותו. באוכלוסיית יצורים, למי שמעמיד הרבה צאצאים יש סיכוי גדול יותר להוריש להם את התכונות התורשתיות שלו.

העקרון באלגוריתמים גנטיים אומר שמבין מרחב הפתרונות, בדיוק כמו בטבע, רק "המתאימים שורדים".
נוכל לשלב אלמנטים של פתרונות אפשרים לבעיה תוך הפעלת הליכים של ברירה מלאכותית כדי "לבחור" מועמדים לדור הבא.
ההצלחה של רעיון זה נובעת מההצלחה של האבולוציה בפתרון בעיות אמיתיות.
בפשטות, ניקח פתרונות אפשריים לבעיה, נבחר פתרונות שנגדיר כ"מתאימים", נערבב אותם ונוסיף רעש נקבל דור חדש של פתרונות הקרוב צעד אחד יותר לפתרון הבעיה.

הגדרות

- גנום ($genome$) – פתרון אפשרי לבעיה. נהוג לייצג כווקטור בינארי, כאשר האינדקס ה-i הוא 1 אם ורק אם בפתרון יש שימוש בתכונה ה-i.
- אוכלוסייה ($population$) – אוסף של גנומים. מייצג את אוסף הפתרונות שאנו בוחנים בדור הנוכחי.
- פונקציית כשירות ($Fitness$) – פונקציה שבהינתן גנום מחזירה את "טיב" הגנום.
- תהליך זיווג ($crossover$) – לקיחת שני גנומים מהאוכלוסייה ויצירת גנום חדש. הגנום החדש מכיל מידע גנטי של שני הוריו (בדרך כל מכיל את הגנים המובחרים של הוריו).
- יצירת מוטציות ($mutation$) – שינוי מספר תכונות של גנום יחיד כדי למנוע חזרה של חלקים שלמים במחרוזת הביטים. מתבצעת באחוזים נמוכים (בהסתברות נמוכה) משום שאם הייתה מתבצעת בסבירות גבוהה, האלגוריתם היה הופך להיות אקראי.

- שיערוך ($evaluation$) – בחירת הגנומים שישתתפו בזיווג לפי פונקציית הכשירות.

באופן פורמלי, אם $f: \mathbb{R}^n \to \mathbb{R}$ פונקציות הכשירות אותה אנו רוצים **למקסם** :

- $\bar{x} = (x_1, .. x_n)$ הוא **גנום** במודל ושייך לתחום של פונקציית הכשירות.
- $x_1, .., x_n$ הם **כרומוזומים**.
- $P_i = \{\bar{x}_1, \dots, \bar{x}_k\}$ היא **האוכלוסיה** בדור i.
- $f(\bar{x}) \in \mathbb{R}$ הוא **השיערוך** של הגנום $\bar{x}$.
- בהינתן $\bar{x}_1, \bar{x}_2$, **הזיווג** שלהם הוא גנום $\bar{x}_3$ המורכב מהכרומוזומים של הוריו.
- **מוטציה** של $\bar{x}$ הוא שינוי בהסתברות קטנה כלשהי p של חלק מהכרומוזומים של $\bar{x}$.

<u>נקביל את המונחים ההגדרות שהצגנו לבעיה שלנו:</u>

- גנום ($genome$) – וקטור בינארי, כאשר האינדקס ה-i הוא 1 אם ורק אם בפתרון יש שימוש ב-feature ה-i. נציין כי אנו נשתמש אך ורק בפתרונות חוקיים לבעיה (עלות הפתרון ≥ העלות המקסימלית).
- אוכלוסייה ($population$) – אוסף של גנומים. מייצג את אוסף הפתרונות שאנו בוחנים בדור הנוכחי. כל הפתרונות חוקיים.
- שיערוך ($evaluation$) – נשתמש במסווג בסיס מסוג ריגרסור לינארי. נייצר מהדוגמאות שניתנו לנו בשלב ה-train סט אימון וסט ולידציה, וכל פעם נאמן מסווג עם סט האימון עבור התכונות שהפתרון הנוכחי מייצג, ונעריך את איכותו לפי ה-accuracy שיקבל על סט הולידציה.

נציין שבספרות שימוש באלגוריתמים גנטיים הוא דרך פופולרית לביצוע feature selection (באופן כללי זו דרך טובה לפתרון בעיית בחירת תת קבוצה מיטבית).
בבסיסה של הבעיה אנו מנסים לפתור עומד אלמנט של בחירת תכונות, ולכן יש הגיון בבחירת שילוב אלגוריתם זה בפתרון שאנו מציעים.

בפתרון שלנו אנו משתמש בספרית deap שמממשת אלגוריתם גנטי ($esSimple$) שמיועד לפתור את בעיית בחירת תת קבוצה מיטבית.

אימון:

- קלט: לפי האלגוריתם הכללי המפורט במבוא.

1. שמור את קבוצת הדוגמאות והתיוגים $\{X, Y\}$.
2. שמור את קבוצת המחירים לתכונות C.

סיווג:

- קלט ופלט: לפי האלגוריתם הכללי המפורט במבוא.

1. הרץ אלגוריתם גנטי למציאת תת קבוצה של קבוצת התכונות ($HOF - hall\ of\ fame$).
   1.1. האוכלוסייה של האלגוריתם תאותחל עם פתרונות חוקיים בלבד.
   1.2. בפונקציית הכשירות יפסלו פתרונות חדשים שאינם חוקיים (יוכלים להגיע מהתליך הזיווג).
   1.3. פונקציית הכשירות תעריך את הדיוק של סיווג באמצעות התכונות הנוכחיות בלבד.
   1.4. נשים לב כי כל פתרון שהאלגוריתם יבחן ישתמש בכל התכונות החינמיות (בהסתברות גבוהה מאוד) מאופן איתחול האוכלוסייה וביצוע זיווג בין פתרונות חוקיים (בהסתברות כי יש מוטציות).
2. מהפתרונות החוקיים ב-HOF צור תת קבוצה של כל הפתרונות שמוערכים בכשירות הגבוהה ביותר.
3. בחר אחד מהפתרונות באופן רנדומלי (ב-HOF כל הפתרונות עם אותו ניקוד, לכן לא אכפת לנו את מי מהם נבחר).
4. הרץ את שלב הסיווג של אלגוריתם הלמידה שבחרת עם התכונות שנמצאות בפתרון שבחרת.
5. החזר את הפלט שהתקבל.

מימשנו את האלגוריתם בקובץ genetic_algorithm.py.

יתרונות וחסרונות האלגוריתם

האלגוריתם הנ"ל משתמש בכל התקציב שברשותו על מנת לרכוש תכונות ולהגדיל את המידע שברשותו.

האלגוריתם משתמש בפתרון בעיה כללית יותר – מציאת תת קבוצה מיטבית כקופסה שחורה.

תת הקבוצה המיטבית מוגדרת לפי פונקציית המחיר שהגדרנו.

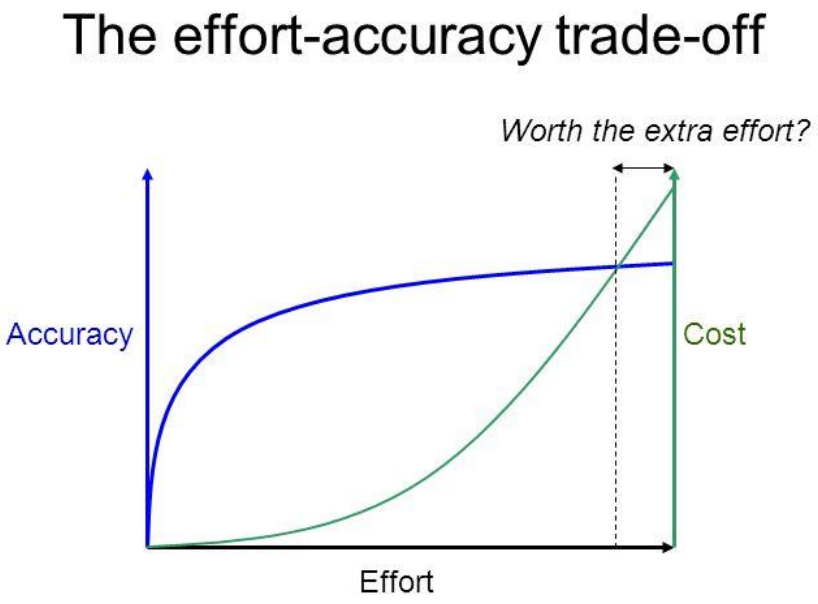
חסרון בולט מאוד של שיטה זו הוא זמן הריצה (ינותח באופן מלא בשלב הניסויים).

בכל הרצה, אנו מפתחים 10 דורות לאוכלוסייה, ובוחנים פתרונות מתוך קבוצה בגודל $2^{\#features}$, כלומר הפתרון הוא מעריכי בגודל הקלט, ורחוק מלהיות אופטימלי במונחים של זמן ריצה.

אנו מצפים שלאלגוריתם זה יהיו ביצועים מצוינים, ולעומת האלגוריתם האחרים, שזמן ריצתו יהיה משמעותית גדול יותר.

בכך אלגוריתם זה מהווה דוגמה ל- accuracy effort tradeoff .

מונח זה מוכר מאוד בתחום ובעצם מדגים את ה-tradeoff ההגיוני בין יעילות האלגוריתם לבין "ההשקעה" באלגוריתם, שבמקרה הזה באה לידי ביטוי בזמן ריצה.



מתודולוגיית הניסויים

בשלב זה נרצה לאמוד את הצלחת הפתרונות שהצענו. אם נחזור לבעיה שממנה התחלנו, בעולם האמיתי, הפתרון יוגדר להיות מוצלח אם הבאים מתקיימים:

- הפתרון שמיש – ניתן לקבל את הכרעת הפתרון בזמן שיוגדר להיות סביר (לדוגמא, אם נחזור לבעיית הרופא והחולה- אנו לא יכולים לתת לאלגוריתם שלנו לרוץ כל זמן שירצה, התשובה צריכה להיות תוך פרק זמן שיהיה רלוונטי לביצוע הבדיקות בזמן).
- הפתרון בעל איכות תוצאה המצדיקה את השימוש בו – אל מול ה-base lines שהגדרנו, האלגוריתמים שלנו משתמשים במשאבים רבים (זיכרון וזמן), ולכן שימוש בהם צריך לתת יעילות שאינה מגיעה בפתרונות הנאיביים. נרצה להצדיק את ה-Trade Of שבין "יוקר" הפתרון אל בין איכותו.

נרצה להגדיר את טיב האלגוריתמים שלנו. לשם כך נגדיר פרמטרים לאיכות פתרון אותם נבחן בניסויים:

1. זמן ריצה – נרצה לבחון את זמן הריצה של האלגוריתמים שלנו ביחס ל-base line שיצרנו.
   נרצה להשוות את שתי הגישות שהצענו במובן של זמני ריצה, וכן לבחון אותן אל מול האלגוריתמים הנאיביים לפתרון הבעיה שהצגנו בהתחלה.
   נציין כי אנו צופים מראש כי זמני הריצה של הפתרונות שלנו יהיו משמעותית ארוכים יותר מזמני הריצה של הפתרונות הנאיביים.
   * באלגוריתם הגנטי אנו בוחנים מצבים חוקיים מתוך קבוצה בגודל $2^{\#features}$, כלומר זמן הריצה הוא מעריכי בכמות התכונות.

2. דיוק הפתרון – בסופו של דבר, נרצה שהאלגוריתם שהצענו יגיע לתוצאת דיוק גבוהה.
   נשים לב שכל אחד מהאלגוריתמים שלנו בנויים על גבי מספר פרמטרים. ראשית נבחן את תרומת כל אחד מהם לדיוק הפתרון.
   לאחר מכן נשקלל מכל התוצאות את הקומבינציות הטובות ביותר ונבדוק מה דיוק כל אלגוריתם.

דברים שנרצה לבדוק בחלק זה :

- היפרפרמטרים לפונקציית Score. נזכיר שבפונקציית ScoreA השתמשנו בסקלר הכופל את אחד הגורמים במשוואה כדי למשקל את החשיבות שלהם. נרצה לבדוק מי מהגורמים הינו בעל השפעה מכרעת יותר בקביעת איכות התכונה, ובשביל זה נבצע כיוונון להיפרפרמטר זה. שלב זה יגודר כולידציה.

- נרצה לבדוק איזו מבין פונקציות Score שהצענו מביאה לדיוק גבוה יותר בפתרון. כזכור, פונקציית ScoreA באה להגדיר תכונה "טובה" כאחת שקורולטיבית לתגית ולא קורולטיבית לתכונות שכבר קיימות בסל התכונות. מנגד, פונקציית ScoreB משתמשת בהגדרה שלנו לוודאות הפתרון. נרצה לראות אילו מבין השניים מצליחה לממש את מטרתה בצורה טובה יותר.
- האלגוריתמים שלנו משתמשים במסווגי בסיס (Classifiers). נרצה לראות האם קיים קשר בין מסווג הבסיס לבין הצלחת אחד האלגוריתמים. אנו צופים כי לא יהיה קיים קשר ישיר, אך במובני זמן ריצה יהיו הבדלים.

על מנת לקבל תוצאות ומסקנות מהימנות, ניקח שלושה סוגים שונים של dataset:

- Cardiovascular- מידע על אנשים החולים במחלת לב כתלות בתכונותיהם (הרגלים, נתונים פיזיים וכ').
- Water- מידע על איכות מים כתלות בחומרים הנמצאים במים.
- Roads- מידע על התרחשות תאונת דרכים בכביש מסוים כתלות בתנאי הדרך.

כל dataset נחלק לשלוש קבוצות:

- קטן – 80 דוגמאות
- בינוני – 200 דוגמאות
- גדול – 600 דוגמאות

את הניסויים נריץ על שלושת סוגי ה-dataset כאשר נבחן את כל הקבוצות (סה"כ תשע הרצות, לכל dataset נריץ שלוש פעמים- פעם אחת על כל קבוצה) ובכך נבחן את התוצאות עבור dataset שונים בגדלים שונים.

מימשנו את הניסויים בקובץ General/experiments.py. על מנת להריצם, יש להריץ את הקובץ experiments.py באמצעות ה-termial ע"י הרצת הפקודה python ./General/experiments.py. או דרך סביבת עבודה IDE כלשהי לבחירתכם. הרצת הניסויים בצורה זו תפלוט את הגרפים המתאימים.

כעת נפרט את תוצאות הניסויים, נצרף את הפלטים הרלוונטים וכן ננתח את הממצאים:

ולידציה: היפרפרמטר לפונקציית המחיר A

רקע

ניזכר כי בפרק על פונקציית המחיר הגדרנו את הפונקציה :

$$\text{ScoreFunctionA} = \frac{corr(f', target) + \gamma \cdot score\_corr(f', features)}{cost(f')}$$

כאשר הפרמטרים הם-

a) קורולטיביות לתגית.

b) קורולטיביות לתכונות קיימות.

c) מחיר.

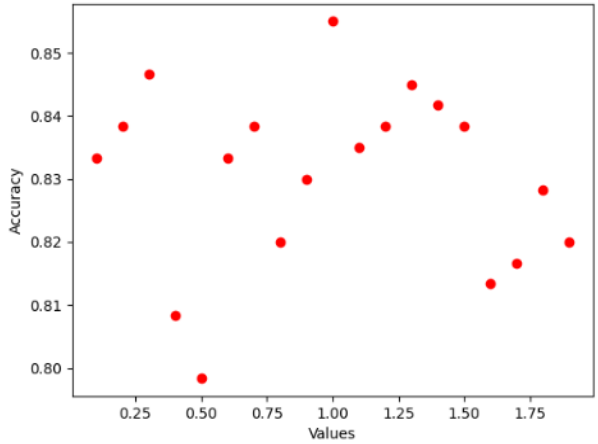
כעת נרצה לכוונן את הפרמטר γ שמציין את היחס בין a ל- b.

אופן ביצוע

הרצנו ולידציה באמצעות Kfold כפי שלמדנו בקורס על Cardiovascular dataset בינוני כדי לכוונן את ההיפרפרמטר הנ"ל. השתמשנו באלגוריתם החיפוש המקומי על מנת להשוות ערכי דיוק של ערכי γ שונים על מנת למצוא את ערך ה-γ שיניב ערך דיוק מקסימלי.

תוצאות

Cardiovascular - בינוני



קיבלנו כי הערך $\gamma = 1$ הינו האופטימלי, לכן נשתמש בו בכל הניסויים שלנו.

ניסוי: פונקציית המחיר הטובה ביותר

רקע לניסוי

במהלך הפרויקט פיתחנו תיאוריה המגדירה ניקוד לכל תכונה בהינתן תת קבוצה קיימת של תכונות.
המטרה הייתה לבדוק את היעילות לאלגוריתם של הוספת תכונה חדשה לסט התכונות הקיימות.
כדי לממש מטרה זו, היה עלינו להגדיר מה היא תכונה "טובה".

נקטנו בשתי גישות שהובילו לפיתוח שתי פונקציות מחיר (Score Function) שונות:

1. Score function A – פונקציה זו מסתכלת על שלושה מדדים.
    a. תכונה היא "טובה" אם היא קורולטיבית לתגית.
    b. תכונה היא "טובה" אם היא לא קורולטיבית לתכונות הקיימות.
    c. תכונה היא "טובה" אם היא זולה.
2. Score function B – פונקציה זו מסתכלת על שני מדדים.
    a. תכונה היא "טובה" אם הוספתה מגדילה את רמת הוודאות (שהוגדרה בפרק Score Function בדו"ח) של האלגוריתם.
    b. תכונה היא "טובה" אם היא זולה.

כעת, נרצה לבדוק איזו מהפונקציות הנ"ל היא הטובה ביותר. נגדיר את הפונקציה הטובה ביותר במובן של אחוזי דיוק.

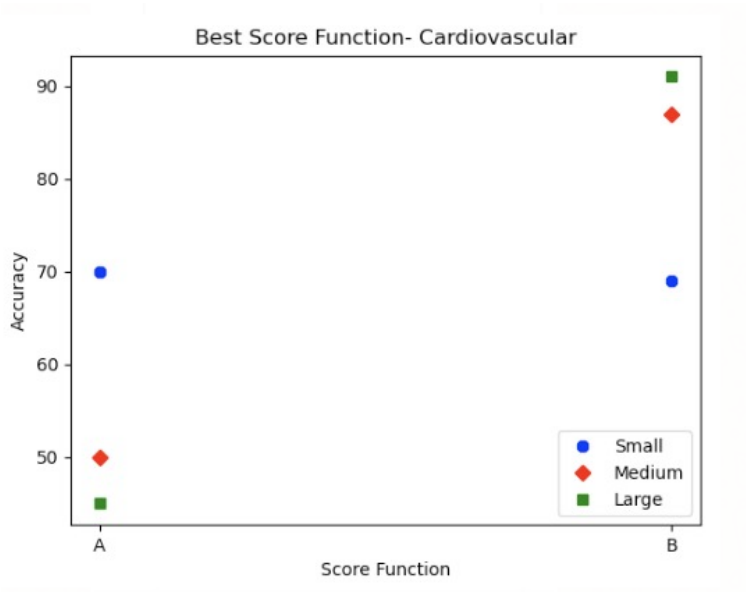
אופן ביצוע הניסוי

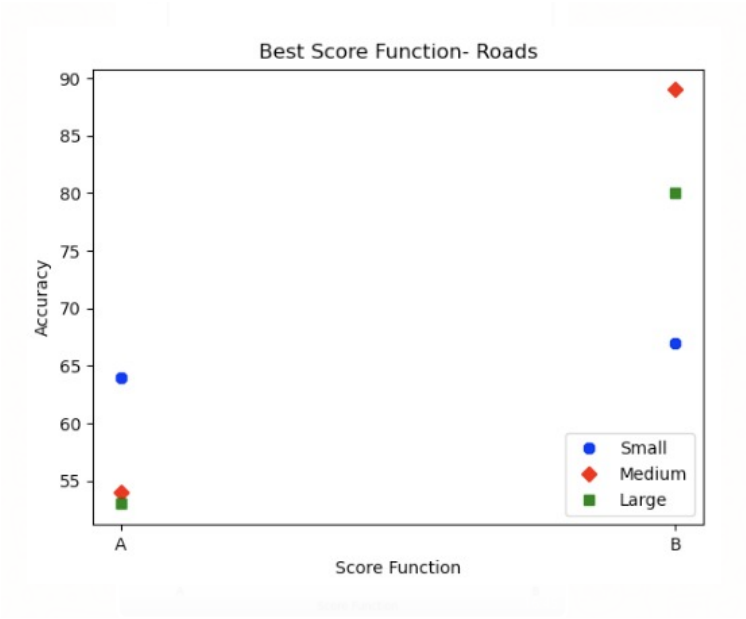
נבדוק את הפונקציות השונות על כל אחד מה-dataset השונים ולכל הגדלים (קטן, בינוני וגדול) וכך נשווה את אחוזי הדיוק של כל האלגוריתמים. שתי הפונקציות ניתנו לאלגוריתם החיפוש המקומי ואותחלו עם פרמטרים זהים.

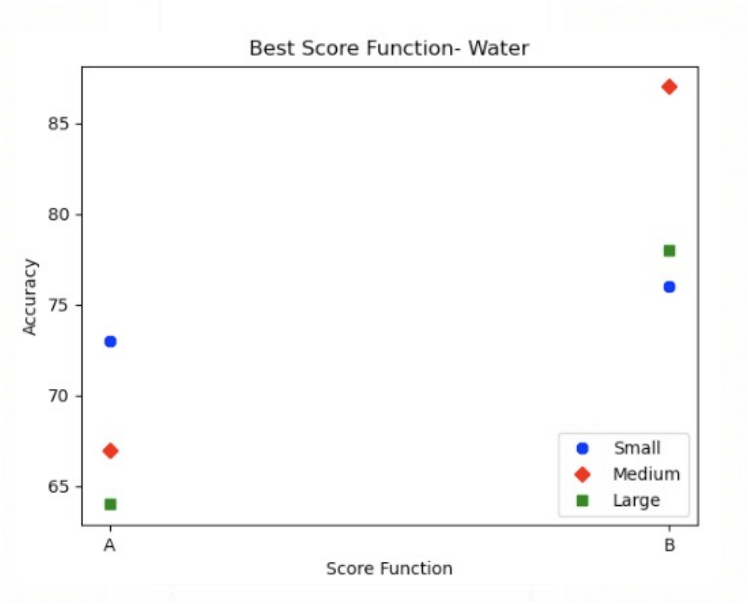
השערות

אנו מניחים שבמובנים של הצלחה בניסוי כפי שהוגדרו (אחוזי דיוק), פונקציה B אמורה להיות טובה יותר כי היא נותנת ניקוד לפי ודאות סיווג.
בנוסף, אנו צופים ביצועים שונים לפונקציה A על גדלים שונים של דאטה סטים בשל הרחבת האפשרות לקורולטיביות להיות משמעותית יותר (ככל שהעמודה גדולה יותר מדד זה מדויק יותר).

תוצאות


Best Score Function- Cardiovascular
Accuracy
90
80
70
60
50
A
B
Score Function
Small
Medium
Large


Best Score Function- Roads
Accuracy
90
85
80
75
70
65
60
55
A
B
Score Function
Small
Medium
Large



בכל אחד מן הגרפים מוצגת תוצאת הדיוק שהשיג האלגוריתם כתלות ב-score function בה השתמש.

ניתוח תוצאות

מהתוצאות עולה כי עבור datasets קטנים הביצועים של הפונקציות במובנים של דיוק הינו זהה.

אך כאשר מספר הדוגמאות עולה, פונקציה B מנצחת באופן משמעותי. בנוסף, נשים לב כי סוג ה-dataset אינו משפיע על התוצאות.

מכך אנו מסיקים מספר מסקנות –

- ההגדרה שביצענו עבור ודאות של שלב באלגוריתם הינה הגיונית ועובדת. אכן קיימת התאמה בין כמות הודאות הגדלה באלגוריתם איטרטיבי לבין אחוז הדיוק.
- ב-datasets קטנים פונקציה A משיגה אחוז דיוק דומה לזה של B. אנו יכולים לשער שזה קורה כי עבור עמודות תכונות קטנות יותר, האקספרסיביות של פונקציה B קטנה יותר כי האימון מתבצע על dataset קטן יותר (החוזקה של פונקציה B מתבטאת כאשר ה-dataset לא קטן).

ניסוי: מסווג הבסיס הטוב ביותר

רקע לניסוי

בכל האלגוריתמים שפיתחנו אנו עושים שימוש במסווג בסיס. מסווג בינארי זה יכול להיות פרמטר המשפיע על ביצועי האלגוריתם. נרצה לחקור את השפעת מסווג הבסיס על ביצועי האלגוריתם מבחינת דיוק.

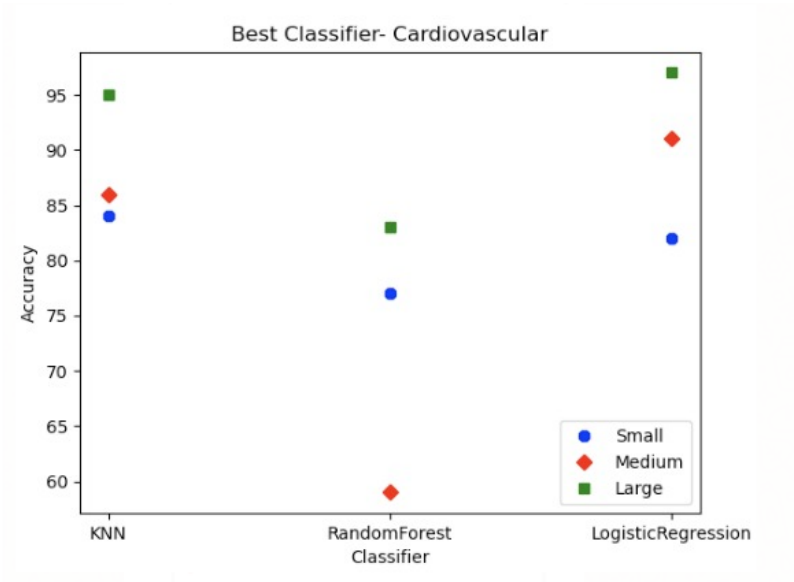
אופן ביצוע הניסוי

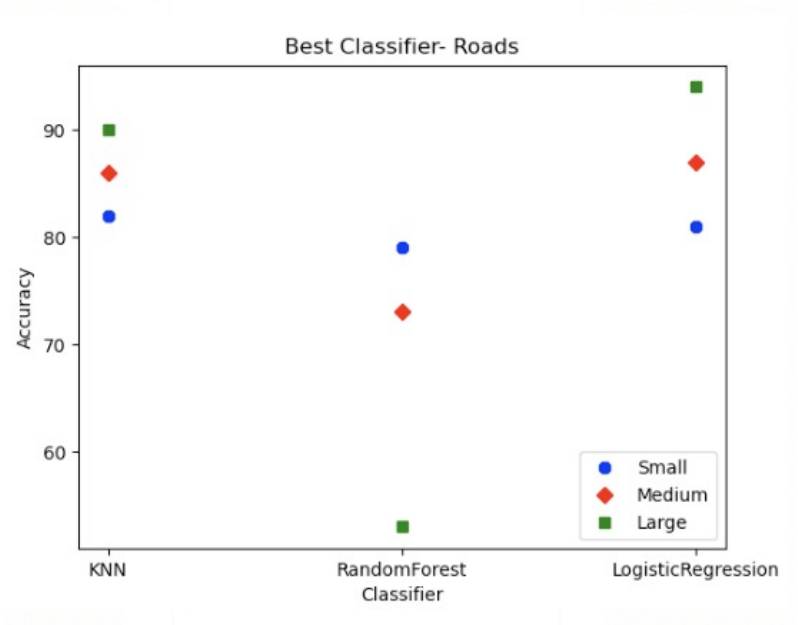
נבדוק את השפעת מסווג הבסיס הניתן לאלגוריתם על כל אחד מה dataset-השונים ולכל הגדלים (קטן, בינוני וגדול) וכך נשווה את אחוזי הדיוק. את הניסוי ביצענו עם אלגוריתם החיפוש המקומי ועם פונקציה B.

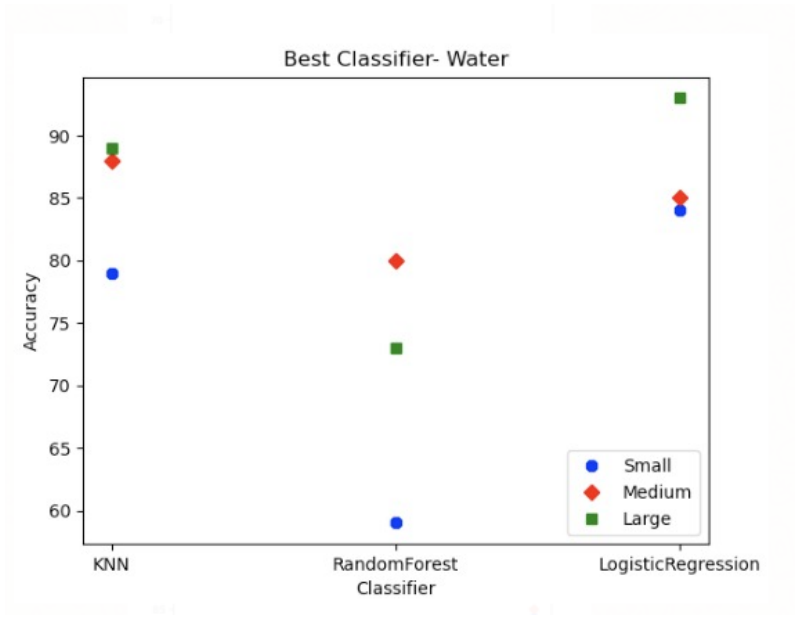
השערות

אין לנו השערות ספציפיות לגבי איזה מסווג אמור לתת ביצועים מרשימים באופן מובהק, אך מניסיון העבר שלנו אנו משערים שה-Logistic Regressor אמור לבצע עבודה טובה כי הוא פחות מוגבל לעומת מסווגים אחרים שבדקנו (KNN מצפה ל-data שפריד מסוג מסוים, וכן יער שבנוי מעצים המפרידים data לינארית בכל מימד).

תוצאות







ניתוח תוצאות

מהתוצאות ניתן להסיק כי כמעט תמיד ה-Random Forest משיג תוצאות פחות טובות משאר המסווגים וכן כי ה-Logistic Regressor ביצע על כל סוגי ה-data וכן על כל הגדלים את התוצאות הטובות ביותר.

נחזור ונאמר כי אנו מבינים כי ניסוי זה אינו חד משמעי וכי יכול להיות שעבור סוגי data שונים יתאימו מסווגים אחרים בשל תכונות ה-data ואופן התפלגותו, והמסקנה הנ"ל הינה עבור שלושת ה-datasets שבדקנו.

ניסוי: אלגוריתם בחירת הפיצ'רים הטוב ביותר

רקע לניסוי

במהלך הפרויקט הצענו מספר אלגוריתמי למידה תחת אילוצים:

אלגוריתמים נאיביים:

1. "האלגוריתם הריק"- סיווג לפי התכונות הנתונות בלבד.
2. "האלגוריתם הרנדומלי"- בוחר תכונות רנדומליות עד אשר מחיר התכונות הנבחרות עובר את התקציב הנתון.
3. "האלגוריתם החמדן"- בוחר תכונות לפי מחירן, מהזולה ביותר ועד היקרה ביותר, עד אשר מחיר התכונות הנבחרות עובר את התקציב הנתון.

אלגוריתם ביניים:

4. "אלגוריתם שונות מקסימלית"- בוחר תכונות לפי גודל השונות שלהן, מהגדול ביותר ועד הקטן ביותר, עד אשר מחיר התכונות הנבחרות עובר את התקציב הנתון.

אלגוריתמים מתוחכמים:

5. "אלגוריתם חיפוש מקומי"- בוחר תכונות לפי הרצת אלגוריתם חיפוש מקומי על פונקציית ה-Score.
6. "האלגוריתם הגנטי"- בוחר תכונות לפי הרצת אלגוריתם גנטי (התכונות המתאימות ביותר "שורדות").

כעת, נרצה לבדוק איזה מהאלגוריתמים הנ"ל הוא הטוב ביותר. נגדיר את האלגוריתם הטוב ביותר במובן של אחוזי דיוק וזמן הריצה.

אופן ביצוע הניסוי

נבדוק את הפונקציות השונות על כל אחד מה-dataset השונים ולכל הגדלים (קטן, בינוני וגדול) וכך נשווה את אחוזי הדיוק וזמן הריצה של כל האלגוריתמים כתלות בגודל ה-dataset בו אנו משתמשים (קטן, בינוני וגדול).
על מנת לבודד את המשתנים בניסוי ולהימנע מרעשים והשפעות הנובעים מהפרמטרים שכל אלגוריתם מקבל, נבחר פרמטרים קבועים לכל האלגוריתמים:

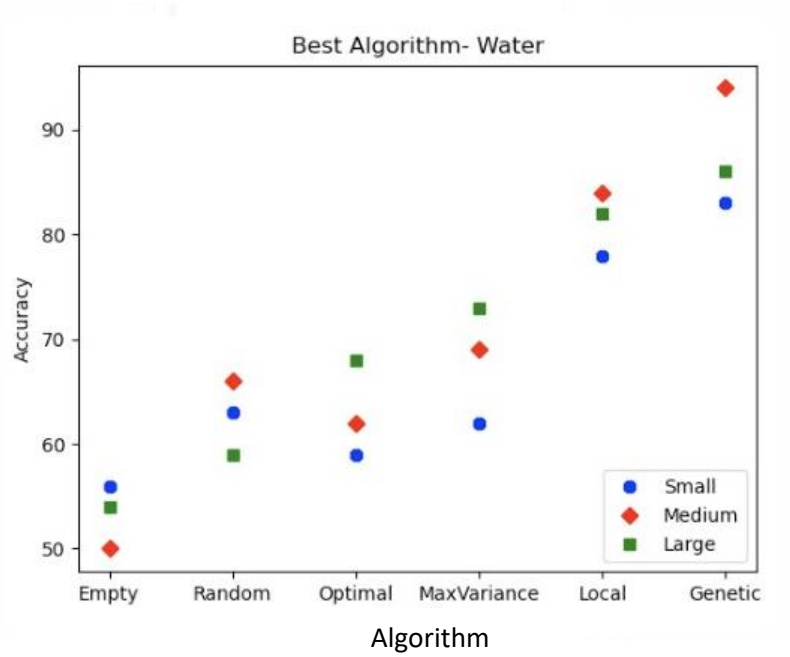
- נקבע שהמסווג שכל האלגוריתמים ישתמשו בו הוא Logistic Regressor.
- נקבע את התכונות הנתונות, מחירי התכונות והמחיר המקסימלי להיות זהים בין כל האלגוריתמים.
- עבור כל האלגוריתמים המשתמשים בפונקציית Score, נבחר כברירת מחדל את ScoreFunctionB.

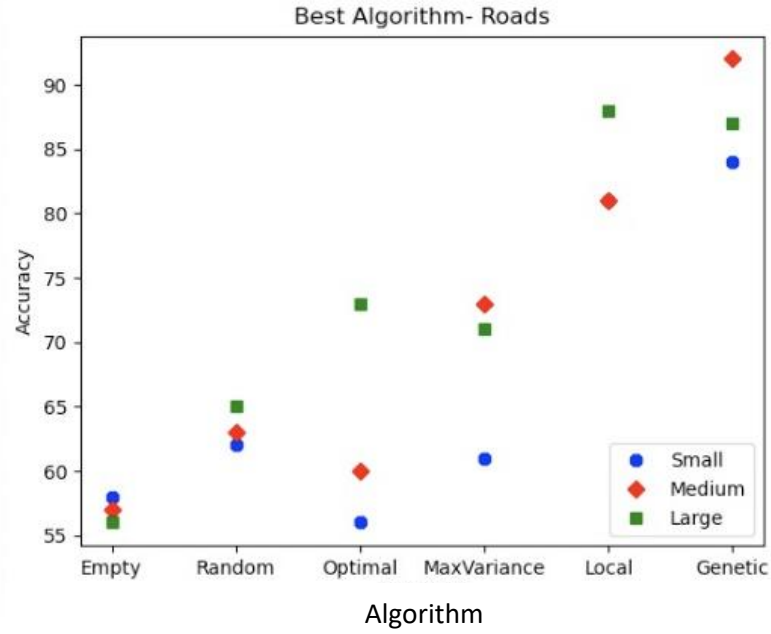
נשים לב כי את ניסוי זמן הריצה נבצע על dataset יחיד, כי אין משמעות להתפלגות הדאטה במדד זה.

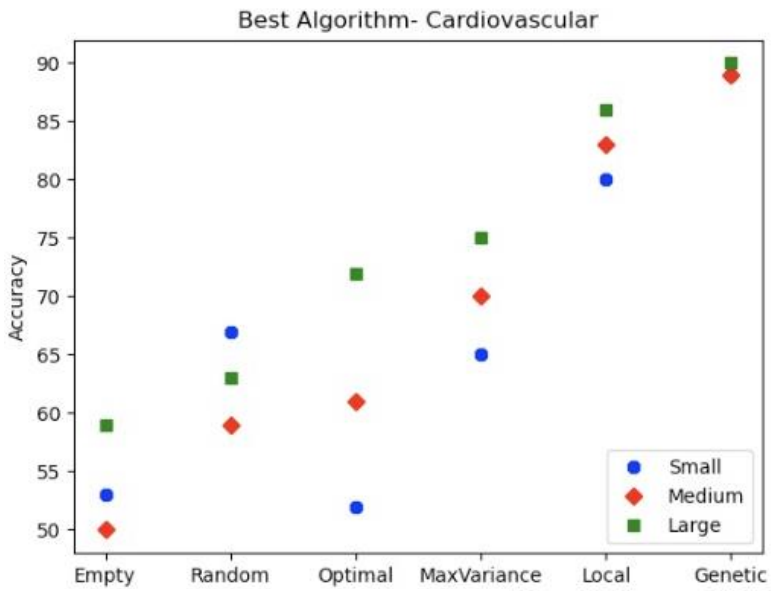
השערות

אנו מצפים שאחוזי הדיוק הטובים ביותר יתקבלו עבור האלגוריתמים המתוחכמים יותר, העושים שימוש בהיוריסטיקות, בכך בוחנים בצורה טובה יותר את הקשר שבין התכונות ומנצלים את כל המידע שביכולתם להשיג. נצפה שהדיוק הנמוך ביותר יתקבל עבור "האלגוריתם הריק", שכן הוא עושה שימוש במספר התכונות הקטן ביותר ובכך מסתפק במידע מאד בסיסי.
לעומת זאת, אנו מצפים שזמן הריצה של האלגוריתמים הנאיביים יהיה קצר משמעותית מזמן הריצה של האלגוריתמים המתוחכמים, שכן האלגוריתמים המתוחכמים כוללים בניית גרף, יצירת מצבים ושימוש בפונקציית ה-Score, שכן כל אלו עלולים לבוא לידי ביטוי בזמן ריצה ארוך.
כמו כן, נצפה שזמן הריצה של כל האלגוריתמים יעלה כתלות בגודל ה-dataset, בפרט נצפה להבדלים גדולים בזמן הריצה של האלגוריתמים המתוחכמים במיוחד.

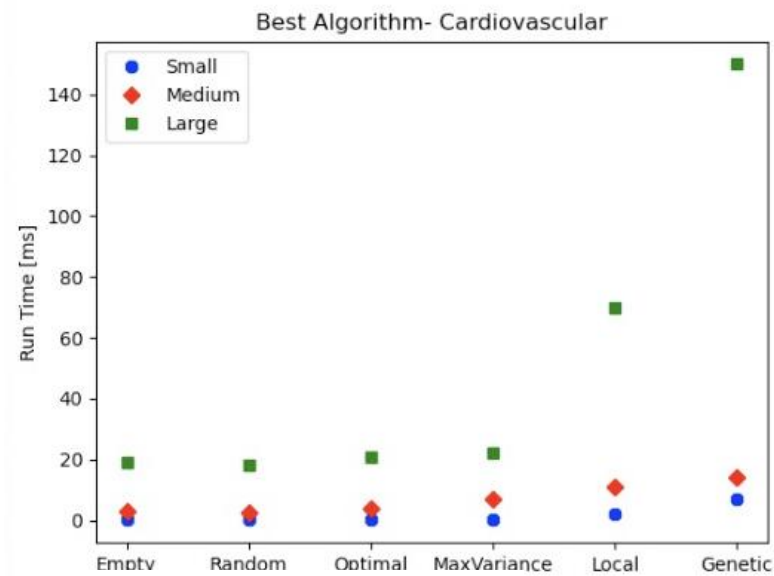
תוצאות


Best Algorithm- Water
Accuracy
90
80
70
60
50
Small
Medium
Large
Empty
Random
Optimal
MaxVariance
Local
Genetic
Algorithm


Best Algorithm- Roads
Accuracy
90
85
80
75
70
65
60
55
Small
Medium
Large
Empty
Random
Optimal
MaxVariance
Local
Genetic
Algorithm



Algorithm



Algorithm

ניתוח תוצאות

ראשית, נשים לב שמבחינת הדיוק, לא התקבלו הבדלים משמעותיים בהרצה על dataset בגדלים שונים. כצפוי, "האלגוריתם הריק" השיג את אחוזי הדיוק הנמוכים ביותר בכל אחת מההרצות- זאת מכיוון שהוא מסווג לפי מספר התכונות הקטן ביותר מבין האלגוריתמים. כמו כן, ניתן לראות שכל האלגוריתמים הנאיביים השיגו אחוזי דיוק קרובים יחסית זה לזה- "האלגוריתם האופטימלי" אמנם מיצה את התקציב הניתן לו לרכישת תכונות, אלא שלא הצליח למנף זאת- כנראה בשל העובדה שבחר תכונות קורלטיביות זו לזו (אם כי אחוזי הדיוק שלו אכן משתנים בין ההרצות על dataset בגדלים שונים). אחוזי הדיוק של "האלגוריתם הרנדומלי" אכן משתנים במקצת בין dataset שונים, אך נוכל לקשור את העובדה הזאת בעובדה שבחירת התכונות נעשית באופן רנדומלי ולכן ההבדל באחוזי הדיוק טמון בכך ולא בגודלו של ה-dataset.
ניתן לראות שהאלגוריתמים המתוחכמים יותר השיגו אחוזי דיוק גבוהים יותר בהשוואה לאלגוריתמים הנאיביים, כאשר "האלגוריתם הגנטי" בעל אחוזי הדיוק הגבוהים ביותר, בכל אחת מההרצות (עם זאת, ייתכן שהתוצאה הנ"ל התקבלה מכיוון שהאלגוריתם לחיפוש מקומי נתקע במקסימום מקומי או שפונקציית ה-Score איננה קעורה- נרצה להתעמק בתופעה הזאת בהמשך כאשר נריץ ניסוי להשוואת סוגי אלגוריתמים לחיפוש מקומי). בנוסף, ראינו שהתפלגות הדאטה אינה גורם המשנה את תוצאות הניסוי באופן משמעותי וכי בכל שלושת הניסויים התקבלו מגמות זהות.

ביחס לזמן הריצה, ניתן לראות שכצפוי- האלגוריתמים הנאיביים היו המהירים ביותר ואף תוצאותיהם קרובות. לעומתם, האלגוריתמים המתוחכמים יותר דרשו זמן ריצה רב יותר, בייחוד האלגוריתם הגנטי שלו זמן הריצה הגדול ביותר. מעניין להבחין שככל שה-dataset גדול יותר, ההבדלים בין זמני הריצה של האלגוריתמים הנאיביים והמתוחכמים גדולים יותר.

לסיכום, נראה שמבחינת ה-Trade Off בין זמן הריצה לאחוזי הדיוק- אלגוריתם שדרש זמן ריצה גדול יותר הצליח להגיע לאחוזי דיוק טובים יותר.

ניסוי: זמן ריצה כתלות בכמות התכונות

רקע לניסוי

בהמשך לניסוי הקודם, נרצה כעת לבחון את זמן הריצה כתלות בכמות התכונות שב-dataset. במיוחד, נרצה לבחון את ההשפעה על פני האלגוריתמים המתוחכמים שהצענו.

אופן ביצוע הניסוי

נבדוק את השימוש באלגוריתמים השונים על ה-Cardiovascular dataset הקטן כך שבכל פעם נגדיל את מספר התכונות שב-dataset וכך נשווה את זמני הריצה המתקבלים כתלות במספר התכונות (כאמור, את מספר הדוגמאות שב-dataset נשאיר קבוע). ניסוי זה מתבצע על dataset יחיד כי התפלגות הדאטה אינה גורם המשפיע על זמן הריצה באופן משמעותי.

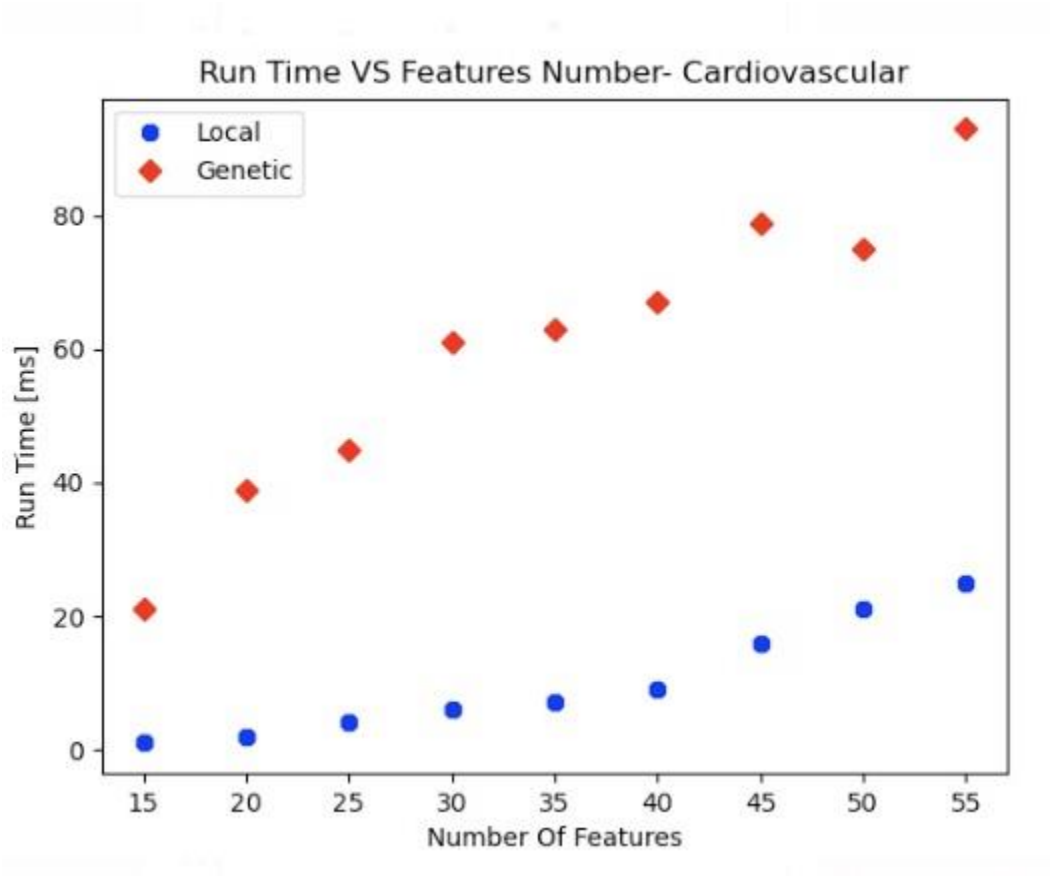
על מנת לבודד את המשתנים בניסוי ולהימנע מרעשים והשפעות הנובעים מהפרמטרים שכל אלגוריתם מקבל, נבחר פרמטרים קבועים לכל האלגוריתמים:

- נקבע שהמסווג שכל האלגוריתמים ישתמשו בו הוא Logistic Regressor.
- נקבע את התכונות הנתונות, מחירי התכונות והמחיר המקסימלי להיות זהים בין כל האלגוריתמים.
- עבור כל האלגוריתמים המשתמשים בפונקציית Score, נבחר כברירת מחדל את ScoreFunctionB.

השערות

אנו מצפים שעבור כל האלגוריתמים הנ"ל, נקבל גרף עולה ממש כתלות במספר התכונות.

תוצאות



ניתוח תוצאות

כצפוי, ניתן לראות שזמן הריצה של כל האלגוריתמים עולה כתלות בכמות התכונות. ניתן לראות שאמנם מגמת כל הגרפים דומים, אך עם זאת קיים הבדל אשר מתבטא בגרף של אלגוריתם החיפוש המקומי שהינו קעור בעוד הגרף של שני האלגוריתם הגנטי קמור. מכאן שהעלייה בכמות התכונות אמורה להשפיע בצורה דרסטית יותר על אלגוריתם החיפוש המקומי (אם כי האלגוריתם הגנטי הוא בעל זמן הריצה הגדול ביותר וכן בעל השיפוע הגדול ביותר).

ניסוי: אלגוריתם החיפוש המקומי הטוב ביותר

רקע לניסוי

במהלך הפרויקט הצענו אלגוריתם המפתח גרף מכוון ומבצע חיפוש מקומי על מרחב החיפוש במטרה למקסם את פונקציית ה-Score שהגדרנו. לאחר שנמצא מצב עבורו ערך ה-Score מקסימלי, נוכל להשתמש בקבוצת התכונות שהמצב מתאר על מנת לסווג על פיהם.

כעת, נרצה לבחון איזה מאלגוריתמי החיפוש המקומי שלמדנו בקורס הוא הטוב ביותר, במונחי אחוזי דיוק, לשמש אותנו במלאכת הסיווג של האלגוריתם שהצענו.

בקורס למדנו מספר אלגוריתמים שימושיים לחיפוש בגרפים שאותם נרצה כעת לבחון:

1. Hill climbing- החיפוש מתחיל במצב התחלתי נתון. בכל שלב מפתחים את כל השכנים ובוחרים את השכן הטוב ביותר- זה בעל ערך ה-Score הגדול ביותר. אם ישנם כמה שכנים טובים ביותר, האלגוריתם יבחר אקראית. האלגוריתם ייעצר אם הגיע למצב מטרה או אם השכן הטוב ביותר איננו טוב יותר מהמצב הנוכחי.
2. Stochastic hill climbing- אלגוריתם חמדן שבוחר תמיד את הצעד התלול ביותר, אך מוכן לצעוד בכיווני שיפור שאינם התלולים ביותר. האלגוריתם בוחר אקראית מבין הצעדים המשפרים עם הסתברות פרופורציונית לעוצמת השיפור.
3. Simulated annealing- אלגוריתם שמתיר צעדים בכיוון מטה (כאלה שמרעים את הערך ההיוריסטי), בכך מונעים היתקעות במקסימום מקומי. ההסתברות לבחירת צעדים כאלו יורדת ככל שמתקדם החיפוש.
4. Random hill climbing- שיפור של אלגוריתם ה-hill climbing. האלגוריתם מריץ את hill-climbing בצורה איטרטיבית ובכל פעם שהחיפוש נתקע, הוא מתחיל מחדש תוך הגרלת מצב התחלה רנדומלי. האלגוריתם שומר את המקסימום הטוב ביותר שמצא ובכך מונע היתקעות במקסימום מקומי.
5. Local beam search- אלגוריתם המבצע חיפוש אלומה ללא התעניינות במסלול.

אופן ביצוע הניסוי

נבדוק את השימוש באלגוריתמים השונים על ה-Cardiovascular dataset לכל הגדלים (קטן, בינוני וגדול) וכך נשווה את אחוזי הדיוק המתקבלים בעת השימוש באלגוריתמים השונים.

על מנת לבודד את המשתנים בניסוי ולהימנע מרעשים והשפעות הנובעים מהפרמטרים שכל אלגוריתם מקבל, נבחר פרמטרים קבועים לכל האלגוריתמים:

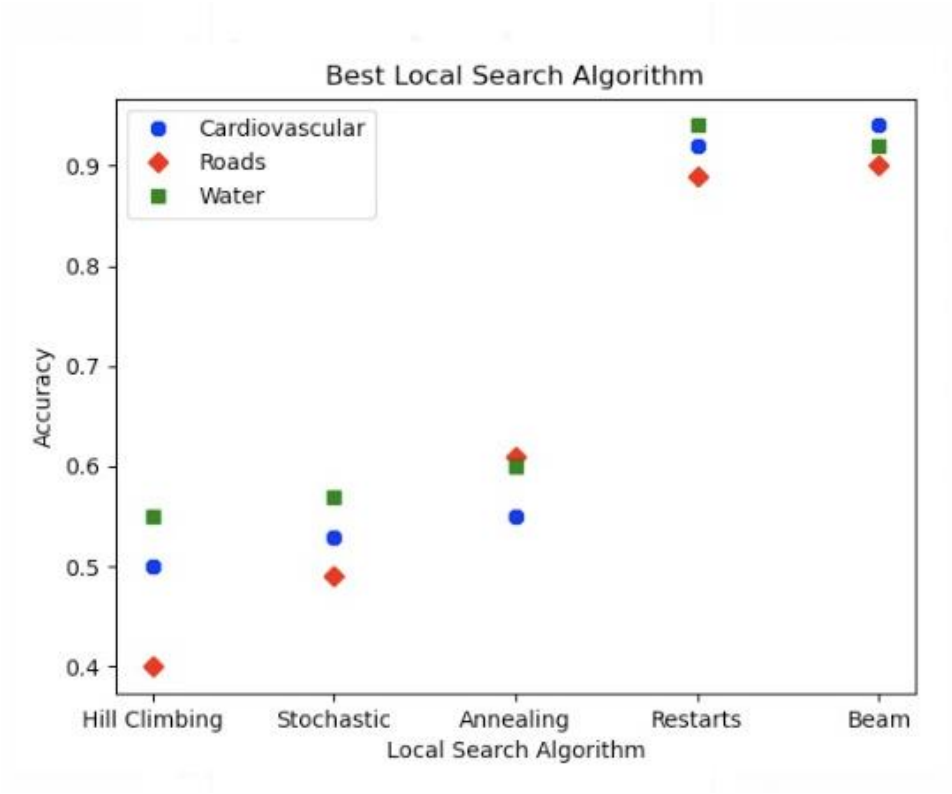
- נקבע שהמסווג שכל האלגוריתמים ישתמשו בו הוא Logistic Regressor.

- נקבע שהתכונות הנתונות, מחירי התכונות והמחיר המקסימלי הניתנים לאלגוריתם שהצענו יהיו זהים בכל אחת מההרצות.
- נבחר כברירת מחדל את ScoreFunctionB כפונקציית ה-Score בה נשתמש.

השערות

אנו מצפים שאחוזי הדיוק הטובים ביותר יתקבלו עבור האלגוריתמים שמונעים היתקעות במקסימום מקומי, זאת משום שאלגוריתמים כאלו מאפשרים מציאה של מקסימום גלובלי ובכך את קבוצת התכונות הטובה ביותר שכדאי לסווג לפיה.

תוצאות



ניתוח תוצאות

כפי שצפינו, נראה כי האלגוריתמים המונעים היתקעות במקסימום מקומי הניבו תוצאות טובות יותר (מלבד Simulated-annealing שהינו אלגוריתם בעל פן הסתברותי ובשל כך רנדומלי- ניתן לחשוב שהתוצאה הפחות טובה התקבלה בשל הרנדומליות). האלגוריתם המוצלח ביותר לשימוש באלגוריתם שהצענו הינו Local Beam Search שנראה שהגיעו לקרוב ל-98% דיוק, בעוד ש-Random hill-climbing הגיע לתוצאה לא רחוקה ממנו.

סיכום הפרויקט

סיכום

את הרעיון לפרויקט שאבנו מבעיה רפואית מציאותית מחיי היום-יום: חולה מגיע לרופא ומבקש ממנו לאבחן מחלה שקיימת או לא קיימת אצלו תחת מסגרת תקציב לביצוע בדיקות רפואיות. לאחר חשיבה וקריאה נוספת של מידע ומחקרים, החלטנו להרחיב את הפרויקט ולהפכו לפרויקט אלגוריתמי כללי. מטרתנו בפרויקט זה הייתה למצוא אלגוריתם למידה העובד תחת אילוצים של תכונות חסרות (מימד חסר של מרחב התכונות).
במהלך הפרויקט הצענו מספר אלגוריתמים שונים, ניתחנו אותם וסיכמנו את יתרונותיהם וחסרונותיהם- כך התקדמנו והצענו אלגוריתמים נוספים במטרה להתמודד עם החסרונות, לשמר את היתרונות ולשפר את האלגוריתמים אחד אחר השני. מאלגוריתמים נאיביים, הגענו לאלגוריתמים מתוחכמים שמבצעים שימוש בכלים שונים שלמדנו במהלך הקורס "מבוא לבינה מלאכותית". לצד הכלים שרכשנו בקורס, ביצענו ניסויים במטרה לאמוד את טיב המסווגים שהצענו ובחנו את השימוש באלגוריתמים השונים שלמדנו על טיב האלגוריתמים שהצענו.
כמו כן, במהלך הפרויקט נחשפנו לכלים טכנולוגיים מתקדמים וספריות מיוחדות שעוסקות בלמידה (כפי שניתן לראות בקובץ environment.yml), בכך התמקצענו בתחום והתנסינו בכתיבת קוד טכנולוגי ויעיל המשלב מסווגים ולמידה באופן כללי.

דיון בתוצאות

לאחר ביצוע הניסויים וניתוח התוצאות, מצאנו שהאלגוריתם הטוב ביותר לפתרון הבעיה בה התעסקנו, מבחינת אחוזי דיוק, הינו האלגוריתם הגנטי (המתבסס על Logistic Regressor). למדנו שהאלגוריתם הגנטי משמש לבחירת תת הקבוצה "החזקה" ביותר (כאשר "חוזק" של קבוצה מתבטא בפונקציית fitness של האלגוריתם), מה שקרוב באופן יחסי לבעיה אותה ניסינו לפתור. נראה שלצד אחוזי הדיוק הרבים שהאלגוריתם הניב, זמן הריצה הארוך שלו (בעיקר בשלב הסיווג) יכול להוות עקב אכילס לשימוש בו.
אנו סבורים ובטוחים שהתבססות על האלגוריתם הנ"ל לצד שדרוג שלו, יכול להוות פריצת דרך של ממש בתחום הסיווג תחת אילוץ גודל של מרחב התכונות.
יחד עם זאת, גם האלגוריתם המבצע חיפוש מקומי במרחב חיפוש הוביל לתוצאות טובות וגם הוא יכול להוות אבן דרך של ממש בתחום.
נשים לב שמבין פונקציות ה-Score שהצענו, פונקציית ScoreB העושה שימוש בהגדרת הוודאות שפיתחנו הינה העדיפה (אם כי זמן החישוב שלה ארוך יותר).

כיוונים להמשך המחקר

הפרויקט שביצענו הינו פרויקט מיוחד וחשוב. הוא מאפשר לבצע סיווג תחת אילוצי תכונות חסרות, מה שיכול להתברר כחשוב וכבסיס בתחום. בפרט, ישנם אפיקים נוספים שאנו חושבים שיכולים להוביל לתוצאות מעניינות ולתת מענה לבעיות מסוג זה:

1. <u>פיתוח אלגוריתמים חדשים:</u> ניתן להמשיך ולהרחיב את המחקר שביצענו על ידי הצעת אלגוריתמים מתוחכמים נוספים שמתמודדים עם החסרונות באלגוריתמים המתוחכמים השונים שהצענו. למשל, ניתן לפתח אלגוריתמים נוספים שמשתמשים באלגוריתמים אחרים שלאו דווקא נלמדים במסגרת הקורס "מבוא לבינה מלאכותית".
לדוגמה, להציע אלגוריתם המתבסס על אלגוריתמי Selector Feature Sequential שהינם משפחה של אלגוריתמים חמדניים המשמשים להורדת מימד מרחב התכונות (כלומר, בהינתן dataset בעל מרחב תכונות ממימד d, האלגוריתמים הללו פולטים את מרחב התכונות הטוב ביותר ממימד k<d). על מנת לפתח אלגוריתם כנ"ל, נדרש ללמוד את אופן פעולת האלגוריתם בהרחבה ולהציע דרך להשתמש בפלט לצד שמירה על מקסימום קורלטיביות בתת קבוצת התכונות המתקבלת כפלט.
בנוסף, ניתן לשלב ולחשוב על אלגוריתם שמתבסס על כלים בלמידה עמוקה (למידה מונחית, בלתי מונחית ולמידת חיזוק) ובפרט ברשת ניורונים. זהו תחום מתפתח שמציע שיפורים ושדרוגים טכנולוגיים רבים וניתן אולי לחשוב על דרך לרתום את התחום הנ"ל לצורך סיווג תחת אילוצי הגודל של מרחב התכונות.

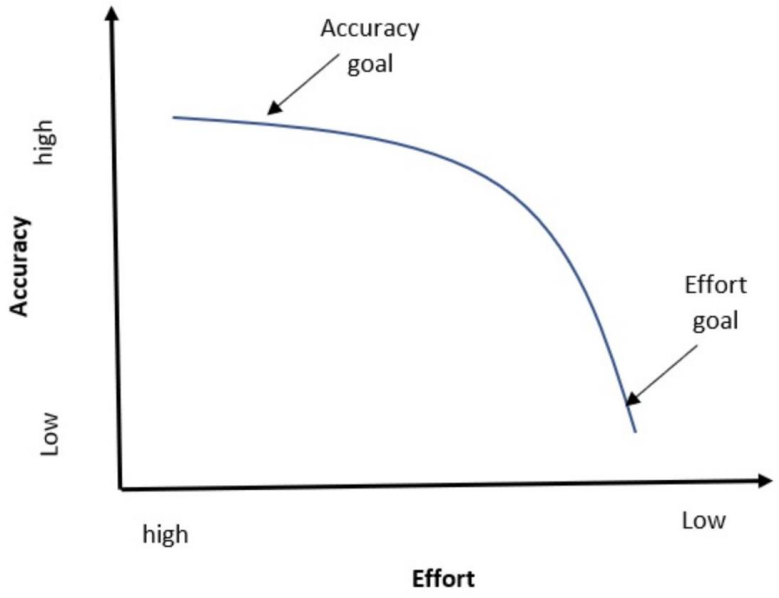
2. <u>שיפור אלגוריתמים קיימים:</u> ניתן לחשוב על דרכים לשפר את הדיוק וזמן הריצה של האלגוריתמים שכבר הצענו. למשל, ניתן למצוא דרך להפחית את זמן ריצת האלגוריתם הגנטי שסיפק את התוצאה הטובה ביותר.

3. <u>שיפור פונקציית ה-Score:</u> אפיק נוסף להמשך מחקר טמון בפונקציית ה-Score. על מנת לשפר את אלגוריתם החיפוש המקומי שהצענו, ניתן לשפר בפרט את פונקציית ה-Score, כך למשל לחשוב על פונקציית Score <u>קעורה</u> בהכרח (לה מובטח מקסימום), פונקציית Score שלה בהכרח מספר מועט של מקסימום מקומי או לחשוב על פונקציית Score חדשה שלוקחת בחשבון פרמטרים אחרים. ניתן בפרט לשפר את זמן הריצה הדרוש לחישוב ה-Score (בפונקציית ה-Score השנייה שהצענו, אנו נדרשים לעשות שימוש במסווג קיים מה שעלול להוביל לזמן ריצה ארוך במיוחד).

4. <u>סיווג שאיננו בהכרח בינארי:</u> על מנת לפשט את הפרויקט ולנתח את העומד בבסיסו, ניתחנו בעיות סיווג עבור תיוגים בינאריים (Classification). ניתן כעת לנתח ולבחון את השפעת האלגוריתמים שהצענו על בעיות סיווג שאינן בינאריות, כלומר בעיות סיווג למספר רב של תיוגים.

5. <u>עיבוד מקדים לתכונות:</u> ניתן לחשוב על ביצוע שלב מקדים טרם הרצת האלגוריתם (Preprocessing) ובמהלכו לבצע Data Exploration, דבר שיכול להוביל "לזריקת" תכונות לא רלוונטיות (עקב מציאת תכונה לא רלוונטית לתגית המטרה, או תכונה שהינה קומבינציה לינארית של תכונות אחרות) טרם הרצת האלגוריתם- בכך לתרום לזמן הריצה של האלגוריתם (הוכחנו במהלך הניסויים שיש קשר בין כמות התכונות לזמן הריצה).

6. <u>ניסיון לסיווג האלגוריתמים על קשת Accuracy Effort Trade Off</u>: אנו במהלך הניסויים התייחסנו לטיב האלגוריתמים במונחי אחוזי דיוק, אלא שאולי נרצה להגדיר את "האלגוריתם הטוב ביותר" כאחוזי דיוק תחת מגבלת זמן ריצה נתון. לכן נרצה לדעת היכן ממוקמים האלגוריתמים שהצענו על העקומה המצורפת ובעתיד לחשוב על שיפורים לאלגוריתמים במונחי Accuracy-Effort ולא רק במונחי אחוזי דיוק.



בנימה אישית

הפרויקט שלנו, שהתחיל מחשיבה על פתרון בעיה רפואית מציאותית מחיי היום-יום, התרחב לפרויקט אלגוריתמי בו ניתן להשתמש בתחומים רבים בחיים במטרה לשפר את החיים ולתרום לאנושות. הפקנו מהפרויקט המון, הן מהפן המחקרי והן מהפן הטכנולוגי. ראינו כיצד הנלמד בקורס "מבוא לבינה מלאכותית" קורם עור וגידים ומשמש אותנו למחקר ופיתוח בחיי היום-יום. אין צל של ספק שהפרויקט בו התנסינו פתח לנו שער למחקר עתידי ועניין רב בתחום, יהווה בסיס להצעות מחקר שנגיש בעתיד ואף ילווה אותנו רחוק בהמשך הקריירה. אנו מרגישים שזכינו לקחת חלק בתחום טכנולוגי מתפתח וכי יש עוד אפיקים רבים למחקר עתידי.