מבוא למערכות לומדות – תרגיל בית 2

מגישים:

שם: אסף חאייק ברוך **ת.ז:** 206783441
שם: בן בנוז **ת.ז:** 207570573

הסברים על התהליך

ניקוי ערכים שגויים:

בתחילת התהליך שלנו אנחנו מוציאים ערכים שגויים ממאגר הנתונים.
לאחר שבחנו את המידע שמנו לב שכל המספרים בו אמורים להיות אי-שליליים ושחלק מהתכונות נמצאות בטווחים אפילו יותר מגבילים – אז החלטנו לקטלג אותם ולמחוק כל כניסה במאגר שאחת התכונות שלה עם ערך לא חוקי (מצורף להגשה קובץ בשם feature_types_and_ranges.txt עם כל המידע הזה).

המרת טיפוסי קטגוריה:

הרבה מעמודות המסד מכילות ערכים שאינם מספריים, ובלתי אפשרי להשתמש בכאלו ערכים ברוב אלגוריתמי הלמידה.
לכן המרנו אותם למספרים במספר אופנים שונים בהתאם לערכים.
את אלו שהיה להם ערכי yes/no הפכנו בהתאם לערכים 0 ו-1, בעוד שאחרים עם תשובות מנוגדות הפכנו ל-1 ול-1− בהתאם לצורך.
חלקם הפכו לקוד מייצג ואת חלקם הפכנו לוקטורי one-hot במקרים שבהם לא היה אפשר לעשות שום דבר אחר ולא היה הגיוני לייצג אותם בקוד שבו ערך אחד יותר גדול מהשני.
גם ההחלטות האלה נמצאות בפירוט מקובץ שהוזכר קודם.

השלמת ערכים חסרים:

בתהליך השלמת המידע החסר השתמשנו ב2 השיטות הבאות:

השיטה הפשוטה ביותר שהיא השלמה לפי ממוצע או לפי רוב, כלומר במקרה של תכונה מספרית שחסרה אצל דוגמא מסוימת נשלים את הערך החסר לפי הממוצע של התכונה הנ"ל אצל כל הדוגמאות, ובמקרה של תכונה קטגוריאלית פשוט נשלים את המידע לפי הקטגוריה שהיא הרוב אצל כל הדוגמאות שהצביעו כמו הדוגמא הנ"ל (מניחים שאם רוב האנשים שהצביעו כמו הדוגמא הזאת אז הם גם קרובים אליו בתכונות)

והשיטה היותר מורכבת בה השתמשנו היא השלמת מידע לפי קורלציה בין תכונות, כלומר ראשית ייצרנו מטריצת קורלציות שבה יש בסקאלה שבין 1 ל1- את רמת קורלציית פירסון שבין כל שני תכונות בdatabase , ואז נעבור על כל הערכים החסרים של כל תכונה בתורה , אם לתכונה יש קורלציה גבוה עם תכונה אחרת (הגדרנו גבוה כגדולה מ0.5 בערך מוחלט) נבדוק לאיזה דוגמא יש את הערך הכי קרוב לדוגמא הנוכחית שלנו בתכונה האחרת ואז נעתיק את הערך שיש לה בתכונה הנוכחית לערך החסר בדוגמא הנוכחית, כלומר בגלל הקורלציה הגבוה בין שתי התכונות שני אנשים שבחרו משהו בתכונה אחת יבחרו דברים דומים בסיכוי גבוה בדוגמא השנייה, ובמקרה שלדוגמא שלנו שחסר לה ערך בתכונה הנוכחית חסר גם ערך בתכונה עם הקורלציה הגבוה ביותר לה אנחנו פשוט ניקח את התכונה הבאה הכי טובה מבחינת קורלציה, ואם אין תכונות שיש להם קורלציה טובה עם הנוכחית או שלדוגמא שלנו חסרים יותר מדי כל הערכים בהן אנחנו פשוט לא נמלא כלום.

- בתהליך השלמת המידע ראשית אנחנו משלימים באמצעות קורלציה שכן זוהי שיטה יותר מיודעת וחכמה להשלים לפיו את המידע מאשר פשוט ממוצע ובנוסף כי השלמה לפי ממוצע פשוט תשלים הכל ולא יהיה מקום להשלמה לפי קורלציה לעשות משהו, ומכיוון שקיימים מקרים שהשלמה לפי קורלציה לא עובדת כי אין תכונות עם קורלציה טובה או שלדוגמא חסרים ערכים בתכונות עם הקורלציה הגבוה לנוכחית, נמלא חורים אלה באמצעות השלמה לפי ממוצע אחרי ההשלמה לפי קורלציה.

- השלמת המידע של סט האימון מתבצעת לפי קורלציה וממוצעים שנלקחים מסט האימון, ובהשלמת המידע של סט הוולידציה השתמשנו גם בהשלמה לפי קורלציה וממוצע של סט האימון כדי לשמור על דמיון בין שני הסטים כמה שיותר שכן בעת הלמידה אנחנו מניחים כי הדוגמאות שהתאמנו עליהם מתנהגות דומה לדוגמאות שניבחן עליהן.

<u>נרמול וסטנדרטיזציה:</u>

החלטנו לעשות זאת לתכונות שלאחר המרת הערכים לא היו בטווחים $[-1,1]$ או $[0,1]$.
התכנות שלא עברו נרמול גם מפורטות בקובץ הטקסט שהוזכר קודם.
יצרנו אובייקט מסוג standardscaler של sklearn שיzfur את התוחלת ואת השונות של כל אחת מהעמודות שכן רצינו לנרמל לפי סט האימון כך שנוכל לעשות זאת לכל סט מתי שנרצה.
את הסטנדרטיזציה עצמה לא עשינו עד שהיינו צריכים להשתמש באלגוריתם relief וב- filter methods.

<u>מתודות feature selection:</u>

אנחנו כתבנו 3 filter methods ו-2 wrapper methods.

1. הראשונה הייתה מתודה שחיסלה תכונות שהיה להן יותר מדי mutual information עם תכונות אחרות (לפי סכום המידע המשותף עם תכונות אחרות שהמידע המשותף איתן גדול מסף מסוים).
עשינו זאת באמצעות חישוב מטריצת mutual information בין כל התכונות ומעבר עליה, כאשר המידע המשותף המנורמל חושב ע"י הפונק' normalized_mutual_info_score של sklearn.
2. השנייה הייתה פונקציה שחיסלה תכונות שהיה להן מעט מדי מידע משותף עם ערך המטרה (גם חושב באמצעות אותה הפונק').
מכיוון שלא רצינו לחסל תכונות מועילות, וידאנו שרק תכונות עם ערך מידע משותף עם המטרה שקטן מסף מסוים יחוסלו.
3. אלגוריתם relief. מומש כפי שלמדנו, נפרט יותר כשנגיע למטלת הבונוס הרלוונטית.
4. אלגוריתם SFS. מומש כפי שלמדנו, עם תוספת של מספר תכונות מקסימלי שהאלגוריתם יעצור בו (אם רוצים להתעלם מהדבר הזה צריך להכניס ערך של אינסוף או מספר התכונות).
נפרט יותר במטלת הבונוס המתאימה.
5. אלגוריתם SBS. מומש כמו SFS רק בכיוון ההפוך, וידאנו שוב שניתן לקבוע ערך של מספר תכונות מקסימלי שהאלגוריתם לא יחזיר יותר תכונות ממנו.

סדר הפעולות שעשינו על התכונות היה:
הסרת תכונות דומות -> הסרת תכונות רחוקות מהמטרה -> relief -> SBS עם עץ החלטה כמסווג.
הסרת התכונות הדומות באה ראשונה כי ראינו שהרבה פעמים הוסרו כמה תכונות דומות ביחד בשלבים האחרים, במקום תכונות שונות שלא רלוונטיות ואז איבדנו הרבה יותר מידע.
אלגוריתם relief היה ה-<> האחרון שהרצנו כי רצינו לחכות עם נרמול הערכים עד לרגע האחרון האפשרי, כדי לוודא שנתפוס תלויות בין תכונות בשלבים שלפני (ובכדי לחסוך קצת בזמן הריצה).
החלטנו להשתמש ב-SBS בסוף בגלל שלא רצינו לזרוק יותר מדי תכונות ולאבד מידע, ו-SFS תמיד בחר באזור ה-7 תכונות ולא היינו בטוחים לגבי זה.
השימוש בעץ החלטה היה בגלל שזה מסווג יחסית מהיר שלא נטה לחסל יותר מדי תכונות.

מטלות הבונוס

תפקיד התכונות השונות ביחס לערך המטרה:

בסך הכל ראינו שלתכונות שמתארות את העיסוק של הבוחר (כמו Occupation) יש השפעה די טובה על הדיוק, ובאופן דומה לתכונות שמראות מה משנה לבוחר וכמה עניין פוליטי יש לו.
המצב הכלכלי והמשפחתי גם נטו להשפיע ולכן תכונות שקשורות להכנסות\הוצאות ולמצב הזוגי מאוד עזרו.
באופן מפתיע הגיל של הבוחר חוסל יחסית מוקדם כתכונה וגם החשיבות של נושאים ספציפיים כמו חינוך וסביבה לא נבחרו כתכונות, וזאת בניגוד לציפיות שלנו (לפעמים מה שנראה ברור לא מתממש במציאות).
באופן כללי דברים שיותר הסבירו על מצבו של הבוחר בחיים האישיים מבחינת מצב כלכלי, עבודה, מצב משפחתי וכמות העניין הפוליטי שלו נטו להיות תכונות רלוונטיות למלגה אליה הוא הצביע.

לגבי אלגוריתם relief:

אלגוריתם relief הוא אלגוריתם שביכולתו למצוא תכונות רלוונטיות בהנחה שכל התכונות בסקאלה דומה (או שגדלי ערכי התכונות משקפים את חשיבותן).
הבעיות שלו הן איטיות עבור מאגר נתונים גדול מספיק (מציאת nearhit ו-nearmiss יכולה לקחת זמן בהתאם למספר התכונות והדגימות) ורגישות לנורמליזציה.
בהרצות שלנו ראינו שהאלגוריתם לא בחר תכונות כמו מין והכנסה, ואלו תכונות שאלגוריתמים אחרים שכתבנו כן בחרו.

לגבי אלגוריתם SFS:

ל-SFS בעיקרון יש כמה יתרונות על מתודות אחרות:
הוא ברצינות בודק רלוונטיות של תכונות למסווגים ויכול למצוא קבוצה קטנה של תכונות שפועלות היטב יחד אך לא טובות לבד.
מצד שני הוא איטי (אימון והפעלת מסווג מספר פעמים בכל איטרציה) ולא בהכרח מודד את טיב התכונות (אם המסווג שהוא משתמש בו שונה מדי מהמסווג שאנו רוצים ללמד).
בעיקרון כדאי להשתמש בו רק אחרי שחיסלנו הרבה תכונות ע"י אלגוריתמים אחרים.
בהרצות שלנו ראינו שהאלג' נוטה לבחור תכונות שיכולות לנבא את המטרה יחסית טוב לבד או קבוצות קטנות של תכונות שיכולות לעשות זאת ביחד.