**מבוא למערכות לומדות**

**תרגיל בית 3**

**פרטי המגישים:**

|  |  |  |
| --- | --- | --- |
| אילנית סמול | 315820258 | ilanitsmul@campus.technion.ac.il |
| שם | ת.ז | אימייל |
|  |  |  |
| אסף אנטר | 301019733 | asafanter@campus.technion.ac.il |
| שם | ת.ז | אימייל |

**מטלת החובה**

**הערות:**

* בהמשך נפרט את תהליך העבודה שלנו כאשר הוא מחולק ל-0 שלבים. מספור השלבים תואם את מספור השלבים בפונקציית   
  ה-main (בקובץ main.py).
* התוצאות שקיבלנו יכולות להשתנות כתלות באופן החלוקה של ה-data לקבוצות train/validation/test וכן באופן פעולת המסווגים שבחרנו (שכן חלקם יכולים להשתמש בפרמטרים רנדומליים המשתנים מריצה לריצה). לכן, כדי שנוכל לנתח את התוצאות, נתייחס לריצה אחת של הקוד שלנו – אשר את הפלט שלה אפשר לראות בקובץ output.txt (לשם נוחות, הפלט מופרד באמצעות כותרות STEP ל-0 חלקים, אחד לכל שלב בתרגיל). לאורך התרגיל, אנו ננתח את התוצאות המפורטות בקובץ זה בלבד.

**שלב 1: טעינת המידע מהקובץ ה-ElectionsData.csv, חלוקתו ל-3 קבוצות של train/validation/test, ועיבוד המידע בהתאם לתרגיל הקודם ולקבוצת ה-features המצומצמת שנתונה בתרגיל.**

* בדומה לתרגיל הקודם, בעזרת הפונקציה read\_csv (המוגדרת ב-pandas) והקובץ ElectionsData.csv טענו את הנתונים לתוך אובייקט DataFrame, ולאחר מכן חילקנו אותו ל-3 קבוצות: 60% עבור training, 20% עבור validation ו-20% עבור test.
* בהתאם לחלוקה זו יצרנו את 3 הקבצים הבאים: "original\_data\_train.csv", "original\_data\_validation.csv"   
  ו-"original\_data\_test.csv", כאשר כל קובץ מכיל את ה- DataFrameהרלוונטי לפי שמו.
* צמצמנו את ה- DataFrameשלנו כך שיכיל רק את 10 ה-features הנתונים בתרגיל, ולאחר מכן ביצענו עליו data preparation המורכב מהשלבים הבאים:

1. תיקון outliers עבור תכונות נומריות: השלמת ערכים באמצעות החציון.
2. השלמת ערכים חסרים: עבור תכונות נומריות – הצבנו את החציון, ועבור תכונות קטגוריאליות – הצבנו את הקטגוריה השכיחה בעמודה.
3. המרת ערכים – נרמול ושינויי טיפוסים: עבור תכונות נומריות – נרמול לפי שיטת ה-min-max, ועבור תכונות קטגוריאליות – המרה למספרים כתלות בסוג הקטגוריה (למשל, תכונות בינאריות הומרו לערכים של 0 ו-1, תכונות בעלות x ערכים הומרו לערכים 1, 2 ועד x).

* בהתאם לחלוקה זו יצרנו את 3 הקבצים הבאים: "prepared\_data\_train.csv", "prepared\_data\_validation.csv"   
  ו-"prepared\_data\_test.csv", כאשר כל קובץ מכיל את ה- DataFrameהרלוונטי לפי שמו.

**שלב 2: אימון מגוון של models עם ה-training set בלבד ובעזרת שיטת cross-validation, כדי לקבוע את ה-hyper-parameters שמשיגים את התוצאות הטובות ביותר עבור על model.**

* סקרנו 0 מסווגים נפוצים של הספרייה "scikit-learn" (אשר מפורטים בקישור [הזה](https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html)):
* לכל model (סוג של מסווג) ביצענו מספר הרצות עם hyper-parameters שונים, ועבור כל ריצה כזו בדקנו מהו ה-score הממוצע המתקבל כאשר משתמשים בשיטת cross-validation (עם פרמטר cv=10) על ה-training set בלבד.
* עבור כל model, בדקנו מהי הריצה שקיבלה את ה-score הממוצע הגבוה ביותר, ובחרנו את ה-hyper-parameters של model זה על פי אותה ריצה.
* איך בחרנו אילו hyper-parameters לבדוק:

1. ראשית, בחרנו לבדוק פרמטרים שאנו מבינים את התפקיד שלהם, ושסביר שישפיעו על ה-score באופן ניכר.
2. לאחר מכן, עבור כל פרמטר p, בדקנו האם כדי להשתמש בערכו הדיפולטי או להשתמש בערך אחר. לצורך כך הגדרנו אובייקט model עם פרמטרים דיפולטיביים פרט לפרמטר p שעבורו ביצענו איטרציה על כל הערכים האפשריים (אם מדובר בערך קטגוריאלי – פשוט עברנו על כולם, ואחרת – עברנו על הערכים שסביר שימקסמו את ה-score). במידה וגילינו שכדאי שלא להשתמש בערך הדיפולטי – המשכנו עם ה-p הזה לשלב הבא.
3. עבור כל הפרמטרים p1,…pn שקיבלנו מהשלב הקודם – הגדרנו אובייקטי model שונים עם כל השילובים האפשריים של פרמטרים אלו, ובדקנו מהו ה-model שמקבל את ה-score המקסימלי.

* הערה: בקוד בחרנו לפרט רק את שלב 3, משום שפירוט של שלב 2 היה מגדיל באופן ניכר את הקוד.
* נדגים את שלב זה על המסווג DecisionTreeClassifier:
* גילינו שכדאי לשנות את הפרמטרים הבאים:
* criterion – הפונקציה שבאמצעותה מודדים את איכות הפיצול. הפרמטר מקבל את הערכים הבאים: "gini" עבור Gini impurity ו-"entropy" עבור information gain. הערך הדיפולטי הינו "gini". בחרנו לבדוק את שני הערכים האפשריים.
* min\_samples\_split – המס' המינימלי של דגימות הדרושות בעלה. הפרמטר מקבל כל ערך הגדול-ממש מ-1. הערך הדיפולטי הינו 2. בחרנו לבדוק את כל הערכים בין 2 ל-20 (לא כולל).
* הגדרנו model של DecisionTreeClassifier עבור כל שילוב של הפרמטרים הנ"ל, כלומר 36 שילובים בסך הכל.
* קיבלנו שה-score המקסימלי מתקבל עבור criterion=entropy ו-min\_samples\_split=6 וערכו 84.98 (לשם השוואה, שימוש בערכים הדיפולטיביים gini=entropy ו-min\_samples\_split=2 הניב score של 0.8306). לעדכן ערכים לפני הגשה.
* בסיום שלב זה קיבלנו רשימה של 0 אובייקטי model, כאשר לכל אחד מוגדרים hyper-parameters שממקסמים את ה-score על פני ה-training set. התוצאות שקיבלנו הן:
* להשלים

**שלב 3: אימון ה-models מהשלב הקודם (לאחר קביעת ה-hyper-parameters עבור כל model) עם כל ה-training set (ללא cross-validation), ובדיקת הביצועים על ה-validation set.**

* צריך לפרט איך בודקים את הביצועים, בהתאם לכל משימה.

**שלב 4: בחירת ה-model הטוב ביותר לכל משימה בהתבסס על השלב הקודם.**

**שלב 5: אימון ה-model הנבחר על ה-training set וה-validation set, ובדיקת הביצועים על ה-test set.**

* צריך לענות על כל המשימות.

**שלב 6: פירוט.**

לקוח מהתרגיל הקודם, צריך לעדכן:

**סיכום**

* כדי לבדוק האם העיבוד שביצענו אכן משפר את תהליך הסיווג, הפעלנו את מסווג KNN על הנתונים שלנו בשתי נקודות בתהליך: לפי הפעלת ה-filter methods (בין שלב 7 לשלב 8) ולאחריהן (לאחר שלב 12).
* התוצאות שקיבלנו הן:
* לפני ביצוע filter method (סה"כ 37 תכונות): הדיוק הינו 38.2%.
* לאחר ביצוע filter method (סה"כ 16 תכונות): הדיוק הינו 76.25%.
* כלומר, אכן קיבלנו שיפור.

**מבנה תיקיית ההגשה**

|  |  |
| --- | --- |
| * הסבר על תהליך העבודה שלנו. | HW2.pdf |
| * קבצי ה-Python שלנו (ב-main נמצאים כל השלבים המתוארים במסמך זה, וב-utils נמצאות פונקציות עזר). | main.py  utils.py |
| * הנתונים המקוריים, מחולקים ל-train, validation ו-test. | original\_data\_train.csv  original\_data\_validation.csv  original\_data\_test.csv |
| * הנתונים לאחר העיבוד שלנו, מחולקים ל-train, validation ו-test. | processed\_data\_train.csv  processed\_data\_validation.csv  processed\_data\_test.csv |
| * תיאור ה-features שבחרנו. | SelectedFeatures.csv |