**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:**

* פתרנו את התרגיל באמצעות חלוקתו ל-0 שלבים. מספור השלבים תואם את מספור השלבים בפונ' main בקובץ modeling.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.
* בסוף דו"ח זה צירפנו הסבר על מבנה תיקיית ההגשה והקבצים שבה.

**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. לפי התרגיל צריך להעביר את זה לשלב האחרון
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 4 מסווגים מוכרים מהספרייה "scikit-learn": RandomForestClassifier, DecisionTreeClassifier, KNeighborsClassifier ו-SVC.
* 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. לבדוק האם חישוב ה-score הוא טוב
* 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-15 (לא כולל).
* הגדרנו model של DecisionTreeClassifier עבור כל שילוב של הפרמטרים הנ"ל, כלומר 28 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 4 אובייקטי model, כאשר לכל אחד מוגדרים ה-hyper-parameters שממקסמים את ה-score על פני ה-training set: לעדכן ערכים לפני הגשה

|  |  |  |
| --- | --- | --- |
| **Score** | **Non-default parameters** | **Model** |
| 0.8911148094252861 | criterion='entropy', min\_samples\_split=4 | RandomForestClassifier |
| 0.8470046373455966 | criterion='entropy', min\_samples\_split=6 | DecisionTreeClassifier |
| 0.8011821343157349 | n\_neighbors=6, weights='distance' | KNeighborsClassifier |
| 0.7975176244891198 | kernel='linear' | SVC |

**שלב 3: אימון ה-models מהשלב הקודם (אחרי שקבענו ה-hyper-parameters עבור כל model) עם כל ה-training set (ללא cross-validation), בדיקת הביצועים על ה-validation set, ובחירת ה-model הטוב ביותר לכל משימה.**

* צריך לפרט אילו מדדים בחרנו: accuracy/error ו-precision/recall עבור כל תכונה.
* לכלול תמונות של confusion matrix ושל histogram.
* להסביר ש-(1) קיבל את הדיוק הכי טוב ואחריו (2). כשמגדירים threshold=0.5 עדיין (1) מנצח אבל (2) מאוד קרוב אליו, כלומר למרות ש(1) לא בטוח בעצמו – הוא בסופו של דבר מקבל את התוצאה הנכונה..
* להסביר במה (3) ו-(4) טועים – לוקחים משני עמודות לכיוון העמודה המנצחת ואז הפער בה יותר גדול.
* לציין את היחס להתפלגות הנתונים האמיתית

**שלב 4: אימון ה-model הנבחר על ה-training set וה-validation set (ביחד), בדיקת הביצועים על ה-test set, ומענה על המשימות.**

* צריך לחזות לאיזה מפלגה כל מצביע ב-test יצביע (לחזות את ה-label של כל שורה ב-test).
* צריך לבנות את ה-confusion matrix (של ה-test) וה-overall test error.

**מתוך התרגיל + פיאצה**

* השאלות שצריך לענות עליהן:

1. לחזות איזו מפלגה תזכה במירב הקולות.
2. לחזות את חלוקת הבוחרים בין מגוון המפלגות (אחוז הבוחרים לכל מפלגה).
3. לכל מפלגה, ספק רשימה של הבוחרים שהכי סביר שיצביעו לה. (ולשים את זה בתוך טבלה כאן, כאשר כל עמודה היא מפלגה, ובכל תא מס' האינדקסים המתאימים למפלגה)

* צריך להסביר בדו"ח:
* את הבחירה שלנו ב-2 מודלים לאמן ולבדוק את הביצועים שלהם.
* את מדדי הביצועים שבחרנו לבדוק.
* את הבחירה שלנו באיזה model להשתמש בקביעת התחזיות הסופיות.
* רשות א': אוטומציה של תהליך בחירת ה-model.
* רשות ב': one size doesn’t fit all – להשתמש ב-model אחר עבור כל אחת מ-3 משימות החובה.
* רשות ג': מצא את הפקטור (מאפיין של בוחר) שע"י שינוי שלו – הכי סביר לשנות את המפלגה שתנצח בבחירות.
* **משימה 3:**
* מה זה most probale voters? המטרה הסופית זה לתת לכל מפלגה שימה של בוחרים שסביר שיצביעו לה. כדאי להשתמש ב- model.predict\_proba(X) ואז לעשות טבלה של N (בוחרים) על M (מפלגות), ולכל אחת לכתוב את הסיכוי. ואז לקבוע את ה-treashhold. בנוגע למשימה עצמה – האינדקסים שצריך להחזיר הם לפי ה-test.csv שלנו.
* המשימות מתייחוסת ל-test, אבל מומלץ לעשות טבלת השוואה עם ה-train וה-validation.

**סיכום**

**מבנה תיקיית ההגשה**

|  |  |
| --- | --- |
| * הסבר על תהליך העבודה שלנו. | HW3.pdf |
| * קבצי ה-python שלנו: ב-modeling נמצאים כל השלבים המתוארים במסמך זה,  ו-prepare\_data אחראי על תהליך הכנת הנתונים כפי שבוצע ב-HW2. | modeling.py  prepare\_data.py |
| * פלט של הרצת פונקציית ה-main בקובץ modeling.py. | output.txt |
| * הנתונים המקוריים, מחולקים ל-train, validation ו-test, כוללים את כל ה-features. | original\_data\_train.csv  original\_data\_validation.csv  original\_data\_test.csv |
| * הנתונים לאחר העיבוד שלנו, מחולקים ל-train, validation ו-test, כוללים רק את 9 ה-features המצוינים בתרגיל וכן 'Vote'. | prepared\_data\_train.csv  processed\_data\_validation.csv  processed\_data\_test.csv |
| * תחזית הבחירות של מצביעי קבוצת ה-test כפי שנחזו ע"י ה-model שבחרנו. | voting\_predictions\_on\_test\_set.csv |
|  |  |