**למידה חישובית – תרגיל בית מס' 1**

בעבודה זו יצרנו עץ רגרסיה ליניארית ב Python.

הקוד מקבל ארבעה פרמטרים:

MIN\_SAMPLES\_TO\_SPLIT – מגדיר את כמות הדגימות (תצפיות) המינימלי הדרוש לפיצול קודקוד.

MAX\_DEPTH – מגדיר את עומק העץ המקסימלי על מנת להימנע מעץ עמוק מידי.

K\_FOLD\_SPLITS – מגדיר את כמו ה folds עבור חלוקת הדאטא לסט אימון וסט בדיקה.

REGRESSION\_MODEL – מגדיר את מודל הרגרסיה ע"פ בחירת המשתמש.

מהלך ריצת הקוד:

* בסיס נתונים:
  + טעינת בסיס נתונים (הנחה: בסיס הנתונים אינו מכיל שורת כותרות).
  + ניקוי בסיס הנתונים ע"י הסרת דגימות עם ערכים חסרים וחלוקת בסיס הנתונים ל X ו Y.
  + הוספת ערכים ייחודיים לכל ערך בכל עמודה נומינלית ב X על מנת להימנע בהמשך מכפילויות בעת הפיכת עמודה נומינלית לעמודה בינארית חדשה.
  + חלוקת המידע לסט אימון וסט בדיקה ע"פ 10 cross fold validation.
* עץ רגרסיה ליניארי:
  + יצירת עץ רגרסיה ליניארית בעל קודקוד שורש יחיד.
  + אימון העץ על X ו Y של סט האימון, כאשר בכל יצירה של קודקוד חדש מחושב עבורו ה MSE הנוכחי בעזרת פונקציית LinearRegression שבספריית scikit-learn וכן מתחילה ריצה רקורסיבית על מנת למצוא את פיצולי העץ, במידה והקודקוד הנוכחי עומד בתנאי שני הפרמטרים שהוגדרו לעיל, MIN\_SAMPLES\_TO\_SPLIT ו MAX\_DEPTH.
  + בכל ריצה כזו (פונקציית find\_best\_fit) נבדקים כל העמודות האפשרויות, כאשר לכל עמודה נבדקים כל מקרי החלוקה האפשריים. במידה ומדובר בעמודה נומרית יבדקו כל מקרי הפיצול הבינאריים האפשריים (קטן שווה וגדול מ) וכאשר מדובר בעמודה נומינלית יפוצל הקודקוד לכל הערכים האפשריים (שווה ל).
    - מציאת ה MSE עבור כל מקרי החלוקה, ה MSE המנורמל לפי כמות ה INSTANCES.
    - במידה ונמצא MSE קטן יותר מה MSE הנוכחי, יעודכן ה MSE הנוכחי וכן עמודה הפיצול וקריטריון הפיצול בהתאם.
    - בסיום בדיקת כל ה MSE האפשריים תיצור הפונקציה ילדים חדשים (קודקודים) לפי העמודה והערך הנבחרים, כאשר עבור עמודה נומרית יוצרו שני ילדים חדשים ועבור עמודה נומינלית יוצרו מספר ילדים כמספר הערכים הייחודיים עבור עמודה זו.
    - במידה ופיצול הקודקוד הנוכחי לא גרמה לפיצול אמיתי (הפיצול החזיר את ערכי הקודקוד הנוכחי), או כאשר הקודקוד הנוכחי לא עומד בתנאי שני הפרמטרים שהוגדרו לעיל יוגדר הקודקוד הנוכחי כעלה.
  + חיזוי ערכים:
    - בדיקת ערך ה MSE עבור תוצאות חיזוי X של סט האימון אל מול Y של הסט.
    - בדיקת ערך ה MSE עבור תוצאות חיזוי X של סט הבדיקה אל מול Y של הסט.
    - לבסוף, לשם השוואה, השתמשנו ב DecisionTreeRegressor מתוך ספריית scikit-learn לה שלחנו את אותם סטי האימון והבדיקה בהם השתמשנו לעץ שלנו וחישבנו את ה MSE של סט האימון ושל סט הבדיקה.

תוצאות:

באופן כללי ראינו כי העץ שבנינו הוביל על העץ של ספריית scikit-learn בעיקר עבור סט האימון, כאשר עבור סט הבדיקה לרוב העץ של scikit-learn היה מוצלח יותר. בנוסף לכך קיים החיסרון העיקרי בבניית העץ שלנו והוא זמן היצירה שגדול משמעותית.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Data base** | **Our model time to train** | **Our model train set MSE** | **Our model test set MSE** | **scikit-learn model time to train** | **scikit-learn model train set MSE** | **scikit-learn model test set MSE** | **Train set difference between us and scikit-learn** | **Test set difference between us and scikit-learn** |
| dataset\_Facebook.csv | 2688.86 | 5.16e-26 | 1883481.8 | 0.018 | 13923.7 | 42979.4 | **-13923.7** | 1840502.39 |
| forestfires.csv | 738.18 | 0.51 | 86883.13 | 0.026 | 1593.99 | 4892.07 | **-1593.47** | 81991.06 |
| imports-85.data | 254.99 | 31127.85 | 5.49e+25 | 0.04 | 1874027.25 | 12693220.72 | **-1842899.4** | 5.49 |
| machine.data | 71.79 | 0.0 | 2303.73 | 0.01 | 2413.98 | 8859.92 | **-2413.98** | **-6556.18** |
| slump\_test.data | 170.96 | 2.54e-29 | 50.88 | 0.01 | 6.78 | 32.74 | **-6.78** | 18.14 |

מסקנות:

נראה שהדרך ליצירת העץ שלנו עדיין לא מושלמת וככל הנראה מובילה ל overfitting שמכשיל את סט הבדיקה. בנוסף זמן יצירת העץ, עקב הצורך במעבר על כל הרשומות מספר רב של פעמים, גדול מידי. אנו מניחים שיש דרך טובה יותר לבניית העץ, אולי ע"י הסרת עמודות שכבר היו בשימוש או ע"פ קריטריון המשלב מדד MSE ופרמטרים נוספים, או ע"י גיזום העץ בצורה מושכלת.