חלק ב' – פיתוח KNN

מימוש KNN-

**תרגיל 1:**הפונקציה *euclidean\_distance* ממומשת בקובץ *functionUtils.py* ומקבלת שתי רשימות של תכונות ומחזירה את המרחק ביניהן לפי הנוסחה המבוקשת.

**תרגיל 2:**שתי המחלקות המבוקשת *knn\_classifier* ו- *knn\_factory* ממומשות בקובץ *classifier.py*, יורשות ממחלקות האב כנדרש. המלקה *knn\_factory* מקבל את הערך K עבור KNN בבנאי שלה ובעת קריאה למתודה train מחזירה מסווג KNN עבור אותו K שקיבלה על ידי קריאה לבנאי של המחלקה *knn\_classifier*.  
המחלקה *knn\_classifier* מקבלת בבנאי שלה סט מידע אימון מתויג ואת מספר השכנים הנוכחי.

הרצת ניסויים:

**תרגיל 3:**

1. הפונקציות *split\_crosscheck\_groups* ו- *load\_k\_fold\_data* ממומשות בקובץ *functionUtils.py.* הפונקציה *split\_crosscheck\_groups* מחלקת את קבוצת האימון לפי מספר החלקים שקיבלה ובפרט שומרת על התפלגות המידע בכל חלק, מאחסת כל אחד מהחלקים על ידי שימוש בחבילה pickle ושומרת על קונבנציית שמות הקבצים שצוינה בהוראות.  
   הפונקציה *load\_k\_fold\_data* טוענת את המידע מקובץ על מבנה נתונים המבוקש וזאת על ידי שימוש בחבילה pickle.
2. הקבצים נוצרו ונשמרו בתיקיית ההגשה הראשית. חשוב לשמור על עקביות זו על מנת שהשוואה בין הפרמטרים ואו המסווגים השונים תהיה הוגנת ולא תהיה מוטעית בגלל דרך החלוקה של קבוצת האימון. בצורה זו אנו מקבעים את התלות של השוואת המסווגים בקבוצת האימון כי אנו משווים את כולם על פי אותה קבוצת אימון, כלומר תהליך הערכת המסווג תלוי רק במסווג עצמו.

**תרגיל 4:**

1. הפונקציה *evaluate* ממומשות בקובץ *functionUtils.py.* פונקציה זו משתמשת בפונקציית עזר בשם *get\_k\_folds\_data* וזאת על מנת לחלק כל אחד מהחלקים של סט האימון לקבוצת אימון וקבוצת ולידציה. לאחר מכן מאמנת את המודל על ידי קבוצת האימון ובודקת אותו על ידי קבוצת הוולידציה ומחזירה את המדדים המבוקשים.

**תרגיל 5:**

1. קטע הקוד המממש את הניסויים נכתב בקובץ *run\_experiments.py*, מצורף *experiments*6.*csv* תחת התיקייה הראשית.
3. מבין המדידות שעשינו עבור k=3 התקבל הערך המקסימלי של דיוק 0.95 כלומר 95%.
4. ערך המינימום הינו: 0.925 ומתקבל כאשר k=13, ערך המקסימום הינו 0.95 ומתקבל כאשר k=3.  
   כפי שניתן לראות מהגרף ישנה מגמת עלייה עד k=3 וישנה מגמת ירידה החל מk=3. מגמת העלייה הראשונית בדיוק נובעת מכיוון שהגדלת הערך של k מונעת תופעות של overfitting אשר מטיבה עם הדיוק על קבוצת המבחן ודווקא פוגעת בקבוצת הוולידציה, הדבר יכול להעיד כי קיים רעש בדגימות שקיבלנו. אמנם הגדלת יתר של k כפי שאנו שמים לב דווקא גורמת לירידה בדיוק מכיוון שהאלגוריתם כעת מסתמך על שכנים יותר "רחוקים" ולכן הדמיון בין האובייקט הנבדק לשכנים אלו קטן וההנחה כי שכנים קרובים הם דומים כבר לא תקפה.  
   את הערכת המסווג ביצענו על ידי k – cross - fold validation , כאשר k=2, כלומר קבוצת אימון וקבוצת מבחן. מבחן הערכה זה קובע כי K=3, הסתמכות על 3 שכנים קרובים יביא למקסימום את דיוק המסווג.

**תרגיל 7:**

1. שתי המחלקות המבוקשת *id3\_classifier* ו- *id3\_factory* ממומשות בקובץ *classifier.py*, על ידי עטיפה של האלגוריתם מהחבילה *"sklearn".*
2. שתי המחלקות המבוקשת *perceptron\_classifier* ו- *perceptron\_factory* ממומשות בקובץ *classifier.py*, על ידי עטיפה של האלגוריתם מהחבילה *"sklearn".*
3. התוצאות הכי טובות התקבלו הם על ידי אלגוריתם KNN **לכל k ביחס לאלגוריתם האחרים שנבדק בפרט עבור k=3**, לדעתנו תוצאה זו נובעת מכך שאכן ההנחה כי אובייקטים דומים הם בעלי סיווג זהה חזקה עבור בעיית למידה זו.לאחר מכן אלגוריתם ה – *perceptron* ואז עץ החלטה *ID3.*

חלק ג' – תחרות ובונוס

בחלק זה נתאר את המסווג אשר בנינו לתחרות ואת בחירת הפרמטרים עבורו.

אנו השתמשנו במסווג אשר מקבל החלטה על סמך וועדה של 3 מסווגים מסוג KNN:

* KNN when k is 1
* KNN when k is 3
* KNN when k is 5

בנוסף על מנת לסנן את התכונות של החולים אשר אינם מוסיפים מידע חדש, רלוונטי לסיווג השתמשנו בעץ ID3 על מנת להבין את משקלי התכונות ומה החשיבות של כל אחת ביחס לאחרת.  
שלבי החשיבה בניית המסווג:

1. נרמול המידע:

כפי שראינו בהרצאה כדאי לנרמל את המידע זאת מכיוון שנרצה למנוע העדפה של תכונה אחת על גבי אחרת רק בגלל הטווח ערכים שלה. לכן נרמלנו את המידע לפי נרמול Z כלומר לפי תוחלת וסטיית התקן.  
נרמול המידע נעשה עבור כל מסווג מסוג ה- KNN.

1. בחירת פרמטרים עבור כל אחד מתתי המסווגים:

* בחירת פרמטר K למסווגים KNN נעשתה בחלק הראשון של התרגיל וכפי שראינו עבור K=3 מתקבל דיוק מקסימלי, אמנם לא בהפרש גדול מ- K=5, K=1 ולכן בחרנו את שלושת ערכים אלו.
* בניית עץ ID3 אשר מאומן ונבדק על ידי *K- fold cross validation* ובנוסף בוצע עבורו אופטימיזציה של פרמטרים על מנת להביא את המסווג הכולל לדיוק מקסימלי. תהליך זה נעשה בעזרת *random search*  על הפרמטרים(נבדקו כ-1,000 קומבינציות אפשרויות) שיכול העץ לקבל, כאשר הערך האופטימלי נבחר על ידי מקסום הדיוק הכולל של המסווג (לא רק העץ).
* הקוד שהשתמשנו לצורך מציאת פרמטרים אופטימליים נמצא בקובץ *hyperparametersTuningContest.py*.

1. קבלת החלטה:

לבסוף עטפנו את שלושת המסווגים במסווג יחיד אשר מסווג את האובייקט על סמן 3 המסווגים שקבענו ומחליט את הסיווג הסופי על פי החלטת הרוב.  
כאשר כל מסווג KNN נותן חשיבות לתכונות האובייקט על ידי שליפת משקל התכונות לפי התרומה שלהם בבניית העץ (*Information Gain*), לדוגמא אם תכונה לא תרמה לבניית העץ, לא השפיעה על *Information Gain* היא בעלת משקל 0 ולכן מסווג KNN לא יתחשב בה. בדרך זו אנו מורידים את המימד של הבעיה.

המוטיבציה לממש מסווג כזה היא שהמסווג טועה אם ורק אם לפחות שני תתי מסווגים טועים.

1. מימוש המסווג לתחרות:

המסווג ממומש על ידי שתי המחלקות:

* *contest\_classifier*
* *contest\_classifier\_factory*

אשר ממומשות בקובץ *classifier.py* ומקיימות את הקונבנציה של מחלקות האב, זאת על מנת שנוכל להעריכו על ידי פונקציית evaluate. כפי שנאמר קודם מימוש המסווג הינו עטיפה לשלושה מסווגים, כאשר ההחלטה על סיווג אובייקט מתבצעת על ידי סיווג הרוב.

1. הערכת המסווג:  
   לאחר בניית המסווג ובחירת הפרמטרים המתאימים הגענו לאחוז דיוק של 97% אשר נמדד בעזרת k-fold-cross-validation כאשר k=3.