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.

**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.
2. מבין המדידות שעשינו עבור k=3 התקבל הערך המקסימלי של דיוק 0.944 כלומר 94.4%.
3. ערך המינימום הינו: 0.935 ומתקבל כאשר k=13, ערך המקסימום הינו 0.944 ומתקבל כאשר k=3. כפי שניתן לראות מהגרף ישנה מגמת עלייה עד k=3 וישנה מגמת ירידה החל מk=3. מגמת העלייה בדיוק הראשונית נובעת מכיוון שהגדלת הערך של k מונעת תופעות של overfitting אשר מטיבה עם הדיוק על קבוצת המבחן. 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.  
   נוסיף כי העלייה בסוף הגרף מערך k=7 לערך k=13 היא מזערית ולא יכולה להעיד כי הדיוק יעלה אם נמשיך להגדיל את k אלא להפך בגלל שאנו מאמנים רק על 500 אובייקטים ובודקים על 500 נצפה לירידה.  
   כפי שניתן לראות הצלחנו למצוא k אופטימלי בעזרת k – cross - fold validation , מבין הערכים שנבדקו.

**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.

חלק ג' – תחרות ובונוס

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

אנו השתמשנו במסווג אשר כולל ועדה של 3 מסווגים: Neural Network, KNN, ID3 .  
על מנת לממש את את כל אחד מתתי המסווגים: Neural Network, KNN, ID3 השתמשנו בחבילת Python: “sklearn” אשר כוללת קוד מוכן של כל מסווגים אלו.

למרות שמסווג מסוג Neural Network לא נלמד בקורס זה, הרקע לבחירתו הוא על ידי חיפוש ברשת ורקע קודם מקורס מערכות לומדות (של הפקולטה לחשמל). קוד זה למעשה בונה מסווג מסוג Neural Network על פי הפרמטרים שהוא מקבל ומאמן את המסווג על פי קבוצת אימון נתונה.  
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 המסווגים: KNN ו- Neural Network.

1. בחירת פרמטרים עבור כל אחד מתתי המסווגים:

* בחירת פרמטר K למסווג KNN נעשתה בחלק הראשון של התרגיל וכפי שראינו עבור K=3 מתקבל דיוק מקסימלי.
* בחירת פרמטרים עבור ID3 ו Neural Network- נעשתה על ידי hyperparameters optimization. כלומר על ידי כל אחד מהמסווגים עם סטים של פרמטרים והערכתו על ידי k-fold-cross validation כאשר k=4.  
  עבור Neural Network נבדקו 384 סטים של פרמטרים ועבור ID3 נבדקו 100 סטים של פרמטרים.
* 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.  
המוטיבציה לממש מסווג כזה היא שהמסווג טועה אם ורק אם לפחות שני תתי מסווגים טועים. מכיוון שהם מסווגים את המידע שלנו בצורה שונה אחד מהשני ולכן כמעט בלתי תלויים אחד בשני הסיכוי לכך קטן מאשר ועדה של מסווגים מאותו סוג.

1. מימוש המסווג לתחרות:

המסווג ממומש על ידי שתי המחלקות *contest\_classifier* ו- *contest\_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.