האקתון IML -2020

מגישים: יונתן גל, גיל גרינברג, ליאור בן יעקוב ושיר ברוש.
מטלה: מס 2- של מי הקוד הזה??

היכרות עם המטלה ותכנון ראשוני

לאחר שפגשנו לראשונה בשתי המטלות התחלנו בשיח וחשיבה על כיווני פתרון ודרכי התמודדות עם שתי המטלות. לאחר חשיבה ארוכה החלטנו כי נבחר במטלה מס 2, הסיבה העיקרית הייתה הרצון להתמודד עם סוג חדש של מידע שלא התעסקנו איתו בעבר, על מנת להרחיב את הידע ולהפוך את ההאקתון לאתגר מרענן.

לאחר בחירת המטלה התחלנו בתכנון כללי של העבודה- מעבר ראשוני ומשותף על הדאטא על מנת לבחון מול מה אנו מתמודדים, האם נוכל להבין איזה דברים חשובים ינחו אותנו בעיבוד ראשוני של הדאטא. העלנו יחד רעיונות לעיבוד חכם ויעיל של הדאטא בנגוע למטלה זו.
התחלקנו ל-2 זוגות והחלטנו שכל זוג יהיה אחראי על תחום: זוג 1- אחראי על עיבוד ראשוני על המידע ויצירת הfeatures, זוג 2- אחראי על חלוקת המידע לtrain, validation, test ויצירת המודל הטוב ביותר.

Data preprocess

הדאטא שקיבלנו במשימה זו היה מורכב מ7 קבצים, כאשר כל קובץ שייך לצוות אחר. מאחר והדאטא הגולמי כפי שהוא אינו מתאים ללמידה היה עלינו לגבש אסטרטגיה בנוגע לאופן שבו נחלץ דגימות מנתונים אלו. בסופו של דבר בחרנו לדגום כל קובץ לפי שורות, כאשר כל דגימה הייתה שרשור של בין שורה ל5 שורות רצופות (באופן אקראי).

כעת היה בידנו וקטור X בעל מימד אחד (string) ווקטור תיוג y. כמובן שמידע זה עדין לא בשל ללמידה יעילה, יש לנו פיצ'ר אחד שהוא string והוא איננו נומרי או קטגורי.

על מנת לעבד את המידע וליצור features מהמילים בכל קובץ השתמשנו ב-
nltk.tokenize.TweetTokenizer() ובכך הפרדנו את הטקסט למילים. הכנסנו את המידע למילון שסופר פר קובץ את מס ההופעות של כל מילה. את המידע הנ"ל יצאנו לאקסל ושם ביצענו כל מיני ניתוחים של ספירה חישובי הסתברויות לכל מילה וקיבלנו הבנה ראשונית על כמות המידע וסדרי הגודל, כאשר ההבנה המרכזית הנה שיש לבצע ניקוי features על ידי שימוש בסטיית תקן.

לאחר מעבר על המידע ראינו כי התווים הנפוצים ביותר הם תווים כגון ","(":":")" וכדומה, המחשבה הראשונית שלנו הייתה שאלו תווים שמופיעים בכל קוד באופן בנאלי ולכן לא יתרמו, החלטנו לנקות את הדאטא מהם, מהר מאוד הבנו שזוהי טעות שכן סטיית התקן של התווים הנ"ל אל מול הממוצע היא הכי גדולה לתווים אלו.

Feature creation / selection

הדאטא שלנו הגיע כדאטא ללא features על מנת לייצר features יצרנו היסטוגרמה של המילים לאחר הסינון משלב העיבוד המקדים, ובעצם יצרנו feature מכל מילה שעברה את שלב העיבוד המקדים. כל sample הוא בין משפט לחמישה משפטים מקטע קוד, ולכל feature מופיע בשורה של הsample כמות הפעמים שהופיעה המילה הספציפית בfeature.

בחרנו לבצע ניקוי features שסטיית התקן שלו קטנה מ60. בחרנו את הערך 60 לאחר הרצות של ניסוי וטעיה במטרה להגיע לכ10,000 features .

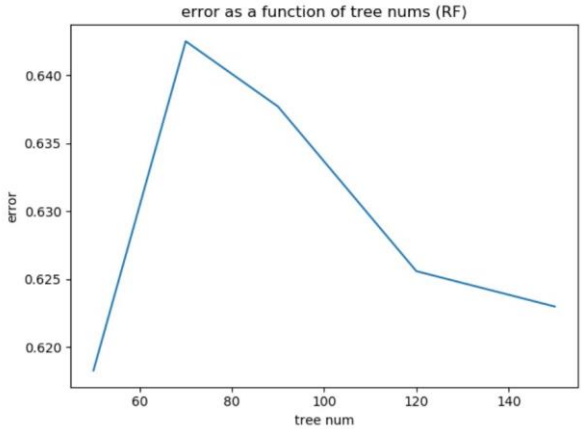
כמובן שאת כל שלבי הpreprocess ביצענו אך ורק על train data לאחר שהזוג שתפקידו היה לדאוג לפיצול המידע הכין את הפונקציה האחראית למשימה.

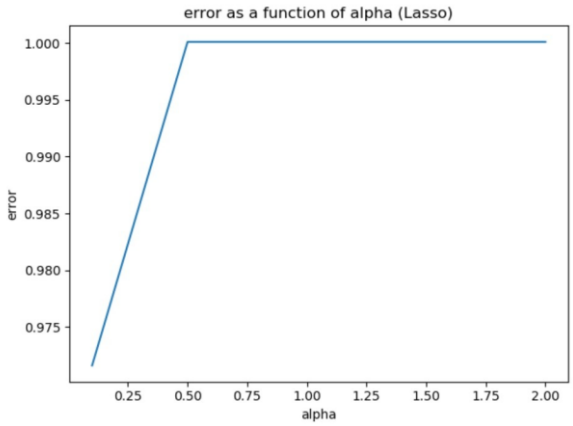
Baseline algorithm

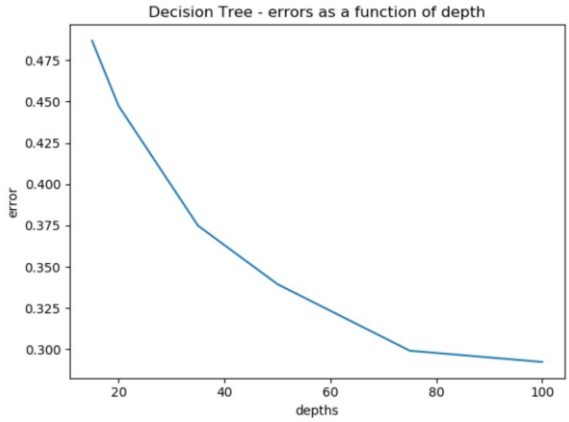
בחרנו לאלגוריתם הbaseline את ה- Decision Tree, הרצנו אותו עבור עומקים שונים על מנת לבחון את ההתנהגות שלו על הtrain data. החלטה זו התקבלה בעיקר על פי אינטואיציה, ומתוך מחשבה כי במודל זה ניתן להגיע לbias גבוה וvariance נמוך.

Model Selection

לאחר בחירת ה Baseline algorithm, אימנו מודלים שונים (כגון Decision Tree, Random forest, Lasso) עם פרמטרים שונים, עבור הtrain data שלנו ובחנו את הביצועים שלהם אל מול הvalidation test הגענו למסקנה כי המודל הטוב ביותר עבור המטלה שלנו (על פי איכות הביצועים שהתקבלו) הוא Decision Tree בעומק של 100. להלן הגרפים בעזרתם בחרנו את המודל הטוב ביותר-







בנוסף ניסנו לשפר את ביצועי המודלים בעזרת adaboost עם בין 15 ל50 חברי ועדה כאשר התוצאה הטובה ביותר הייתה עבור הDecision Tree הגענו לכ60% הצלחה. ניסיון זה לא צלח שכן קיבלנו שגיאה טובה יותר על הvalidation set עבור Decision Tree פשוט בעומק 100.
כפי שניתן לראות ברוב המודלים קיבלנו שגיאה די גדולה, מלבד המודל של Decision Tree שהיה בעל השגיאה הטובה ביותר.

לבסוף, לאחר ביצוע כל התהליך ובחירת המודל הנכון ביותר, אימנו מחדש את המודל על כל הדאטא כלומר על- train, validation and test.
תהליך העבודה היה מאתגר מעשיר וללא ספק חווייתי, למדנו המון ואנו מקווים שהתוצרים שלנו מהווים אסמכתא לתהליך שעברנו.