דו"ח מסכם

פרויקט מסכם בלמידת מכונה

זיהוי קבצים זדוניים

30/06/2023

אריאל חדוות, 323842914

יובל רובינשטיין, 209668763

**אקספלורציה:**

בהתחלה חקרנו את התפלגות הפיצ'רים השונים שיש בנתונים. ניסינו לראות כיצד הם מתנהגים וראינו שברוב הפיצ'רים קשה לזהות זאת ללא טרנספורמציית לוג, לכן לצד כל גרף צפיפות לפיצ'ר, הוספנו את הגרף המתאים ללוג שלו. לדוגמה, הפיצ'ר printables ללא טרנספורמציית לוג נראה כמו "פס" ולאחר טרנספורמציית הלוג הפיצ'ר נראה כי הוא מתפלג נורמלית (ראה נספח 1). בנוסף גם ניתן לראות כי פיצ'ר A אינו זקוק להתפלגות לוג ומתפלג נורמלית כמו שהוא (ראה נספח 2). לכן הכנו רשימה של הפיצ'רים שמתפלגים נורמלית לפי הגרפים. לאחר מכן חקרנו באמצעות אמצעים ויזואליים וסטטיסטים את הפיצ'רים בדאטה שלנו. ניסינו למצוא קשרים בינים לבין עצמם, את השפעתם על ה-label וקבלת מידע נוסף שניתן להבין "בין השורות" לאחר עיבודים מסוימים. החלטנו לחקור את הפיצ'רים הבוליאניים (פיצ'רים ה-has\_...) ולבדוק באמצעות מבחן חי בריבוע האם יש תלות בינים לבין ה-label. הגענו למסקנה ע"י PV שקטן מ-0.05 שכל פיצ'ר has כזה משפיע. לכן, החלטנו לחקור את האינטראקציות בין כל שני פיצ'רים בוליאניים (באמצעות "וגם"), והגענו למסקנה שההשפעה שלהם משתנה בהתאם ל-label באמצעות מפות חום (ראה נספח 3). לאחר מכן המשכנו לעבור על פיצ'רים שמעניינים אותנו בדאטה ולנסות להבין אותם באמצעות ויזואליזציות כאלה ואחרות. חקרנו את פיצ'ר הsymbols- וראינו כי רובו אפסים אך כן יש שינוי בהתפלגות בין קוד זדוני ללא זדוני( ראה נספח 4). בנוסף, חקרנו את הפיצ'ר file\_type\_proba\_trid וניסינו להבין האם יש קשר בין הסתברות גבוהה או נמוכה שקובץ מסוים הוא סוג הקובץ שנאמר תחת file\_type לבין היותו קובץ זדוני וראינו שאין קשר ברור ממש (ראה נספח 5). בנוסף, מאחר והפיצ'ר הינו קטגוריאלי אך עם הרבה אפשרויות, רצינו לחקור גם את המילים הכי נפוצות בכל הקטגוריות (ראה נספח 6).

**עיבוד מקדים:**

את חלק העיבוד המקדים התחלנו בעדכון וביצוע טרנספורמציות על הפיצ'רים הנמצאים בדאטה-סט (ראה Step 1 of pre-processing במחברת). התחלנו בהתמודדות עם פיצ'ר קטגוריאלי C, שבחלק האקספלורציה ראינו כי הוא מכיל כמות יחסית לא גדולה מידיי של מילים המאפיינים אותו (ראה נספח 7), ועל כן החלטנו להפוך משתנה זה לדאמי באמצעות OneHotEncoder, כאשר הקבוצה "המושמטת" היא קבוצת ה-unknown. לאחר מכך, ביצענו טרנספורמציות לוג עבור הפיצ'רים שבאקספלורציה ראינו כי המעבר ללוג מגדיל את השונות או מאפשר התפלגות נורמלית ונוחה יותר לעיבוד כי זה יכול לחזק קשרים בין פיצ'רים ואפילו להקל על זיהוי וטיפול בחריגים (ראה נספח לדוגמה 8). לאחר מכן, עברנו לטפל בפיצ'ר קטגוריאלי נוסף file\_type\_trid המכיל יחסית הרבה קטגוריות (ראה נספח 9), לכן החלטנו לא להשתמש בשיטה של מעבר למשתני דאמי מאחר ומספר הפיצ'רים יגדל משמעותית, דבר המגדיל את השונות ויגרום למודל להיות מורכב מידיי מה שיצור חשש שהמודל יותאם מידיי לנתוני האימון. לכן החלטנו להשתמש בשיטת ה- tf-idf מאחר וכל קטגוריה בתוך הפיצ'ר הינה משפט, כאשר כל משפט (המייצג סוג של קובץ) מכיל מילים שעשויות להופיע בכמה סוגי משפטים. החלטנו לקחת את ה- score שקיבלנו על כל מילה מה- tf-idf ועליהם לבצע רגרסיה לוגיסטית על ה label כך שנקבל פיצ'ר חדש שנוסיף לדאטה הנקרא sus\_type\_proba, המייצג את ההסתברות שקובץ הוא זדוני לפי ה - file\_type\_trid והמילים שמופיעות בו. בעת העבודה, התלבטנו כיצד להתמודד עם הפיצ'ר הקטגוריאלי file\_type\_trid. תחילה חשבנו, לקחת כמה פיצ'רים שהמייצגים את המילים עם הקורלציה החזקה ביותר ל-label עם הציון שקיבלו מה- tf-idf, אבל לא היינו בטוחים כמה מילים לקחת ובסופו של דבר הרגשנו שהוספנו כמות גדולה מידיי של פיצ'רים לדאטה שלנו, כאשר הרגרסיה הלוגיסטית פשוט נותנת לנו פיצ'ר אחד ופותרת בעיה זו, ומתחשבת גם בקומבינציות של המילים.

לאחר מכן, חשבנו שהשלב הנכון הבא יהיה הוספת פיצ'רים חדשים ע"י הבנה של עסקית על הנתונים ואינטראקציה ביניהם (ראה Step 2 of pre-processing במחברת). הוספנו שני פיצ'רים שראינו לנכון להוסיף כרגע ובנוסף היה לנו רעיון בהוספת פיצ'רים המייצגים "וגם" בין הפיצ'רים הבוליאניים אבל, ראינו לנכון להוסיף פיצ'רים אלו לאחר השמטת התצפיות החריגות וערכי ה-Nan על מנת לא ליצור תוצאות שקריות עבור אינטראקציות אלו (עקב עיבוד כזה או אחר במילוי ערכי ה-Nan). פיצ'ר ראשון שהוספנו לדאטה שלנו הוא code\_density המורכב מחלוקה של vsize ב- size ומתארת את יעילות הזיכרון של קובץ ההפעלה, פיצ'ר שעשוי להיות מתואם עם קבצים זדוניים. לאחר טרנספורמציה לוג על פיצ'ר זה ראינו כי הוא מתפלג נורמלית, אך עם שונות די קטנה (ראה נספח 10). פיצ'ר נוסף שהוספנו נקרא i\_e\_ratio המייצג את יחס יבוא-יצוא שיש בקובץ. לאחר קריאה, הבנו כי קבצים זדוניים יכולים לנצל יחס יצוא-יבוא גבוה (המרמז על תלות גדולה יותר ביבוא) על ידי מינוף נקודות תורפה במשאבים או רכיבים מיובאים, שעלולים להוביל לפרצות אבטחה.

כעת, (ראה step 3 of pre-processing במחברת) נוריד את הפיצ'רים ה"מיותרים", תחילה נוריד את הפיצ'רים שאי אפשר לחשב איתם / למדל איתם, כגון sha256 וfile\_type\_trid שטיפלנו בו מקודם. לאחר מכן חשבנו להוריד הפיצ'רים עם שונות קטנה, שהם: symbols, registry, exports (ראה נספח 11). אבל לאחר שראינו שהתפלגות הפיצ'רים משתנה בהתאם ל-label, החלטנו להשאיר אותם, וראינו שזה משפר את טיב ביצועים של המודל. בנוסף נוריד את הפיצ'רים בעלי קורלציה גבוהה לפיצ'רים אחרים, שניתן לראות זאת באמצעות מטריצת הקורלציה (ראה נספח 12), הורדנו את: size, vsize, numstrings, imports, exports . כאשר החלטנו איזה פיצ'רים להוריד ווידאנו שאין להם קורלציה ממש גבוהה עם ה-label ובמידה וכן שקלנו זאת שוב.

בחלק זה בדקנו האם קיימים נתונים חריגים בדאטה שלנו (ראה Step 4 of pre-processing במחברת). תחילה הסתמכנו על כך שיש לנו פיצ'רים שמתפלגים נורמלית, והשתמשו בכלי הסטטיסטי stats.zscore והעפנו תצפיות על סמך הסטייה שלהם מהממוצע. השלב הבא הוא להבין איזה מבין השיטות הקלאסטרינג: DBSCAN, isolation forest, Kmeans היא הטובה ביותר עבור הפיצ'רים שאינם מתפלגים נורמלית. לבסוף החלטנו ללכת על שיטת ה- isolation forest מאחר ולאחר בדיקת הגרפים המתאימים (ראה נספח 13) עבור היפר-הפרמטר כמה נתונים מהדאטה הם חריגים וצריך להסירם, ראינו כי DBSCAN ו- Kmeans חוזים יותר מידיי נתונים חריגים (מגיעים לאלפי תצפיות ובמקרים חריגים גם ליותר מרבע הנתונים שלנו), מה שלדעתנו נראה מוגזם ודבר שעשוי לגרום להורדה בביצועי חיזוי המודל העתידי שלנו.

בחלק זה בדקנו האם קיימים נתונים חסרים בדאטה (ראה Step 5 of pre-processing במחברת), כלומר ערכי Nan אשר לא מועילים לנו בחיזוי העתידי ועלינו לטפל בהם. החלטנו לחלק זאת לטיפול בערכים החסרים עבור הפיצ'רים הבוליאניים ועבור הפיצ'רים הנומריים. עבור כל פיצ'ר בוליאני ניקח את הערך הנפוץ ביותר, כלומר אם רוב התצפיות עבור אותו ערך בוליאני הן אפסים אז נמלא את כלל ערכי ה nan באפסים גם כן. זה מאפשר לנו להניח הנחה מושכלת לגבי הערכים החסרים בהתבסס על השכיחות של הערך השכיח ביותר. גישה זו עוזרת לשמר את התפלגות ומאפיינים הכוללים של הנתונים, מה שהופך אותם לבחירה סבירה לטיפול בערכים בוליאניים חסרים. לאחר שטיפלנו בערכים החסרים בפיצ'רים הבוליאנים, ניתן לבצע את האינטראקציות שחשבנו עליהם בשלב האקספלורציה, שזה הוספת פיצ'רים המייצגים "וגם", כלומר האם קיים שילוב "מחשיד" בין שני פיצ'רים בוליאניים. נשתמש ברשימת המילונים שיצרנו באקספלורציה (interactions) המייצגת את אותם שילובים וכאן ניצור את הפיצ'רים המתאימים באמצעות "וגם". לאחר מכן, נטפל בערכים החסרים עבור הפיצ'רים הנומריים ששם השתמשנו בכלי הנקרא SimpleImputer המאפשר החלפת הערכים החסרים בדאטה בסטטיסטיקה המחושבת ע"י אימון על הערכים שאינם חסרים. אנחנו החלטנו להשתמש באסטרטגיית החציון המחליפה ערכים חסרים בחציון של הערכים הלא חסרים באותה עמודה. בדקנו וראינו שאסטרטגיה זו נותנת לנו ביצועים גבוהים יותר של המודל הסופי בהשוואה לממוצע.

בחלק זה נתעסק בנרמול הדאטה (ראה Step 6 of pre-processing במחברת) שחשוב במיוחד עבור אלגוריתמים שאינם מבוססי עץ, כמו K-NN, נורמליזציה תשחק תפקיד גדול, מכיוון שחישוב מרחק / כל סוג של אפקט (ערכים גדלים / יורדים) היינו רוצים שכל התכונות יגיעו לקנה מידה או טווח דומה, כך שניתן יהיה להשוות ולנתח אותן במגרש שווה, ובכך להעלות את אחוזי ההצלחה של המודל שלנו. החלטנו להשתמש בשיטת ה- Standard-scaler לביצוע הנרמול מאחר וזה אינטואיטיבי. לצורך העניין, מה שמעניין אותנו זה לא רק הערכים עצמם, אלא גם אפקט / חריגה מהממוצע בכל פיצ'ר. ובנוסף, עבור מודלים בהם נצטרך לבצע הקטנת ממדים באמצעות PCA, Standard-scaler עובד הכי טוב עם שיטה זו.

בנוסף, לפני שנמשיך אל שלב הקטנת הממדים ע"י שימוש ב – PCA, נרצה לראות את משמעות הפיצ'רים שיש לנו כרגע. הרי לאחר שימוש ב- PCA אנחנו מאבדים את המשמעות העסקית שכל פיצ'ר מייצג. השתמשנו במודל ExtraTreesClassifier זהו מודל שיודע להתמודד עם ממדים רבים במהירות יחסית שכן הוא בוחר באקראי תת-קבוצה של תכונות בכל צומת של עצי ההחלטה, מה שיכול לסייע בהפחתת התאמת יתר ולשפר הכללה בעת הבנת החשיבות של התכונות. היה לנו חשוב לראות האם לאינטראקציות ולתוספת הפיצ'רים החדשים שעשינו בנתונים יש השפעה על החיזוי והאם ביצענו תוספת חכמה של פיצ'רים (ראה נספח 14). כפי שניתן לראות, חלק מהאינטראקציות ומשתני הדאמי מופיעים אפילו בטופ 10 המשתנים הכי משפיעים על ה-label.

בשלב האחרון של העיבוד המקדים (ראה step 7 of pre-processing במחברת), החלטנו לדון בממדי הדאטה שלנו. ממבט ראשון נראה כי ממדיות הבעיה כן יכולה להיות גדולה מדי, מצב שנובע גם מהוספת פיצ'רים חדשים, לדוגמה האינטראקציות בין המשתנים, וגם כי מלכתחילה אנחנו עובדים עם דאטה העשוי להכיל כמות גדולה מדי של ממדים. מצב זה יכול להוות בעיה בביצועים של המודלים שלנו, ככל שהממדיות גדולה יותר יש יותר "רעש" ואנחנו יכולים להגיע לאוברפיטינג, בנוסף ככל שיש יותר ממדים, קשה יותר לזהות כל מיני דפוסים בדאטה שלנו, מה שיכול להקשות על אימון המודל או הדיוק. כדי לדעת אם אנחנו עם יותר מדי ממדים, נשתמש בשתי שיטות: הראשונה,  Cross-validation והשנייה, אמידת טיב הביצועים וקבלת ציון על בסיס סט validation ששמנו בצד ולא נעזרנו בו בעת האימון (הוצאת חריגים וטיפול בערכים חסרים). כדי לצמצם את הממדים, נשתמש ב2 שיטות נפרדות בהתאם למודלים, הראשונה PCA עבור מודלים לא מבוססי עצים. השיטה השנייה, עבור מודלים מבוססי עצים, נשתמש במגבלות כגון: max\_features, max\_depth ועוד, שנראה בשלב המודלינג. החלטנו לבחור בPCA ולא Feature selection משיקולים של זמן ריצה, כאשר ניסנו לבצע backwards selection, זמן הריצה חרג מהמותר. בחרנו לעשות PCA כך ש90% מהשונות תהיה מוסברת, מאחר והסתכלנו על הגרף המראה כמות הממדים פר שונות מוסברת וראינו כי אחרי 0.9, כמות הממדים עולה משמעותית (ראה נספח 15). בנוסף חשבנו לבצע גם PCA על הפיצ'רים הנומריים בלבד (ללא הבוליאנים), על מנת לשמור חלק מהמשמעות של הפיצ'רים. חשבנו שאולי זה יסייע לנו בחלק מהמודלים אך ראינו שהביצועים לא כל כך טובים בהשוואה לPCA הכללי.

**הרצת והערכת המודלים:**

על מנת למצוא את היפר-הפרמטרים הטובים ביותר עבר כל מודל ולבדוק את הביצועים שלו נעזר בפונקציה (ראה במחברת getBestModel) אשר מקבלת אומדן ומיליון של פרמטרים וערכים, ומבצעת grid-search עבור כל שילוב של פרמטרים (מאלה שהבאנו לה) ומשתמשת באימות kfold על מנת למצוא את השילוב הטוב ביותר שנותן את התוצאות המדויקות ביותר, כך נקבל את ההסתברות המדויקת ככל הניתן לזיהוי קבצים זדוניים / לא זדוניים עבור כל מודל. בחנו להשתמש בשיטת ה- grid-search מאחר והיא מבצעת בדיקת אוטומטית ועוזרת לנו למצוא את שילוב היפר-הפרמטרים הכי טובים יחסית בקלות.

בחלק המודלים הפשוטים החלטנו לבצע רגרסיה לוגיסטית ו-KNN .  
רגרסיה לוגיסטית - עבור הרגרסיה הלוגיסטית, אנו נותנים טווח של Cs (טווח עבור עונשים / רגולריזציות), כמו כן עונשים שונים (לאסו או רידג'), class\_weight עשוי להשפיע על הביצועים ו-fit\_intercept (האם צריך חותך ברגרסיה, השפעה של הטיה שיכולה לסייע לנו להוריד את השונות) להיות נכון או לא נכון, מכיוון שאנו עשויים להזדקק לתוספת של הטיה. במודל זה הגענו להיפר פרמטרים C=1, class\_weight = balanced, fit\_intercept=True, penalty = l1, כלומר הגענו לC הכי גבוה שנתנו והענשה לאסו, כלומר הגענו למצב של overfitting בגלל ה-C הגבוה אבל תיקנו זאת ע"י לאסו.  
KNN - עבור מודל ה- KNN אנחנו נותנים את מספר ההיפר-פרמטר של השכנים (n\_neighbors). במודל זה הגענו לכמות שכנים של 11, שלפי הgridsearch הוא הערך האידאלי בין שונות והטיה. (יותר שכנים = פחות שונות אך יותר הטיה).

בחלק המודלים המודלים המורכבים החלטנו לבצע Gradient Boosting וMLP .  
Gradient Boosting - עבור מודל זה אנחנו נותנים טווח של learning rate (האלפא), ככל שאלפא יותר קטנה צריך יותר עצים, נקבע ל- 0.05 גם מתוך ניסוי וטעיה וגם מתוך האינטואיציה שזה מאוזן בין מהירות לזהירות, max\_feauresכמות פיצ'רים מקסימלית כדי למצוא את האמצע בין הורדה של ההטיה ולמנוע גדילה משמעותית בשונות, (יותר פיצ'רים = פחות הטיה ויותר שונות), עומק עץ מקסימלי (יותר עומק = פחות הטיה ויותר שונות) כאשר קיבענו את כמות העצים ל-500 מכיוון שאנחנו לא חוששים מoverfitting כי זה יחסית נדיר לקבל זאת ב- boosting אך בכל זאת נתנו לאלפא טווחים על מנת להגיע לאלפא האידאלית לכמות העצים הזו כדי לחסוך זמן ריצה ושכל המחברת תרוץ תוך שעה. הגענו להיפר פרמטרים: מקסימום פיצ'רים = 35% מהפיצ'רים המקוריים, עומק מקסימלי = 15.  
 MLP- עבור מודל זה נתנו טווח עבור ערכים אפשריים ל alpha (הכיווץ) וערכים אפשריים ל- learning\_rate\_init (גודל הצעד ההתחלתי שגם משפיע על זמן ההתכנסות) וכמות השכבות (יותר שכבות =יותר שונות אך פחות הטיה). הגענו לפרמטרים אלפא = 0.01, 200 שכבות וlearning\_rate\_init = 0.01.

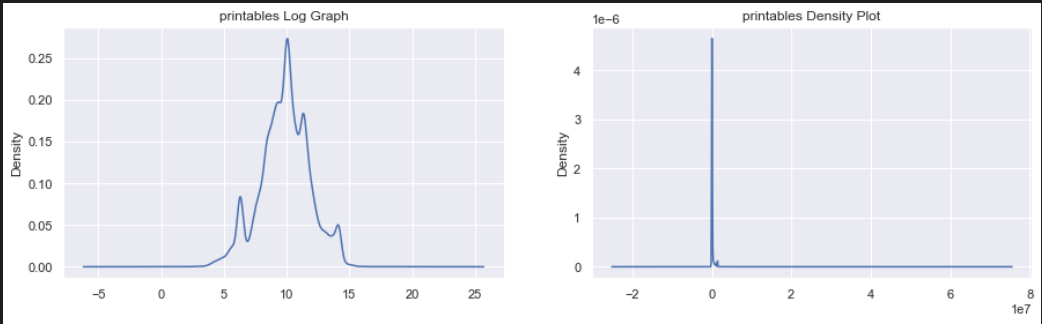
בחלק זה בחרנו להציג את ה- Confusion Matrix עבור המודל הטוב ביותר שלנו, ה- GradientBoosting. ה- Confusion Matrix מספקת לנו עבור נתוני המבחן רמת דיוק של 0.925 כלומר מתוך כמה שהמודל חזה, הוא צדק בכ 92.5% מהמקרים. סה"כ מתוך 12000 התצפיות של נתוני המבחן (ששמנו בצד), המודל חזה נכון ואמר שהקובץ אכן זדוני עבור 5447 תצפיות (TP), המודל חזה לא נכון ואמר שהקובץ זדוני למרות שהוא אינו זדוני עבור 393 תצפיות (FP), המודל חזה נכון ואמר שהמודל אינו זדוני עבור 5662 תצפיות (TN) והמודל חזה לא נכון ואמר שהקובץ אינו זדוני למרות שהוא זדוני עבור 498 תצפיות (FN) (ראה נספח 16). לפי הconfusion matrix אנו מבינים שברוב המחולט של המקרים נקבל את התוצאה האמיתית (עבור threshold>=0.5), אנחנו כן יכולים להסיק על כך שהמודל שלנו טוב, מכיוון שהתפלגות של ה-label הוא די סימטרי לפי האקספלורציה (חצי זדוני וחצי לא) כלומר לא צריכים לחשוש ממצב קיצוני שבו אפשר רנדומלית לתת תוצאה והדיוק יהיה גבוה. כשאנחנו מסתכלים על גרף הROC לכל מודל על פי KFold, ניתן לראות כי המודל Gradient Boosting מבצע הכי טוב עם בערך AUC = 0.98 לכל פולד (ראה נספח 17) וגם עבור הAUC של הvalidation set ששמנו בצד מקבלים גם בערך אותו דבר (ההבדל מאוד זניח).

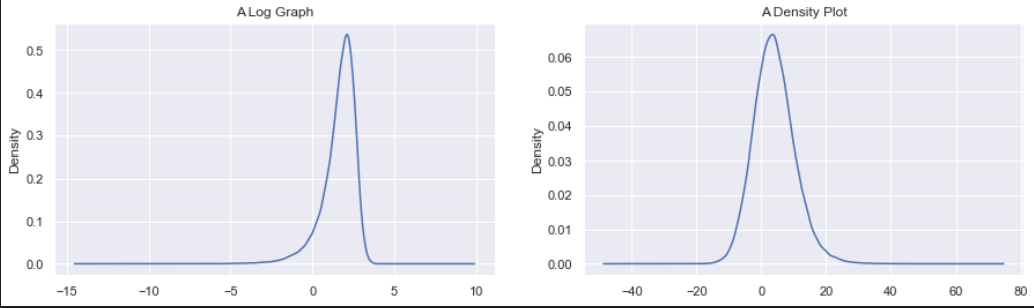
הפערי הביצועים בין ה- train ל- :test ניתן לראות כי ישנו פער קטן של כ- 0.02 בין ה- AUC של ה- train לבין ה-AUC של ה- test (ראה נספח 18), כאשר אנו מסתכלים על המודל בו נשתמש לחיזוי Gradient Boosting . כאשר יש פער קטן זה מצביע על כך שהמודל שלנו מתפקד יחסית טוב עבור נתוני האימון וגם עבור נתוני הבדיקה, דבר המקטין את הסיכויים להיות ב- overfitting . במהלך הפרויקט דאגנו לחלק את הדאטה שלנו לנתוני אימון ומבחן ע"י Test-Train split כבר מתחילת שלב העיבוד המקדים על מנת להתאמן ולצבור מידע על נתוני האימון בלבד, על מנת לבדוק את הביצועים שלנו על נתוני המבחן שבצד ולמנוע מצב של "זיהום" שלהם או "מודל מושלם" שלא באמת יחזה שום דבר ריאלי כאשר יקבל תצפיות חדשות.

**Pipeline:**

במהלך העבודה דאגנו ליצור תשתית נוחה לצורך ביצוע ה .pipeline דאנו ככל הניתן ליצור פונקציות עבור כל שלב בעיבוד המקדים על מנת שב pipeline נוכל לקרוא להן עם פרמטרי הקלט של נתוני האימון והמבחן בלבד. בסוף ה- pipeline היה לנו חשוב לוודא כי אין לנו שינוי גדול מידיי בין תוצאות החיזוי שקיבלנו בעת הרצת הפרויקט עצמו לבין הרצת ה,pipeline דבר שעשוי להיות מחשיד ואף מעיד על התאמת יתר (בכל זאת יהיה שינוי קטן כי ב pipeline מריצים את המודל על כל הדאטה ובשאר הפרויקט מבצעים Test-Train split). בדקנו זאת על ידי גרף צפיפות והפרש התחזיות וראינו כי אכן השונות קטנה ואין שינויים ממש משמעותיים בין התחזיות, כלומר המודל שלנו עובד בצורה יחסית אמינה (ראה נספח 19).

**נספחים:**





A picture containing text, screenshot, diagram, number

Description automatically generated A picture containing text, screenshot, number, diagram

Description automatically generated

A picture containing text, screenshot, number, diagram

Description automatically generated A picture containing text, screenshot, number, diagram

Description automatically generated A picture containing text, screenshot, number, diagram

Description automatically generated A picture containing text, screenshot, number, diagram

Description automatically generated A picture containing text, screenshot, number, diagram

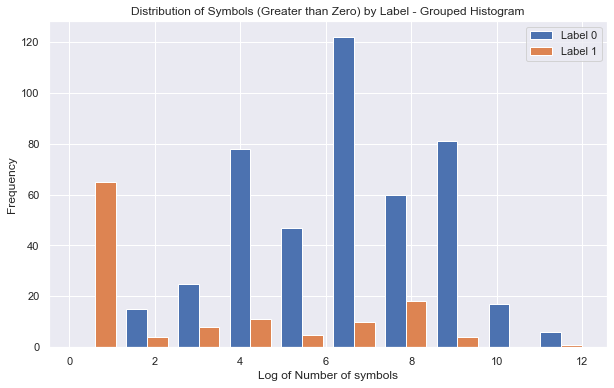
Description automatically generated A picture containing text, screenshot, number, font

Description automatically generated A picture containing text, screenshot, number, font

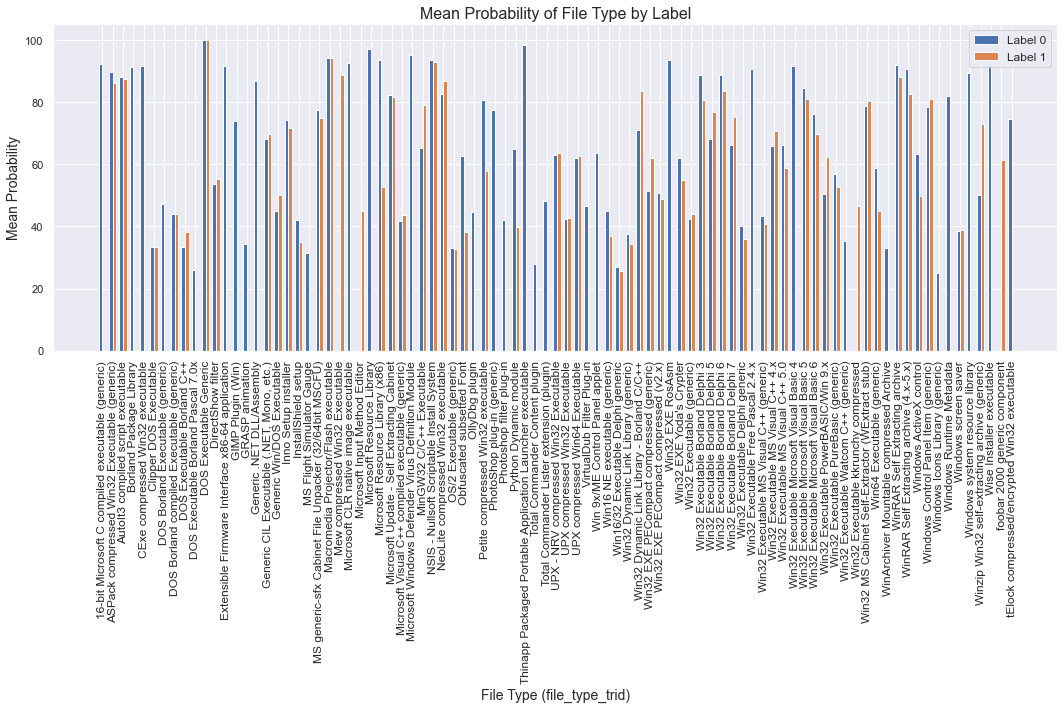
Description automatically generated A picture containing text, screenshot, number, diagram

Description automatically generated









A picture containing circle, screenshot, diagram, colorfulness

Description automatically generated

A picture containing text, screenshot, diagram, plot

Description automatically generated

A picture containing plot, line, screenshot, diagram

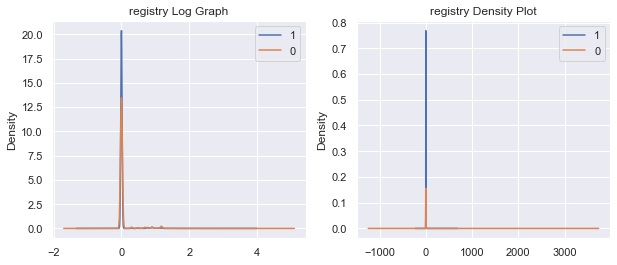
Description automatically generated

A picture containing text, screenshot, line, parallel

Description automatically generated

A picture containing text, screenshot, plot, line

Description automatically generated



A picture containing text, screenshot, plot, number

Description automatically generated

A picture containing text, screenshot, plot, line

Description automatically generated

*A picture containing text, screenshot, colorfulness, rectangle

Description automatically generated*

A picture containing text, screenshot, line, plot

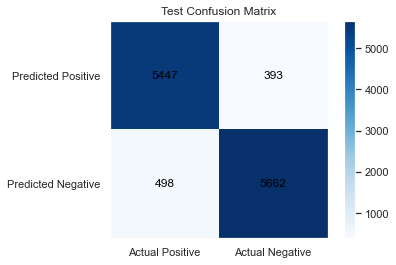
Description automatically generated

A picture containing text, screenshot, diagram, line

Description automatically generated

A picture containing text, screenshot, line, plot

Description automatically generated



A picture containing text, line, plot, screenshot

Description automatically generated

A picture containing text, line, plot, screenshot

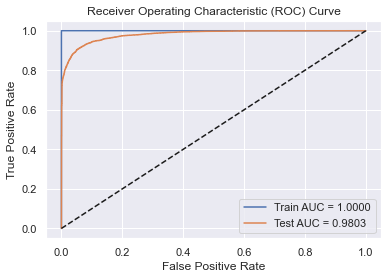
Description automatically generated

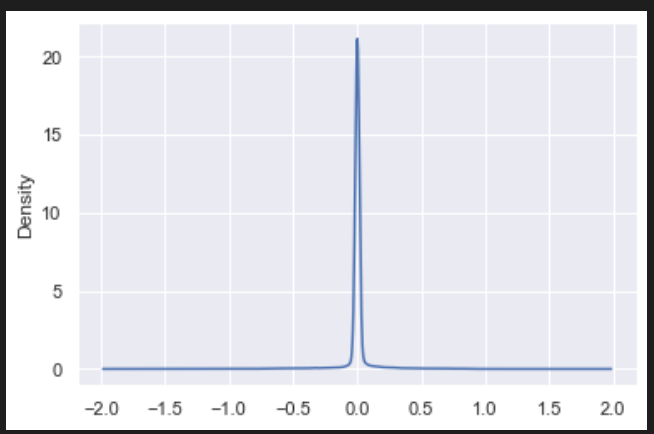
A picture containing text, screenshot, line, plot

Description automatically generated

A picture containing text, line, screenshot, plot

Description automatically generated





**נספח: כלים שלא נלמדו בקורס:**

1. חי בריבוע – בחלקים מהאקספלורציה ניתן לראות שכתוב "p-value", זה בעצם הערך המינימלי של רמת המובהקות בו ניתן לדחות את ההנחה ששני פיצ'רים בלתי תלויים. רמת המובהקות היא ההסתברות לטעות מסוג ראשון (false positive), ככל ש ה"p-value" קטן יותר, כך רמת המובהקות נמוכה יותר, ולכן רמת הבטחון גבוהה יותר ולכן ניתן להיות יותר בטוחים בהנחה ששני פיצ'רים תלויים זה בזה, זה נעשה בפועל על ידי השימוש בpd.crosstab שסופר לנו כל שילוב, וchi2\_contingency שמביא לנו גם את הערך של חי בריבוע וגם הp-value . בחרנו להשתמש בזה כדי להבין בעיקר איזה פיצ'רים ה-label תלוי בהם ואיזה לא, ועל כן להבין על איזה פיצ'רים לא נוותר.
2. SimpleImputer – מחלקה בsklearn המאפשרת החלפת ערכים חסרים בערך קבוע או בסטטיסטיקה שניתן לעשות על הדאטה המאומן. בחרנו להשתמש במחלקה הזו על מנת להתמודד עם הערכים החסרים, ולהחליף את כלל הערכים החסרים בחציון של ערכי האימון (ללא ערכי קיצון שהורדנו), המודל הזה מאפשר לנו לבצע fit על דאטה האימון ואז להשתמש במודל המאומן (לעשות transform) על מנת לטפל בערכים חסרים בtest. כפי שציינו קודם, בחרנו להשתמש בחציון מאחר וזה שיפר לנו את ביצועי המודל.
3. ExtraTreeClassifier – מדובר במודל של עצי החלטה שדומה יחסית לRandomForest, אבל בשונה ממנו הוא מפצל את העלים בצורה רנדומלית (שעוזר למנוע overfitting) ויודע לההתמודד עם מימדים יחסית גדולים במהירות ויותר חסין ל"רעש" מאשר RandomForest, מה שעוזר לבצע בדיקה זריזה לפני ביצוע הPCA על מנת להבין האם

הפיצ'רים החדשים שהוספנו מועילים או לא.

**נספח: הבנה על בחירת היפר-הפרמטרים:**

בעת ביצוע המודלים: gradient boosting ו- MLP החלטנו לנסות ערכי היפר פרמטרים שונים, לתת לקוד לרוץ (יחסית הרבה זמן) ולבסוף לקחת את שילוב היפר הפרמטרים הטוב ביותר שהתקבל מפונקציית getBestModel מה grid search.

עבור ה- gradient boosting נתנו טווח 'learning\_rate' : [0.05, 0.1, 0.15] ו- 'n\_estimators' : [500, 1000] וקיבלנו כי השילוב המדויק ביותר הינו learning\_rate = 0.05 ו n\_estimators=500

עבור ה- MLP נתנו טווח 'hidden\_layer\_sizes' : [100, 200] והגענו לזה ש- 200 הכי טוב.

**נספח: תרומת כל אחד מאיתנו לפרויקט:**

מרבית העבודה נעשתה בצורה משותפת, כאשר נפגשנו פרונטלית לשבת על הפרויקט מתחילתו ועד סופו, התחלקנו בעבודה כך שפעם אחת אחד כתב את הקוד והשני חקר את הדאטה ולהיפך. בנוסף פתחנו GIT עבור הפרויקט כך שכל חבר צוות יוכל להיות מעודכן באופן שוטף בעבודה. התחלנו את העבודה בכך שכל אחד חקר בעצמו את הפיצ'רים ולקח את זה לכיוון שלו, אריאל חקרה את התלות בין הפיצ'רים הבוליאנים ל-label ויובל חקר את האינטרקציות בין הפיצ'רים הבוליאנים, בנוסף שאר הוויזואליות הן שילוב של שנינו כאשר כל אחד חקר את הדאטה לפי ראות עיניו. לאחר מכן בעיבוד המקדים, חשבנו ביחד מה הסדר הנכון של השלבים. יובל עבד על התמודד עם משתנים קטגוריאליים, חלק מהפיצ'רים החדשים, ביצוע PCA ונרמול. אריאל עבדה על טיפול הערכים החסרים, מחיקת פיצ'רים, חלק מהפיצ'רים החדשים ובדיקת חשיבות של הפיצ'רים. ביחד עבדנו על טיפול בנתונים החריגים ושילבנו בין הקוד שכל אחד כתב ובין הרעיונות של שנינו, וכמובן יצא שבסוף כל אחד מאיתנו עבד ושיפר את מה שהשני עשה. בשלב הרצת המודלים חשבנו ביחד איזה מודלים הכי נכון לבצע וכל אחד עשה חקירה אישית משלו על איזה טווח של היפר פרמטרטים הכי טוב לתת. חשבנו ביחד לכתוב פונקציה שמבצעת gridsearch ולאחר מכן ביצענו את המודלים ביחד. כל אחד מאיתנו בזמן האישי שלו ניסה לשחק עם טווחי ההיפר פרמטרים שנתנו למודלים. אריאל עבדה על הconfusion matrix ויובל עבד על הערכת המודלים באמצעות ROC ו KFold. לאחר מכן עבדנו ביחד על ביצוע הפרדיקציה, והפייפליין (כאשר עבדנו ביחד על הפיכת רוב הפרויקט שלנו לפונקציות וייעול העבודה).