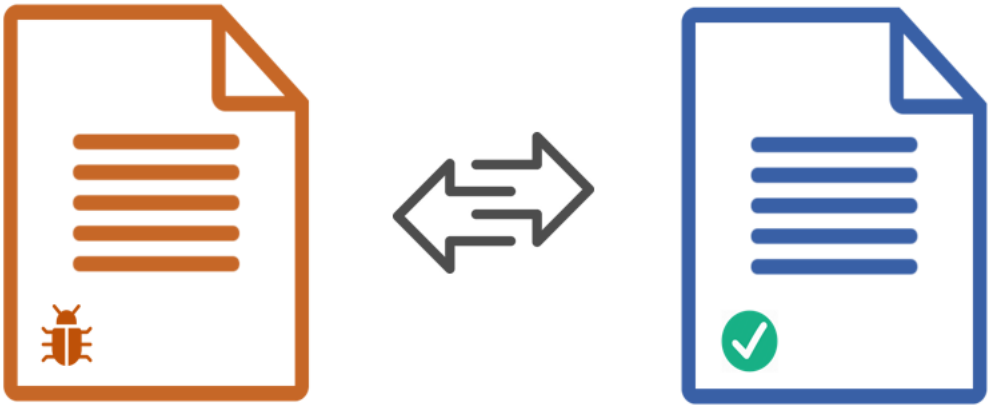
פרויקט מסכם – מבוא ללמידת מכונה

- חיזוי קבצים זדוניים -

מרצה – דור בנק
מתרגל – אילן וסילבסקי
תאריך הגשה – 30.06.2023



מגישים:
איתן ויובל בקירוב

למידת מכונה – פרויקט סופי

תקציר פרויקט

משימתנו בפרויקט היא לפתור בעיה בינארית, בעזרת ניתוח סטטי של הקבצים, בה היה עלינו לסווג רשומות לשתי קטגוריות - קובץ זדוניי (malicious 1) או לא קובץ זדוני (benign 0) על סמך מספר פיצ'רים בדאטה סט. חלק מהפיצ'רים ידועים וחלקם אנונימיים.

לביצוע הפרויקט ניתנו בידינו 60,000 תצפיות מסווגות לקבצים זדוניים וקבצים שאינם זדוניים.
ראשית, ביצענו אקספלורציה על הנתונים הקיימים, חקרנו אותם והגענו למסקנות כלליות על אופי התפלגות הפיצ'רים, התנהגותם הקורלטיבית ונתוניהם הסטטיסטים.
לאחר הוויזואליזציה המעמיקה, חילקנו את הנתונים ל- data ול- validation והתחלנו בהליך העיבוד המקדים בו יישמנו את המסקנות שעלו מחקר הנתונים. בין היתר השלמנו ערכים חסרים, התמודדנו עם ערכים חריגים והוספנו פיצ'רים רלוונטיים.
אחרי שחקרנו את הנתונים ועיבדנו אותם הגיע הזמן להריץ את מודלי למידת המכונה שלמדנו לאורך הקורס. הרצנו ארבעה מודלים, שניים ראשוניים ושניים מתקדמים כדי להחיל עליהם את סט הנתונים ומצאנו את ההיפר-פרמטריים המיטביים.
להערכת טיב המודלים נעזרנו ב- confusion matrix וב- K-Fold cross validation, ובעזרת ציון ה - AUC בחרנו לבסוף במודל ה – Random Forest.
סיימנו עם ביצוע פרדיקציה על נתוני קובץ הטסט הלא ידוע.

חלק ראשון – אקספלורציה

בחלק זה נחקור את הנתונים, נתמקד בהתפלגות הפיצ'רים, התנהגותם הקורלטיבית נתוניהם הסטטיסטים.
בהסתכלות ראשונית, נראה שבדאטה סט 60,000 תצפיות (שורות) ו 24 מאפיינים של הקובץ (עמודות) – 23 פיצ'רים שהם למעשה תכונות הקובץ, ועמודה נוספת - לייבל (label) שאומרת לנו האם הקובץ הוא קובץ זדוני או קובץ לא זדוני. נפריד את כל 23 הפיצ'רים מעמודת הלייבל, וזה יהיה מעתה דאטה האימון שאיתו נעבוד.

תחילה, נרצה לוודא שכל תצפית בדאטה הינה ייחודית, לכן נסיר כפילויות כדי להבטיח שכל שורה מופיעה פעם אחת בקובץ. מהסתכלות על הדאטה נוכל לראות שישנם בדיוק 30,000 קבצים זדוניים ו30,000 קבצים לא זדוניים.
ממבט על סוגי הפיצ'רים נראה שחלק מהפיצ'רים הם אובייקט 'object', חלקם מספר עשרוני 'float64' וחלקם מספר שלם 'int64'. כלומר, הדאטה שלנו מחולק לערכים מספריים או אובייקטים.
הפיצ'רים sha256, file_type_trid, C הם אובייקט. הפיצ'רים vsize, imports, exports, has_debug, has_relocations, has_resources, has_signature, has_tls, symbols, numstrings, paths, urls, registry, MZ, printables, avlength, file_type_prob_trid, A, B הם מספריים עשרוניים. הפיצ'ר size הוא מספר שלם. נראה גם שישנם מספרים שערכיהם הם 0/1, כלומר בינאריים, וישנם מספרים אשר פרוסים על טווחים רחבים של מספרים. לצורך נוחות העבודה עם הנתונים, נגדיר את כל המספרים להיות מסוג 'float64', את הבוליאניים להיות מסוג 'bool' ואת האובייקטים להיות מסוג 'category'.

מהסתכלות על נתונים סטטיסטים של הפיצ'רים השונים הצלחנו להבין הרבה על כל פיצ'ר, על טווח הערכים בו, על כמות הערכים החסרים בו, על ממוצע הערכים בו, השונות, הערך המקסימלי והמינימלי ועל אחוזוני הערכים בו. בעזרת כל אלו נצליח לעבוד בצורה חכמה ונוחה יותר בהמשך.
מהסתכלות על כמות הערכים השונים בכל פיצ'ר למדנו בעיקר על הפיצ'רים הקטגוריאליים. לפיצ'ר 'sha256' ישנם 60,000 ערכים שונים, מה שיכול להצביע עליו כעל פיצ'ר מייצג וייחודי לכל קובץ. עוד ראינו כי לפיצ'ר 'file_type_trid' 89 קטגוריות שונות. לפיצ'ר 'C' 7 קטגוריות שונות. ניתן על אלו את הדעת בהמשך.

התפלגות הפיצ'רים

תחילה נצייר היסטוגרמה לכל פיצ'ר מספרי על מנת לראות כיצד הוא מתפלג (נספח 1.1). ממבט ראשוני ניתן לראות שמרבית הפיצ'רים נראים כמו מאין קו אנכי על ערך 0 בציר ה-x ונסיק כי הערכים החריגים (outliers) שישנם בפיצ'רים השונים מרחיבים מאוד את ההיסטוגרמה כך שקשה להבחין בהתפלגות האמיתית של הפיצ'רים השונים. נתמודד עם הערכים החריגים בהמשך. ניעזר בטרנספורמציית לוג על הפיצ'רים המספריים כדי לכווץ את הערכים כדי שנוכל לראות מקרוב יותר את התפלגויות השונות. מהסתכלות לאחר הטרנספורמציה (נספחים 1.2.1-.1.2.15) נראה שרק הגרף של פיצ'ר 'A' מתנהג כמו התפלגות נורמאלית. אולי התמודדותנו בהמשך עם ערכיים חריגים יעזרו לראות טוב יותר את התפלגות הפיצ'רים הנוספים.

מתאם בין פיצ'רים

נרצה לראות האם קיים מתאם בין הפיצ'רים השונים. ניעזר במדד VIF (Variance Inflation Factor) שיעזור לנו להסתכל בכלליות על כל הפיצ'רים יחד. ציון VIF גבוה מצביע על מולטיקולינאריות גבוהה בין הפיצ'ר לכל שאר הפיצ'רים. ה-VIF עבור כל משתנה מחושב כיחס בין השונות של מקדם הרגרסיה המשוער, לשונות של המקדם אם לפיצ'ר הזה לא היה מתאם עם הפיצ'רים האחרים. ציון VIF קטן מ-1 מצביע על מתאם חלש. ציון בין 1-5 מצביע על מתאם בינוני ואילו ציון גבוה מ-5 מצביע על מתאם גבוה! (נספח - 11 - VIF) הפיצ'רים שמדד ה- VIF מגדיר כבעלי מולטיקולינאריות גבוהה הם: 'file_type_prob_trid', 'numstrings', 'has_resources', 'size', ו – 'B'. לאחר מחקר קטן שביצענו הבנו כי דרך זו לא מספיקה בהכרח כדי לקבוע שפיצ'ר עלול להזיק למודל ויש להסירו.

דרך נוספת שאימצנו לבדיקת המתאם בין הפיצ'רים הוא מטריצת הקורלציות (נספח 1.3) הבודקת מתאם לינארי בין כל שני פיצ'רים. מספר גבוה יותר במשבצת המטריצה מצביע על מתאם גבוה יותר בין שני הפיצ'רים. (+ נספחים 1.4.1-1.4.2)

נוכחנו לגלות שישנו מתאם גבוה של 0.9 בין 'size' ל-'numstrings', מתאם גבוה של 0.7 בין 'size' ו-'MZ' ומתאם של 0.6 בין 'numstrings' ו-'MZ'. בשל הופעת הפיצ'רים המספרים 'size' ו – 'numstrings' גם במדד ה- VIF וגם במטריצת הקורלציות, ואת 'MZ' במטריצת הקורלציות, כבעלי מתאמים גבוהים, נשקול בהמשך להיפתר מאחד או מכמה מהם כדי להפחית את כמות המימדים ולמזער מולטיקולינאריות לטובת שיפור ביצועי המודל.

ביצענו הסתכלויות נוספות על קורלציה בין הפיצ'רים השונים לייבל כדי לנסות ללמוד משהו על השפעתם של הפיצ'רים על זדוניות הקובץ אך ללא מסקנות מיוחדות. (נספח 1.5)

התמודדות עם ערכים חסרים

מהסתכלות ראשונית נראה שישנם כ-34,810 ערכים חסרים בדאטה סט. לפיצ'רים 'file_type_trid', 'size', 'sha256' ו-'file_type_prob_trid' אין ערכים חסרים כלל. לכל היותר 7 אחוז מהתאים בכל פיצ'ר הם בעלי ערכים חסרים. ערכים חסרים יכולים להשפיע על הניתוח והמידול בהמשך לכן עלינו למלא אותם בחוכמה, נעשה זאת בעיבוד המקדים.

התמודדות עם פיצ'רים קטגוריאליים

את הניתוחים נעשה על הפיצ'רים הקטגוריאליים 'file_type_trid' ו-'C' בלבד (נספחים 1.6.1 – 1.6.2), שכן הפיצ'ר 'sha256' הוא פיצ'ר מייצג וייחודי לכל תצפית ולכן לא יוכל לעזור לנו במיוחד. לפני שנתחיל ננסה להבין איך הפיצ'רים הקטגוריאליים "מתנהגים". ראשית, נבחין שלפיצ'ר 'file_type_trid' קטגוריות רבות שמופיעות יחסית מעט פעמים (נספח 1.7). בהתמודדותנו עם הפיצ'רים הקטגוריאליים בהמשך, נרצה לפרק את הפיצ'רים למשתני דמי.

הפיצ'ר הקטגוריאלי הנוסף 'C' הוא פיצ'ר חסוי (anonymous) שיעניין אותנו להבין כיצד מתנהג והאם יכול לעזור לנו ללמוד משהו על הלייבל שנצטרך לחזות בהמשך. בדקנו האם אחת הקטגוריות בו מאפיינת בעיקר סוג אחד של קבצים (זדוני או לא זדוני) אך נכחנו לגלות שכל קטגוריה מאפיינת כמות דיי דומה של קבצים זדוניים וקבצים שאינם זדוניים (נספח 1.8).

בדומה לפיצ'רים הקטגוריאליים, חשבנו שנכון יהיה לבדוק על הפיצ'רים הבוליאניים האם אחת משתי הקטגוריות בו (1/0) יכולות לאפיין קובץ זדוני או לא (נספח 1.9). בעזרת גרפים ראינו שאם הפיצ'רים 'has_debug' ו- 'has_signature' הם 1 (כלומר ישנו חלק של דיבאג בקובץ ו/או ישנה חתימה בקובץ), מרבית הקבצים לא זדוניים, דבר שיכול לעזור לנו רבות בסיווג. נדון בכל אלו בחלקים הבאים.

בנוסף, הסתכלות על פיצ'רים מספריים עזרה להבין שישנם פיצ'רים שכמות גדולה או קטנה שלהם יכולה לחזות את סוג הקובץ (זדוני/לא זדוני). לדוגמא: גודל קובץ, כמות הפונקציות המיובאות והמיוצאות וכמות הסמלים בקובץ (נספחים 1.10.1 – 1.10.4).

מבדיקה האם ממוצע גודל הקובץ, או ממוצע מספר הפונקציות המיובאות יכול לאפיין קובץ זדוני, התקשנו להסיק מסקנות ברורות (נספחים 1.10.4 ו-1.10.2). בניגוד לכך, נכחנו לגלות שממוצע הפונקציות המיוצאות גבוה במיוחד לקבצים לא זדוניים בהשוואה לקבצים זדוניים (נספח 1.10.3). כך גם עבור ממוצע כמות הסמלים (נספח 1.10.1). מסקנות חשובות שנביע דעתנו עליהן בהמשך.

בהמשך למסקנותינו כאן, ניצור בהמשך פיצ'ר חדש שיהיה פרופורציה בין פיצ'רים ויוכל גם הוא לעזור לנו בסיווג הקבצים.

הסתכלות על חריגים

נעשה בחלק זה הסתכלות נוספת על התפלגות הפיצ'רים המספריים השונים בעזרת בוקס-פלוטים. ראשית, ניתן לראות שלמרבית הפיצ'רים ישנם ערכים חריגים (נספח 1.11). בנוסף, בדומה למסקנות על התפלגות הפיצ'רים שביצענו מוקדם יותר, ניתן לראות שהפיצ'ר 'A' מתפלג נורמאלית. גם כאן, בדומה למקודם, נעשה טרנספורמציה של לוג על הנתונים כדי לנסות לראות טוב יותר את ההתפלגויות (נספח 1.12). ניתן לראות שלאחר הטרנספורמציה, גם הפיצ'רים 'numstrings', 'vsize', 'size' ו – 'printables' מתפלגים נורמאלית בגלל חלוקת ה-boxplot באופן כזה בו חציון הנתונים דיי במרכז והערכים החריגים בשני צידי התיבה באופן דיי דומה.

בשביל לחזק את השערותינו נעזרנו בחלק זה בשיטת QQ-Plot, כשיטה שלא נלמדה בקורס, בנוסף למדד ה VIF שצוין לפני כן. גרף הQQ מאפשר לבדוק אם התפלגות הערכים היא כמו התפלגות אחרת, ובמקרה שלנו, כמו התפלגות נורמאלית. אם נוצר קו ישר אלכסוני עולה משמאל לימין אז אפשר לקבוע כי אכן הפיצ'ר מתפלג נורמאלית (נספח 1.13). ובאמת, לאחר בדיקה עם QQ-Plot אפשר לסכם ולומר כי הפיצ'רים האלו מתפלגים נורמלית גם כן. לבסוף, נעזרנו בכמה ויזואליזציות שונות כדי לראות אם קיימים נתונים חריגים שאולי כדאי יהיה להוריד בהמשך (נספח 1.14). בחלק השני נתייחס לסוגי הפיצ'רים בנפרד ונטפל בהם בהתאם.

חלק שני – עיבוד מקדים

בחלק זה ניישם את המסקנות שהגענו אליהן בחלק הקודם. נתחיל בלחלק את הדאטה לאימון ולאימות (train and validation) כך ש-80 אחוז מהדאטה יהיה אימון ו 20 אחוז מהדאטה יהיה האימות.
נתחיל במילוי ערכים חסרים כדי שהדאטה יהיה מלא. חשבנו על מספר גישות למילוי ערכים חסרים: חציון, סיווג לפי רוב וקורלציה בין פיצ'רים.

הגישה הראשונה שנאמץ על הדאטה שלנו היא שיטת הקורלציה. בהמשך למסקנות בחלק הקודם, ראינו שישנם 3 פיצ'רים בעלי מתאם גבוה יחסית ביניהם והם 'MZ', 'numstrings', 'size'. בגישה זו, נמלא ערכים חסרים רק לשלושת פיצ'רים אלו. בהמשך לנאמר למעלה, קיימת היררכיה בין שלושת הפיצ'רים הללו לפי גודל המתאם ביניהם, ובדיוק לפי היררכיה זו נשלים את הערכים החסרים. (מתאם 0.9 בין 'size', 'numstrings') > (מתאם 0.7 בין 'size', 'MZ') > (מתאם 0.6 בין 'numstrings', 'MZ').
כלומר, במקרה בו size הוא תא ריק ניעזר ב numstrings שבאותה שורה לחישוב הערך היחסי שהיה אמור להיות ב size. במקרה בו גם numstrings חסר בשורה הנוכחית, ניעזר בקורלציה של size עם MZ להשלמת הערך החסר ב size. במידה ושלושת הערכים בשורה ריקים, ניעזר בגישה השנייה שנפרט מייד. אופן השלמת התאים החסרים בגישה זו יתבצע כך:
תא ריק = ערך מטריצת הקורלציות (בין 0-1, במקרה שלנו 0.9/0.7/0.6) * (ערך הפיצ'ר המתואם באותה שורה בהתאם להיררכיה שתיארתי).
בגישה השנייה למילוי ערכים חסרים ניעזר בחציון ובשיטת הרוב שציינו למעלה. ערכים מספריים חסרים שלא טופלו בגישה הראשונה יושלמו לפי חציון הנתונים באותו פיצ'ר. משתנים בוליאניים מזוהים בגישת העיבוד המקדים כערכים מספריים לכל דבר (0/1) ומסווגים בשיטת החציון גם כן. ערכים חסרים של משתנים קטגוריאליים יושלמו בשיטת הרוב, זאת אומרת, ערכים אלו יושלמו להיות הקטגוריה הנפוצה ביותר לאותו פיצ'ר. בסיום חלק זה לא נותרים ערכים חסרים בדאטה סט.

מייד לאחר מילוי הערכים החסרים, נתעסק ביצירת פיצ'ר חדש בשם 'proportion_imports' . **הפיצ'ר יהיה פרופורציה בין מספר הפונקציות המיובאות (imports) לבין כלל הפונקציות (imports + exports).** בהמשך למסקנות מהחלק הקודם, חשבנו לבדוק את התנהגות הפונקציות המיובאות והמיוצאות בקבצים כי יכולות לעזור לנו בסיווג הקובץ לזדוני או לא זדוני. פיצ'ר הפרופורציה הזה בעצם בודק את הישענות הקובץ על פונקציות חיצוניות (exports) שיכול במידה רבה להצביע על קובץ זדוני.
על מנת למנוע בעיות חישוביות, כאשר המכנה הוא 0 נגדיר אותו להיות 1 כדי להימנע מחילוק ב 0.

בחלק הבא בעיבוד המקדים, נתמודד עם הפיצ'רים הקטגוריאליים שניתחנו בחלק הקודם. הבעיה שהייתה לנו היא שהיו המון קטגוריות שונות בפיצ'ר 'file_type_trid'. בעיה זו גם יצרה בעיה עם קובץ הטסט, שמכיוון שישנן המון קטגוריות, קיימת סבירות שבטסט יהיו קטגוריות שלא הכרנו לפני כן באימון.
כדי להתמודד עם בעיות אלו, החלטנו שכל הקטגוריות שמופיעות בפחות מ-1 אחוז מהתצפיות תשויכנה להיות בקטגוריה אחת שנקראת 'other'. בזאת נצמצם במידה רבה את כמות הקטגוריות, ונקל משמעותית על מימדי המודל.
בשלב הבא, נשתמש בשיטת ה- One-Hot-Encoding שבאמצעותה נפרוס את הקטגוריות השונות של הפיצ'רים למשתני דמי עם ערכים 0/1. לשיטה זו כמה יתרונות, ראשית, הפיכת פיצ'רים קטגוריאליים לפיצ'רים מספריים מאפשר לאלגוריתמי למידת המכונה לגשת לכל האינפורמציה בפיצ'רים הללו. בנוסף, זה שומר על מובהקות ויחס שווה בין כל הקטגוריות.
לאחר פרישת הפיצ'רים נשאר עם הרבה פחות קטגוריות, כך ש- 2883 תצפיות יסווגו להיות קטגוריה 'other'.
סה"כ, נותרנו עם 50 פיצ'רים כך שכולם בעלי ערכים מספריים למעט מזהה הקובץ 'sha256'. בזאת נשלים את ההתמודדות עם פיצ'רים קטגוריאליים.
מניתוח הפיצ'רים שבוצע קודם לכן, גילינו שישנם פיצ'רים שמתפלגים נורמאלית וחלק שלא. על הפיצ'רים הנורמליים נעשה טרנספורמציית לוג שתמזער את השפעת התצפיות החריגות. ניעזר בשיטת ה – IQR לזיהוי החריגים הנותרים ונעדכן אותם להיות הערך הגבולי הקיצוני האפשרי הקרוב ביותר לערך החריג של אותו הפיצ'ר.
על הפיצ'רים שאינם מתפלגים נורמאלית, ניישם את אלגוריתם ה – Isolation Forest שנלמד בהרצאה 12. אלגוריתם זה מתאמן ולומד את הגבולות של כל פיצ'ר על סט האימון ועל סמך האימון מסוגל לתת ציון לכל ערך בדאטה סט על היותו חריג או לא. אם ציון ה-Isolation Forest

מצביע על תצפית שמכילה ערכים חריגים – נסיר אותה מהדאטה סט. על מנת למנוע overfitting בכל האפשר, העדפנו לסווג רק חלק קטן של ערכים כחריגים, ולכן בחרנו בהיפר-פרמטר contamination להיות 0.01.
ניעזר בגרפים מתאימים כדי לראות את השינויים השונים (נספחים 2.1 – 2.2).

לאחר שטיפלנו בנתונים החריגים ניסינו להבין האם בעיית המימדיות גדולה מידי. לפני שעונים על השאלה הזו נבין איך בכלל אומדים את בעיית המימדיות.
קיימות מספר שיטות: מדד feature-to-sample אשר מחשב את היחס בין כמות פיצ'רים לכמות השורות. בנוסף, בדיקת אחוז הערכים האפסיים\קרובים לאפס שקיימים בפיצ'רים שאולי מלמדים על מימדים מיותרים שניתן להוריד. עוד, ניתן לבדוק את כמות התרומה של כל פיצ'ר, כך לדוגמה פיצ'רים עם קורלציה גבוהה לא תורמים למודל ואולי כדאי להסירם. נוסף לאלו, זמן ריצה גדול\ הקצאת משאבים גדולה להרצה יכולים להעיד על בעיית מימדיות. ולבסוף, תרומת הפיצ'רים למודל.
לאחר בחינת כל אחת מהדרכים הגענו למסקנה גורפת כי אכן בעיית המימדיות לא מובהקת כאן: כמות הפיצ'רים ביחס לכמות הדגימות קטנה מאוד, רק שני פיצ'רים הם בעלי אחוז גבוה של ערכים אפסיים, ובהמשך נראה גם שהמודלים איכותיים ותרומת מרבית הפיצ'רים משמעותית.
לפני שנעבור להקטנת המימדיות נשאל – למה מימדיות גדולה היא בעיה?
אחת הסיבות העיקריות שהוצגו לעיל היא בעיית זמן הריצה של המודלים. כמות גדולה של פיצ'רים גורמת לזמן ריצה רב. בנוסף, לעיתים, כמות גדולה של פיצ'רים הופכת את הנתונים להיות דלילים - דבר המקשה על מציאת דפוסים משמעותיים ועלול ליצור בעיית התאמת יתר (Overfitting).

רגע לפני הורדת המימדיות, שאלנו את עצמינו האם המידע מנורמל. אף פיצ'ר אינו מנורמל. חשוב היה לנו לנרמל את הדאטה מכמה סיבות: השוואה – כך ניתן להשוות בין הערכים השונים בפיצ'רים, ולבטל את החריגות שאולי קיימות עדיין בדאטה. מניעת הטיה – אם הפיצ'רים בסקאלה שונה, עלולה להיווצר הטיה בנתונים. פרשנות – נרמול מאפשר לנו ולמודלים השונים להבין את היחסים בין הפיצ'רים בצורה טובה יותר.
הנרמול היחידי שנפעיל יהיה standard scaler וזאת על מנת לשמור על עקביות בנתונים ולא ליצור הטיה רק בחלקם.

כדי למנוע את בעיית המימדיות פעלנו בשתי גישות:
הורדת פיצ'רים בצורה ידנית: לאורך הפרויקט הגענו לשלוש מסקנות חשובות על הפיצ'רים שלנו: אחוז הערכים האפסיים בפיצ'רים symbols ו- registry הוא מעל 0.9%. בנוסף, קיים קשר גדול בין symbols לממוצע הקבצים הלא זדוניים. עוד, יש מתאם גדול בין size, numstrings ו- MZ. לאור המסקנות, החלטנו להשמיט את הפיצ'רים registry וMZ.
גישה נוספת היא ה-PCA. גישה זו תעזור לנו להקטין את המימדיות של הבעיה עוד יותר, ולהשיג 99% משונות הנתונים (נספח 2.3).
למרות יתרונותיה של גישת ה- PCA, החלטנו שלא להשתמש בה במודל הסופי וזאת כיוון שלמרות יתרונה הגדול, אנחנו מאבדים את משמעות הפיצ'רים בפרויקט.
רגע לפני הרצת המודלים יישמנו את כל ההחלטות הסופיות שקיבלנו במהלך האקספלורציה והעיבוד המקדים לפונקציית Preprocess, אשר מאפשרת להריץ את כל הפעולות בבת אחת.

חלק שלישי – הרצת מודלים

המודלים הראשוניים שבחרנו הם KNN ורגרסיה לוגיסטית.

- **KNN** – Test AUC – 0.955: (נספח 3.1)
  - n_neighbors = 15 -עבור מודל זה בחרנו כמות גדולה יחסית של שכנים על מנת לאפשר למודל להיות עמיד ולתפוס דפוסים כלליים ולוקאליים בדאטה במורכבות בינונית.
  - weights = distance – בחרנו בdistance כיוון שהנתונים מנורמלים ובמצב זה הוא יעבוד טוב יותר מ-uniform.
  - power - השתמשנו ב Grid Search CV על מנת למצוא את ההיפר פרמטר הטוב ביותר מבין 1 / 2.
- **Logistic Regression** – Test AUC – 0.874: (נספח 3.2) (+ נספח 10 – היפר פרמטרים)
  - C = 10 - כיוון שהדאטה לא מורכב מדי אפשר להקטין את הרגולריזציה כך ש - 10 הוא מדד שמציע איזון בין שונות להטיה.
  - penalty - נתנו לGrid Search לבחור בין שתי טכניקות הרגולריזציה – L1\L2.
  - solver - אפשרנו לGS לבחור בין שני האלגוריתמים – Liblinear\lbfgs.
  - max_iter = 100 - כיוון שנרצה לתת זמן למודל להתכנס בחרנו במקסימום איטרציות – 100.

המודלים המורכבים שבחרנו הם ANN ו- Random Forest:

- **ANN** – Test AUC – 0.957: (נספח 3.3)
  hidden_layer_sizes = 64 - העדפנו להגדיל את היכולת ללמוד דפוסים מורכבים.
  activation - נתנו לGS לבחור בין שתי הפונקציות העדיפות – logistic\relu.
  solver = adam - בחרנו בadam כיוון שזהו אלגוריתם טוב לדאטה סטים בינוניים עד גדולים.

- **Random Forest** – Test AUC – 0.977: (נספח 3.4) כאן בחרנו שלא להשתמש בGS עקב זמן הריצה הגדול שמתווסף עם בחירת היפר פרמטרים נוספים ולכן:
  n_estimators = 300 - העדפנו לשפר את המודל ככל האפשר וכיוון שכל תת-עץ מתאמן על דאטה סט אחר בחרנו ב300 עצי החלטה.
  max_depth = None - על מנת לאפשר למודל לתפוס קשרים מורכבים יותר העדפנו לא להגביל את עומק העץ (Default).
  min_samples_split = 10 - מאותן סיבות שציינו לפני כן בחרנו בחלוקה גדולה יחסית – 10.
  min_samples_leaf = 4 - אותן סיבות גם כאן – 4.

לאחר הרצת המודלים הגענו למסקנה כי Random Forest הוא המודל המועדף יותר מכמה סיבות:
ציון ה AUC שלו היה הגבוה ביותר וללא overfitting. בנוסף, מודל זה ידוע ביכולות הטובות שלו ללמוד נתונים ולחזות נכון.

לאחר בחירת המודל, המשכנו במציאת הפיצ'רים שתורמים הכי הרבה למודל הנבחר. לשם כך השתמשנו בFeature Importance של המודל וראינו כי הפיצ'רים המשפיעים ביותר הם (נספח 3.5):

1. 10.65% – Avlength - ממוצע אורך המילים בקובץ יכול לתת תובנות טובות על מורכבות או מבנה הקבצים. קבצים זדוניים יכולים להכיל דפוסים מורכבים שיכולים להיתפס על ידי פיצ'ר זה.
2. 10.50% – B - למרות שלא הצלחנו להבין את מהות הפיצ'ר הזה, ההחלטה להשאיר אותו נראית מבטיחה.
3. 9.56% – imports - כמות הפונקציות המיובאות לקובץ יכולות להעיד על התלות שלו בקבצים או נתיבים חיצוניים שאולי יכולים להיות זדוניים. קבצים זדוניים יכולים להכיל דפוסים ששונים יחסית מאלו של קבצים שאינם זדוניים ולכן הפיצ'ר הזה יכול לתרום רבות.
4. 7.53% – Urls - בדומה לפונקציות המיובאות, נוכחות קישורים למקורות חיצוניים יכולה להעיד על קשר לקבצים זדוניים.
5. 6.9% – File_type_prob_trid – הסתברות סוג הקובץ מאפשר לקבל הרבה מידע על טבע הקובץ. סוגים מסוימים של קבצים יכולים להיות כבר מסווגים כמסוכנים.

חלק רביעי – הערכת המודלים

Confusion matrix – מהסתכלות על הטבלה ניתן לראות כי המודל מצליח לסווג בצורה טובה קבצים זדוניים (נספח 4.1). נוכל לראות זאת בערכים הגבוהים שקיימים ב-TN ו-TP. עם זאת, ישנם עדיין סיווגים מוטעים של המודל, אם כי לא באופן מובהק. מבחינת דיוק (Precision) קיבלנו 0.93, כלומר 93% מהדגימות שסווגו כזדוניות אכן היו זדוניות. בנוסף, עבור חולשת המודל, (Recall/ Sensitivity) המודל זיהה 91% מהחריגות האפשריות. כלומר 9% מהקבצים הזדוניים לא זוהו. (ניתן לקרוא בהרחבה על כל המסקנות מהמודל במחברת) .
לאחר מכן, עברנו להעריך את המודלים שלנו על ידי KFold Cross Validation ובניית פלט ROC לכל מודל עם 5 ההרצות השונות שלו (נספחים 4.2 – 4.6). ניתן להסיק כי כל המודלים פועלים בצורה טובה יחסית, וכי המודל Random Forest הוא אכן המודל העדיף כאשר מודדים לפי ציון הAUC.
ניתן לראות כי המודל הוא בעל הבדל קטן בין AUC Train לבין AUC Test (0.03), וזה מעיד על כך שהמודל אינו Overfitted. בנוסף, הפערים בין תוצאות הtrain ל-validation בין המודלים לבין עצמם גם הם אינם מובהקים, דרך נוספת לדעת כי אין למודלים overfitting.
על מנת להגדיל את ההכללה של המודל, אנו יכולים לממש את כל השלבים שבוצעו בשלב העיבוד המקדים: נורמליזציה, הסרת חריגים והקטנת המימדיות, וזה מה שביצענו.
כמו כן, עם ההרצות השונות של המודל ראינו שהפער בין תוצאות הtrain לtest גדלות, וזאת כיוון שטיפלנו בסט האימון יתר על המידה, ולא בצורה שתואמת את סט האימות. לכן, ראינו לנכון לנסות ולמזער פערים כאלו על ידי צמצום הסרת החריגים מסט האימון למשל.

חלק חמישי – ביצוע פרדיקציה

בחלק זה איגדנו את כל הפעולות הסופיות לפונקציית Pipeline אחת אשר מאפשרת את שחזור תוצאות המודל הנבחר בצורה מהירה ונקייה.

לאחר אימון המודל, הזנו את ההסתברויות של כל קובץ לייבל (זדוני/לא זדוני) על ידי המודל בסט הטסט (test.csv).

חלוקת עבודה

אנחנו אחים תאומים ולכן את כל העבודה עשינו ביחד פנים מול פנים.

יובל בקירוב היה אחראי על חלק האקספלורציה, חקר הנתונים ויישומם באופן כללי.

איתן בקירוב היה אחראי על בחירת מודלים והרצתם, התאמת היפר-פרמטריים מיטביים ומסגור שלד הקוד.

בחלק האקספלורציה והעיבוד המקדים יובל היה אחראי על חקירת הנתונים, השלמת נתונים חסרים, התמודדות עם פיצ'רים קטגוריאליים והוספת פיצ'רים חדשים. איתן היה אחראי על נתונים חריגים, נרמול, בעיית הממדיות וההתמודדות איתה. KFold ו- Confusion Matrix עשינו יחדיו.

הרצת הבדיקות וההתאמות הרלוונטיות בוצעו גם כן יחדיו בסיעור מוחות.

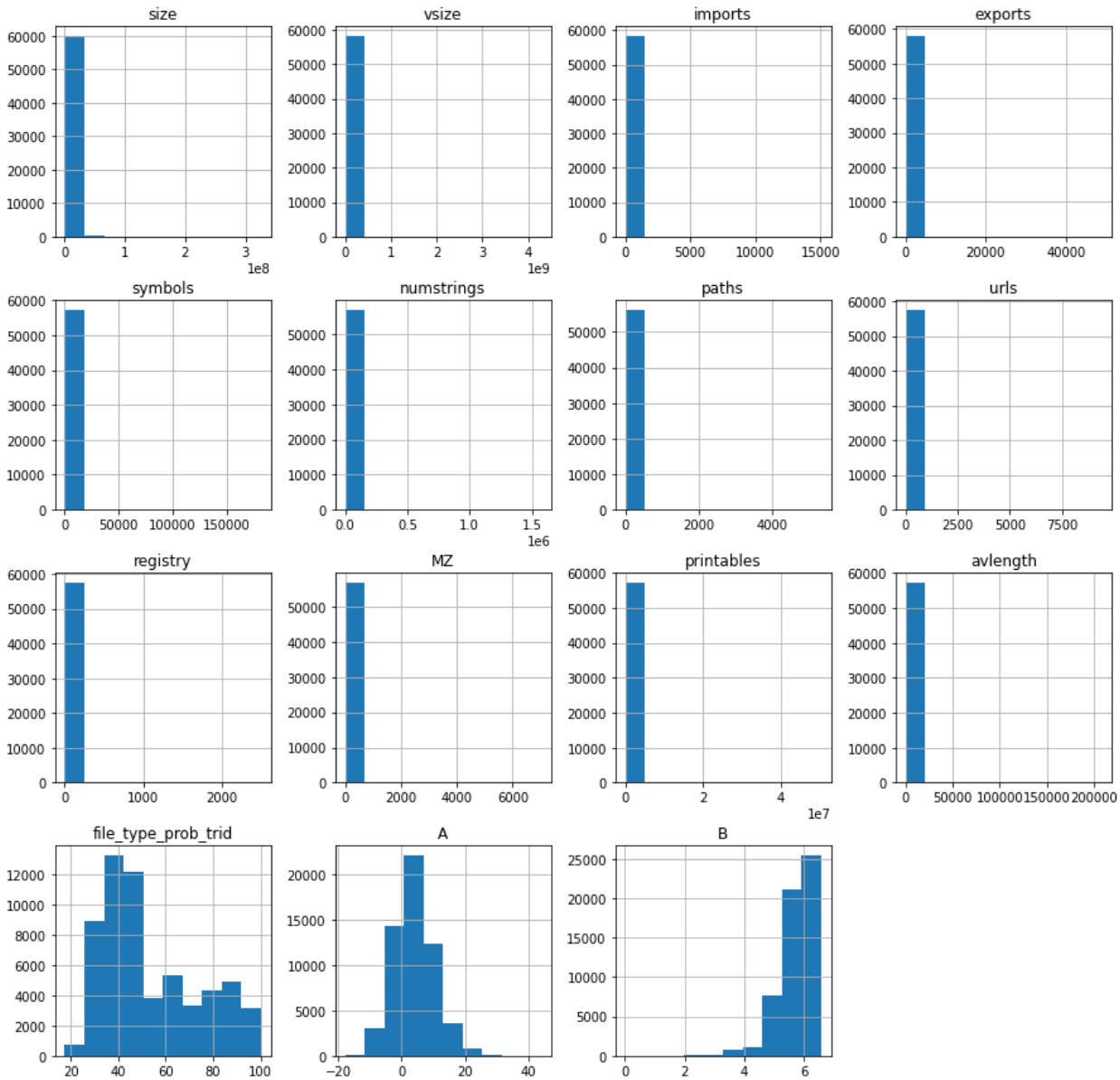
דו"ח מסכם נכתב יחד.

נספחים

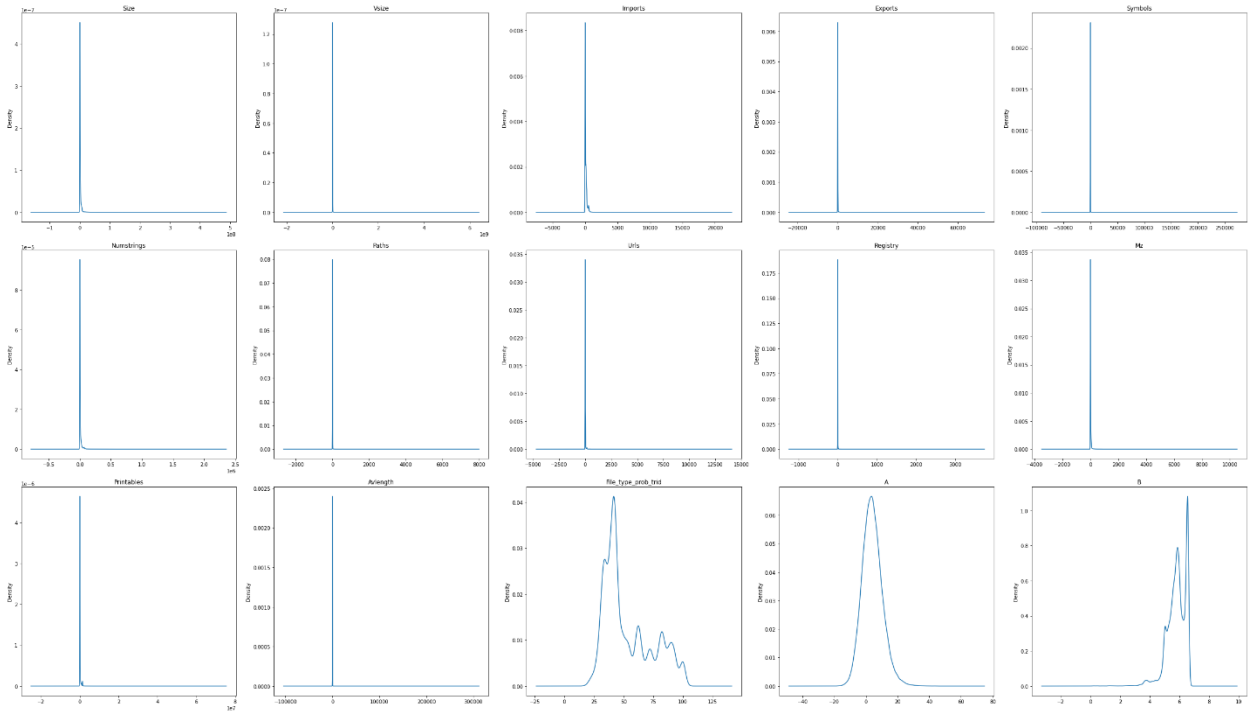
היפר פרמטרים שניתנו לGrid Search ברגרסיה לוגיסטית: (נספח 10)

```
# param_grid = {
#     'C': [10.0],  # Inverse of regularization strength
#     'penalty': ['l1', 'l2'],  # Regularization penalty (L1 or L2)
#     'solver': ['liblinear', 'lbfgs'],  # Solver algorithm
#     'max_iter': [100],  # Maximum number of iterations
#     'n_jobs': [-1]
# }
```

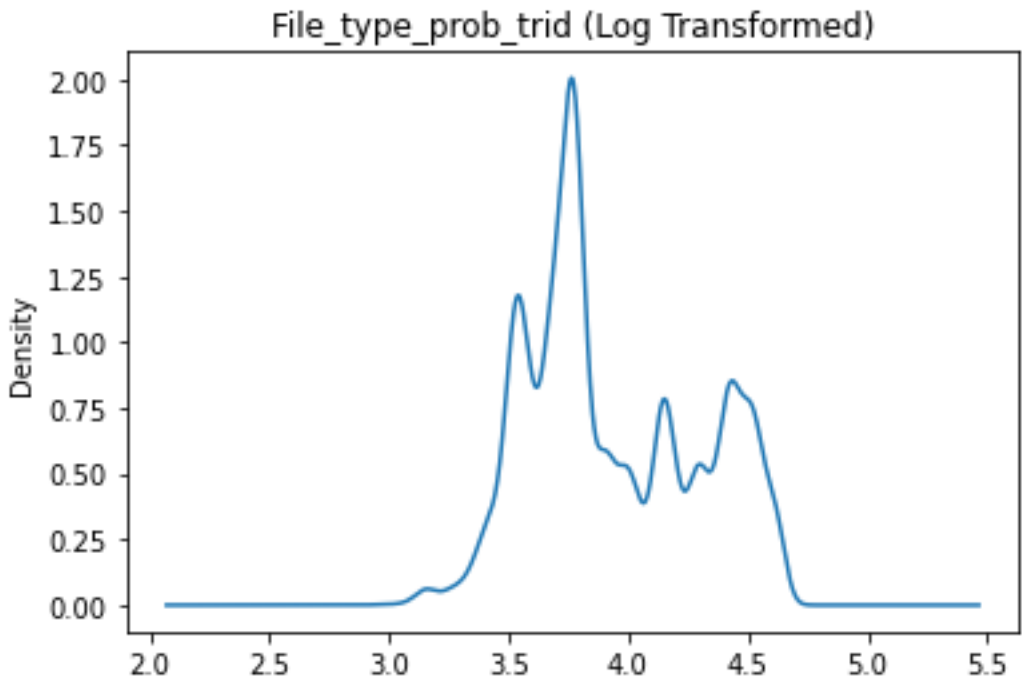
נספח 1.1

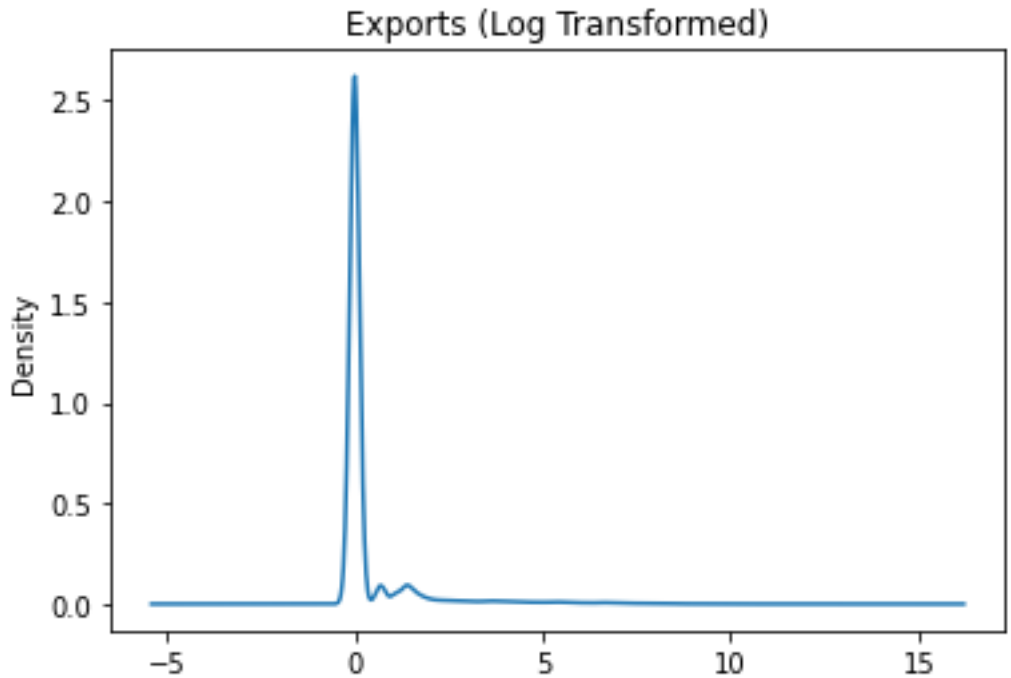


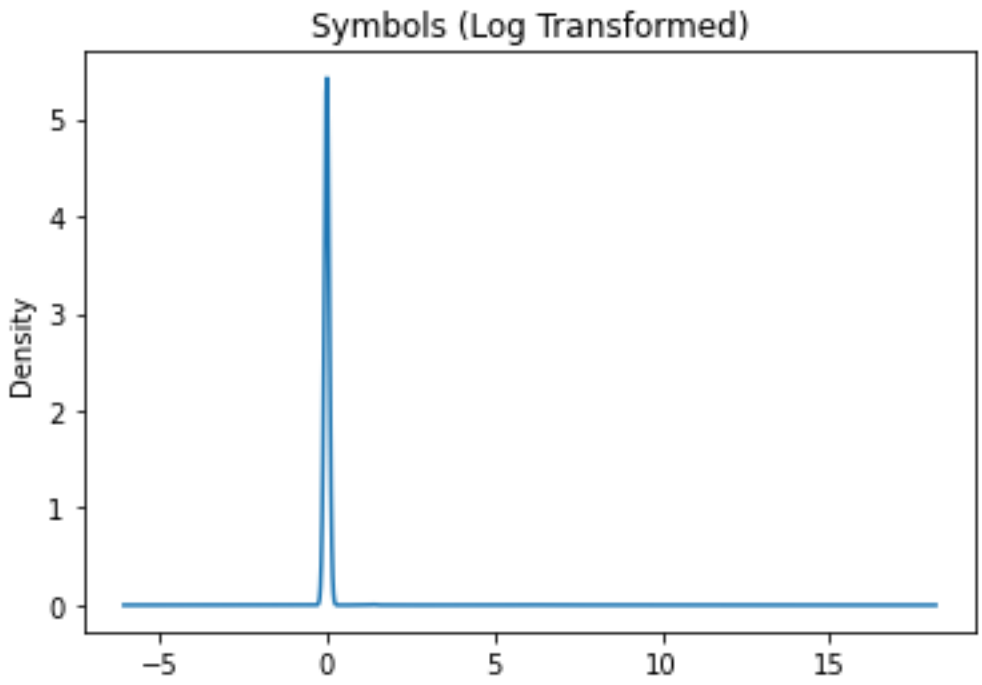
נספח 1.2

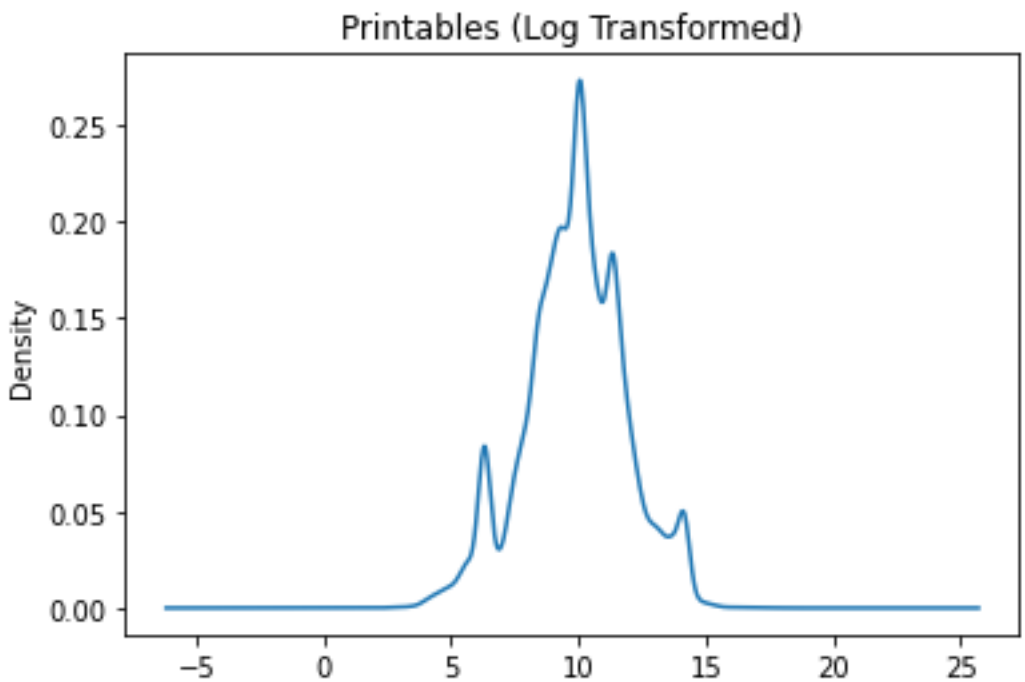


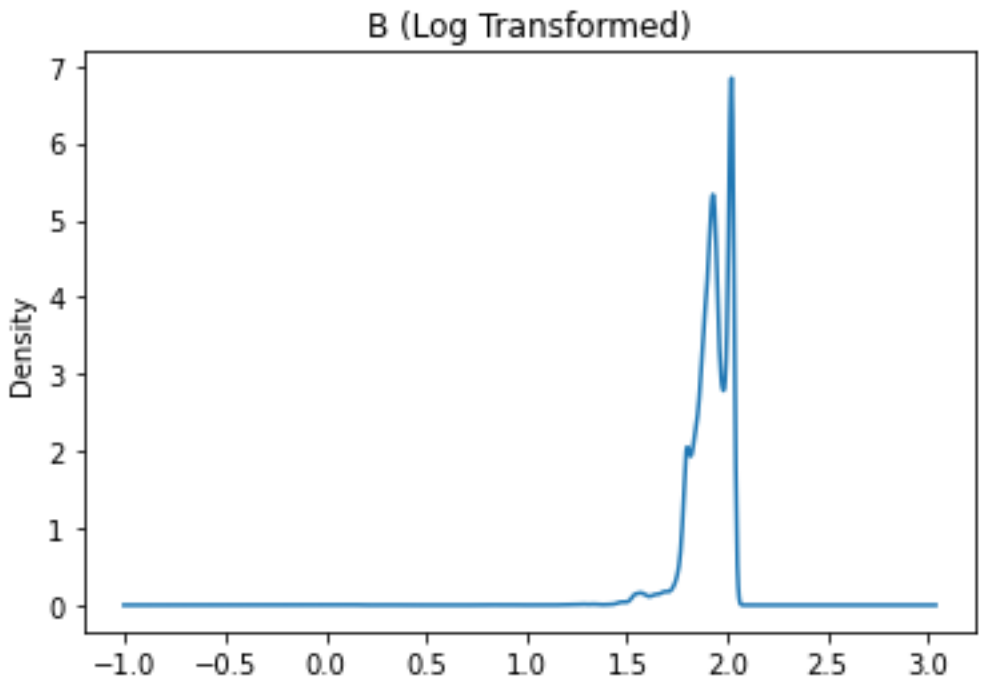
נספחים 1.2.1 – 1.2.15

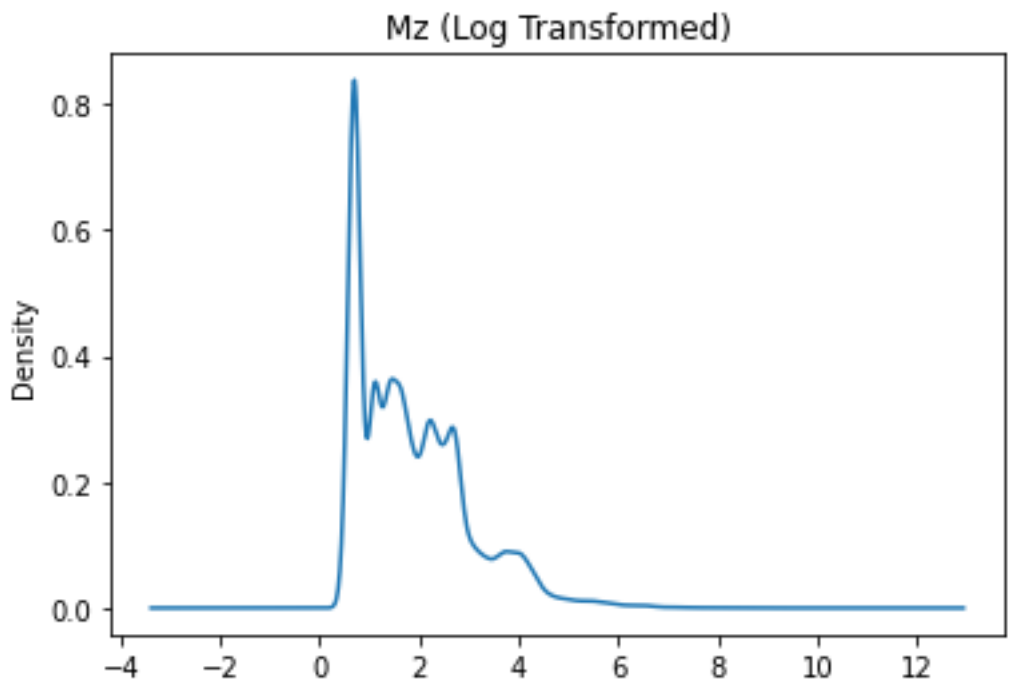

File_type_prob_trid (Log Transformed)
Density
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5

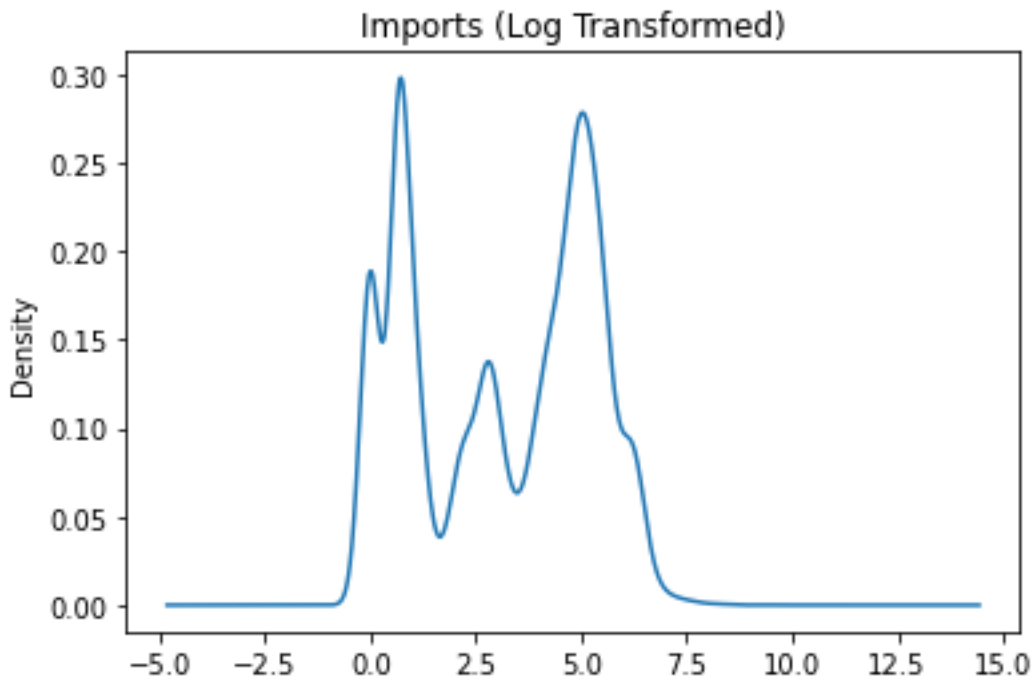

Exports (Log Transformed)
Density
2.5
2.0
1.5
1.0
0.5
0.0
−5
0
5
10
15

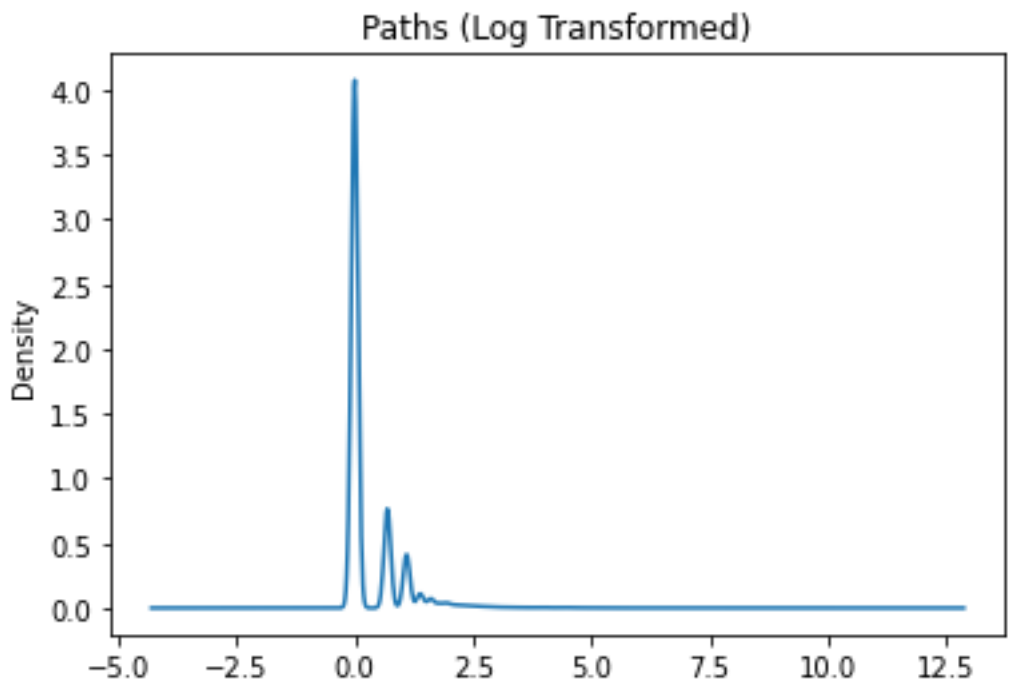

Symbols (Log Transformed)
Density
0
1
2
3
4
5
−5
0
5
10
15

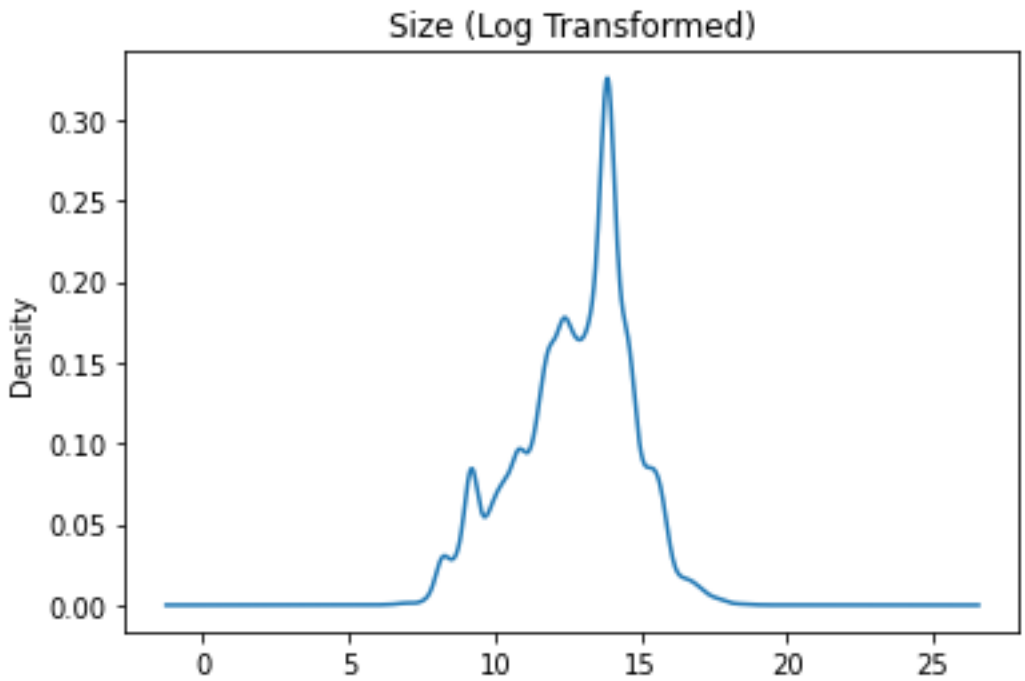

Printables (Log Transformed)
Density
0.00
0.05
0.10
0.15
0.20
0.25
−5
0
5
10
15
20
25

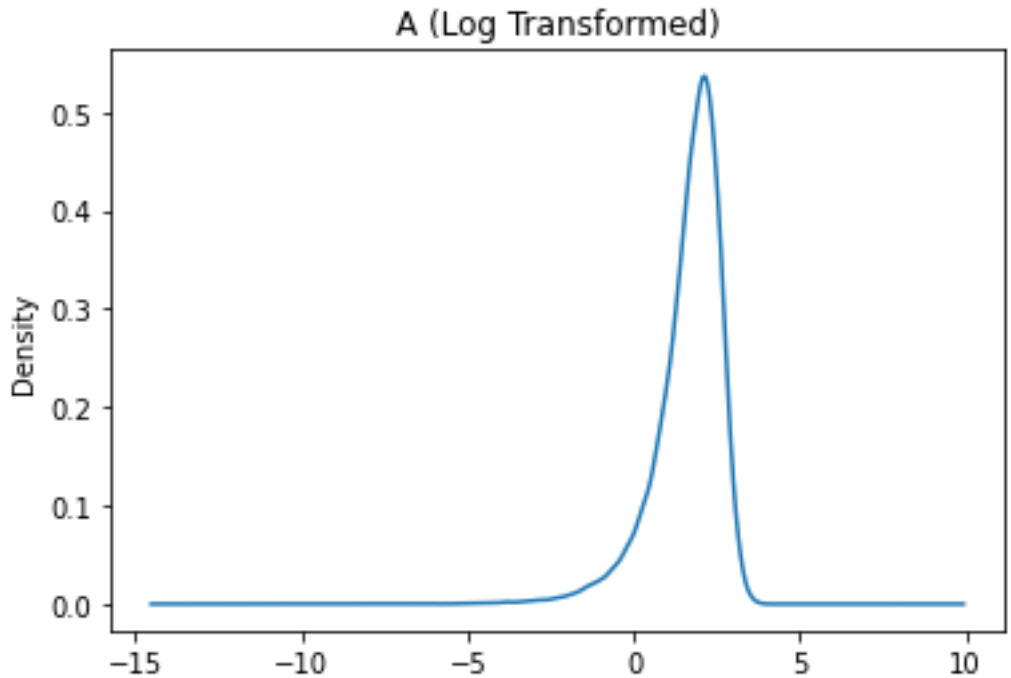

B (Log Transformed)
Density
7
6
5
4
3
2
1
0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0

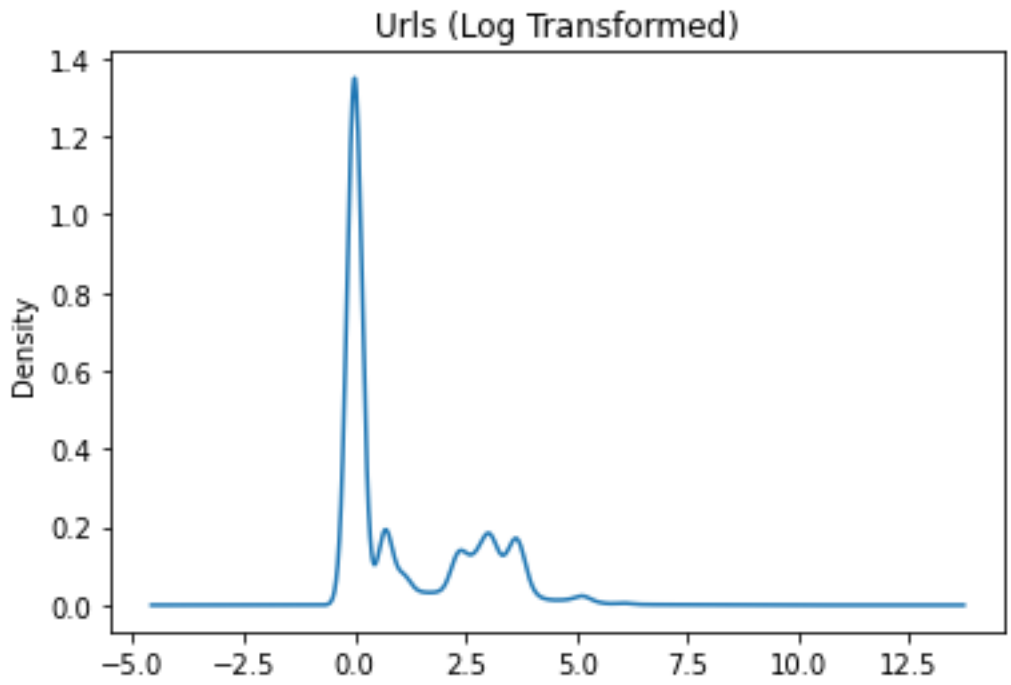

Mz (Log Transformed)
Density
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4
6
8
10
12

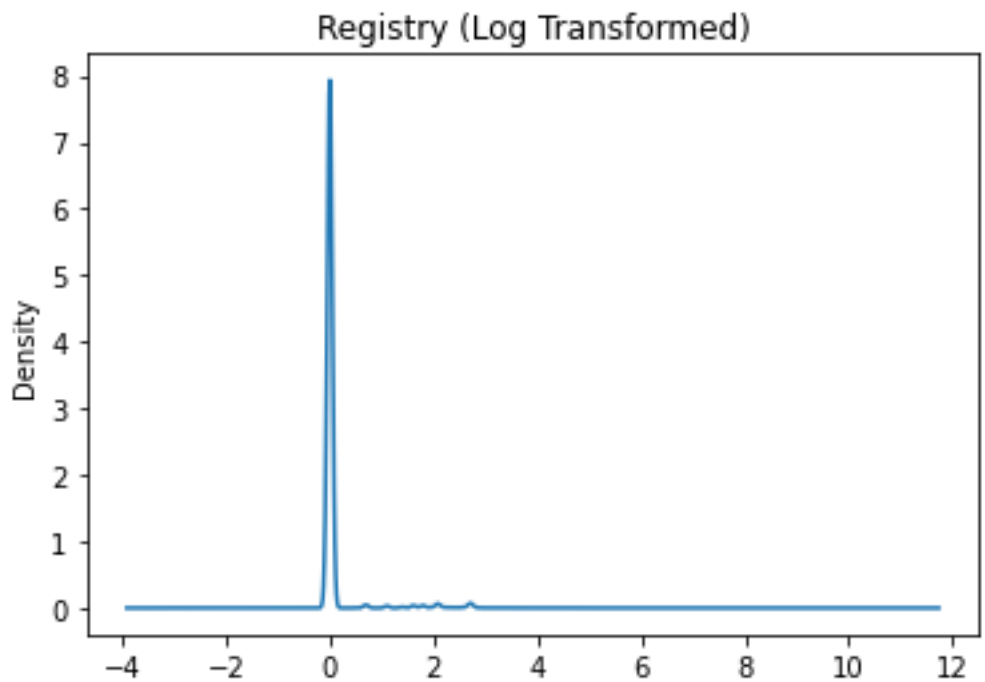

Imports (Log Transformed)
Density
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0

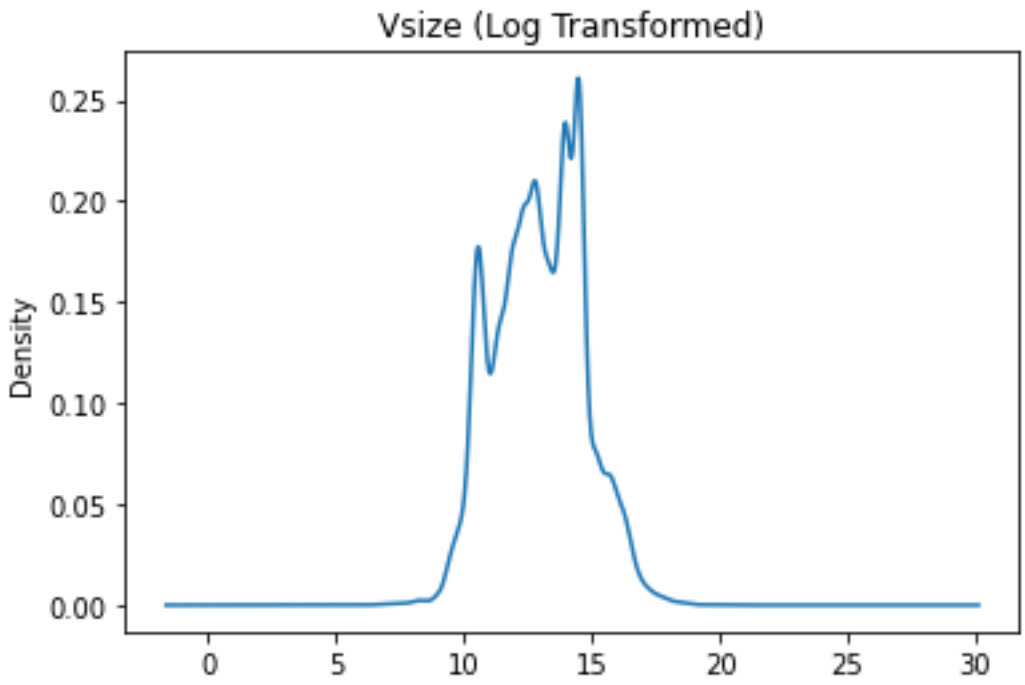

Paths (Log Transformed)
Density
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5

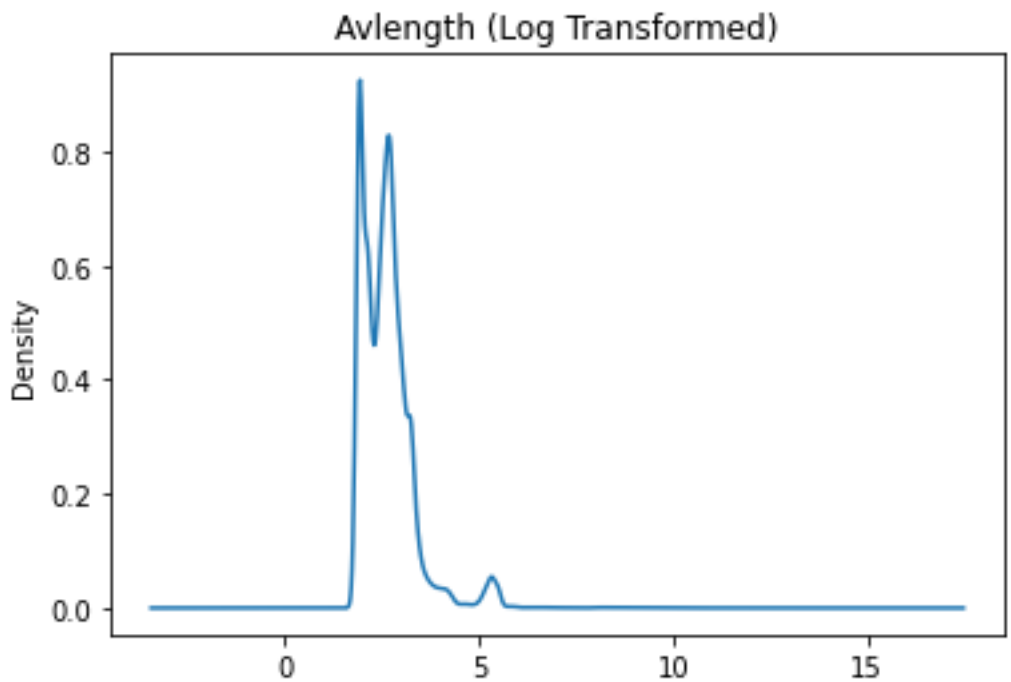

Size (Log Transformed)
Density
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
25

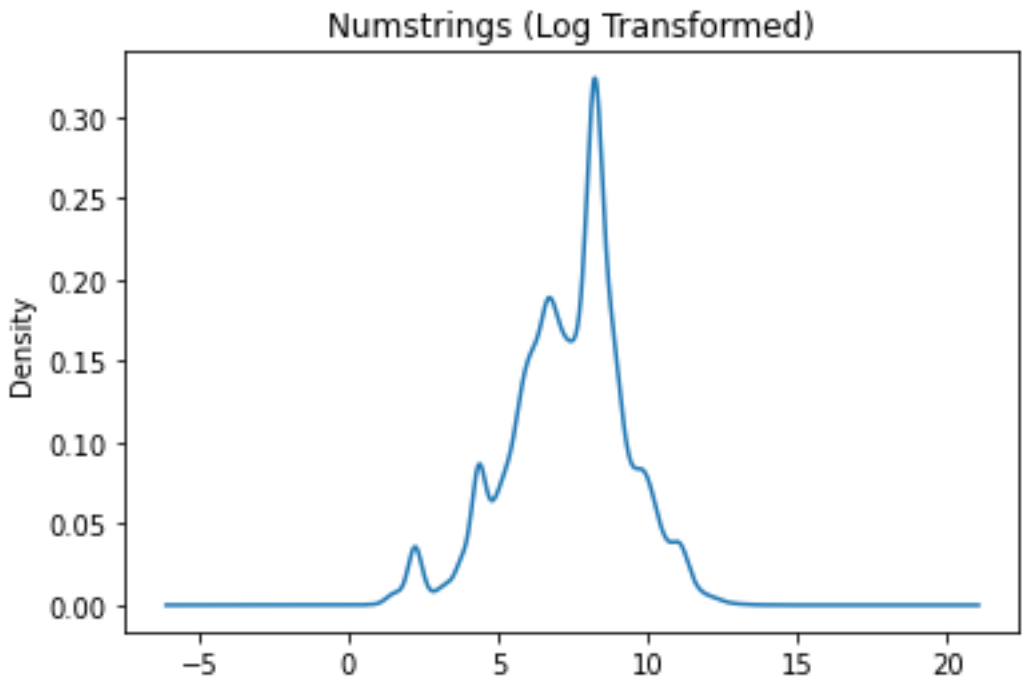

A (Log Transformed)
Density
0.5
0.4
0.3
0.2
0.1
0.0
−15
−10
−5
0
5
10


Urls (Log Transformed)
Density
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5

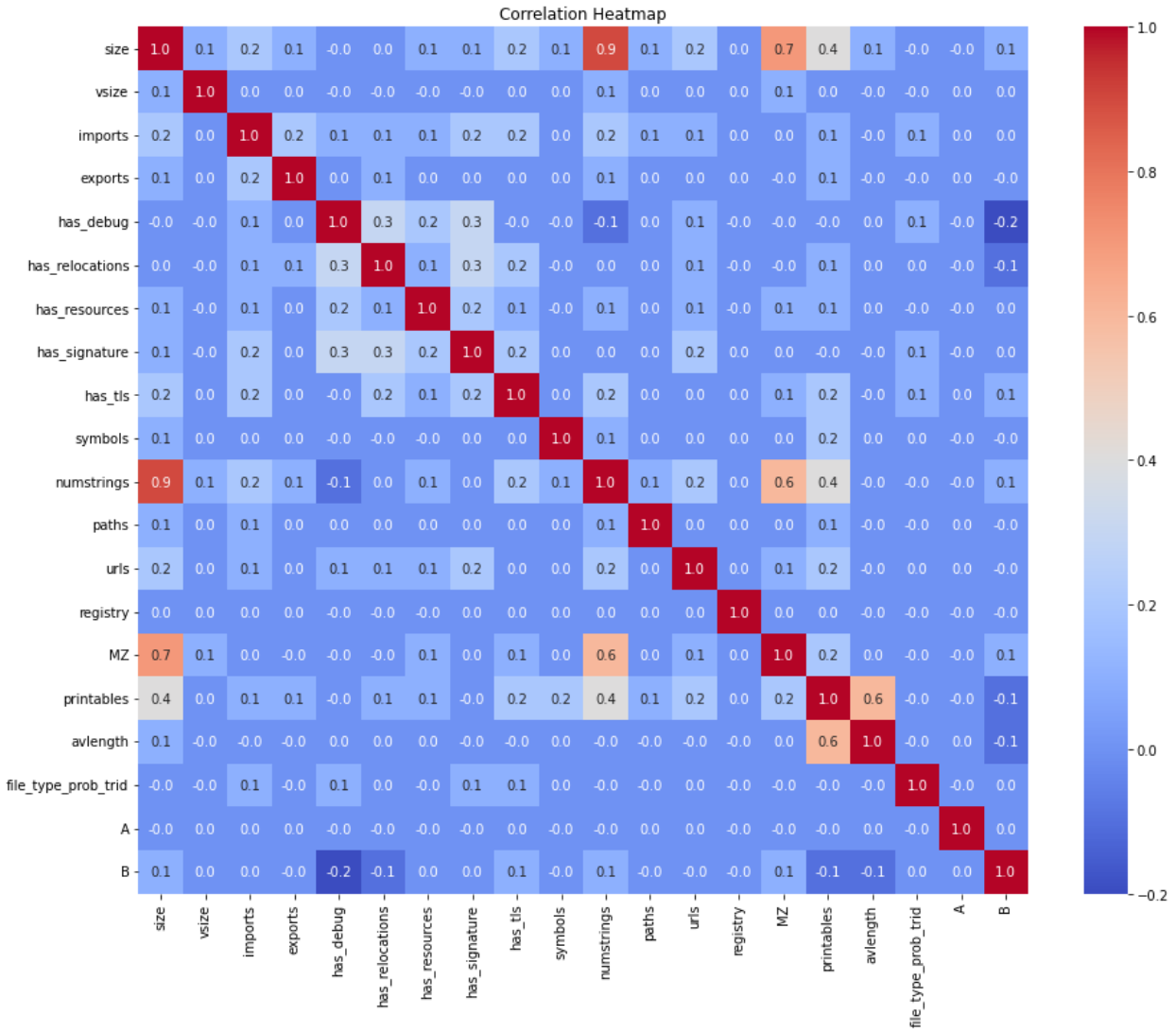

Registry (Log Transformed)
Density
8
7
6
5
4
3
2
1
0
−4
−2
0
2
4
6
8
10
12


Vsize (Log Transformed)
Density
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
25
30

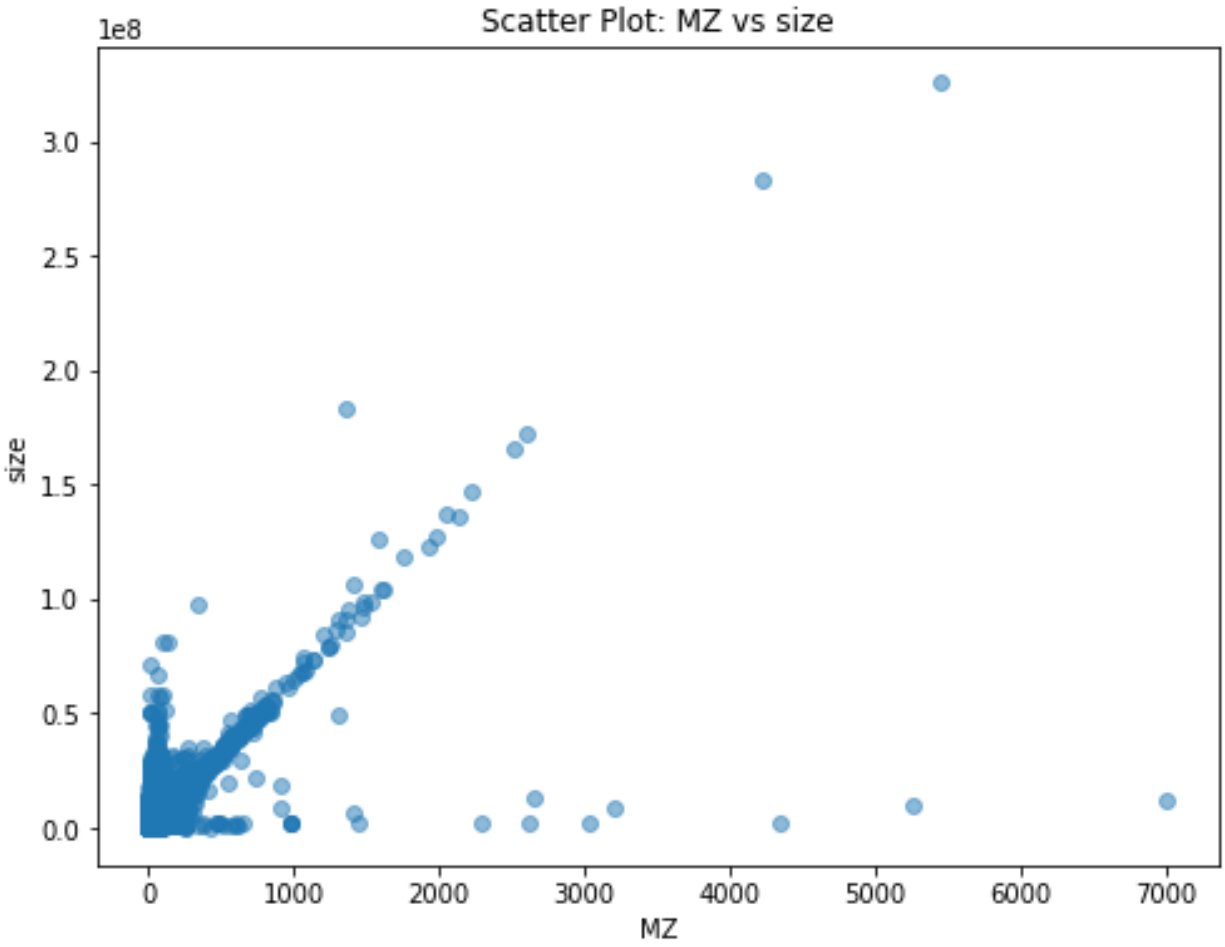

Avlength (Log Transformed)
Density
0.8
0.6
0.4
0.2
0.0
0
5
10
15

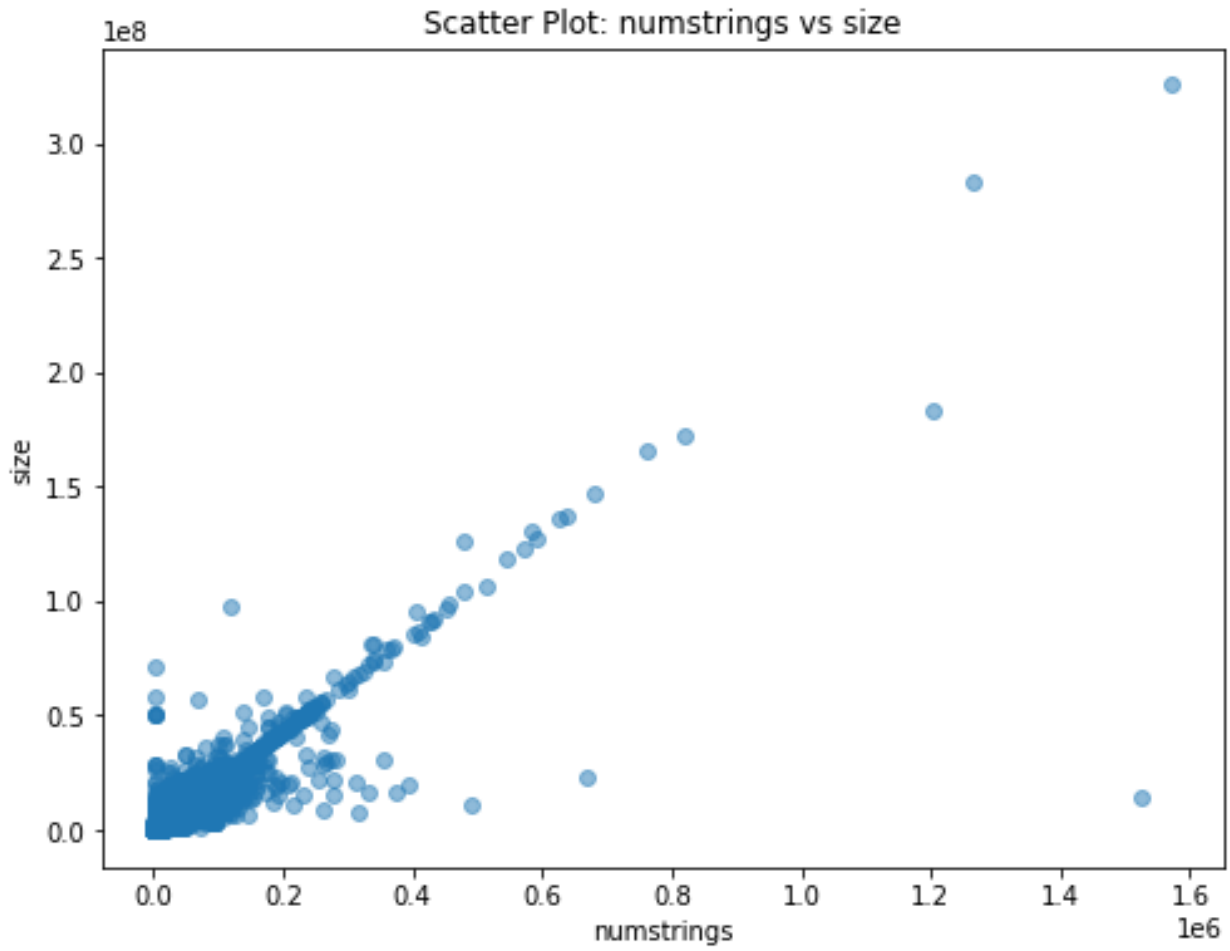

Numstrings (Log Transformed)
Density
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−5
0
5
10
15
20

נספח 1.3

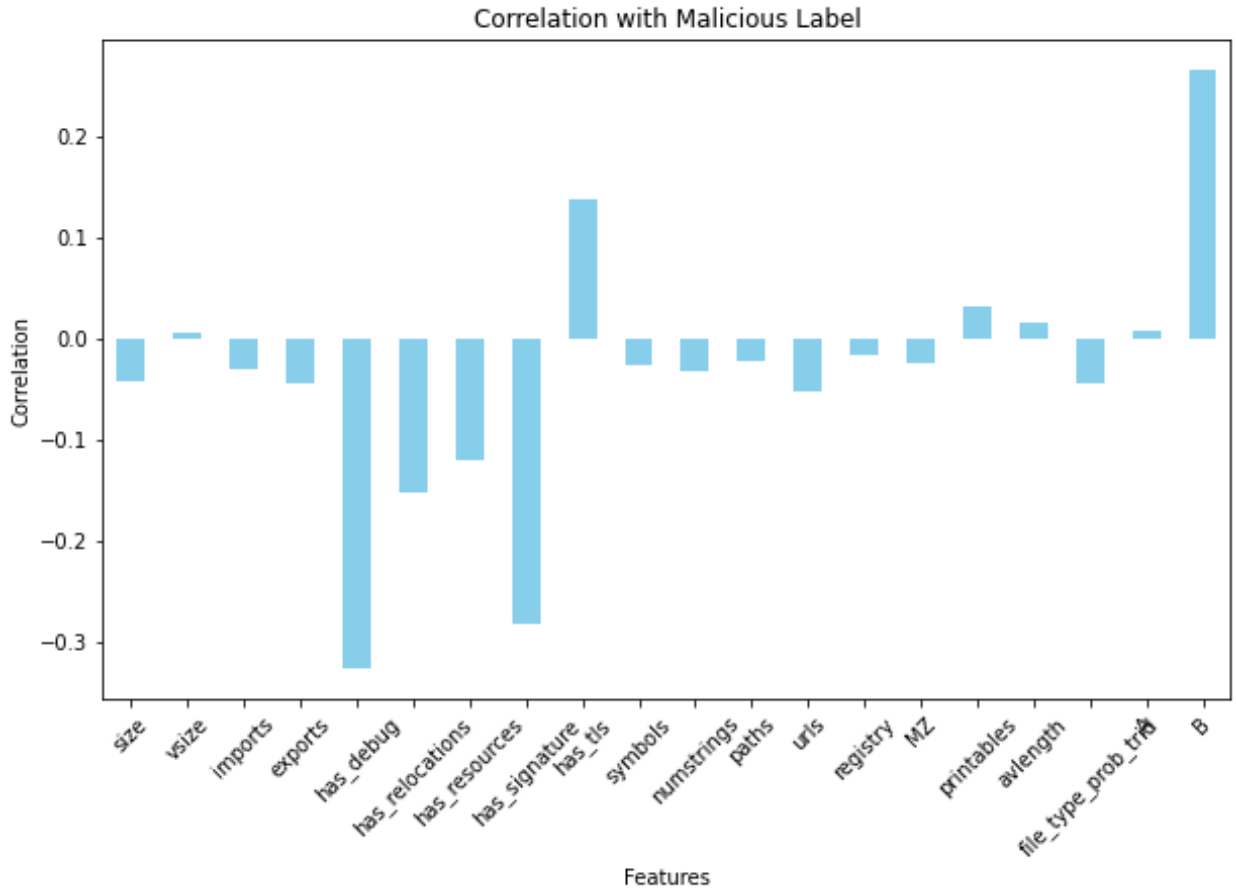

Correlation Heatmap
size 1.0 0.1 0.2 0.1 -0.0 0.0 0.1 0.1 0.2 0.1 0.9 0.1 0.2 0.0 0.7 0.4 0.1 -0.0 -0.0 0.1
vsize 0.1 1.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0 0.1 0.0 0.0 0.0 0.1 0.0 -0.0 -0.0 0.0 0.0
imports 0.2 0.0 1.0 0.2 0.1 0.1 0.1 0.2 0.2 0.0 0.2 0.1 0.1 0.0 0.0 0.1 -0.0 0.1 0.0 0.0
exports 0.1 0.0 0.2 1.0 0.0 0.1 0.0 0.0 0.0 0.0 0.1 0.0 0.0 0.0 -0.0 0.1 -0.0 -0.0 0.0 -0.0
has_debug -0.0 -0.0 0.1 0.0 1.0 0.3 0.2 0.3 -0.0 -0.0 -0.1 0.0 0.1 -0.0 -0.0 -0.0 0.0 0.1 -0.0 -0.2
has_relocations 0.0 -0.0 0.1 0.1 0.3 1.0 0.1 0.3 0.2 -0.0 0.0 0.0 0.1 -0.0 -0.0 0.1 0.0 0.0 -0.0 -0.1
has_resources 0.1 -0.0 0.1 0.0 0.2 0.1 1.0 0.2 0.1 -0.0 0.1 0.0 0.1 -0.0 0.1 0.1 0.0 -0.0 -0.0 0.0
has_signature 0.1 -0.0 0.2 0.0 0.3 0.3 0.2 1.0 0.2 0.0 0.0 0.0 0.2 0.0 0.0 -0.0 -0.0 0.1 -0.0 0.0
has_tls 0.2 0.0 0.2 0.0 -0.0 0.2 0.1 0.2 1.0 0.0 0.2 0.0 0.0 0.0 0.1 0.2 -0.0 0.1 0.0 0.1
symbols 0.1 0.0 0.0 0.0 -0.0 -0.0 -0.0 0.0 0.0 1.0 0.1 0.0 0.0 0.0 0.0 0.2 0.0 0.0 -0.0 -0.0
numstrings 0.9 0.1 0.2 0.1 -0.1 0.0 0.1 0.0 0.2 0.1 1.0 0.1 0.2 0.0 0.6 0.4 -0.0 -0.0 -0.0 0.1
paths 0.1 0.0 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.1 1.0 0.0 0.0 0.0 0.1 -0.0 -0.0 0.0 -0.0
urls 0.2 0.0 0.1 0.0 0.1 0.1 0.1 0.2 0.0 0.0 0.2 0.0 1.0 0.0 0.1 0.2 -0.0 0.0 0.0 -0.0
registry 0.0 0.0 0.0 0.0 -0.0 -0.0 -0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0
MZ 0.7 0.1 0.0 -0.0 -0.0 -0.0 0.1 0.0 0.1 0.0 0.6 0.0 0.1 0.0 1.0 0.2 0.0 -0.0 -0.0 0.1
printables 0.4 0.0 0.1 0.1 -0.0 0.1 0.1 -0.0 0.2 0.2 0.4 0.1 0.2 0.0 0.2 1.0 0.6 -0.0 -0.0 -0.1
avlength 0.1 -0.0 -0.0 -0.0 0.0 0.0 0.0 -0.0 -0.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 0.6 1.0 -0.0 0.0 -0.1
file_type_prob_trid -0.0 -0.0 0.1 -0.0 0.1 0.0 -0.0 0.1 0.1 0.0 -0.0 -0.0 0.0 -0.0 -0.0 -0.0 -0.0 1.0 -0.0 0.0
A -0.0 0.0 0.0 0.0 -0.0 -0.0 -0.0 -0.0 0.0 -0.0 -0.0 0.0 0.0 -0.0 -0.0 -0.0 0.0 -0.0 1.0 0.0
B 0.1 0.0 0.0 -0.0 -0.2 -0.1 0.0 0.0 0.1 -0.0 0.1 -0.0 -0.0 -0.0 0.1 -0.1 -0.1 0.0 0.0 1.0
size vsize imports exports has_debug has_relocations has_resources has_signature has_tls symbols numstrings paths urls registry MZ printables avlength file_type_prob_trid A B
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

נספחים 1.4.1 – 1.4.2

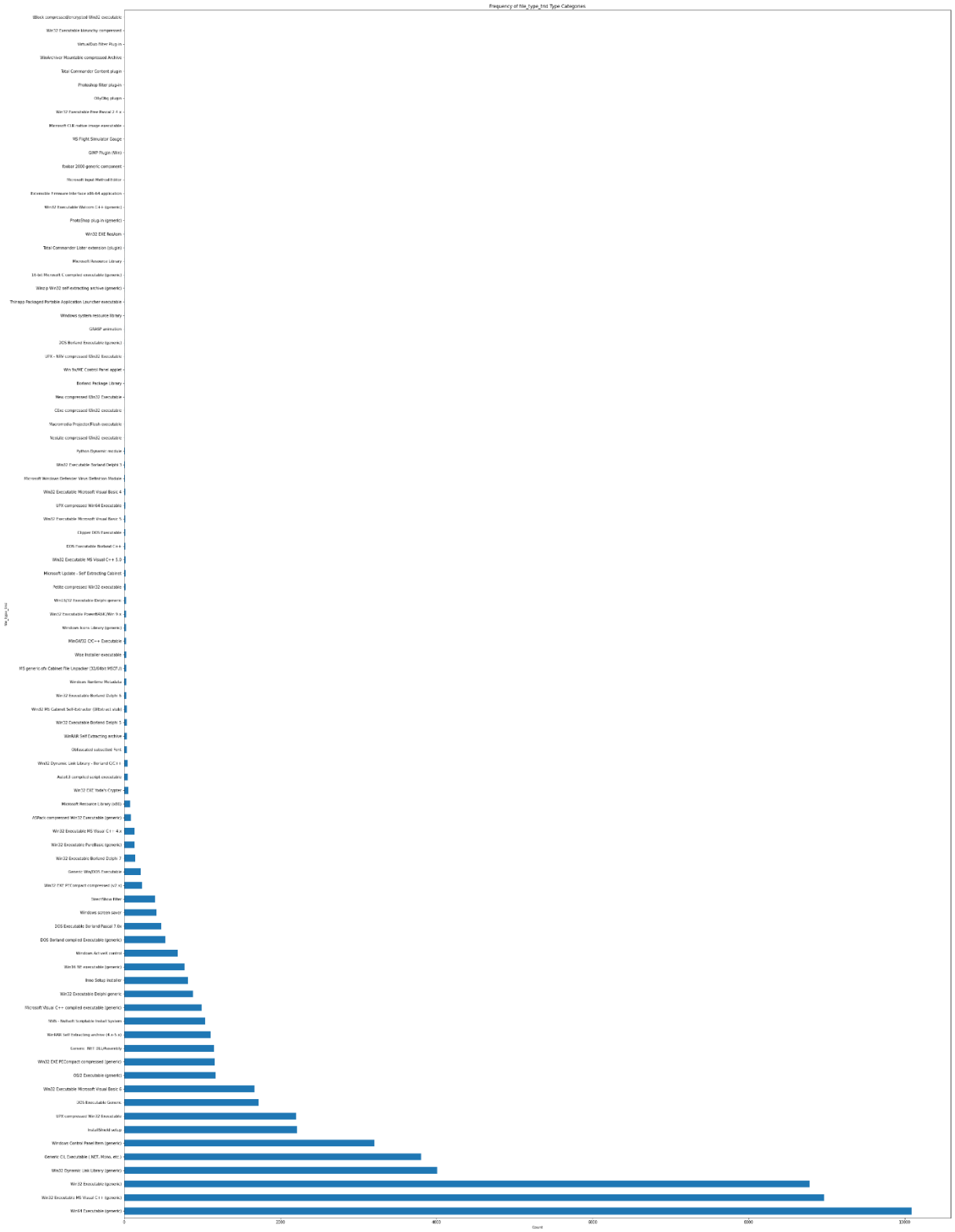



Scatter Plot: numstrings vs size
1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
size
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1e6
numstrings

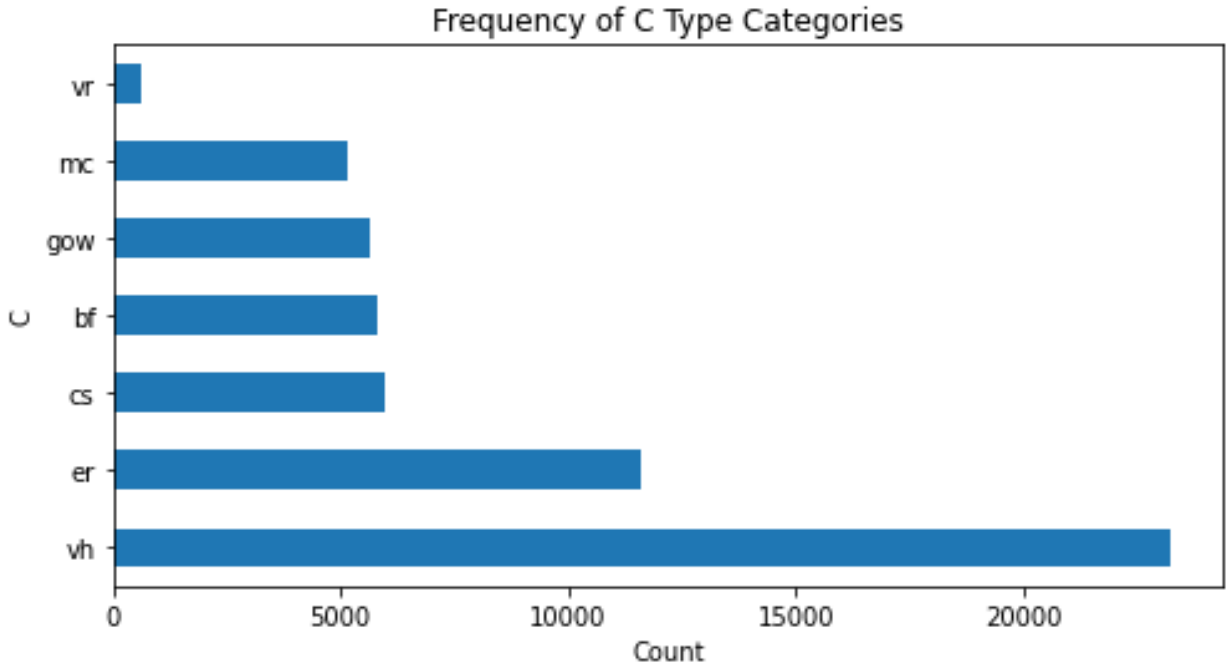
נספח 1.5


Correlation with Malicious Label
Correlation
0.2
0.1
0.0
−0.1
−0.2
−0.3
size
vsize
imports
exports
has_debug
has_relocations
has_resources
has_signature
has_tls
symbols
numstrings
paths
urls
registry
MZ
printables
avlength
file_type_prob_trid
B
Features

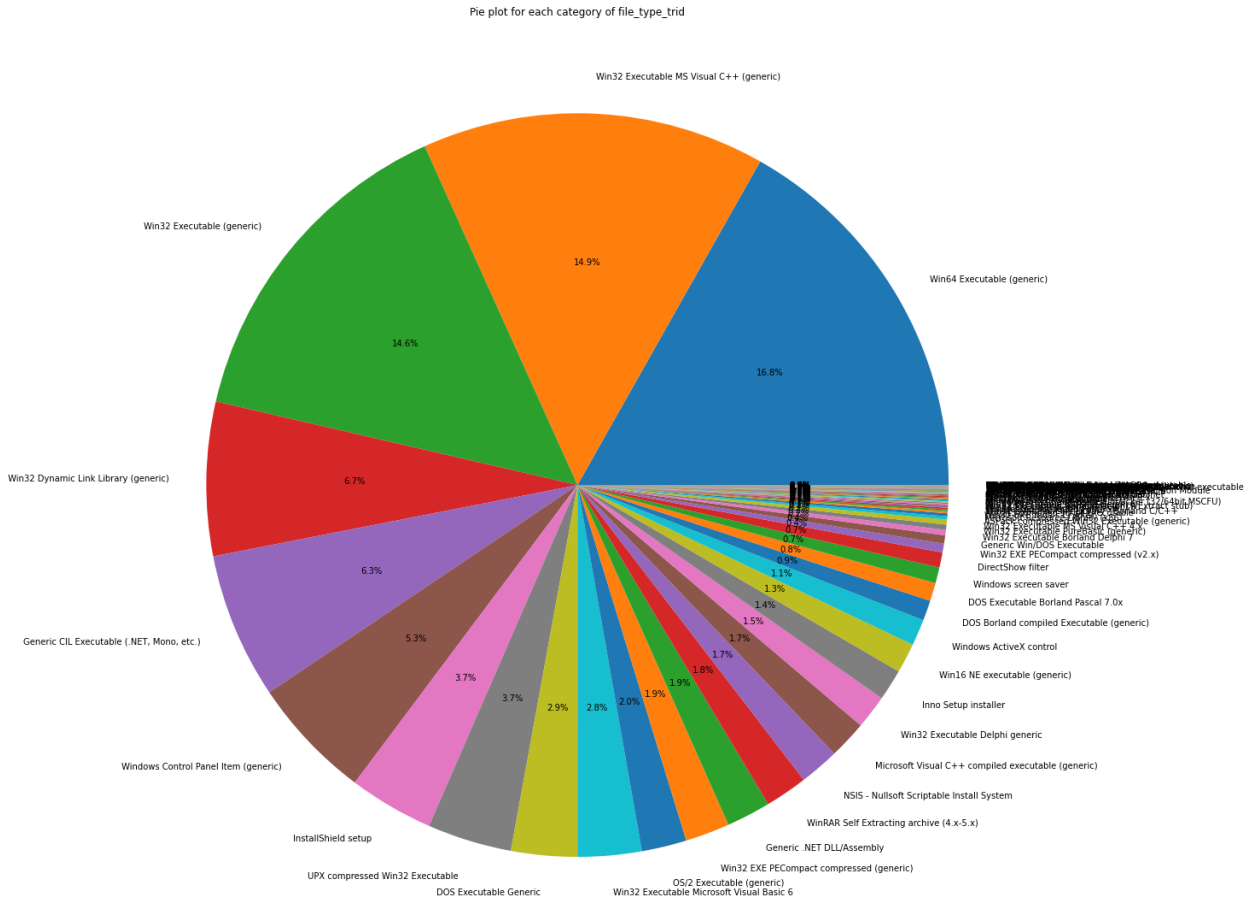
נספח 1.6.1


Frequency of file_type_trid Type Categories
EBlock compressed/encrypted Win32 executable
Win32 Executable Kinetochy compressed
VirtualDub filter Plug-in
WinArchiver Mountable compressed Archive
Total Commander Content plugin
Photoshop filter plugin
CityMap plugin
Win32 Executable Free Pascal 2.4.x
Microsoft CLR native image executable
MS Flight Simulator Gauge
GIMP Plug-in (Win)
Bobcat 2000 generic component
Microsoft Input Method Editor
Extensible Firmware Interface x86-64 application
Win32 Executable Watcom C++ (generic)
PhotoShop plug-in (generic)
Win32 EXE ReSalvo
Total Commander Lister extension (plugin)
Microsoft Resource Library
16-bit Microsoft C compiled executable (generic)
Wise® Win32 self-extracting archive (generic)
Thinapp Packaged Portable Application Launcher executable
Windows system resource library
GRASP animation
DOS Borland Executable (generic)
UPX - NRV compressed Win32 Executable
Win 9x/NT Control Panel applet
Borland Package Library
Neo compressed Elks32 Executable
Cdec compressed Win32 executable
Macromedia Projector/Flash executable
Tcscomp compressed Win32 executable
Python Dynamic module
Win32 Executable Borland Delphi 7
Microsoft Windows Defender Virus Definition Module
Win32 Executable Microsoft Visual Basic 4
UPX compressed Win64 Executable
Win32 Executable Microsoft Visual Basic 5
Clipper DOS Executable
EDS Executable Borland C++
Win32 Executable MS Visual C++ 5.0
Microsoft Update - Self Extracting Cabinet
Petite compressed Win32 executable
Win32/32 Executable Delphi generic
Win32 Executable PowerBASIC/Win 9.x
Windows Icons Library (generic)
MacOS/32 C/C++ Executable
Wise Installer executable
MS generic cfb Cabinet File Unpacker (32/64bit MSCF.dll)
Windows Runtime Metadata
Win32 Executable Borland Delphi 6
Win32 MS Cabinet Self-Extractor (Wfdextract.exe)
Win32 Executable Borland Delphi 5
WinRAR Self Extracting archive
Obfuscated subscribed Perl
Win32 Dynamic Link Library - Borland C/C++
AutoIt3 compiled script executable
Win32 EXE Yoda's Crypter
Microsoft Resource Library (x86)
ASPack compressed Win32 Executable (generic)
Win32 Executable MS Visual C++ 4.x
Win32 Executable PureBasic (generic)
Win32 Executable Borland Delphi 7
Generic Win/DOS Executable
Win32 EXE PECompact compressed (v2.x)
TrueMotion filter
Windows screen saver
DOS Executable Borland Pascal 7.0x
DOS Borland compiled Executable (generic)
Windows ActiveX control
Win16 NE executable (generic)
Inno Setup Installer
Win32 Executable Delphi generic
Microsoft Visual C++ compiled executable (generic)
NSIS - Nullsoft Scriptable Install System
WinRAR Self Extracting archive (4.x-5.x)
Generic .NET CIL/Assembly
Win32 EXE PECompact compressed (generic)
OS/2 Executable (generic)
Win32 Executable Microsoft Visual Basic 6
DOS Executable Generic
UPX compressed Win32 Executable
InstallShield setup
Windows Control Panel Item (generic)
Generic CIL Executable (.NET, Mono, etc.)
Win32 Dynamic Link Library (generic)
Win32 Executable (generic)
Win32 Executable MS Visual C++ (generic)
Win64 Executable (generic)
File_type_trid
0
2000
4000
6000
8000
10000
Count

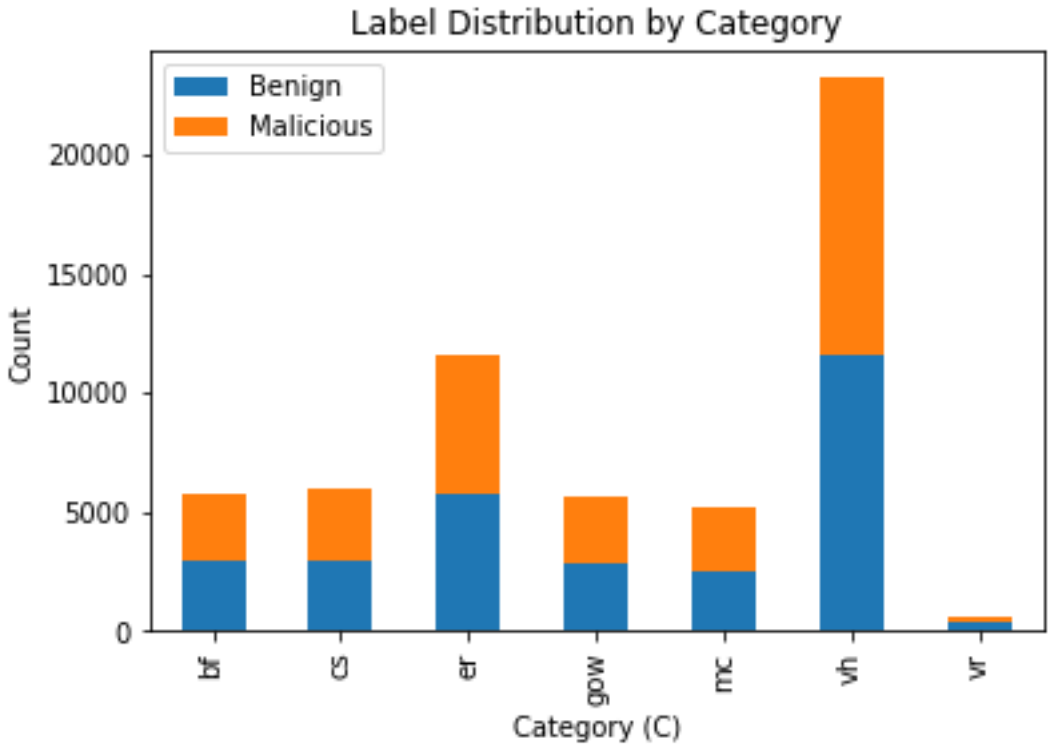
נספח 1.6.2



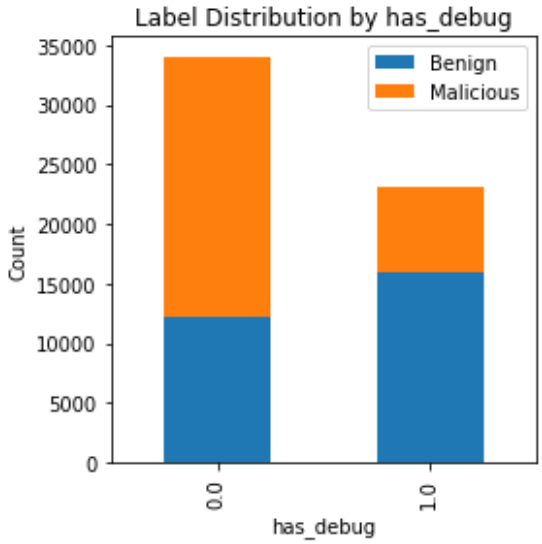
נספח 1.7

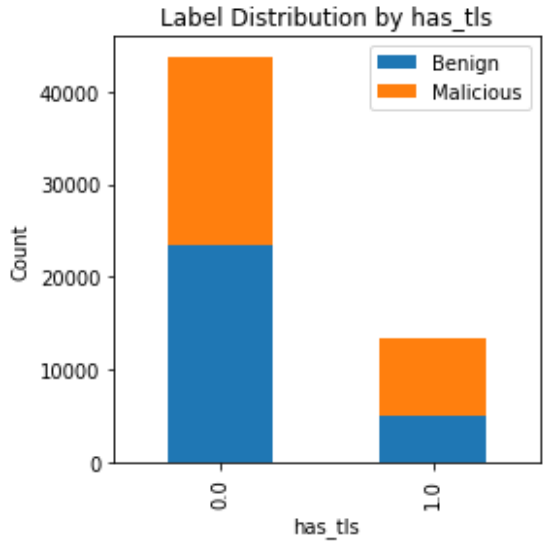


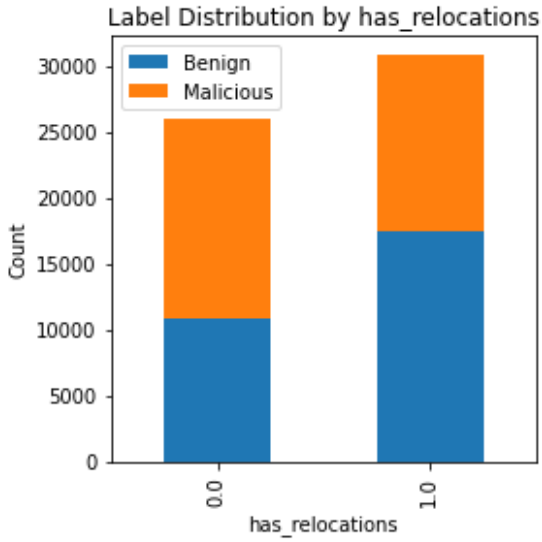
נספח 1.8

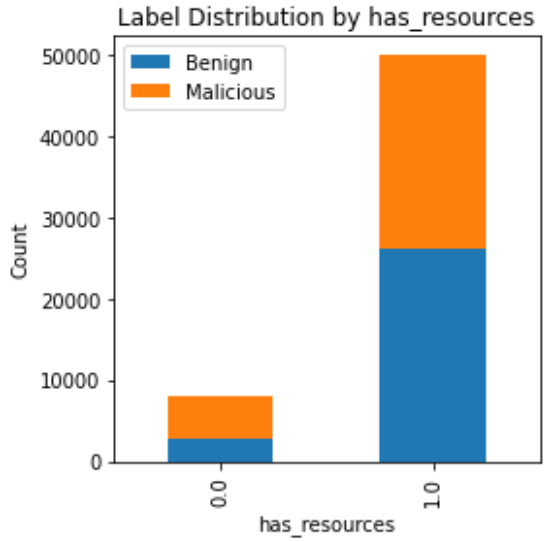

Label Distribution by Category
Benign
Malicious
Count
0
5000
10000
15000
20000
bf
cs
er
gow
mc
vh
vr
Category (C)

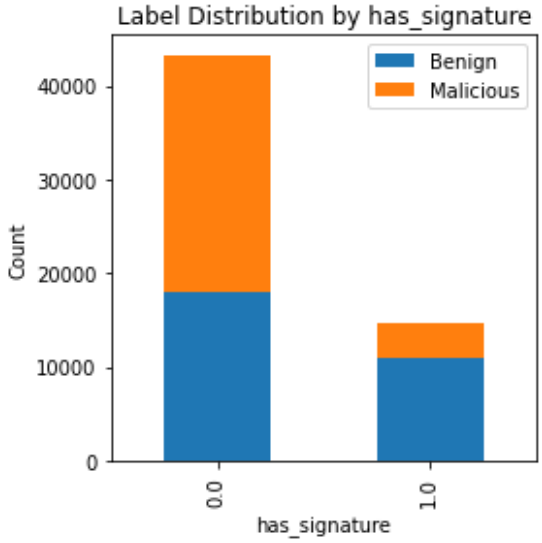
נספח 1.9


Label Distribution by has_debug
Benign
Malicious
Count
has_debug
0.0
1.0

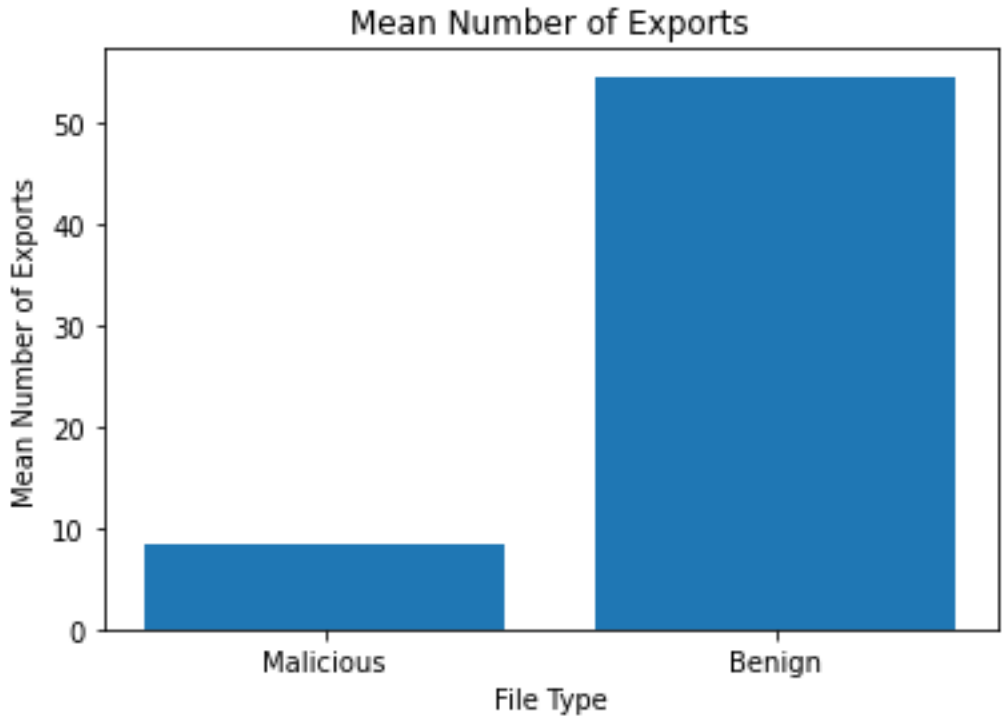

Label Distribution by has_tls
Benign
Malicious
Count
has_tls
0.0
1.0

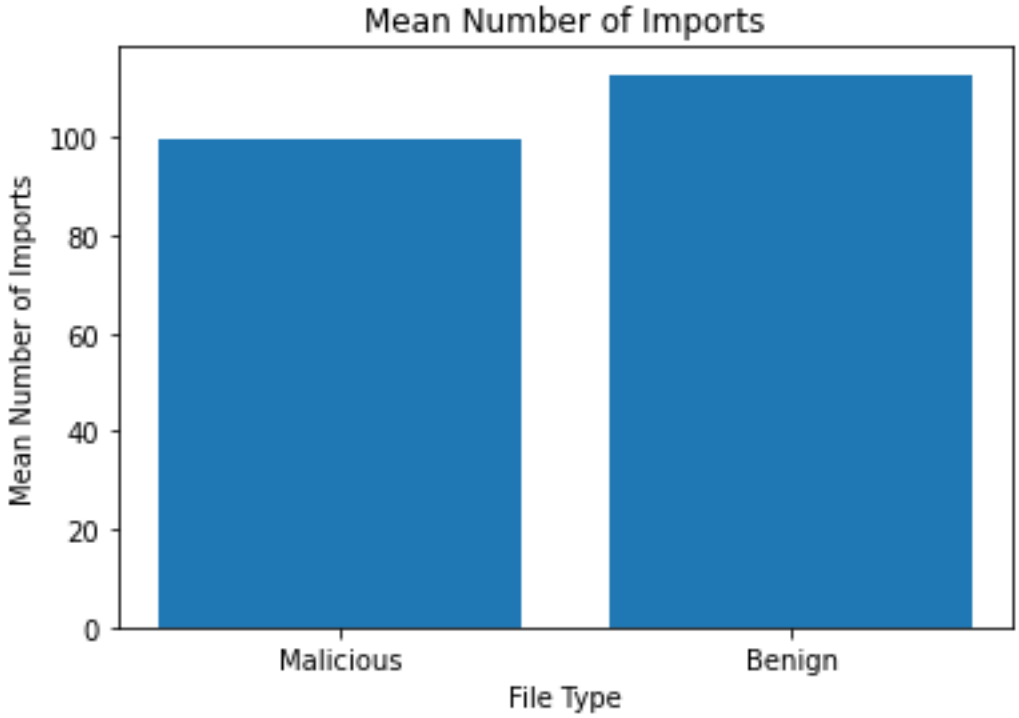

Label Distribution by has_relocations
Benign
Malicious
Count
has_relocations
0.0
1.0

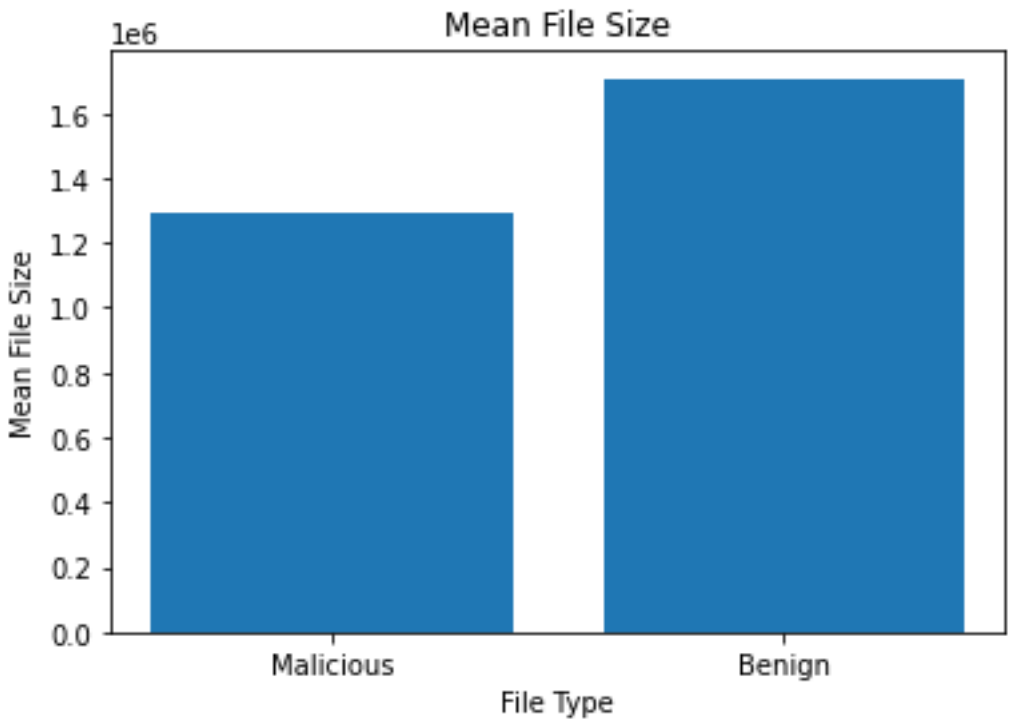

Label Distribution by has_resources
Benign
Malicious
Count
has_resources
0.0
1.0

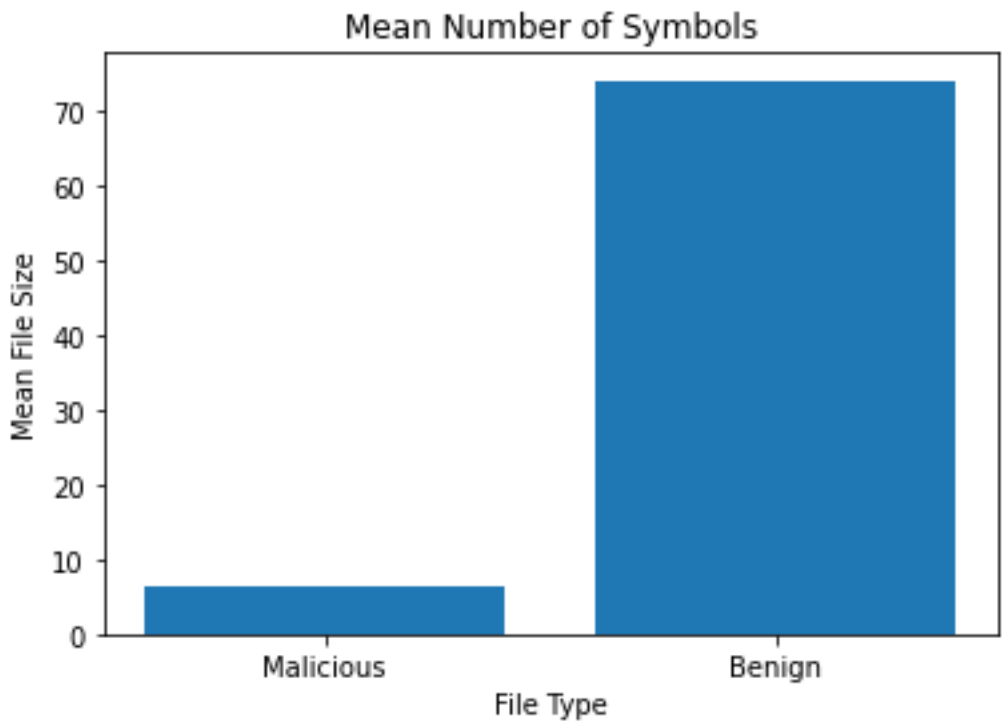

Label Distribution by has_signature
Benign
Malicious
Count
has_signature
0.0
1.0

נספחים 1.10.1 – 1.10.4

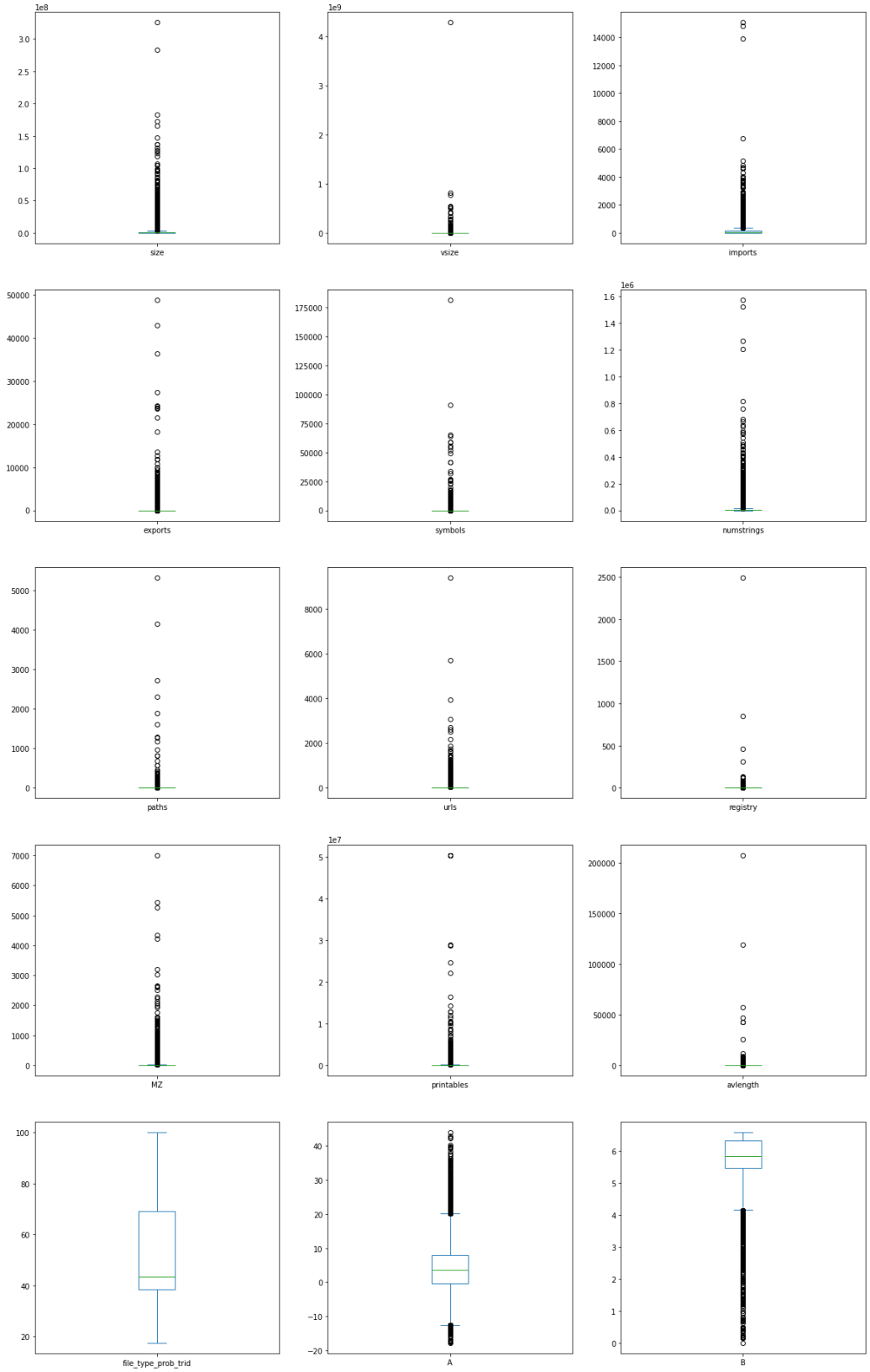




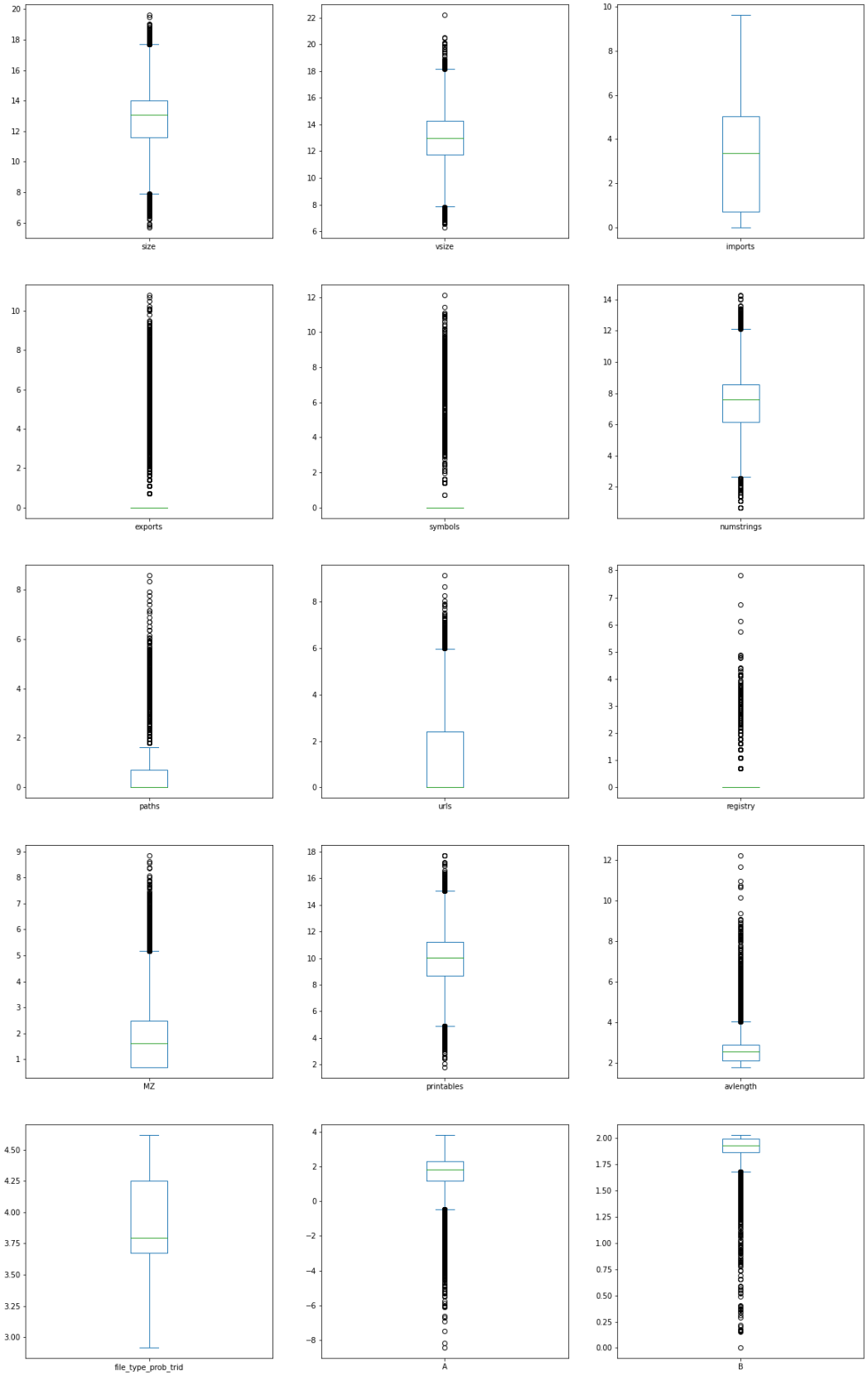

Mean File Size
1e6
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Mean File Size
Malicious
Benign
File Type


Mean Number of Symbols
70
60
50
40
30
20
10
0
Mean File Size
Malicious
Benign
File Type

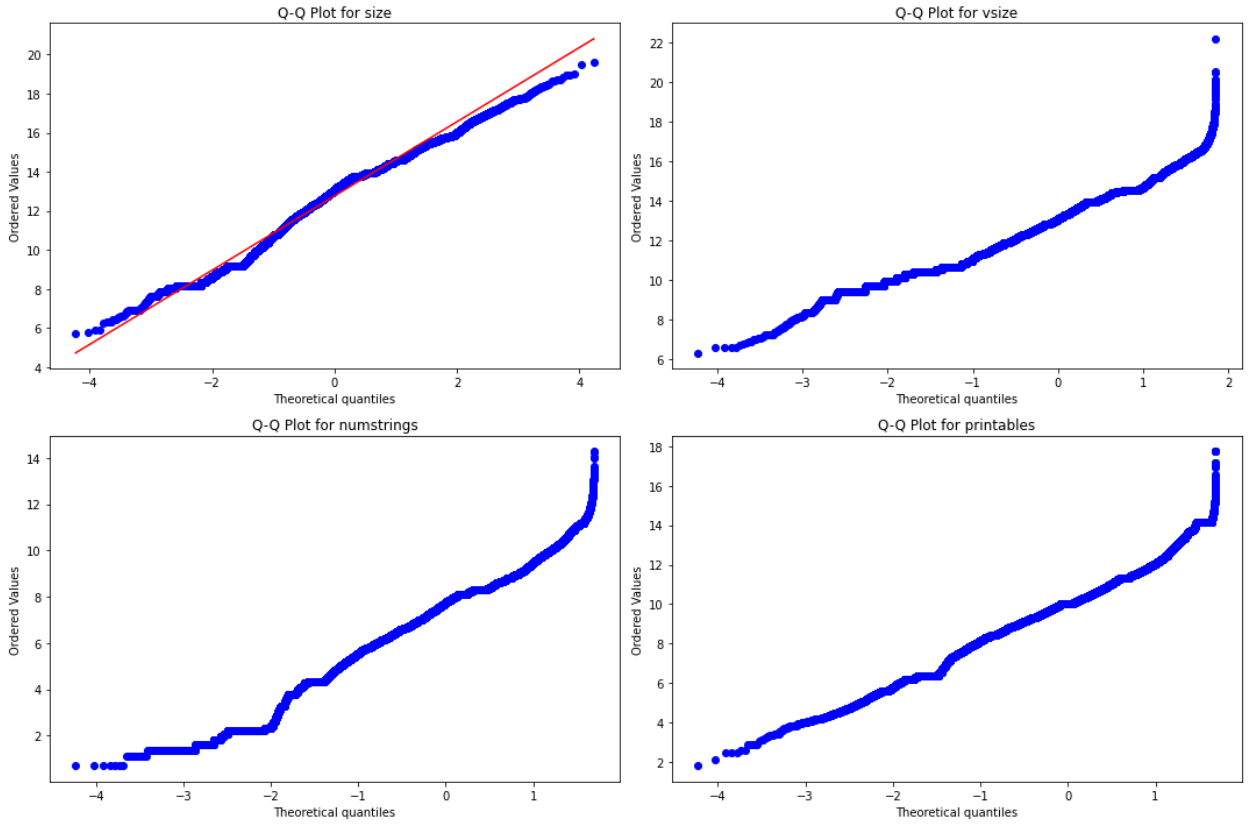
נספח 1.11


1e8
size
1e9
vsize
imports
exports
symbols
1e6
numstrings
paths
urls
registry
MZ
1e7
printables
avlength
file_type_prob_trid
A
B

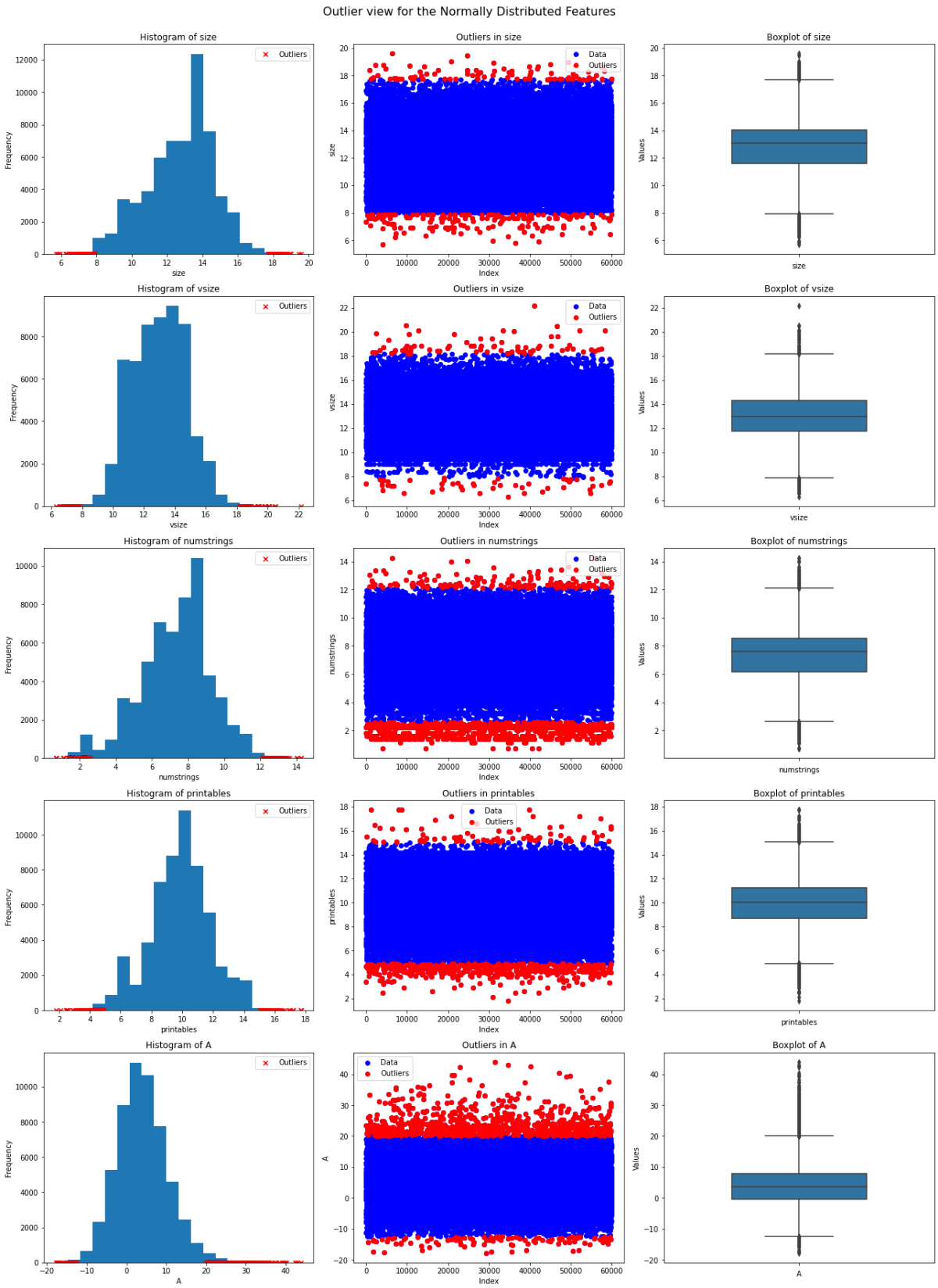
נספח 1.12


size
vsize
imports
exports
symbols
numstrings
paths
urls
registry
MZ
printables
avlength
file_type_prob_trid
A
B

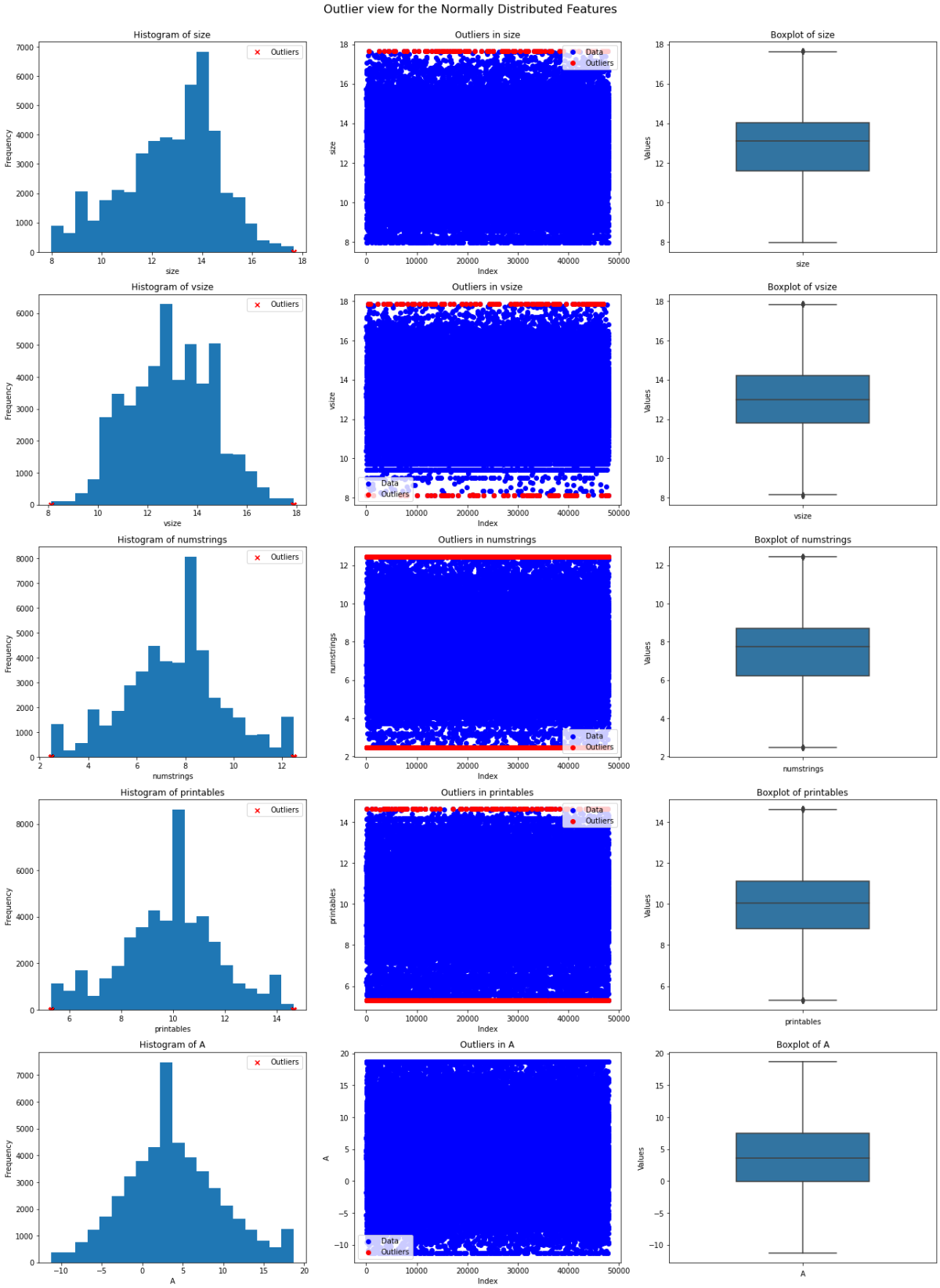
נספח 1.13


Q-Q Plot for size
Ordered Values
Theoretical quantiles
Q-Q Plot for vsize
Ordered Values
Theoretical quantiles
Q-Q Plot for numstrings
Ordered Values
Theoretical quantiles
Q-Q Plot for printables
Ordered Values
Theoretical quantiles

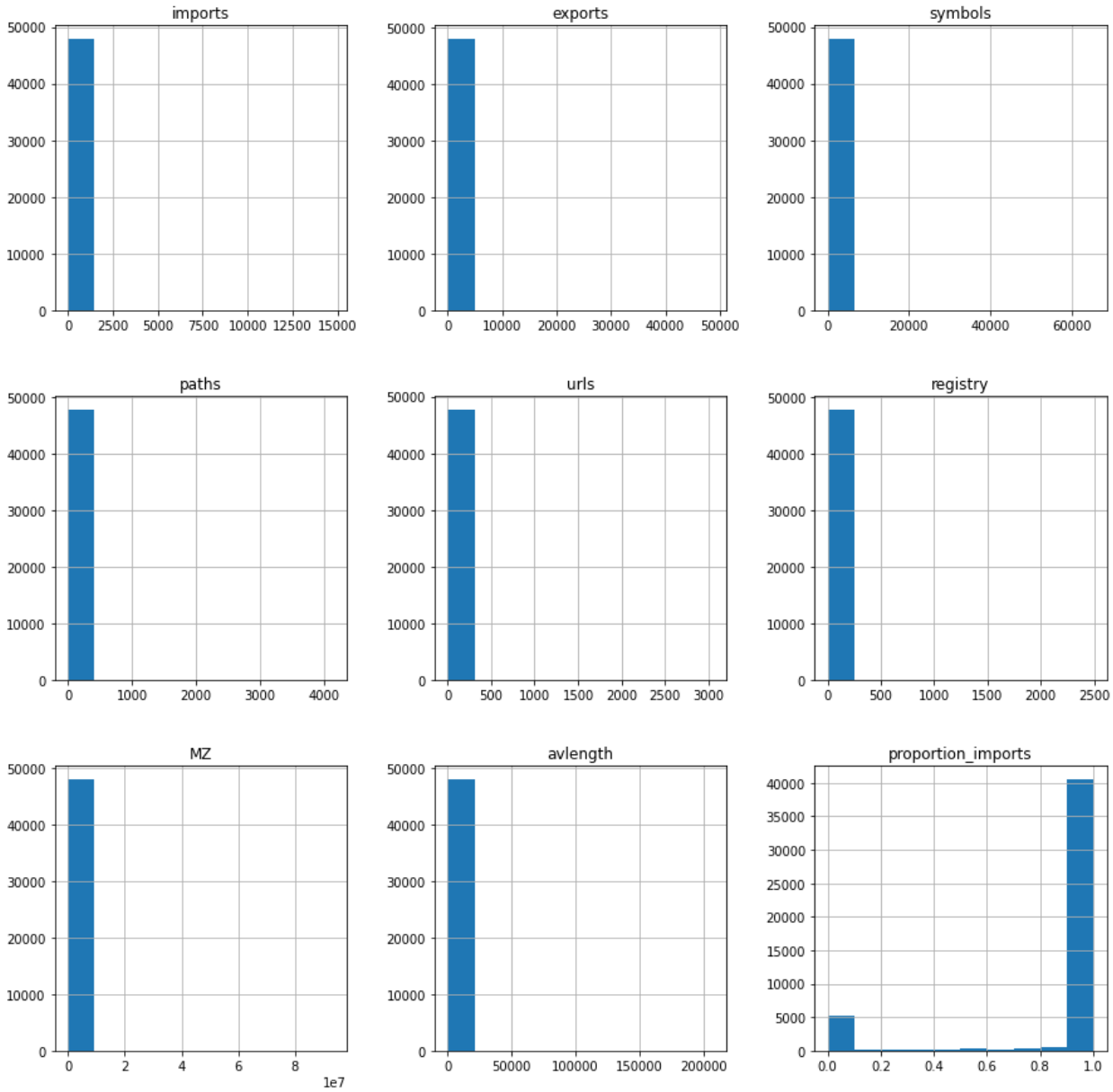
נספח 1.14


Outlier view for the Normally Distributed Features
Histogram of size
Outliers in size
Boxplot of size
Histogram of vsize
Outliers in vsize
Boxplot of vsize
Histogram of numstrings
Outliers in numstrings
Boxplot of numstrings
Histogram of printables
Outliers in printables
Boxplot of printables
Histogram of A
Outliers in A
Boxplot of A
Outliers
Data
Frequency
Index
Values
size
vsize
numstrings
printables
A

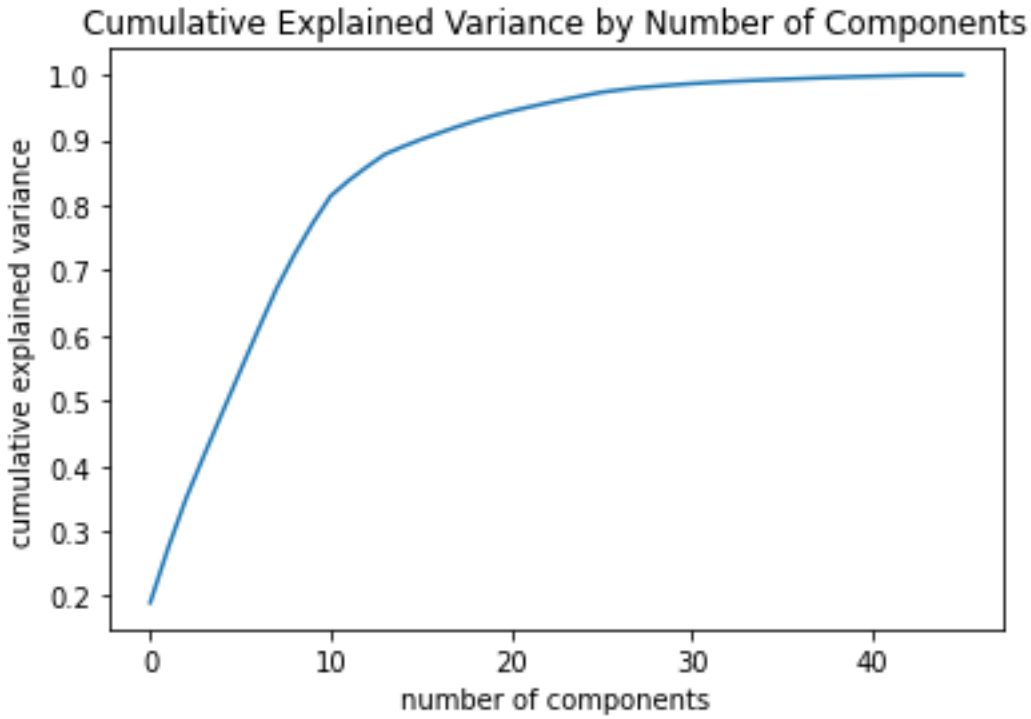
נספח 2.1


Outlier view for the Normally Distributed Features
Histogram of size
Outliers in size
Boxplot of size
Histogram of vsize
Outliers in vsize
Boxplot of vsize
Histogram of numstrings
Outliers in numstrings
Boxplot of numstrings
Histogram of printables
Outliers in printables
Boxplot of printables
Histogram of A
Outliers in A
Boxplot of A
Outliers
Data
Frequency
Values
Index
size
vsize
numstrings
printables
A

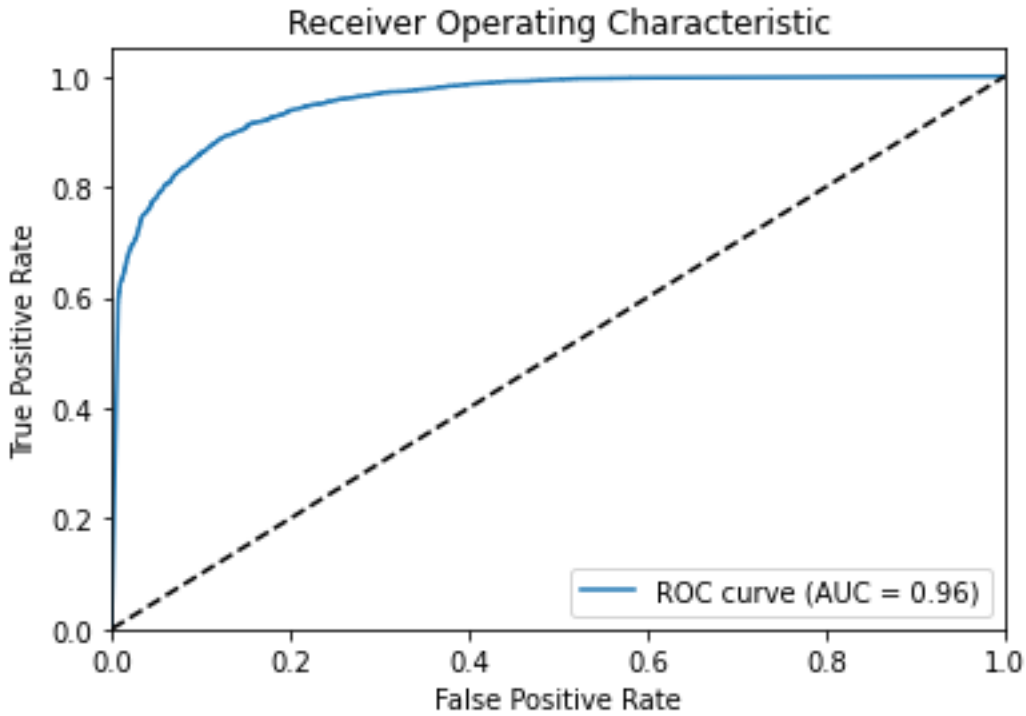
נספח 2.2


imports
exports
symbols
paths
urls
registry
MZ
1e7
avlength
proportion_imports

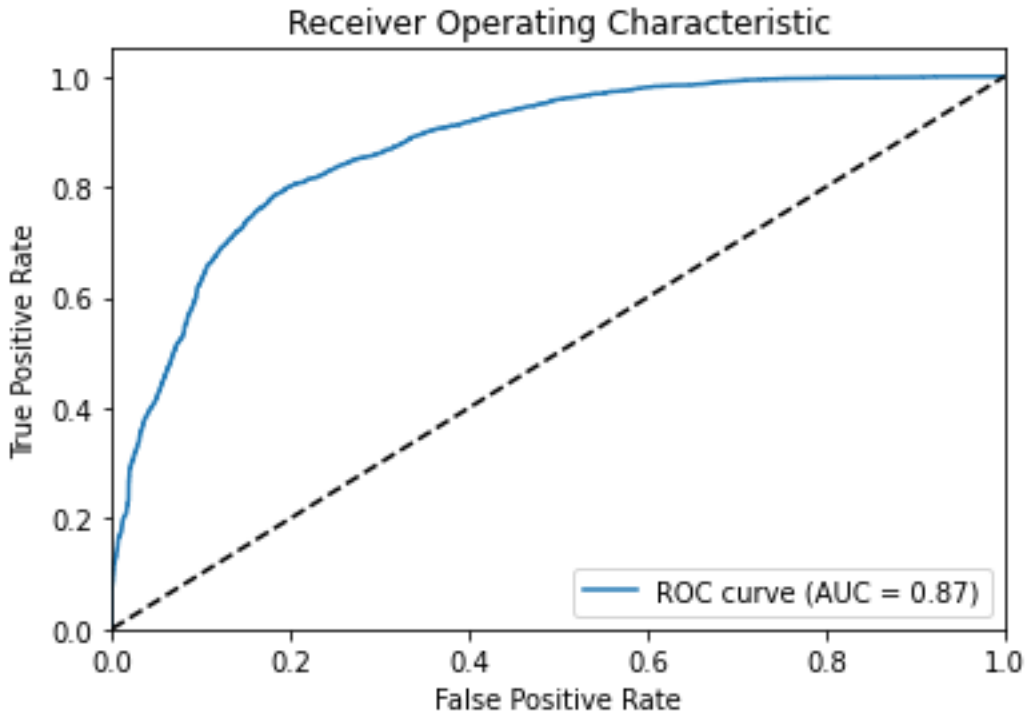
נספח 2.3



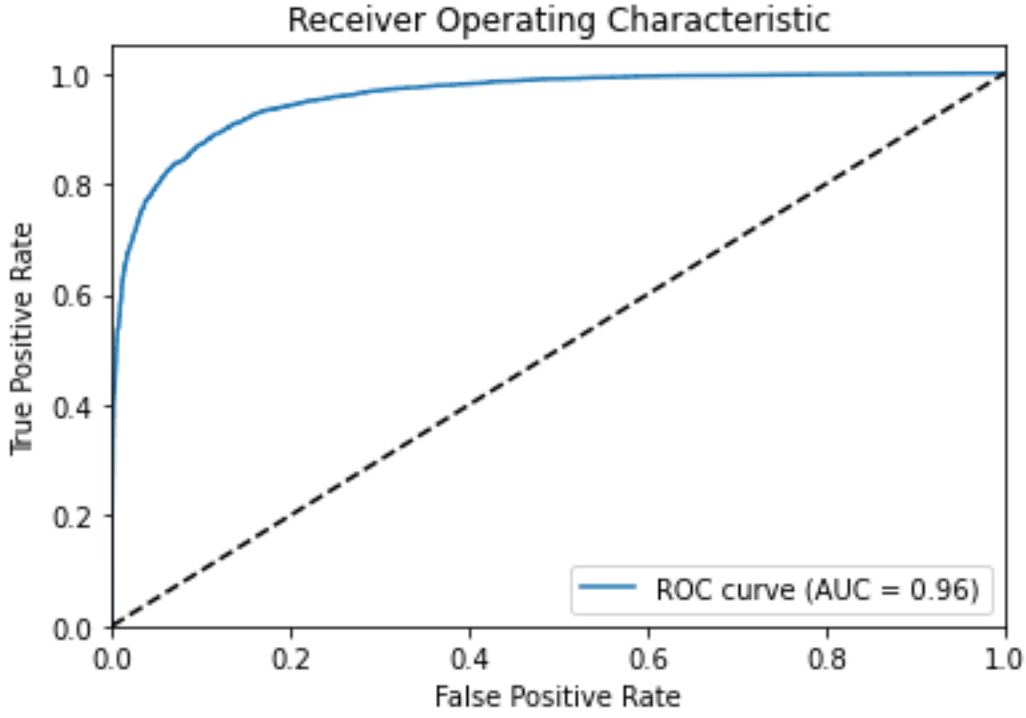
נספח 3.1



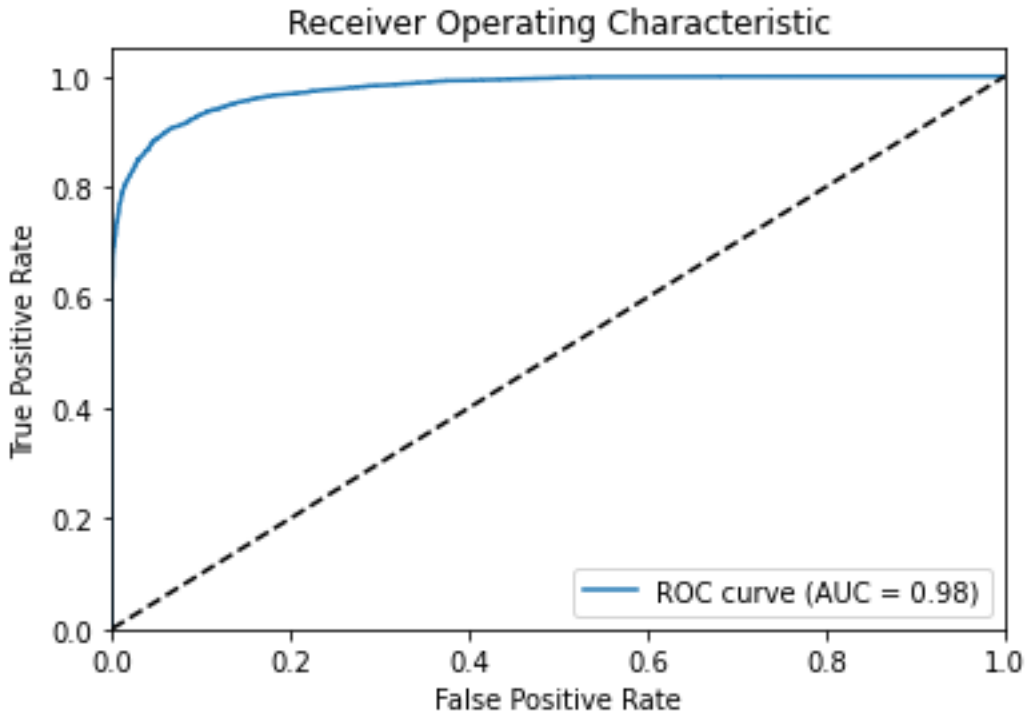
נספח 3.2



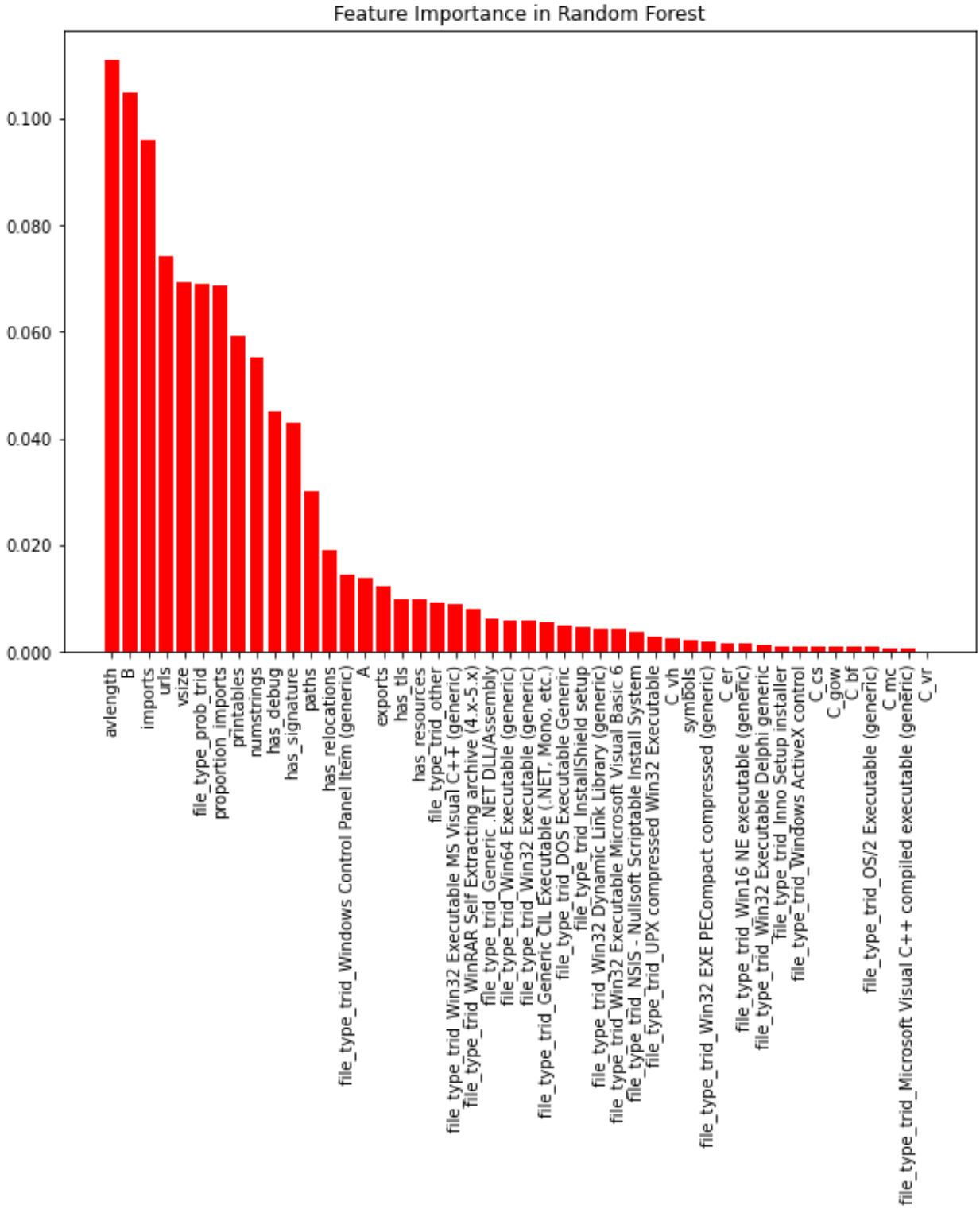
נספח 3.3



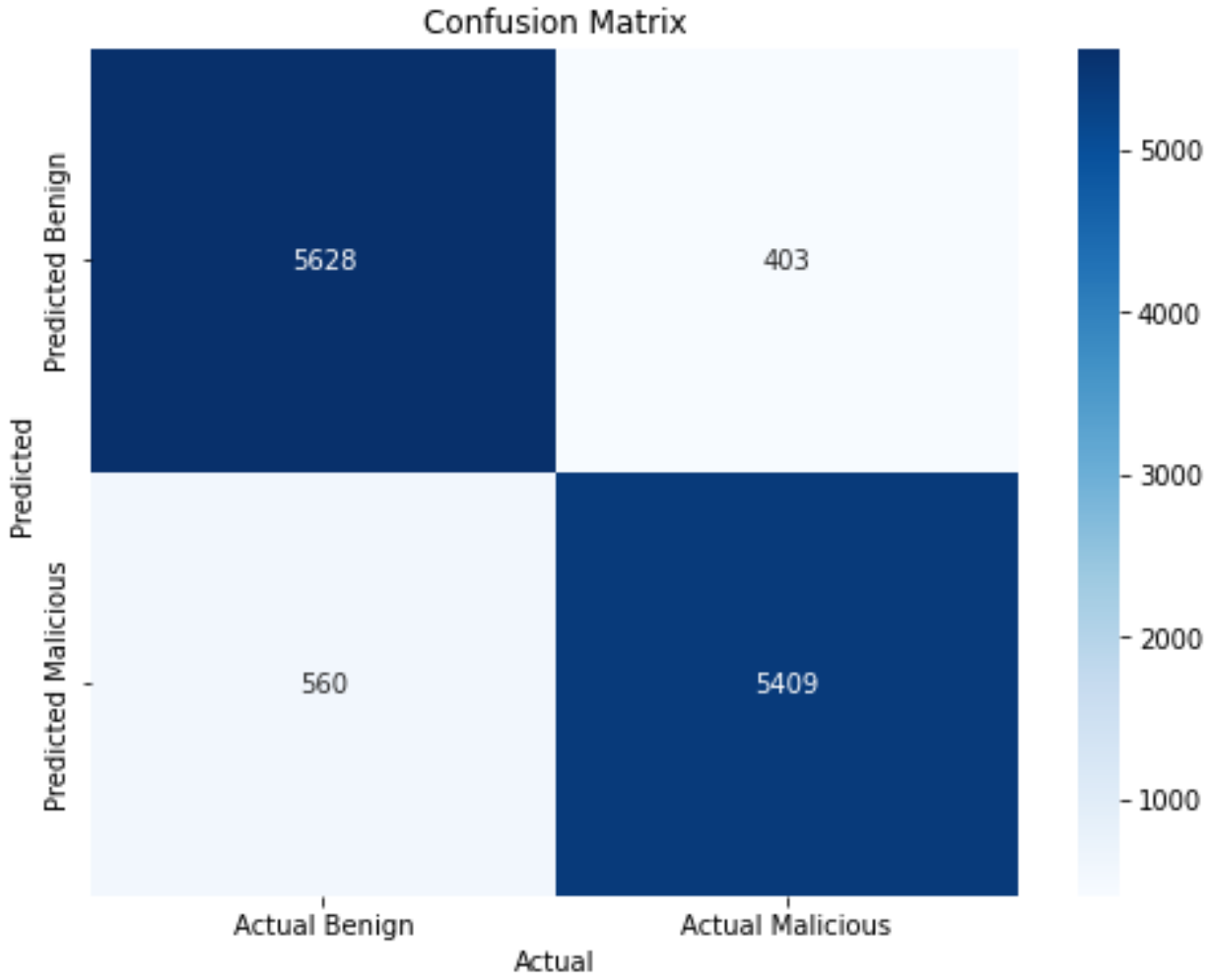
נספח 3.4


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.98)

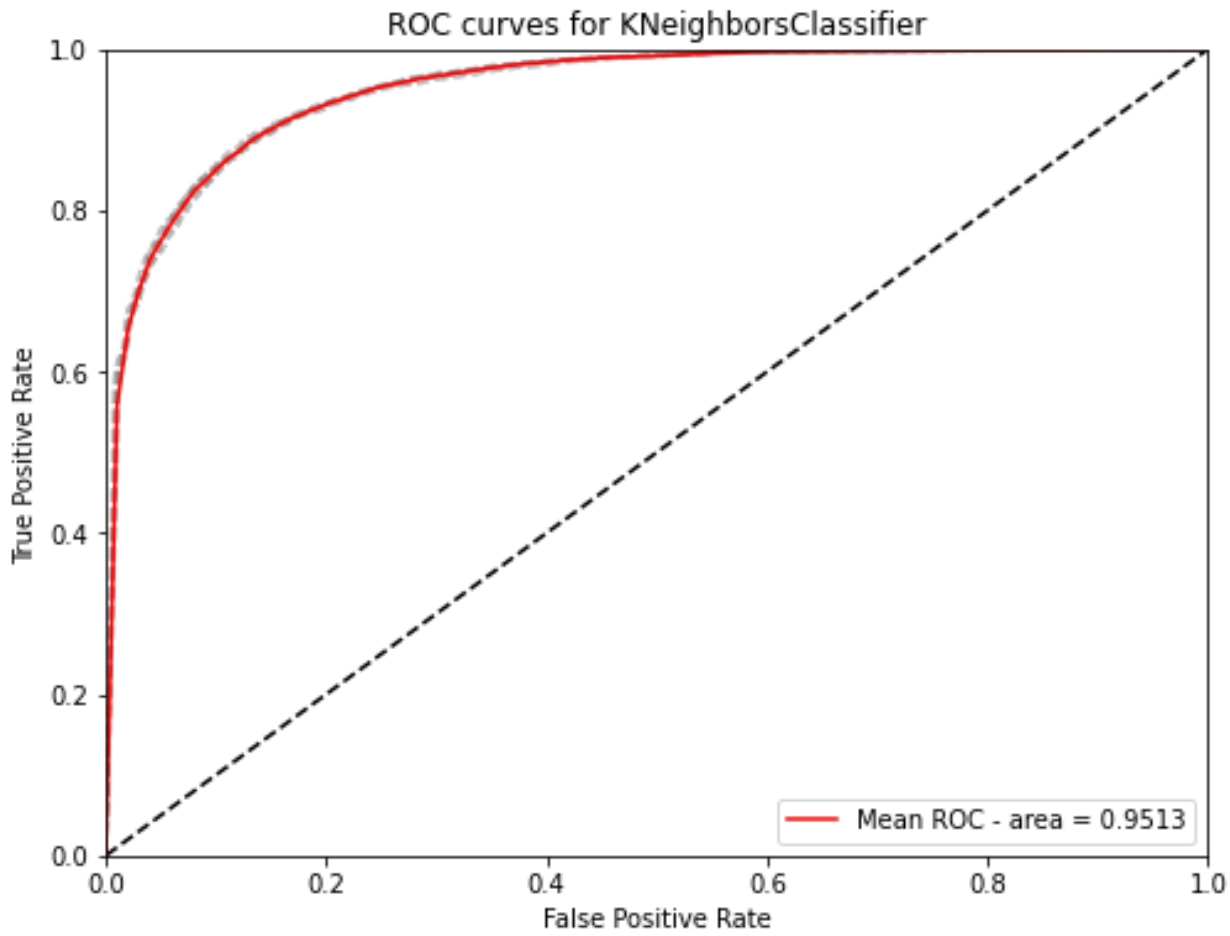
נספח 3.5


Feature Importance in Random Forest
0.100
0.080
0.060
0.040
0.020
0.000
avlength
B
imports
urls
vsize
file_type_prob_trid
proportion_imports
printables
numstrings
has_debug
has_signature
paths
has_relocations
file_type_trid_Windows Control Panel Item (generic)
A
exports
has_tls
has_resources
file_type_trid_other
file_type_trid_Win32 Executable MS Visual C++ (generic)
file_type_trid_WinRAR Self Extracting archive (4.x-5.x)
file_type_trid_Generic .NET DLL/Assembly
file_type_trid_Win64 Executable (generic)
file_type_trid_Win32 Executable (generic)
file_type_trid_Generic CIL Executable (.NET, Mono, etc.)
file_type_trid_DOS Executable Generic
file_type_trid_InstallShield setup
file_type_trid_Win32 Dynamic Link Library (generic)
file_type_trid_Win32 Executable Microsoft Visual Basic 6
file_type_trid_NSIS - Nullsoft Scriptable Install System
file_type_trid_UPX compressed Win32 Executable
C_vh
symbols
file_type_trid_Win32 EXE PECompact compressed (generic)
C_er
file_type_trid_Win16 NE executable (generic)
file_type_trid_Win32 Executable Delphi generic
file_type_trid_Inno Setup installer
file_type_trid_Windows ActiveX control
C_cs
C_gow
C_bf
file_type_trid_OS/2 Executable (generic)
C_mc
file_type_trid_Microsoft Visual C++ compiled executable (generic)
C_vr

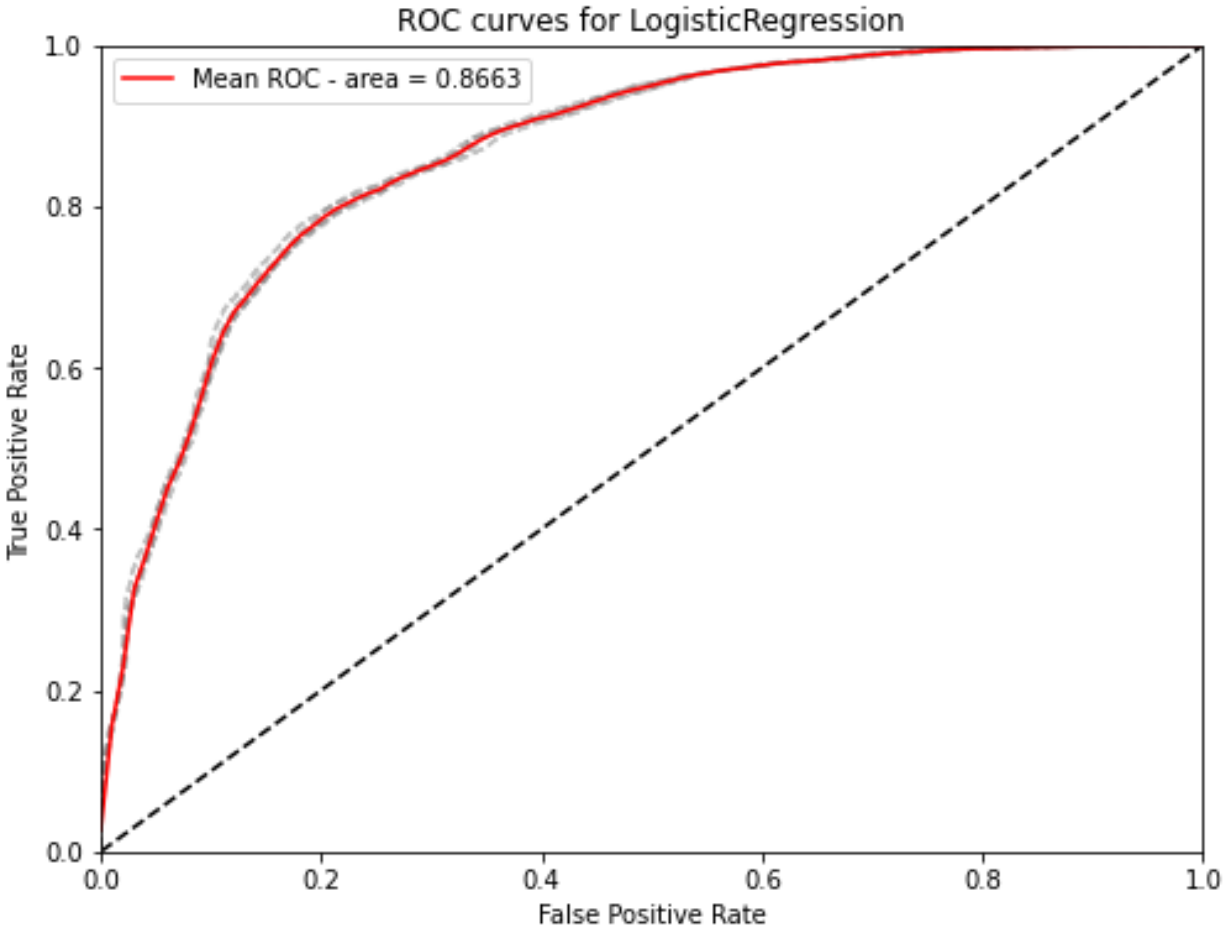
נספח 4.1


Confusion Matrix
Predicted Benign
Predicted Malicious
Predicted
5628
403
560
5409
Actual Benign
Actual Malicious
Actual
5000
4000
3000
2000
1000

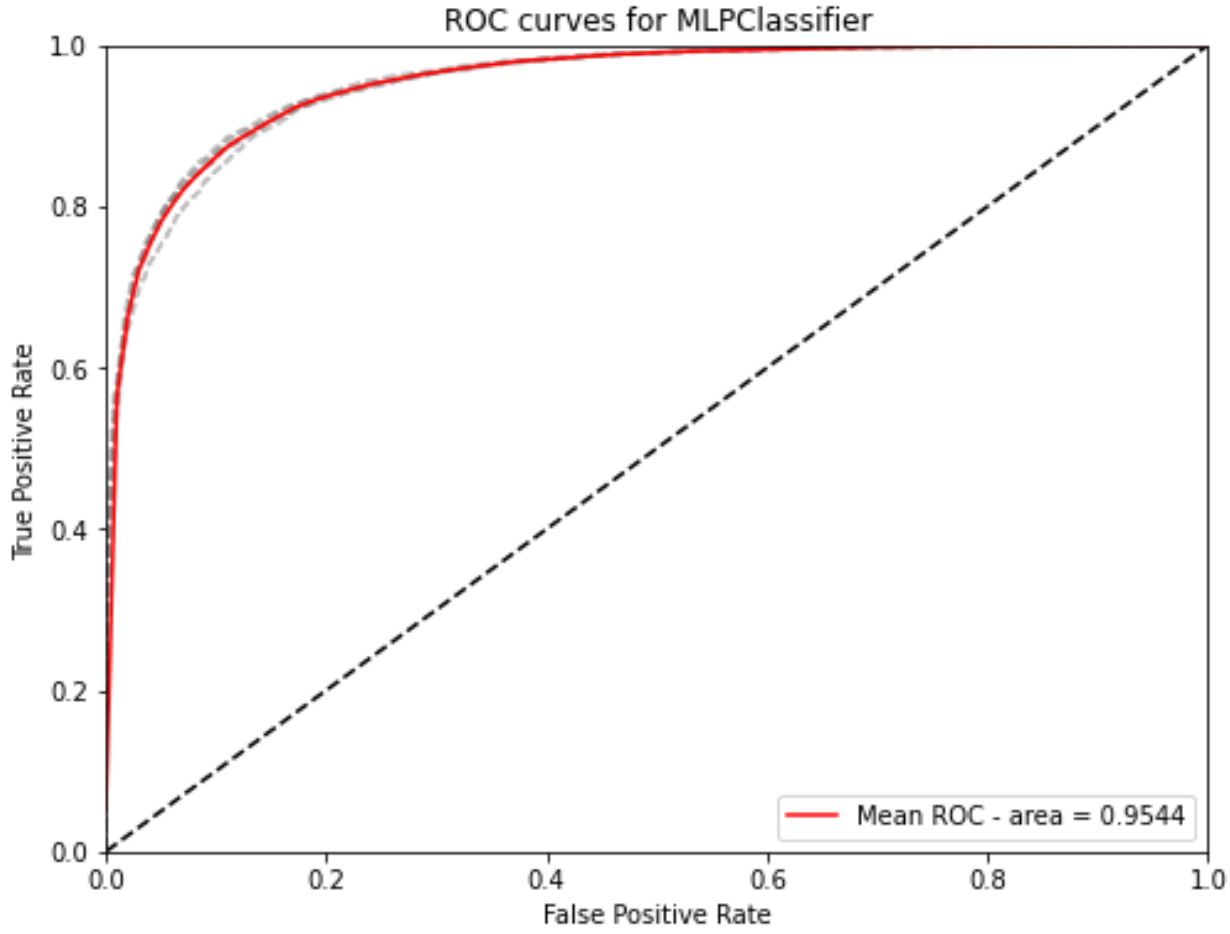
נספח 4.2


ROC curves for KNeighborsClassifier
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Mean ROC - area = 0.9513

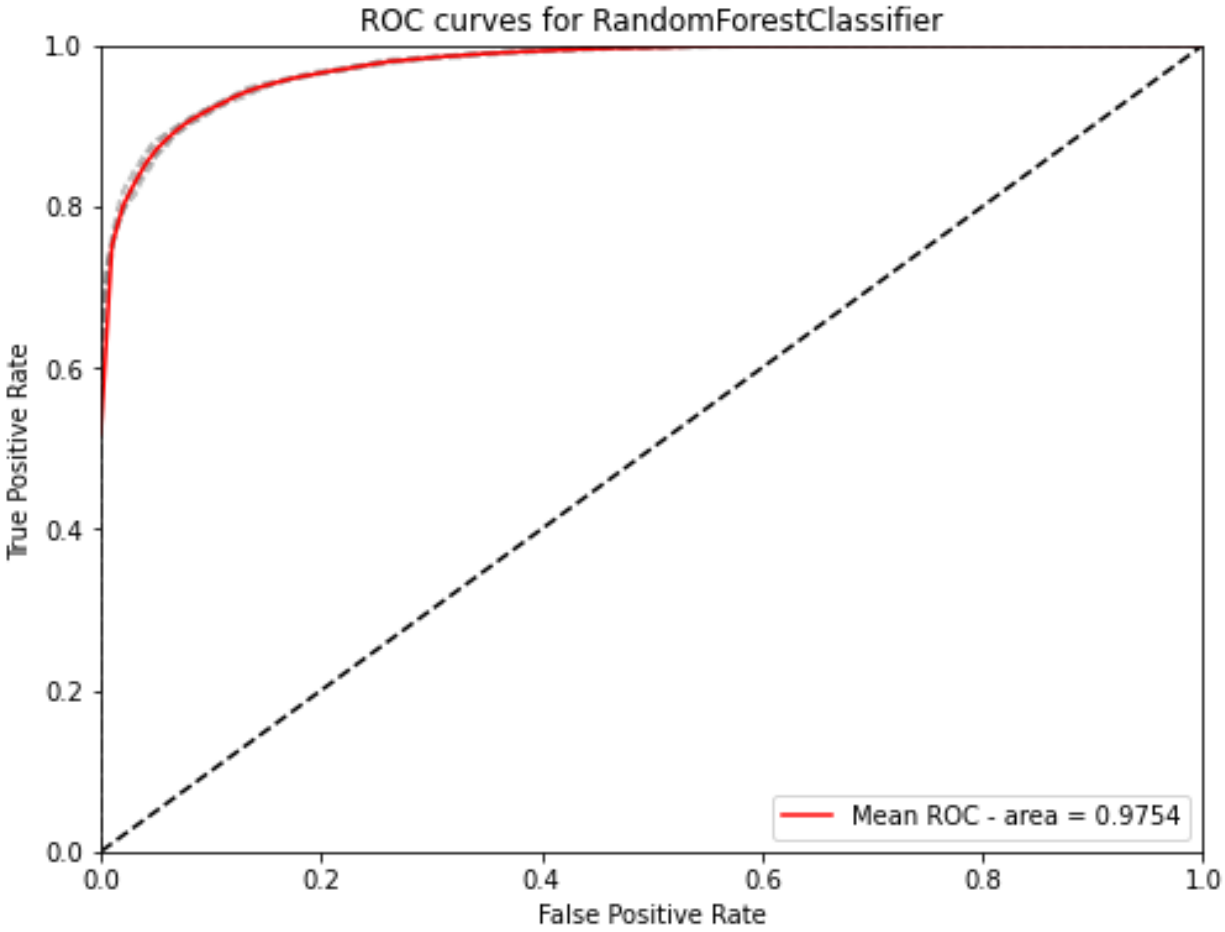
נספח 4.3


ROC curves for LogisticRegression
Mean ROC - area = 0.8663
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

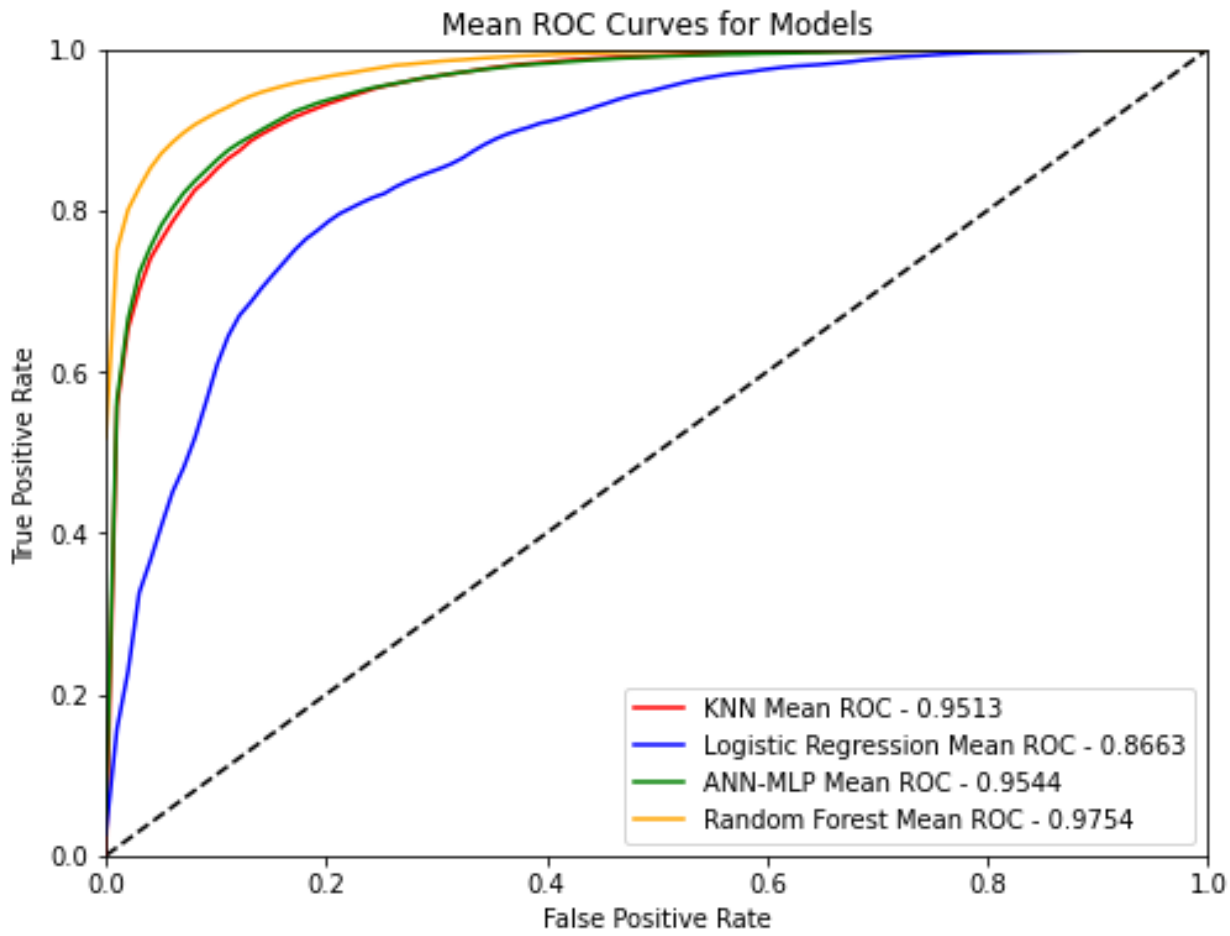
נספח 4.4


ROC curves for MLPClassifier
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Mean ROC - area = 0.9544

נספח 4.5


ROC curves for RandomForestClassifier
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Mean ROC - area = 0.9754

נספח 4.6


Mean ROC Curves for Models
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
KNN Mean ROC - 0.9513
Logistic Regression Mean ROC - 0.8663
ANN-MLP Mean ROC - 0.9544
Random Forest Mean ROC - 0.9754

נספח 11 - VIF

VIF for all features except category features:

	Variable	VIF
0	size	12.355045
1	vsize	1.013361
2	imports	1.479693
3	exports	1.059537
4	has_debug	2.164934
5	has_relocations	2.669602
6	has_resources	7.216717
7	has_signature	1.689708
8	has_tls	1.548797
9	symbols	1.172485
10	numstrings	12.103221
11	paths	1.011277
12	urls	1.328562
13	registry	1.007875
14	MZ	1.853792
15	printables	3.450308
16	avlength	2.674772
17	file_type_prob_trid	6.775868
18	A	1.377063
19	B	12.090778

הסוף :)