למידה חישובית – פרוייקט מסכם

דו"ח מסקנות

מגישים: אור אוקסנברג, 312460132

סהר בריבי, 311232730

1. שם האלגוריתם:

RandomForest

1. מאמר:

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.‏

<http://machinelearning202.pbworks.com/w/file/fetch/60606349/breiman_randomforests.pdf>

1. מוטיבציה לאלגוריתם:

“Significant improvements in classification accuracy have resulted from growing an ensemble of trees and letting them vote for the most popular class.” (Breiman, 2001)

אלגוריתמי אנסמבל הוכחו ככאלה אשר משפרים את דיוק הסיווג. שימוש באוסף של עצי החלטה ויצירת voting על התוצאות של כל אחד מהם, מביאה לשיפור הדיוק. אחת מהבעיות עם עצי החלטה בודדים היא שכאשר אין מגבלה על העץ, אנחנו יכולים להגיע בקלות למצב של overfitting – RandomForest הוא רק אחת מהדרכים לפתור בעיה זו.

1. תיאור קצר:

“A random forest is a classifier consisting of a collection of tree-structured classifiers

{h(x,Θk ), k=1, ...} where the {Θk} are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x .” (Breiman, 2001)

1. פסאודו קוד:

***N***

1. הסבר האלגוריתם:

תנאי התחלה לבניית העץ הוא דאטאסט S כלשהו, בעל F פיצ'רים. בנוסף, עלינו להחליט על גודל היער שאנחנו רוצים לבנות, כלומר לבחור את כמות העצים שהיער יכיל, אותה נסמן בB.

הפונקציה הראשית, RandomForest, מקבלת את S ואת F (שורה 1). תחילה נייצר קבוצה ריקה H, שאליה נכניס את העצים לאחר יצירתם. עבור כל עץ מבין B העצים, נבצע דגימה של הדאטסאט באמצעות bootstrap – נסמן אותה כ *(שורה 4). כעת, נשתמש בפונקציית על מנת לייצר את העץ, כאשר הפונקציה מקבלת את* F*, כמות הפיצ'רים של הדאטה, ואת ,* דגימת הדאטה שיצרנו (שורה 5). בפונקציה תחילה יוצרים קודקוד ריק וממלאים אותו עם הדאטה שמגיע לאותו השלב בריקורסיה, כאשר תהליך זה חוזר על עצמו עבור כל העצים ביער, כאשר בכל שלב אנחנו מוסיפים את העץ שנוצר ליער H שבנינו.

הפונקציה היא הפונקציה שאחראית על יצירת העץ, תוך יצירת קדקוד ריק בכל שלב והשמת הדאטה הרלוונטי לאותו קדקוד, והיא עובדת בצורה רקורסיבית. בכל איטרציה, אנחנו בוחרים חלק מהפי'צרים בצורה רנדומלית. מתוך הפיצ'רים שנבחרו, אנו בוחרים את הפיצ'ר בעל הinformation gain הגבוה ביותר (יכול להיות מחושב בדרכים שונות כמו ג'יני, אנתרופיה ועוד), ליצירת הnode של העץ (שורה 16,17). עבור כל node, נפצל את הדאטה לפי התנאי של אותו node, ונקרא שוב לפונקציה. בתחילת הפונקציה נבדוק האם יש לנו דאטה של יותר מclass אחד – זהו תנאי העצירה שלנו – במידה וקיים יותר מclass אחד נמשיך ביצירת העץ, אחרת נעצור (שורה 11).

1. דוגמאות:

האלגוריתם משתמש בbootstrap על מנת לבחור חלק מהדאטה, שבעזרתו יבנה עץ ההחלטה הנוכחי. בכל איטרציה נבחר חלק מהדאטה ונוצר עץ החלטה נוסף. מספר האיטרציות נקבע לפי כמות עצי ההחלטה הנבחרת. כלומר, בכל שלב האלגוריתם בונה עץ החלטה נוסף.

על מנת להציג תהליך זה עשינו שימוש בדאטאסט של acute-inflammation, ויצרנו 10 עצים. נציג את חלקם (תמונה1, תמונה2, תמונה 3). כל עץ החלטה נבנה בעזרת הפונקציה RandomizedTreeLearn אשר מקבלת חלק מהדאטה וכמות פיצ'רים ומחזירה עץ החלטה. תהליך בניית העץ נעשה בצורה איטרטיבית – בכל שלב נבחרת כמות פיצ'רים מצומצמת, ובעזרת מדדים שונים, ביניהם ג'יני, אנתרופיה ועוד, נבחר הפיצ'ר הטוב ביותר לפי המדד. בעצים המוצגים בתמונות לעיל נעשה שימוש במדד ג'יני בבנייתם.

A picture containing object, clock

Description automatically generated

תמונה 1 – עץ החלטה ראשון

A close up of a device

Description automatically generated

תמונה 2 – עץ החלטה 2

A picture containing clock

Description automatically generated

תמונה 3 – עץ החלטה 5

1. יתרונות:

* מדובר בשיטת אנסמבל הטובה יותר מעץ החלטה בודד, מאחר והיא מקטינה את הסיכוי לover-fitting ע"י מיצוע של התוצאות (בעזרת voting). הקונספט העיקרי מאחורי האלגוריתם של RandomForest הוא “The wisdom of crowds”.
* האלגוריתם מתאים הן לבעיות סיווג והן לבעיות רגרסיה
* לא קיימת מגבלה על כמות הפיצ'רים איתה האלגוריתם יודע להתמודד.
* האלגוריתם הוא בעל explainability גבוהה – העובדה שקיימים מספר עצים וניתן לדעת מה תהליך ההחלטה עבור כל פרדיקציה בכל עץ מסייעת להבנת המודל.

1. חסרונות:

* האלגוריתם אמנם מתאים גם לבעיות רגרסיה, אך הביצועים שלו על בעיות אלו פחות טובים.
* זמן אימון ארוך יותר – מאחר והאלגוריתם מייצר כמות גדולה של עצים ומקבל החלטה לפי הרוב, זמן האימון שלו גבוה ארוך יותר מעץ החלטה רגיל.
* סיבוכיות – יצירת כמות העצים שהאלגוריתם דורש (בדיפולט יצירה של 100 עצים), דורשת יותר משאבים מאשר יצירת עץ החלטה בודד.

1. פרטי הניסויים:

**אלגוריתמים:**

* RandomForest
* InfiniteBoost
* NGBoost
* KTBoost

**מדדי השוואה לבחינה:**

* Accuracy
* TPR
* FPR
* Precision
* השטח מתחת לעקומת ROC – AUC
* השטח מתחת לעקומת Precision-Recall
* זמן הריצה הנדרש לאימון מודל
* זמן הריצה הנדרש לInference של 1000 רשומות

אלו הם מדדי ההשוואה שאספנו עבור כל אחד מהאלגוריתמים שרצו על כל אחד מהדאטאסטים. עם זאת, עבור המבחן הסטטיסטי, מבחן פרידמן, בחרנו להשתמש במדד AUC על מנת לבצע את ההשוואה. בנוסף, עבור קבלת ההחלטה על האלגוריתם המוביל לטובת הרצת הMetaClassifier, עשינו שימוש גם כן בAUC.

**דאטאסטים:**

בעבודה בחרנו לבצע משימת סיווג ולכן בחנו את האלגוריתמים על 150 הדאטסטים המופיעים [כאן](https://drive.google.com/file/d/11l8Nnpt5_bb6eIvfHsKiNJfR8oa7bIoB/view).

**היפר פרמטרים:**

הרצנו RandomizedSearchCV לו נתנו את הטווחים של ההיפר-פרמטרים אותם אנו מעוניינים לבדוק. בחלק זה נפרט עבור כל אלגוריתם את ההיפר פרמטרים שבדקנו וכן את הטווחים עבור כל אחד מהם.

* RandomForest
  + Max\_depth - הטווח הנבחר הוא 3-15.
  + n\_estimators - הטווח הנבחר הוא בין 5-20.
* InfiniteBoost
  + n\_estimators – הטווח הנבחר הוא בין 5-20.
  + Max\_leaf\_nodes – הטווח הנבחר הוא 2-9.
* NGBoost
  + n\_estimators - הטווח הנבחר הוא בין 5-20.
  + Base\_\_max\_depth - הטווח הנבחר הוא 4-13.
* KTBoost
  + Max\_depth - הטווח הנבחר הוא 2-8.
  + n\_estimators - הטווח הנבחר הוא בין 5-20.

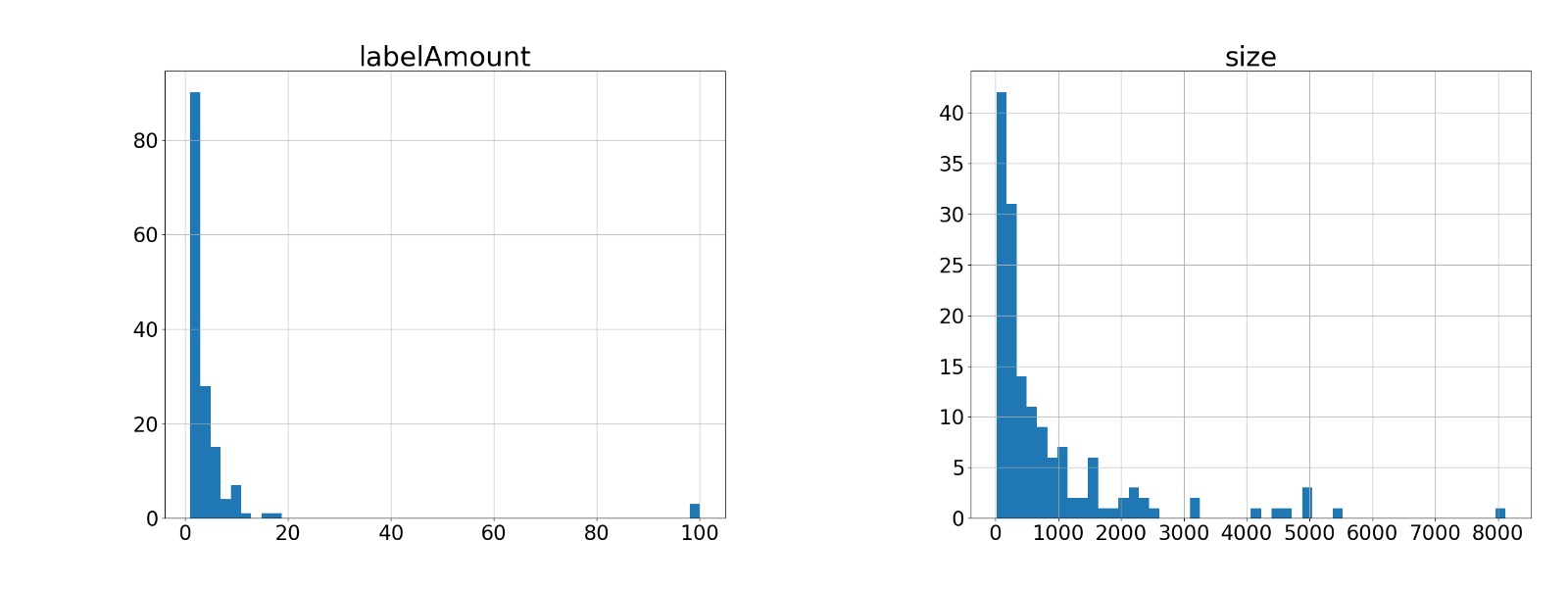
ההיפר פרמטרים נבחרו לטובת התמודדות עם זמני ריצה ארוכים. למרות שהדיפולט של n\_estimators הוא גבוה יותר עבור האלגוריתמים שבחרנו, לא בחרנו בערכים גבוהים יותר בגלל זמני ריצה ארוכים מדי. בRandomForest בחרנו בn\_estimators שהוא נמוך מהדיפולט מ2 סיבות עיקריות. הראשונה היא שרצינו להיות סביב אותו טווח ערכים של שאר האלגוריתמים, והשניה היא שבגרסאות הקודמות של האלגוריתם, הדיפולט היה 10.

**תיאור הליך הניסויים ותוצאות:**

מטרת הפרוייקט היא לבחון את ביצועיהן של שיטות Ensemble שונות, אל מול שיטות ידועות ופופולריות. בעבודתינו, בחרנו שלושה אלגוריתמים פחות ידועים - InfinteBoost, KTBoost, NGBoost, והשווינו אותם לאלגוריתם המוכר והידוע RandomForest.

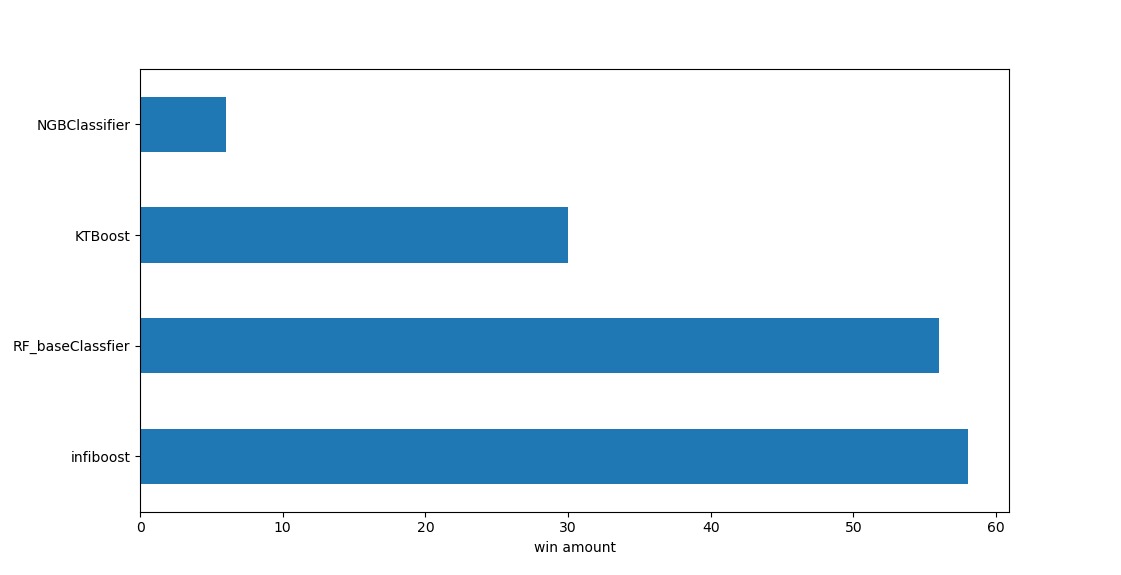
לטובת בחינת הביצועים, אנחנו בחרנו לבצע משימת סיווג, והרצנו את ארבעת האלגוריתמים על 150 דטסטים, כפי שמפורט בחלק של הניסויים. עבור כל דאטאסטים הרצנו 10 folds cross validation חיצוני, כלומר עבור כל דאטאסט היו לנו 10 רשומות עבור כל אלגוריתם.

ניתוח קטן של הדאטאסטים מראה שרוב הדאטסטים הם בעלי סיווג בינארי, אבל ניתן לראות מהגרף השמאלי למטה, שיש גם דאטאסטים בהם אנחנו מתמודדים עם בעיית multi-class. את המדדים חישבנו בעזרת one-vs-all, ע"י שימוש בmacro average. בנוסף, רצינו לראות את התפלגות הדאטאסטים לפי גודלם. הגרף הימני למטה מציג התפלגות זו. רוב הדאטאסטים איתם עבדנו הם קטנים, בעלי פחות מ1000 רשומות.



לאחר הרצת האלגוריתמים השונים עבור כל הדטסטים, ואיסוף המדדים השונים, הרצנו את מבחן פרידמן, על מנת לבדוק האם ההפרשים הקיימים בין האלגוריתמים אכם מובהקים סטטיסטית. תוצאות מבחן פרידמן מתוארות בחלק הבא.

לאחר מכן, הרצנו metaclassifier, שמטרתו לחזות איזו שיטה תהיה הטובה ביותר על בסיס תכונות Meta-Features אשר חושבו מראש עבור כל דאטאסט. על מנת לקבוע מיהו האלגוריתם המוביל עבור כל דאטאסט, בחרנו להשתמש במדד הAUC. השווינו בין מדדי הAUC השונים שנתן כל אלגוריתם לכל דאטאסט, והאלגוריתם בעל הAUC הגבוה ביותר הוא המוביל עבור אותו דאטאסט.



ההתפלגות המוצגת כאן מראה את כמות הדאטסטים עבורם כל אלגוריתם ניצח. ניתן לראות מהגרף, שהאלגוריתמים המובילים עבור רוב הדאטסטים הם InfiniteBoost ו- RandomForest, כאשר InfinteBoost מוביל במעט.

בעזרת המידע על האלגוריתם המוביל, וכן הMetaFeatures, הרצנו את הMetaClassifier בשיטה של Leave-one-out וחישבנו את מדדי החשיבות ואת מדד SHAP של המודל, אשר את המידע עליהם ניתן למצוא בחלק מדדי metaclassifier.

1. תוצאות מבחן פרידמן:

התוצאות שאספנו בחלק ג' של הפרוייקט מכילות 10 שורות עבור כל אלגוריתם עבור כל דאטאסט (חוץ מדאטאסט אחד שהוא קטן מדי, שעבורו ביצענו פחות CV). על מנת לבצע את מבחן פרידמן, בחרנו את מדד AUC שלפיו נשווה את מדדי הדיוק, על מנת לבדוק האם ההפרשים מובהקים סטטיסטית. ביצענו מיצוע של הCV עבור כל דאטאסט, כך שקיבלנו עבור כל אלגוריתם 150 רשומות – רשומה עבור כל דאטאסט עם הAUC הממוצע עבורו. על מידע זה ביצענו את מבחן פרידמן על מנת לבחון האם ההפרשים בAUC אכן מובהקים סטטיסטית. התוצאה היא:

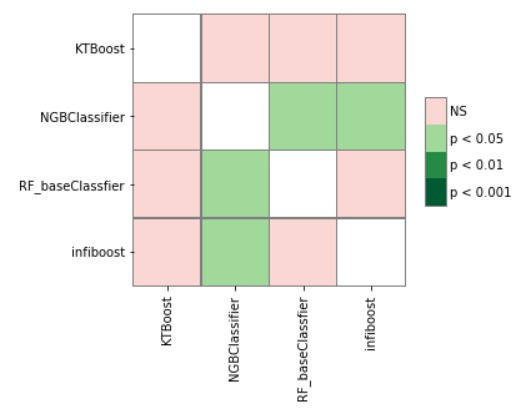
FriedmanchisquareResult(statistic=149.50544959128092, pvalue=3.3685962664244784e-32)

כפי שניתן לראות, הערך של p-value הוא מאוד קטן, ולכן נדחה את השערת האפס. מכאן, שההפרשים בAUC בין האלגוריתמים אכן מובהקים סטטיסטית. לכן, נרצה לבצע מבחני post-hoc, Nemenyi test , אשר יראו את דירוג ההבדלים בין האלגוריתמים.

חישוב מהיר של ממוצע הAUC עבור האלגוריתמים מייצר את הטבלה הבאה:

|  |  |
| --- | --- |
| AlgoName | AUC |
| RF | 0.849869 |
| infiniteBoost | 0.847212 |
| KTBoost | 0.845046 |
| NGBoost | 0.804866 |

הטבלה מראה לנו כי RandomForest הוא המוביל, אחרי בהפרש ממש קטן Infiniteboost וKTboost, ולבסוף בהפרש של כ-4% אלגוריתם NGBoost. עם זאת, נתונים אלו אינם מספיקים ומבחני הpost-hoc הם אלו שמראים לנו את ההבדלים האמיתיים בין האלגוריתמים, אם אכן קיימים כאלו.

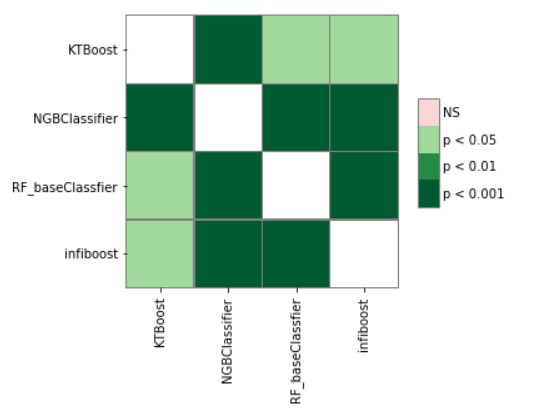
התמונה הבאה מציגה את המובהקות של ההבדלים בין האלגוריתמים. ניתן לראות כי קיים הבדל מובהק בין NGBoost לבין RandomForest ו-InfiniteBoost, הבדל שניתן לראות גם מסיכום הטבלה המופיעה מעלה. נתון מפתיע הוא שאין הבדל מובהק בין KTBoost לבין NGBoost, למרות שקיים הבדל במיצוע הAUC של כ-4%. בנוסף, ניתן לראות כי אין הבדל מובהק בהבדלים בין KTboost, InfiniteBoost וRandomForest.

לאחר שבדקנו את המובהקות הסטטיסטית עבור AUC, רצינו לראות האם אנחנו רואים מובהקות סטטיסטית גם במדד הזמן. לכן, הסתכלנו על הTraining time של כל האלגוריתמים בשניות, ביצענו את המיצוע באותו תהליך שביצענו עבור הAUC והרצנו את מבחן פרידמן שוב. כפי שניתן לראות מהתוצאות, גם כאן קיימת מובהקות סטטיסטית וניתן להגיד שקיימים הפרשים ממשיים בזמנים.

FriedmanchisquareResult(statistic=350.9070247933883, pvalue=9.490218571476184e-76)

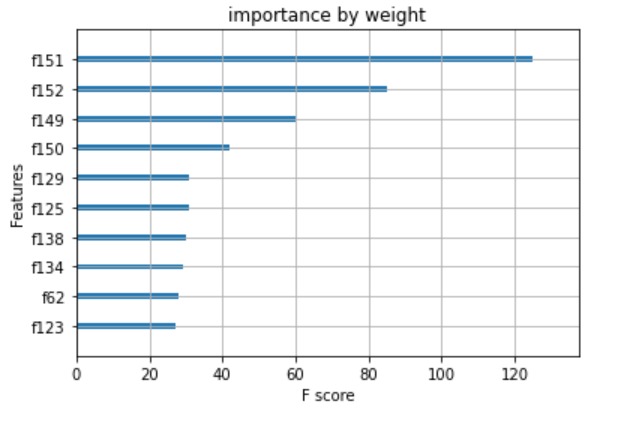
|  |  |
| --- | --- |
| AlgoName | Training\_Time |
| infiniteBoost | 10.072 |
| RF | 12.70667 |
| KTBoost | 37.59556 |
| NGBClassifier | 198.4938 |

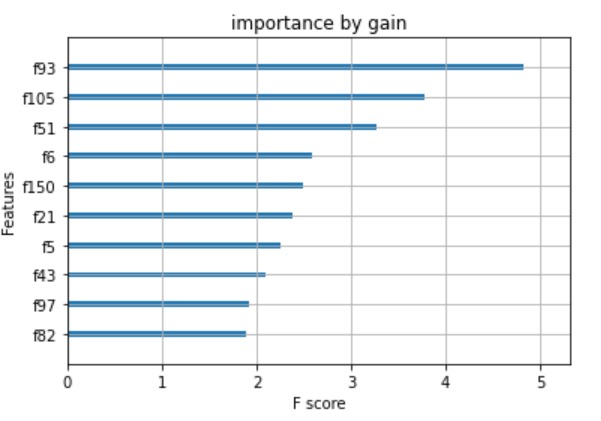
חישוב מהיר של מיצוע הזמנים עבור האלגוריתמים מציג את הטבלה הבאה: InfiniteBoost הוא בעל זמני האימון הקצרים ביותר, אחריו RandomForest, ואחריהם שני האלגוריתמים האחרים. כאשר NGBoost הוא האלגוריתם בעל זמני האימון הגבוהים ביותר, בפער משמעותי משאר האלגוריתמים.

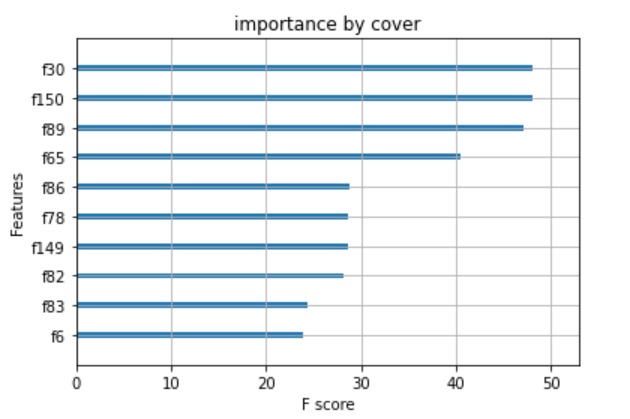
התמונה הבאה מציגה את המובהקות של ההבדלים בין האלגוריתמים. אם בהשוואה בין הAUC לא ראינו יותר מדי הבדלים מובהקים בין האלגוריתמים, כאן ניתן לראות שכל ההבדלים בין האלגוריתמים הם מובהקים, חלקם יותר וחלקם פחות. התמונה מביאה אותנו להסיק, שדירוג האלגוריתמים לפי זמני האימון הוא אכן כפי שמוצג בטבלה – InfiniteBoost הוא בעל זמני האימון הקצרים ביותר, מעט אחריו מגיע RandomForest, אחריהם KTBoost ואחרון בפער משמעותי הוא אלגוריתם NGBoost.

1. מדדי metaclassifier:

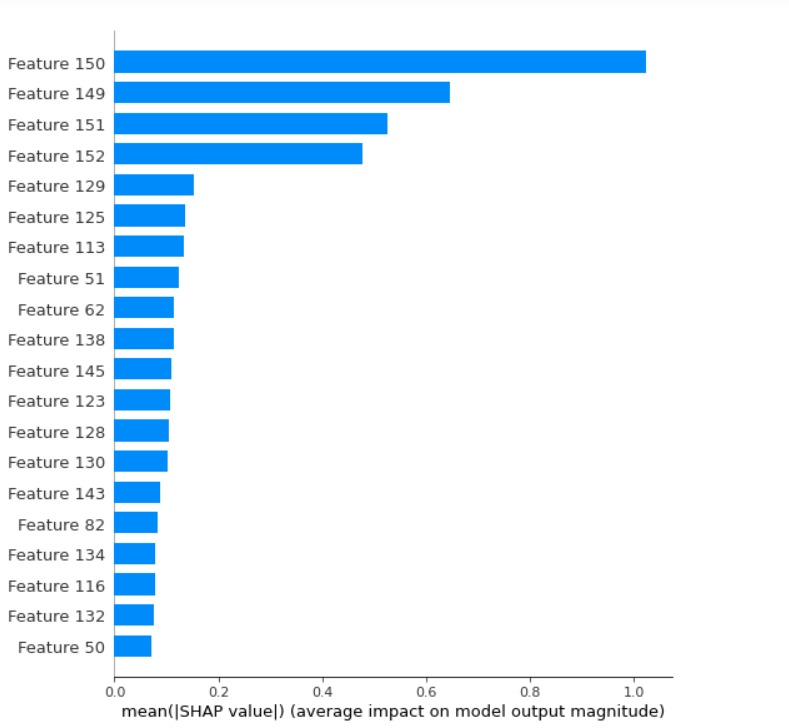
**מדדי החשיבות (Importance):**







**מדד SHAP**



ניתן לראות שבכל אחד מהמדדים השונים קיבלנו חשיבות שונה לפיצ'רים השונים. עם זאת, קיימים מספר פיצ'רים המופיעים ביותר ממדד חשיבות אחד. לדוגמא, פיצ'ר 150 מופיע בכל מדדי הImportance (בחשיבות שונה), וכן במדד SHAP כפיצ'ר בעל החשיבות הגבוהה ביותר. פיצ'ר נוסף שמופיע ביותר ממדד אחד הוא 149, אשר מופיע בכל המדדים חוץ מgain. כמו כן, פיצ'ר 150 מופיע במדד SHAP ובמדד importance by weight.

1. מסקנות:

כפי שציינו, מטרת הפרוייקט היא לבחון את ביצועיהן של שיטות Ensemble שונות, אל מול שיטות ידועות ופופולריות. התהליך שביצענו וכן סקירה קצרה של התוצאות מופיעות בחלק פרטי הניסויים. בחלק זה, נציג את המסקנות שלנו מהתהליך.

הרצת המודלים ואיסוף הפרמטרים השונים, ביניהם AUC, איפשרו לנו לבצע השוואה על המדדים. מיצוע של AUC, עבור כל אחד מהאלגוריתמים, מוצג בטבלה בחלק של תוצאות מבחן פרידמן. נתונים אלו בשילוב עם המבחנים הסטטיסטים הראו לנו כי קיימים אלגוריתמים בעלי AUC גבוה בצורה מובהקת סטטיסטית מאלגוריתמים אחרים. תוצאות המבחנים מראות כי InfiniteBoost וכן RandomForest הם בעלי הAUC הגבוה ביותר, כאשר אין הבדל מובהק ביניהם. בנוסף, אין הבדל מובהק ביניהם גם לKTBoost, אך גם לא קיים הבדל מובהק בין KTBoost לNGBoost, אבל כן קיים הבדל מובהק בין NGBoost לבין שאר האחרים. לכן, אנחנו מסיקים ש כי InfiniteBoost וכן RandomForest הם האלגוריתמים המובילים, וNGBoost הוא האלגוריתם הכי פחות מוצלח, בהשוואה שאנחנו ביצענו.

בנוסף, רצינו לבחון גם את זמני האימון של האלגוריתמים השונים. ביצענו מבחנים סטטיסטים על ערכים אלו, וראינו כי קיימת מובהקות סטטיסטית בהבדלים בין האלגוריתמים, כאשר InfiniteBoost הוא המהיר ביותר, ואחריו RandomForest. ניתן לראות, שהאלגוריתמים בעלי הAUC הגבוה ביותר בצורה מובהקת סטטיסטית הם האלגוריתמים בעלי זמני האימון הקצרים ביותר. בנוסף, האלגוריתם בעל הAUC הנמוך ביותר מאלו שהשווינו, הוא גם בעל זמני האימון האיטיים ביותר, בהפרש משמעותי.

בנוסף, כבר מתוצאות הניסויים ניתן לראות כי InfiniteBoost ו- RandomForest הם האלגוריתמים המובילים עבור מירב הדאטאסטים. הגרף שמוצג בחלק פרטי הניסויים, מציג כי InfiniteBoost מוביל על RandomForest במספר מועט של דאטאסטים, אך כמו שכבר ראינו במבחנים הסטטיסטים, אין באמת הבדל מובהק בין האלגוריתמים כאשר מסתכלים על AUC.

הMetaClassifier חוזה מי מהאלגוריתמים הוא האלגוריתם המוביל עבור כל דאטאסט בהינתן Meta-features שחושבו מראש. חישוב הAUC הממוצע של הMetaClassifier הוא 0.713 וה-accuracy הממוצע הוא 0.74, שתי תוצאות הגבוהות מRandom. כלומר, המודל מצליח לחזות בצורה יחסית טובה את האלגוריתם המנצח, ולכן ניתן להגיד שתכונות הדאטסטים משפיעות על האלגוריתם המוביל עבור אותו דאטאסט. ניתן לחזק טענה זו בעזרת הסתכלות על מדדי החשיבות של המודל - מהמדדים ניתן לראות כי התפלגות ההשפעה של הפיצ'רים אינה אחידה. כלומר, קיימים Meta-Features אשר משפיעים יותר מהאחרים על החלטות המודל.

מכאן אנו סוברים, שלמאפייני הדאטאסט יש השפעה על האלגוריתם הטוב ביותר עבור אותו דאטסט. על מנת לבחון השערה זו, יש לבצע ניתוחים נוספים על הפיצ'רים בעלי החשיבות הגבוהה ביותר – לדוגמא, אם היינו יודעים שגודל הדאטסט הוא בעל השפעה היינו מריצים את כל האלגוריתמים על דטסטים קטנים/ גדולים בלבד ובודקים את הAUC של המודלים.

1. ציטוטים:
2. Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test*, *25*(2), 197-227.‏
3. Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.‏
4. Duan, T., Avati, A., Ding, D. Y., Basu, S., Ng, A. Y., & Schuler, A. (2019). Ngboost: Natural gradient boosting for probabilistic prediction. *arXiv preprint arXiv:1910.03225*.‏
5. Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R news*, *2*(3), 18-22.‏
6. Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012, July). How many trees in a random forest?. In *International workshop on machine learning and data mining in pattern recognition* (pp. 154-168). Springer, Berlin, Heidelberg.‏
7. Rogozhnikov, A., & Likhomanenko, T. (2017). InfiniteBoost: building infinite ensembles with gradient descent. *arXiv preprint arXiv:1706.01109*.‏
8. Sigrist, F. (2019). KTBoost: Combined Kernel and Tree Boosting. *arXiv preprint arXiv:1902.03999*.‏