קבוצה 35
אוניברסיטת תל אביב – הנדסת תעו"נ
מבוא ללמידת מכונה
יוני 2020

פרויקט בקורס מבוא ללמידת מכונה

הנדסת תעשייה וניהול שנה ג'

קבוצה 35

יובל אפל (3115456699), יותם דרעי (311551170) ורפאל שוכהנדלר (3114022000)

קבוצה 35
אוניברסיטת תל אביב – הנדסת תעו"נ
מבוא ללמידת מכונה
יוני 2020

**פרויקט במבוא ללמידת מכונה – תחזית בינארית**

**תקציר מנהלים:**

פרויקט זה עוסק בבעיית קלסיפיקציה בינארית באמצעות סט נתונים המכיל כעשרים אלף רשומות ועשרים וחמישה פיצ'רים. תחילה חקרנו את מבנה הנתונים, תוך מתן דגש על חקירת סוג הנתונים, התפלגותם של הנתונים המספריים והמרת הנתונים הקטגוריאליים למספרים בכדי להגיע לתחזיות אופטימליות, באמצעות שימוש בויזואליזציה עשירה. לאחר מכן פנינו לביצוע עיבוד מקדים בכדי להכין את הדאטה לקראת תהליך הלמידה, במהלכו מילאנו תאים ריקים, והסרנו נתונים חריגים בכדי לשפר את מודלי החיזוי. בתום תהליך ה – "Preprocessing", הרצנו מספר סוגים שונים של מודלים המשמשים לחיזוי תוך בחינה ולקיחה של הפרמטרים האופטימליים עבור כל מודל. לאחר קבלת התוצאות ביצענו השוואה בין ביצועי המודלים בעזרת גרף ROC ומדד AUC. בסוף התהליך ביצענו חיזוי עם המודל שנתן את התחזית הטובה ביותר עבור סט האימון (train) על סט המבחן (test) וההסתברות לסיווג נתונות בקובץ נפרד.

**מבוא:**

- נייבא את כל הפקודות והחבילות הדרושות.
- נמדוד את זמן תחילת ריצת המודל, על מנת לחשב כי אכן משך הריצה היה לא יותר מאשר שעה.
- נקרא את קבצי ה train וה test באמצעות פקודת read_csv. את קובץ ה train נחלק ל x,y . כאשר x הוא הפיצ'רים ו y הוא ה label כפי שלמדנו במהלך הקורס. בקובץ ה test אין label ולכן נקרא אותו כמו שהוא ישר אל המשתנה test_x.

**חלק א':**

- נציג pie chart על מנת לקבל מושג ראשוני כמה רשומות יש מכל סוג (כלומר עם label 0 או 1).
- נשנה את הקונפיגרציה display_max_columns של pandas כדי שנוכל להציג את כל העמודות.
- נריץ את הפקודה ('np.warnings.filterwarnings('ignore . בגלל שיש ערכים חסרים ב data, לפעמים קופצת אזהרה. (למשל, כשמנסים למצוא מקסימום/מינימום כאשר יש ערך חסר). פקודה זו תכבה את האזהרות הללו.
- נדפיס את ה head, tail כדי לקבל מושג לגבי איך הdata נראה.
- נדפיס את סוגי המשתנים של כל עמודה (dtypes) כדי לדעת מה הם (וכך נוכל לקבל מושג ראשוני לגבי מה מהם קטגוריאלים, מספריים וכו').
- נדפיס את פלט הפקודה describe אשר מאפשרת לנו לראות נתונים סטטיסטיים על כל עמודה ב dataframe: מספר הערכים (הלא ריקים) בכל עמודה, תוחלת, סטיית תקן, ערך מינימום, רבעון תחתון, חציון, רבעון עליון, וערך מקסימום. נדפיס את פלט פקודה זו בצורת

Transform כדי להקל על הקריאה ועל ההבנה שלה (כעת כל עמודה מתוארת ע"י שורה בטבלה).

- נדפיס את פקודת describe עבור העמודות המספריות והקטגוריאליות בנפרד, כיוון שעבור משתנים מספריים ניתן להדפיס סוג שונה של נתונים לעומת עבור משתנים קטגוריאלים (שם נספרים הערכים הייחודים, מוחזר הערך שמופיע הכי הרבה פעמים וכו').
- הערה: עמודה 14 מופיעה בתור משתנה קטגוריאלי אף על פי שהיא משתנה מספרי, בגלל התווים 'mm' שמופיעות עבור הערכים בה. בחרנו לא 'לנקות' אותה מתווים אלו בשלב זה, כיוון שלהבנתנו שלב זה יגיע ב preprocessing.
- נייצר heatmap של dataframe מסוג corr ל data המקורי. כלומר, נציג את הקשר (Correspondence) בין כל פיצ'ר בטבלה, ובין כל פיצ'ר אל ה label. נשים לב כי השתמשנו אך ורק בעמודות בעלות נתונים מספריים. כמו כן, נשים לב כי השתמשנו בdataframe השלם של train (ולא רק ל-x), כדי לקבל מושג לגבי הקשר ל label (אשר לא קיים ב train_x).

- נציג היסטוגרמה של כל פיצ'ר. כך נראה מה השכיחות של כל ערך/טווח ערכים. נשים לב כי יש יישום שונה עבור משתנים מספריים ועבור משתנים קטגוריאליים. (בנוסף, עבור פיצ'רים 14 ו22 היה צורך ב"טיפול מיוחד", כדי שה xticks – הערכים לאורך ציר הx יהיו קריאים ולא צפופים מדי).

**<u>חלק ב'</u>**

- המימדיות של הבעיה אכן נראית גדולה מדי. מימדיות גדולה מדי יוצרת רעש, אשר מפריע למודלי החיזוי הסטטיסטיים (דוגמה היא ה curse of dimensionality). ניתן לזהות מימדיות גדולה מדי על ידי שימוש בבדיקות וטכניקות – למשל, ניתן לבדוק את ה corr, קורלציה בין פיצ'רים (כאשר היא גבוהה וקרובה ל1, ניתן להבין כי אפשר לצמצם פיצ'רים אלו לפיצ'ר אחד), דרך נוספת היא על -ידי PCA אשר באמצעותה אנו בוחרים את מספר הפיצ'רים המינימלי באמצעותו אנו מסבירים 95% מהשונות של הדאטה המקורי. בכך למעשה אנו מורידים את המימדיות של הבעיה המקורית.
- לאחר בחינת הנתונים עלה כי עמודה 14 מכילה ערכים מספריים המלווים ביחידות מידה, מה שהפך אותה לאובייקט שאינו כמותי ופגע ביכולת לבצע ניתוחים מספריים, ולכן מחקנו את התוספת של יחידות המידה "mm" (עבור שני סטי הנתונים).
- בדקנו את כמות התאים הריקים בנתונים, והחלטנו כי כל מאפיין אשר כמות התאים הריקים בו עולה על 50% ימחק, אך לאחר בדיקה גילינו כי אין מאפיינים שעומדים בתנאי זה. עבור המאפיינים הכמותיים בחרנו לבצע השמה של הערך החציוני של כל מאפיין ולעומת זאת עבור המאפיינים הקטגוריאליים בחרנו להשים את הערך השכיח ביותר.
- ביצענו הסרת חריגים עבור המאפיינים הכמותיים באמצעות שיטת zscore ו-3 סטיות תקן, כלומר כל רשומה שהכילה ערך שחורג ב-3 ס"ת או יותר מהערך הממוצע של העמודה, הוסרה. את הסרה זו ביצענו אך ורק על סט ה train, מכיוון שעבור סט ה test נצטרך לחזות גם עבור רשומות אשר יש בהן נתונים חריגים. כאשר ניקינו שורות מסט ה x של ה train, ניקינו את אותן שורות גם מסט ה y (בעזרת mask) – כדי לשמר את הקשר בין train_x, train_y.

- ביצענו סטנדרטיזציה של הנתונים (עבור העמודות המספריות) כך שנתוני כל העמודות המספריות יהיו בעלי ממוצע 0 וס"ת 1, מה שמצמצם את השפעת ערכי העמודה על השגיאה הסופית על המודל. בנוסף, המרנו את העמודות הקטגוריאליות למספרים באמצעות אינדיקטורים. את פעולות אלו ביצענו על שני סטי הנתונים.
- גילינו כי עמודה 'a25_6' קיימת בTest אך אינה קיימת ב Train (לאחר הוצאת החריגים מהtrain אין ערך שמגיע לעמודה זו), ולכן מחקנו את עמודה זו גם מה Test, כדי לשמור על מבנה אחיד בשני סטי הנתונים (לא מחקנו שורות כך שלא איבדנו רשומות מסט ה test, כנדרש).
- על מנת להקטין את מימדיות הבעיה, החלטנו להשתמש בטכניקת PCA. בחרנו לקחת את הפיצ'רים אשר מסבירים 95% מהשונות, ומצאנו כי עלינו להשתמש ב53 פיצ'רים.
- יצרנו 3 עמודות חדשות – 3 פיצ'רים חדשים – כל פיצ'ר הוא מכפלה של 2 פיצ'רים אחרים.

**<u>חלק ג'</u>**

- בחלק זה בחרנו מודלים והחלנו אותם על סט ה train. המודלים שבחרנו הם:
  1. GaussianNB
  2. KNeighborsClassifier
  3. RandomForestClassifier
  4. MLPClassifier
- עבור 2 המודלים המתקדמים יותר (MLPClassifier , RandomForestClassifier) השתמשנו בשיטת grid_search, כדי למצוא את הפרמטרים הטובים ביותר עבור המודל. סיפקנו לכל מודל פרמטרים אפשריים (עליהם תבוצע אופטימיזציה לפי סט ה train), תוך מחשבה על ה tradeoff בין דיוק רב יותר (ע"י שימוש במספר רב של פרמטרים) לבין זמן ריצת הקוד. לבסוף, השתמשנו בכל מודל עם ה best_params, כלומר עם הפרמטרים הטובים ביותר שמצא ה grid_search.
  הפרמטרים הטובים ביותר שהותאמו עבור מודל 3 הם:
  {max_depth': 6, 'n_estimators': 300, 'random_state': 0'}
  הפרמטרים הטובים ביותר שהותאמו עבור מודל 4 הם:
  activation': 'logistic', 'batch_size': 50, 'hidden_layer_sizes': (50, 50), 'learning_rate_init': '} {0.1, 'max_iter': 1500}

- השתמשנו בשיטת Kfold cross validation להערכת כל מודל, כאשר עבור כל מודל הדפסנו עקומת ROC. בחרנו במודל עם מדד AUC ROC הגבוה ביותר – מודל MLPClassifier. (ערך המדד היה 0.8654).

**חלק ד'**

כעת, ביצענו הערכת מודלים:

- עבור כל מודל, קיבלנו את מדד ה AUC שלו מפונקציית KFoldPlot.
- חישבנו עבור כל מודל מדד Weighted Accuracy, כאשר בהתאם להנחיות משקלנו טעויות FN כבעלות משקל פי 5 מאשר טעויות FP.
- עבור המודל הנבחר, MLPClassifier, הדפסנו confusion_matrix. בגרף זה מוצגות כמות ה FN,FP, TP,TN שפרידקציה זו ביצעה. הסבר על משמעות כל תא:

| סימון | ערך חזוי | ערך אמת |
|---|---|---|
| TP | 1 | 1 |
| TN | 0 | 0 |
| FP | 1 | 0 |
| FN | 0 | 1 |

נשאף למקסם את ערכי TN,TP ולצמצם את ערכי הטעויות.

- עבור כל מודל מבין הארבעה, בדקנו האם הוא Overfitted, כלומר האם הפרדיקציה עבור הtrain מספקת תוצאות טובות באופן מובהק בהשוואה לסט ה test.
מהבדיקה שערכנו עלה כי מודל 2 (KNeighborsClassifier) בלבד הוא Overfitted.

**חלק ה':**

שמרנו את הפרדיקציה מתוך המודל שבחרנו וייצאנו את התוצאות לקובץ csv.

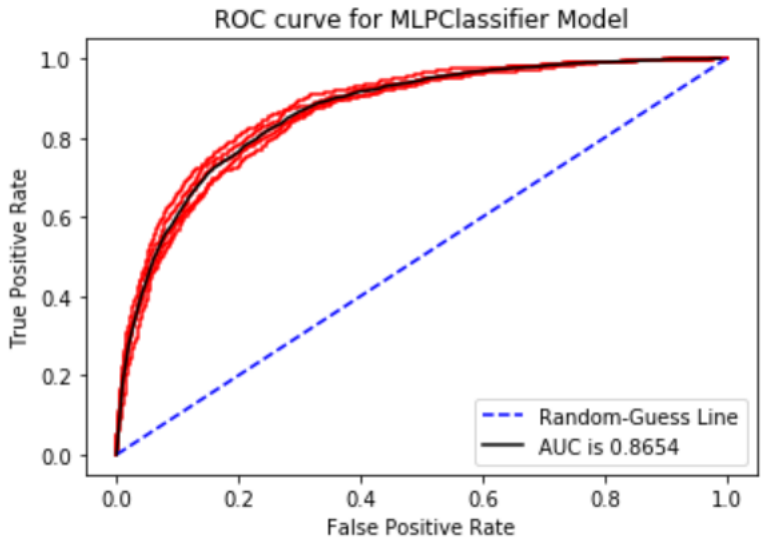
**סיכום:**

בתחילתו של התהליך קיבלנו שני סטים של נתונים – קובץ אחד לאימון המודלים, והשני ליצירת החיזויים (שניהם מסוג csv). פיצלנו את סט האימון לtrain וvalidation כדי להבין את איכות התחזיות במהלך בדיקת ביצועי המודלים. חקרנו את סט האימון בכדי להבין את סוג הנתונים, את התפלגותם ואת הקשרים בין הפיצ'רים. המשכנו בניקוי סט הנתונים והשלמת ערכים חסרים בהתאמה למסקנות אשר עלו מהחקירה ולאחר מכן נרמלנו את העמודות הנומריות וביצענו הורדת מימדיות באמצעות אלגוריתם PCA. לאחר השארת הפיצ'רים המסבירים 95% מהשונות, הוספנו שלושה פיצ'רים אשר חושבו על סמך הפיצ'רים שהושארו.
מכאן פנינו לבחינת מספר סוגים של מודלי חיזוי במטרה למצוא את האחד אשר יחזה בצורה אופטימלית את הסיווג של הדאטה בקובץ המבחן (test) . בדקנו את ביצועי החיזוי של שני מודלים פשוטים, ושניים מורכבים יותר. המודלים הפשוטים הינם KNN ו- Gaussian Naïve

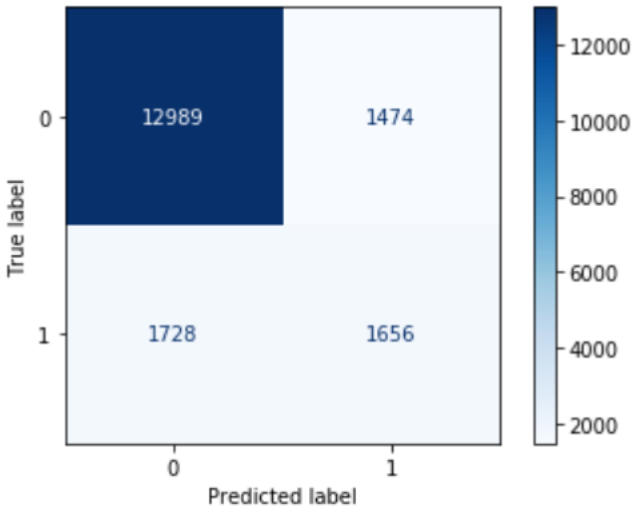
Base, והמודלים המורכבים הינם Random forest ו - MLP Classifier. במהלך ניתוח התוצאות עלה כי המודל היחיד אשר עומד בתנאים שהגדרנו עבור overfitting הינו KNN, ובנוסף זהו המודל שהניב את התוצאות הנמוכות ביותר עבור נתוני Validation על פי מדד AUC. נתון זה עולה בקנה אחד עם הטענה כי Overfitting הינה תופעה שלילית בלמידת מכונה, כיוון שהמודל לומד בצורה טובה "מדי" את סט האימון, ובכך פוגם ביכולתו לחזות את הסיווג עבור רשומות חדשות. המודל שהניב את התוצאות הטובות ביותר הינו MLPClassifier, אשר משתמש ברשת נוירונים במבנה האופטימלי שמצאנו באמצעות פונקציית GridSearch המבצעת אופטימיזציה על ההיפר-פרמטרים של המודל. בעזרת מודל זה חזינו תוצאות (Label) עבור סט ה test, ואותם שמרנו לקובץ submission.csv.

**נספחים:**

גרף ROC עבור המודל אותו בחרנו:




Confusion Matrix עבור המודל אותו בחרנו:




טבלה בה הצגנו עבור כל מודל האם הוא Overfitted:

| | Classifier | train-auc_value | validation-auc_value | difference | overfit |
|---|---|---|---|---|---|
| 0 | GaussianNB | 0.818549 | 0.814205 | 0.004343 | False |
| 1 | KNeighborsClassifier | 0.932750 | 0.797848 | 0.134903 | True |
| 2 | RandomForestClassifier | 0.891524 | 0.851063 | 0.040461 | False |
| 3 | MLPClassifier | 0.891546 | 0.862061 | 0.029486 | False |