כריית מידע וייצוג מידע - חלק 2

מגישים:

אראל דקל - 326064888

בועז גורביץ' - 325813970

shutterstock.com · 1484974583

תיקון חלק א'

בחלק זה של הפרויקט המטרה היא ליצור מודל שיכול לסווג את המידע ולנחש בדיוק גבוהה האם סטארטאפ יצליח או יכשל, אך ראשית נחזור לחלק הראשון של הפרויקט ונתקן בעיות שנוצרו על מנת שבחלק זה המידע שלנו יהיה תקין.

היו לנו מספר בעיות בחלק הראשון וקודם כל נשנה את קובץ ה preprocessing ונסביר להלן את כל השינויים שביצענו ואת הסיבה לכך.

לא הורדנו את ה attributes: city, state_code על מנת שנוכל לדעת בדיוק באיזה עיר ומדינה אנו נמצאים.
את הגרפים של ההתלפגויות של כל ה attributes הצגנו במקום כkde כhistplot בשביל שהגרפים של הערכים הבדידים יהיו ברורים.

את ערכי הגלאים השלילים לא הורדנו מכיוון שיכול להיות שהם מייצגים דברים שקרו לפני שהסטארטאפ רשמית הוקם. בנוסף לא שינינו את עמודה roundA מכיוון שהשינוי שלה הורס את העמודה.

הורדנו ערכים חרגים רק עבור הtotal_founding מכיוון שרק עבורו באמת היה אפשר לראות סוג של sגרף דלתא והיה ברור שיש שם ערך חריג. במקום להוריד ערכים בכל attribute.

עשינו בדיקות שהערכים של המידע אכן תקינים: שהגיל של דברים שהוגדרו כfirst אכן קטנים מגלאים של דברים שהוגדרו כlast, שהתאריך שנשארו "foundation_date" אכן ערכים של תאריכים תקינים, ועוד בדיקות שהיו כבר לפני השינויים ושורות שהיו חריגות מחקנו.

כעת עשינו בדיקה עבור עריכים שהם abnormal ע"י בדיקת ערכים שמרחקים מהממוצע גדול מ std*5 ראינו שמדובר ב14 שורות ולכן זה שרירותי יחסית לכמות המידע ולכן מחקנו אותם.

כעת עברו ערכי ה milestone_age שהיו חסרים מכיוון שלא היו שום milestones לסטארטאפ, המרנו אותם ל0, ולפחות נוכל ככה לזהות עדיין לאיזה סטארטאפים לא היה milestone.

כעת עבור attributes שהם date/string המרנו אותם לint ולא נרמלנו אותם:
עבור עמודת ה"foundation_date" המרנו אותה למספר הימים שחלפו מאותו הdate לdate המינימלי בסט המידע. עבור שאר העמודות הקטגוריאליות, "category" ,"city" ,"state_code" המרנו אותם למספר סופי ויחיסית קטן של קטגוריות הכי שכיחות ואלא שלא היו בקטגוריות האלה החלפנו לקטגורית "other" ואז המרנו את הattributes האלה לערכים של int. עבור "category" לקחנו את ה8 הכי שכיחים, עבור "state_code" לקחנו את ה10 הכי שכיחים, ועבור "city" לקחנו את ה20 הכי שכיחים.

איזון המידע

בחלק הראשון של הפרויקט שבו עיבדנו את המידע, מחקנו שורות רבות ועמודות רבות, יצא מצב שאיבדנו מידע ויצרנו חוסר איזון ביחס בין סטארטאפים שהצליחו לבין כאלה שלא הצליחו. חוסר איזון שכזה יכול לגרום להטייה של המודלים שנלמד. לדוגמא, מודלים קונבנציונלים נוטים לסווג את המידע לקבוצת הרוב וכך לפספס מידע שצריך להיות מסווג למיעוט. לשם כך נשתמש באלגוריתמים אשר יתקנו את הבעיה הזאת.

במקרה שלנו החלטנו להפעיל 2 אלגוריתמים. תחילה התחלנו עם יחס של 36%-64% שזהו יחס לא מאוזן. הפעלנו את אלגוריתם ADASYN בשביל לעשות over-sampling ולאחר מכן השתמשנו באלגוריתם Tomek-Link אשר עוזר למזער את הרעשים שיש לנו בדאטה. זאת, בדומה למה שראינו בהרצאה (SMOTE & Tomek-Link). לאחר איזון המידע בדרך זו הגענו ליחס של 45%-55%.

המוטיבציה להשתמש ב-Tomek Link הוא שאם יש 2 נקודות מקלאסים שונים כך שהנקודות מאוד קרובות יכול ליצור עיוותים בקו ההחלטות ובכך להוריד את הביצועים של המסווג. כלומר, במילים פשוטות אלגוריתם זה מבצע "ניקוי רעשים" לדאטה שלנו. אלגוריתם ADASYN עוזר ליצור דאטה סינטטי עבור במיעוט שקשה יותר ללמד. השילוב של שני האלגוריתמים מאזן את הדאטה על ידי דאטה סינטטי ולאחר מכן מנקה רעשים וכך מאפשר להגיע לתוצאות טובות יותר במודל.

Decision Tree Classifier

ראשית, בתור המסווג הראשון שלנו בנינו עץ החלטות ללא עדכון פרמטרים וקיבלנו דיוק נמוך על הtest בשל כך שהעץ שנוצר היה overfitting עד כדי כך שהדיוק על הtrain היה 1.

לכן החלטנו ליצור כמה מסווגים חדשים שבכל אחד מהם שנינו את הפרמטר min_samples_leaf וקבענו את ה splitter להיות ע"פ ה 'best', ואת הפרמטר לפיצול הצמתים ע"פ entropy.
הראנו את כל מטריקות הביצועים והסברנו את משמעותם במחברת, בחרנו להסתכל על הaccurecy ועל ה precision מכיוון שלדעתנו הכי חשוב זה שמה שאנחנו מסווגים כמצליח באמת יהיה מצליח.

מתוצאות המסווג ראינו שהדיוק הכי גבוה הוא 80% עבור min_samples_leaf = 31 אך בדקנו זאת עבור train_test_split, ולכן החלטנו לעשות אותה בדיקה עבור kfolds, בתקווה לקבל מודל שיהיה יותר נכון ומדיוק.
כעת קיבלנו שהדיוק היה 78% אך ערך מספר ה min_samples_leaf הניב תוצאה הכי טובה עבור 17, ראינו באמצעות גרף את ההשפעה שלו על הaccurecy וה precision וראינו כי אכן זה הערך הטוב ביותר עבור ה min_samples_leaf.

כעת ניסינו להפעיל bagging_classifier על הdecision_tree_classifier.
נסביר על אלגוריתם הקלסיפיקציה של ה bagging, האלגוריתם לוקח את הdata ומחלק אותו לקבוצות כך שבכל קבוצה יש מידע שונה אבל מידע יכול לחזור על עצמו בקבוצה שונה, הוא מפריד לm קבוצות ומייצר m מודלים, ואז משלב את המודלים למודל אחד ממוצע, ההדבל הוא שהמידע שלנו יכול לחזור על עצמו בקבוצות שונות.
המודל החדש נתן לנו accuracy של 81% וprecision של 86% ולכן הוא היה המודל מסוג ה decision tree classifier הכי טוב שלנו.

Random Forest

לאחר שניסינו למצוא decision tree classifier החלטנו לנסות ליצור random forest classifier אשר מחלק את המידע ומאמן מספר עצי החלטות ולבסוף עושה להן ממוצע, (שונה מ bagging).

ראשית, על הrandom forest עשינו חיפוש RandomizedSearchCV בשביל למצוא פרמטרים מקסימלים עבור העצים. קיבלנו דיוק של 85% ולכן החלטנו שיהיה טוב אם נחזיר את הפרמטרים הכי טובים שה RandomizedSearchCV מצא ונפעיל GridSearchCV, באותה סביבה של פרמטרים.

אכן לאחר שביצענו את ה GridSearchCV על איזור הפרמטרים שהכנסנו לו, מצאנו שהדיוק גדל ל85.5%. מאחר וראינו כי קיבלנו מודל טוב מאוד, החלטנו להסתכל גם לתוך שאר מטריקות הביצועים.

ראינו כי כל מטריקות הביצועים גבוהות מאוד במיוחד הprecision שהגיע ל88%, בנוסף בנינו גרף של trp כתלות ב fpr וראינו שה auc = 0.92, שגם זה מייצג מודל מדויק וטוב, ולכן החלטנו שזה יהיה מודל ה Random Forest הכי טוב שלנו.

XGBoost

אלגוריתם זה הוא אלגוריתם מסוג חיזוק גרדיאנט שכולל ענישה חכמה של עצים, כיווץ פרופורציונלי של צמתי עלים, בחירת מאפיינים אוטומטית ועוד. אלגוריתם זה מועדף עבור צוותים מנצחים רבים של תחרויות למידת מכונה.

תחילה פיצלנו את המידע שלנו לTrain ו-Test כך שה-test הינו 25% מהדאטה שלנו. זאת עשינו על מנת למדוד את הביצועים שלנו ולזהות overfitting של המסווג.

השתמשנו ב-RandomizedSearchCV כדי למצוא את הפרמטרים האופטימליים עבור XGBClassifier. מדדנו את ההצלחה בעזרת accuracy ועשינו קרוס ולידציה 5.

לאחר הרצה של המודלים ובחירת המודל הטוב ביותר הרצנו את המודל על הTest כדי שנוכל להעריך את ההצלחה של המודל על מידע שלא ראה. קיבלנו דיוק של 84%. לא הסתפקנו בתוצאה זו והחלטנו למצוא את ההיפר-פרמטרים בצורה שונה.

השתמשנו באלגוריתם Bayesian Optimization על מנת למצוא את ההיפר פרמטרים האופטימליים עבור המודל. אלגוריתם משתמש במשפט בייס כדי למצוא את אופטימום גלובלי בצורה אפקטיבית ויעילה. האלגוריתם יוצר מודל הסתברותי של פונקצית ה-objective שמקבלת סט של היפר-פרמטרים, יוצרת מודל (במקרה שלנו XGBoostClassifier) עם הפרמטרים האלה, מאמנת את המודל, מחשבת את הדיוק ומחזירה את הנגדי של הציון הדיוק (שעלינו להקטין במהלך האופטימיזציה). את המינימום אנחנו מחפשים בעזרת אלגוריתם ששמו Tree-structured Parzen Estimator.

לאחר הרצת אלגוריתם זה למציאת היפר-פרמטרים קיבלנו מודל אשר מחזיר לנו דיוק של 89%! זוהי תוצאה מאוד טובה ולכן החלטנו להסתכל על כל הפרמטרים שמאפיינים את המודל. מצאנו את עקומת ה-ROC, וראינו את FPR, TPR, precision, f1_score.

לבסוף בדקנו את המודל שמצאנו כאשר אימנו אותו על data מאוזן ועשינו predict על data לא מאוזן, וראינו שהדיוק עדיין גבוה.

Logistic Regression

רגרסיה לוגיסטית היא מודל סטטיסטי המתאר קשר בין משתנה איכותי/קטגורי לבין משתנים אחרים. שימוש נפוץ במודל הוא כאשר המשתנה המוסבר הוא בינארי (במקרה שלנו אם העסק הצליח או לא). לכן החלטנו לבדוק האם מודל זה יוכל להניב תוצאות טובות עבורנו.

תחילה הרצנו את הרגרסיה הלוגיסטית עם קרוס ולידציה 5 וקיבלנו דיוק של 70.5%. זוהי תוצאה לא כל כך טובה לכן עלינו לבצע אופטימיזציה למודל ולמצוא את ההיפר פרמטרים שיביאו את המודל ליכולתו האופטימלית.

בדומה למודל ה-XGBoost חילקנו את הדאטה שלנו ל-train ו-test כדי לבחון את המודל שלנו בסוף ריצתו. השתמשנו שוב באלגוריתם Bayesian Optimization על מנת למצוא את ההיפר-פרמטרים האופטימליים, כאשר האלגוריתם שמוצא את המינימום של פונקציית ה-objective היא Tree-structured Parzen Estimator. בצורה זו הצלחנו להגיע לדיוק של 78% שזהו שיפור משמעותי מהמודל הבסיסי.

K-Nearest Neighbors

בהנתן קלט של דוגמא חדשה, האלגוריתם משייך אותה לקבוצה. הדוגמה משויכת למחלקה הנפוצה ביותר בקרב k השכנים הקרובים ביותר. אלגוריתם זה הוא בסיסי ויכול לשמש לרגרסיה או לסיווגים (כמו במקרה שלנו).

תחילה, כמו בהרבה מודלים חילקנו את הדאטה לtrain ו-test כדי למדוד את הביצועים שלנו בצורה נוחה. לאחר מכן, בדקנו ערכי K שונים עבור מטריקות שונות וראינו כיצד עבור מטריקה מסוימת ה-K משפיעה על רמת הדיוק של המודל. הצגנו זאת באמצעות הצגת גרף לכל מטריקה. קיבלנו דיוקים לא כל כך טובים.

אחר כך בדקנו עבור מטריקת מנהטן כיצד משפיע leaf_size ו-K. במודל זה קיבלנו מודל עם 70.4% דיוק שזהו שיפור משמעותי ממה שקיבלנו לפני כן, אך עדיין לא מספיק או מתקרב למודלים אחרים שהצגנו לאורך הפרויקט.

Prediction

לבסוף בחרנו את המודל הכי טוב שיצא לנו - best_XGbooster להיות המודל והעלנו את החיזוי של הtest לקובץ בשם Target.csv כאשר מדובר בעמודה של 0 ו 1, והשורה הראשונה ריקה.