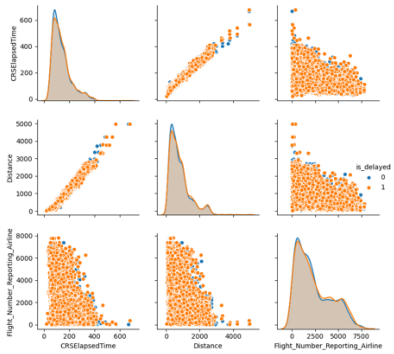
Introduction to Machine Learning (67577 – Hackaton Project

א. רקע:

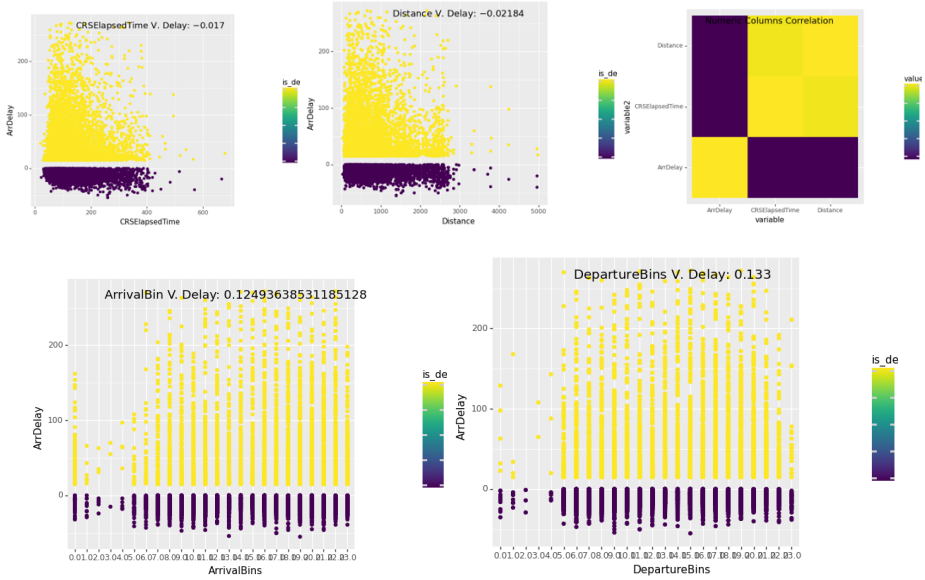
במשימה זו קיבלנו טבלת נתונים על טיסות בארה"ב בשנים האחרונות. הטבלה מכילה 539,878 דגימות כאשר לכל דגימה 17 פיצ'רים שונים, חלקם דיסקרטיים וחלקם רציפים. בנוסף, קיבלנו מידע על מזג האוויר שנמדד בתחנות שונות ברחבי ארה"ב. מיזגנו את המידע הזה עם זה שלנו על פי התאריך והתחנה בו התבצע המדידה. הוספנו רק את הפיצ'רים שחשבנו שישפרו את הניבוי – טמפרטורה מקסימלית, כמות משקעים ורוח ממוצעת.

ב. אבחון ה-data:

תחילה, התחלנו לחקור כל תכונה ותכונה (feature and feature). יצרנו Scatterplot Matrix המבצעת רגרסיה לינארית על כל שני פיצ'רים של ה-data כדי להסיק על קשרים שונים ביניהם.



קל לראות כי יש קורלציה גבוהה מאוד (כמעט 1) בין מרחק הטיסה לאורכה. כלומר שני הפיצ'רים יתרמו לניבוי בצורה זהה ולכן מספיק להשתמש רק באחד מהם. בנוסף, קיימת קורלציה שלילית בין פיצ'רים אלו למשך האיחור, ככל שהטיסה ארוכה יותר נראה שיש פחות איחורים, והאיחורים הקיימים קטנים יותר. בהמשך:



בהסתמך על גרפים אלו (ושהיו לפניהם, אלו התוצאות הסופיות כמובן) הבנו מה היחסים בין כל feature ל-feature, וכיצד כדאי לבצע עיבוד מקדים אופטימלי.

ג. ביצוע עיבוד מקדים:

בשלב זה התחלנו לבצע pre-processing. נשים לב שישנו כמובן דמיון בין ה-train לבין הנתונים שנקבל ב-test "הלא ידוע", לכן גם אלו וגם אלו זקוקים לעבור הליך עיבוד מקדים דומה. ברם, יש הבדלים קטנים בין הנתונים שאותם לקחנו בחשבון. כך, יצרנו מסלול (pipeline) כדלקמן: עבור ה-train data: הסרת ערכים שחשיבותם לא תורמת הרבה למידע. פקטוריזציה של שדה ה-DelayFactor כדי להבחין בסיבה וללמד את המודל ושימוש ב-pipeline הכללי. עבור ה-test data: שימוש ב-pipeline הכללי וסינכרון שדות חסרים עם ה-train data.

עתה, באשר ל-pipeline העיקרי, הוא כולל את המשימות הבאות: א) תיקון פורמט התאריך שהתקבל לפורמט אחיד. ב) איחוד נתוני מזג האוויר, אם קיימים. ג) הוספת מידע על חופשות בארה"ב (הבחנו שמדובר במדד שתורם לחיווי). ד) יצירת dummies. ה) חלוקת זמני הטיסה לסלים (bins) של שעתיים, דבר שסייע גם כן לחיווי. ו) יצירת תכונה המבררת האם הטיסה לוקאלית (דהיינו, מאותה המדינה). ז) הסרת תכונות לא נחוצות.

בחירת פי'צרים: בבחירת הפיצ'רים, חשבנו קודם אילו פיצ'רים ניתן להוריד. החלטנו להוריד הפיצ'רים כדוגמת הקשורים לשדה ההמראה והנחיתה המרחיבים מעבר לשם השדה עצמו (העיר והמדינה בה השדה נמצא). הורדנו את מספר הזנב של המטוסים כיוון שהדרך היחידה בה פיצ'ר זה מוסיף מידע היא על ידי הפיכתו ל dummy values. לאחר בדיקה, שמנו לב כי מדובר במעל 8,000 מספרים שונים ולכן החלטנו להוריד פיצ'ר זה.

לאחר מכן, הפכנו חלק מהפיצ'רים לקטגוריאליים. פעולה זו אינטואיטיבית מאוד עבור פיצ'רים כמו חברת התעופה, שדה המראה ושדה נחיתה. כפי שניתן לראות מתיעוד ה-pipeline שלעיל, הוספנו לדאטא שלנו עוד שני פיצ'רים בינאריים. הראשון מתאר האם יום הטיסה הוא יום חופש בארה"ב. ביום כזה נצפה לעומסי טיסות ומעט עובדים ולכן לעיכובים רבים. הפיצ'ר השני מציין האם שדה ההמראה ושדה הנחיתה נמצאים באותה מדינה. יכול להיות שיש יותר בירוקרטיה בטיסות בין מדינות ועל כן איחורים רבים יותר

ד. החיווי ושמירת הנתונים:

בשלב זה השתמשנו ב-LassoCV כדי לבצע רגרסיה ולהשיג את החיווי עבור ה-ArrDelay. נציין שאחר כך עשינו שימוש עם PolynomialFeatures דבר שעבד הרבה יותר טוב, אך מפאת זמן לא הספקנו להטמיע את שיטה זו– ובחרנו להשאיר את הקוד הרלבנטי תחת do_train_. ניסינו כמובן גם שיטות אחרות, שהיו פחות אופטימליות.

בהמשך, בשלב ה-classification, השתמשנו ב-OneVsRestClassifier כדי להגיע לסיווג של סיבת האיחור. בחנו שיטות אחרות, כמו RandomForestClassifier אך קיבלנו ב-OneVsRestClassifier את התוצאה הטובה ביותר. לבסוף, עשינו שימוש ב-joblib כדי "להקפיא את המודל" ולשמור אותו (פרמטר דחיסה של 9).

ה. שלב החיווי:

שלב החיווי אינטואיטיבי מאוד. בתחילה, מעבירים את הנתונים ב-test data pipeline שצויין בפרק ב' לעיל. משהנתונים עברו pre-process אפשר להשתמש במודלים "שהוקפאו" בשלב הקודם, כדי לקבל את החיווי.