אגוֹדה - משימה תיאור כללי של הפתרון

כאשר ניגשנו למשימה הבנו שנוכל להשתמש במגוון הספריות הזמינות למתכנתי פייתון כיום, ביניהן ספריות כגון sklearn והסקנו שמוטב יהיה להשקיע את מירב מאמצינו בניקוי, סידור והנדוס הדאטה אשר עמד לרשותו ולא להשקיע בניסיון לתכנן ולממש מודל למידה חדשני.
כאשר ניגשנו למשימת הינדוס הדאטה, התחלנו בניסיון להבין את הפיצ'רים השונים, את המשמעות של כל פיצ'ר וניסינו לשער לגבי כל פיצ'ר האם הוא יכול להיות גורם משמעותי בחיזוי הפיצ'רים אותם רצינו לחזות (שיערנו למשל שמדיניות הביטולים של ההזמנה עשויה להשפיע על האם ההזמנה תבוטל או לא, אך הפיצ'ר המתאר את מדיניות הביטולים היה מסובך ודרש הינדוס, אך כך נרחיב מאוחר יותר).
לאחר שחשבנו ביחד וכתבנו את השערותינו, ניגשנו לבחון אותן ולראות אם פיספסנו פיצ'רים אחרים שעשויים להיות משמעותיים, לשם כך חישבנו קורולציה בין כל פיצ'ר ופיצ'ר שאותו רצינו לשערך (אך לא לפני שהמרנו את העמודה cancellation_datetime שהכילה תאריכים של ביטולים לעמודה המכילה ערכים בוליאניים המייצגים ביטול \ לא ביטול).
לאחר מכן בחרנו מספר מודלים ובחנו את הביצועים שלהם על ה train_set החדש באמצעות cross_validation והמטריקה f1_score, אך לאחר שהגענו לתוצאות מעט מאכזבות, הבנו שנדרשת עבודת הינדוס יותר משמעותית ולא נוכל להסתמך על מספר פיצ'רים בודדים ונצטרך להנדס פיצ'רים חדשים.

<u>הסבר על ה preprocessing:</u>
קידוד משתנים קטגוריים:
העמודות: language, hotel_country_code, accommadation_type_name, origin_country_code, guest_nationality_country_name, customer_nationality, original_payment_method, original_payment_type ו-original_payment_currency מקודדות למשתנים קטגוריים.

טיפול במשתנים בינאריים:
העמודות: request_nonesmoke, request_airport, request_latecheckin, request_highfloor, request_twinbeds, request_earlycheckin, request_largebed ו-pay_now עוברים טרנספורמציה באמצעות פונקציית היישום עם פונקציית lambda למיפוי ערכים שאינם 0 ו-1 ל-0.

ניתוח ויצירת עמודות חדשות:
night_amount מחושב על ידי הפחתת תאריך יציאה מתאריך צ'ק-out.
מספר עמודות (days_cancellation_1, percentage_cancellation_1, days_cancellation_2, percentage_cancellation_2, ו-no_show_percentage) נוצרות באמצעות חלוקה של העמודה cancellation_policy_code לימים ואחוזי החזרה.
cancellation_policy_code משתנה על ידי חילוץ הערך המספרי.
has_request מחושב על ידי סיכום עמודות הבקשות הבינאריות.

טרנספורמציות נוספות:
distance_booking_checkin מחושב על ידי המרחק בין ההזמנה לצ'ק אין.
guest_amount מחושבת על ידי סיכום מספר המבוגרים והילדים.
hotel_live_date מחושב על ידי הפחתת hotel_live_date מתאריך הצ'ק אין.
עמודות תאריך (תאריך_checkin, checkout_date ו-booking_datetime) משתנות לערכי יום בשנה.
costumer_guest_same_nation נוצר על ידי השוואת השוויון של customer_nationality עם guest_nationality_country_name ו-origin_country_code.

שחרור עמודות:
העמודות: did_cancel, h_customer_id, cancellation_datetime, hotel_brand_code, hotel_chain_code ו-charge_option יוסרו מהדאטא.

תהליך בחירת המודלים

1. במשימה 1 התבקשנו לסווג האם הזמנה תתבטל או לא. בהנחה שסביר שהדאטה שקיבלנו לא מופרד ליניארית נוכל לצמצם את הפתרונות האפשריים למודלים שהם supervised classification learning. למשל:
    a. logistic regression
    b. K-Nearest Neighboors
    c. Random Forests

כשלב ראשון, חיפשנו מודל בסיסי ביותר שייתן לנו אינדיקציה ורפרנס בעתיד
לשם כך בחרנו מודל שכל מה שהוא עושה על הדאטה זה להחזיר וקטור רנדומלי שכל תא בו הוא 1 בהסתברות 0.5 ללא תלות בפיצ'רים.
מודל בסיסי זה קיבל F1=0.34. בהמשך נוכל להסיק שמודל שעושה פחות מזה, יותר גרוע ממודל רנדומי.
בשלב השני,חילקנו את הדאטה ל-80 אחוז אימון ו20 אחוז טסט שאותו אנו שומרים בצד.
בחרנו את המודלים לעיל ועל כל אחד מהם הרצנו Cross Validation עם 5 פולדים על היפרפרמטרים קבועים כדי להבין פחות או יותר את היעילות של כל מודל בצורה שטחית.
משלב זה, הסקנו כי logistic regression וKNN הרבה פחות יעילים במדד הF1 מאשר Random Forests.
להלן תוצאות של GridSearchCV עבור שניהם:

```
GridSearchCV for Logistic Regression- Best Parameters: {'C': 1.0, 'penalty': 'l1', 'solver': 'liblinear'}
GridSearchCV for Logistic Regression - Best F1 Score: 0.4813514576759704
--------------------------------------------
GridSearchCV for KNN- Best Parameters: {'n_neighbors': 3, 'p': 1, 'weights': 'distance'}
GridSearchCV for KNN - Best F1 Score: 0.42978775529390073
```

נראה כי הם יותר טובים ממודל רנדומי אך פחות טוב מ-Random Forest שהצליח להשיג F1=0.61.
בשלב הבא, רצינו לבדוק האם ישנו שיביא ביצועים טובים יותר. לאחר חיפושים באינטרנט מצאנו שני אלגוריתמי אנסמבל על עצים שרצינו לבדוק:

1. lightGBM
2. XGBoost

כדי להבין איך להשתמש במודלים הללו נעזרנו בchatGPT באופן הבא.
ביקשנו ממנו הסבר קצר על כל אחד המודלים ואופן השימוש בהם. לדוגמא ביקשנו גריד של פרמטרים למודל וביצענו תחילה GridSearchCV שלקח המון זמן ולכן עברנו לבדוק עם RandomizedSearchCV. בעזרת הקונפיגורציה שנבחרה עבור כל אחד מהם אימנו את המודל ובדקנו את הF1.

לאחר הרבה הרצות ובדיקות שונות על הולידציה התוצאה הכי טובה שקיבלנו על הטריין הייתה F1=0.85 עם recall מאוד גבוה (שזה מעיד על מעט מאוד FN שבמקרה שלנו קריטים) עבור XGBoost והחלטנו להריץ לראשונה על הטסט ולבדוק.

אלו התוצאות :

```
Train Results:
    best params:  {'subsample': 1.0, 'reg_lambda': 0, 'reg_alpha': 0.5, 'n_estimators': 300, 'max_depth': 7, 'learning_rate': 0.1, 'gamma': 0.2, 'colsample_bytree': 1.0}
    best score:  0.6073765787855273
    F1:  0.8568721160184575
    Precision: 0.8968520914187149
    Recall:  0.8203044884436381
Test Results:
    F1:  0.5852664576802508
    Precision:  0.5669602186456119
    Recall:  0.6047942986718496
```

נשים לב כי F1=0.585 שזה יחסית נמוך למה שציפינו. ייתכן כי יש מעט overfit.
החלטנו בסופו של דבר כי זו תוצאה יחסית טובה וכדי שלא נגיע ל overfit על הטסט בחרנו להגיש את המודל הזה.
במשימה השנייה היינו צריכים להעריך כמה עלתה ההזמנה במקור. זוהי בעיית רגרסיה קלאסית ולכן התחלנו להריץ רגרסיה ליניארית ולהבין מי נגד מי. קיבלנו RMSE= 475.
בדקנו האם רגולריזציה תשפיע (תעניש מודל מסובך) עם מודלים כמו Ridge ו-Lasso שראינו בכיתה.
לאחר מס' בדיקות עם פרמטרים שונים ראינו כי אכן ה-RMSE יורד אך לא בצורה מספקת.
חיפשנו באינטרנט אפשרויות נוספות וניסינו Random Forest Regressor שנתן את התוצאה הטובה ביותר עד כה עם RMSE=382 אז נשארנו איתו.