Assignment 2 - category embedding part 2

העבודה חולקה ל2 מחברות שונות באופן הבא:

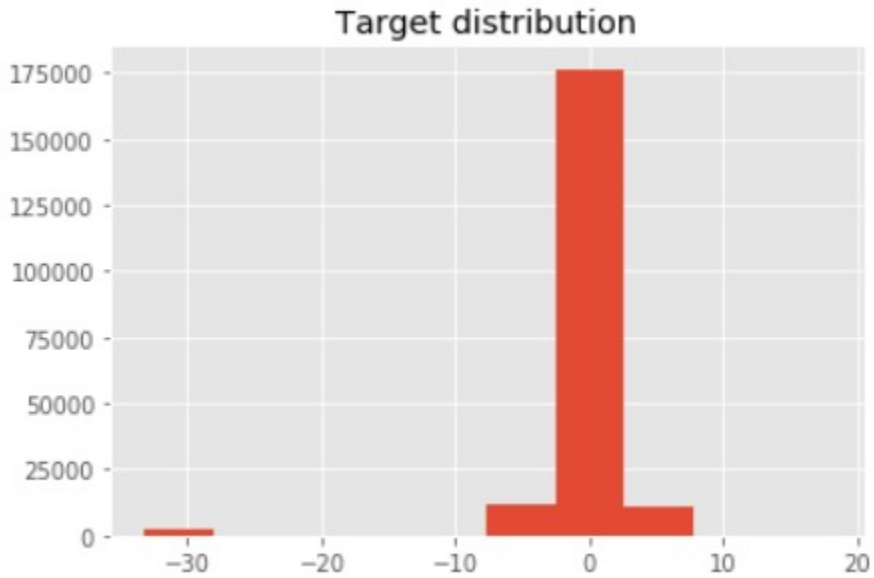
1. https://drive.google.com/open?id=17FP__VNy5umAfqqG-mbZqSZVdZV-lFmD
מחברת המכילה את ההשתתפות בתחרות יחד עם category embedding
2. https://drive.google.com/open?id=1yU87QA84Z_0hhLGjKL5leNEL6DAS5pSv
מחברת המכילה את יצירת הפיצרים החדשים.

Category embedding

חלק זה מהווה קפיצה למים, כאשר אנו נדרשים להשתתף בתחרות ELO בKaggle אשר מטרתה היא ביצוע חיזוי לערך דירוג הנאמנות של הלקוח.

תחילה נרשמנו לתחרות תחת השם **BGU-DL-ELEN** על שמם של הכותבים כמובן **EL**עדן לוי , **EN**אליאור נחמיה. יש לציין שמדובר בתחרות הראשונה שלנו בקאגל שזה צעד קטן לקאגל וצעד גדול עבורנו.

המידע שניתן בתחרות: המידע מחולק ל5 קבצים שונים, כאשר השניים המרכזים הם train test המכילים תאריך פתיחת החשבון מס הכרטיס ושלושה משתנים קטגוריאלים אנונימים , כמו כן ישנו משתנה המטרה בקובץ האימון. בנוסף ישנם שני קבצים אודות טרנזקציות היסטוריות וחדשות בהתאמה המכילות מגוון נתונים לכל טרנזקציה שבוצעה. ואחרון ישנו קובץ סוחרים המכין לידע על החנויות עצמן שנסחרו ע"י המשתמשים.
נתון מעניין על הדגימות הוא שהרוב המוחלט נמצא בטווח של (-7,7) אך כ K2 דגימות נמצא ב-33, ככול הנראה הדגימות הקיצוניות הן מדווחות כfraud ולכן הדירוג שלהן כה נמוך.



ההפרש הגדול בדירוג של דגימה רגילה להונאה עלול לפגוע ביכולת שלנו לבצע חיזוי שכזה וכמו כן הכמות המועטת של דגימות הונאה לא מאפשרת ללמוד כיצד לזהותן (בהמשך ניסינו לבצע זאת ונחלנו הפסד).
כמובן שנעשה עוד חיפוש מעמיק יחד גם עם הדיונים האחרים בקאגל על המידע ושאבנו משם המון ידע ורעיונות לעבודה שלנו.

תחילה בחנו מספר מודלים לרגרסיה על המידע הגולמי בtrain והגענו לbenchmark ראשוני ע"י מודל Lasso לרגרסיה:

```
clf = Lasso(alpha=0.2)
clf.fit(x_train,y_train)

clf_pred = clf.predict(x_test)
rms = sqrt(mean_squared_error(y_test, clf_pred))
print(rms)
```

```
3.8008292845419827
```

ניתן לראות שהמודל מגיע ל3.8 rmse על הטסט הפנימי שלנו. אך כאשר אנו מגישים את הפרדיקציה לkaggle אנו מגלים רק דיוק של 3.940 .
אז נכון מדובר רק בהגשה ראשונית אך יש כאן נקודה חשובה והיא שהוולידציה שלנו אינה איכותית מספיק. כאשר אנו בדקים את הסיבה לכך ישנה סיבה אחת מרכזית אשר משנה את התמונה הגדולה מעט, נזכר שוב בדגימות ההונאה אשר בעלות פרדיקציה מאוד קיצונית ביחס לשאר ולכן נבין שדגימות אלו מקשות עלינו לקבל הערכה נכונה. כמו כן נשים לב שהמודל הנ"ל עדיין פחות טוב מההגשה של אפסים בלבד (0,0). נעזור כוח ונמשיך לשלב הבא.

בשלב זה אנו ננסה ליצור ייצוג emmbedding למשתנים הקטגוריאלים שלנו. כדי לבצע זאת נדרש עיבוד מקדים למידע:

```
data = X_train[:]
F1 = {p:i for (i,p) in enumerate(data['feature_1'].unique())}
F2 = {p:i for (i,p) in enumerate(data['feature_2'].unique())}
F3 = {p:i for (i,p) in enumerate(data['feature_3'].unique())}

processed_data = data.copy()
processed_data['F1'] = [F1[x] for x in data['feature_1']]
processed_data['F2'] = [F2[x] for x in data['feature_2']]
processed_data['F3'] = [F3[x] for x in data['feature_3']]
processed_data = processed_data.loc[:,'F1':'F3']

target = Y_train[:]

F1_inp = Input(shape=(1,),dtype='int64')
F2_inp = Input(shape=(1,),dtype='int64')
F3_inp = Input(shape=(1,),dtype='int64')

F1_emb = Embedding(len(F1),4,input_length=1, embeddings_regularizer=l2(1e-6))(F1_inp)
F2_emb = Embedding(len(F2),2,input_length=1, embeddings_regularizer=l2(1e-6))(F2_inp)
F3_emb = Embedding(len(F3),2,input_length=1, embeddings_regularizer=l2(1e-6))(F3_inp)
```

כאן אנו יוצרים ייצוג חדש לכל אחת מהקטגוריות 1 2 3 בתקווה שנקבל ערך משמעותי יותר שהרשת תוכל להיעזר בה.

```
x = concatenate([F1_emb,F2_emb,F3_emb])

x = Flatten()(x)

x = Dense(64,activation='relu')(x)
x = Dense(128,activation='relu')(x)
x = Dense(64,activation='relu')(x)
x = Dense(1, activation='linear')(x)
nn_model = Model([F1_inp,F2_inp,F3_inp],x)
nn_model.compile(loss = 'mse',optimizer='adam')
```

```
history1 = nn_model.fit([processed_data['F1'],processed_data['F2'],processed_data['F3']],target,validation_split=0.2,epochs=10)
```

הרשת הנל משיגה val_loss - 3.831 :loss: 3.918
סברה שלנו לתוצאה זו היא שהמשתנים הקטגוריאלים שניתנו לנו אינם מספיק מסבירים למשתנה המטרה ולכן האמבדינג אינו מניב ידע חדש.

כעת מתחיל השלב המעניין בתרגיל שבו אנו נדרשים לאסוף פיצרים נוספים משאר הקבצים או מכל מקור ידע אחר. יש לציין שבשלב זה ביצענו בנייה ואגרגציה של משתנים במספר רמות שונות של מקורות מידע וגם של תוכן. בחרנו לעבוד בצורה איטרטיבית כך שבכל שלב ננסה לייצר עוד מידע חיוני למודל. יתרון של שיטה זו הוא שתמיד יש לנו פיתרון ביד ואנו יכולים לשלוט בידע שאנו מוסיפים וגם להבין האם בכלל הידע הנוסף היה נחוץ או לא.

המחברת של יצירת הקבצים מכילה יצירה של 4 קבצים שונים:

- גרסה 1V - בגרסא זו אנו מוסיפים מידע מקובץ הטרנזקציות החדשות ע"י אגריגציה לפי מספר הכרטיס יחד עם סכימה לפי התמונה-

```
agg_func = {

    'purchase_amount': ['sum', 'mean', 'max', 'min', 'std'],
    'installments': ['sum', 'mean', 'max', 'min', 'std'],
    'purchase_month': ['mean', 'max', 'min', 'std'],
    'purchase_date': [np.ptp, 'max', 'min'],
    'month_lag': ['min', 'max']

}
```

גרסה זו מכילה 26 משתנים מסבירים.

- גרסה 2V - בגרסה זו אנו מוסיפים משתנים מקובץ הטרנזקציות ההיסטוריות שוב ע"י אגריגציה לפי מספר הכרטיס יחד עם סכימה לפי התמונה-

```
agg_func = {
    'authorized_flag': ['sum', 'mean'],
    'purchase_amount': ['sum', 'mean', 'max', 'min', 'std'],
    'installments': ['sum', 'mean', 'max', 'min', 'std'],
    'purchase_month': ['mean', 'max', 'min', 'std'],
    'purchase_date': [np.ptp, 'max', 'min'],
    'month_lag': ['min', 'max']
}
```

ניתן לראות כי הפעם הוספנו את אישור התשלום, זאת בגלל שבטבלה זו משתנה זה כן מעיד ולא קבוע על 1 כמו בטבלת הטרנזקציות החדשות. גרסה זו מכילה 28 משתנים מסבירים.

- גרסה 3V - בגרסה זו מיזגנו את שתי הגרסאות הקודמות 2V1 V לקובץ אחד המכיל 50 משתנים מסבירים.
- גרסה 4V - בגרסה זו בחרנו להוסיף לגרסה 3V הקודמת את כל המשתנים הקטגוריאלים שזנחנו עד כה כ one hot וקטור. גרסה זו היא הגדולה ביותר ומכילה 93 משתנים מסבירים.

*באמת שניסינו בכל כוחנו אך לא הצלחנו לבצע אמבדינג יחד עם משתנים נוספים כאינפוט לאותה הרשת.. כנראה שמשהו בהבנה שלנו לביצוע סעיף זה היתה לא טובה. כל צורה שניסינו כשלה ולא קיבלנו תוצאות כלל.

מקווה שההשקעה שלי בתחרות תפצה על הבעיה מקודם..

כפי שכבר ציינו ההתקדמות בתחרות היתה מונעת לפי כמות הידע בכל שלב ונסיונות לאפטם את ניצול הידע בכל שלב לפני המעבר לשלב הבא.

התחלנו עם יצירת הקובץ 1V אשר הכיל אגרגציה יחד עם הטרנזקציות החדשות רק עבור משתנים כמותיים. יצרנו 3 רשתות שונות בעומקים שונים ובדקנו אותן תחת המידע מ1V. עבור רמת הידע הנ"ל יצרנו קפיצת מדרגה ראשונה והשגנו 3.830 rmse בהגשה עצמה בקאגל!

לאחר מכן עברנו למידע 2V אשר מכיל אגרגציה מקובץ טרנזקציות היסטוריות שוב בדקנו על 3 הרשתות וקיבלנו שוב שיפור אך אומנם קטן יותר 3.824 rmse! , ניתן לראות שלא חל שינוי משמעותי כל כך אך הוא דיי הגיוני בהתחשב בכך שהשינוי בין הגרסאות היה רק בכמות הדאטה שהן הכילו ולא במידע נוסף שונה.

כעת רצינו לשלב את המידע של 2V1 V ולכן יצרנו את 3V אשר איחד את שניהם כלומר כעת יש לנו את כל המידע על הטרנזקציות עצמן. תחת 3V הצלחנו להשיג . 3.787 rmse !!
כעת רצינו להוסיף את המידע הקטגוריאלי, ולכן הוספנו ל3V את כל הקטגוריות הנדרשות וכך יצרנו את 4V, הרשתות שלנו השיגו שיפור קטן של 3.778 rmse!! , אך ניתן לראות שהשיטה של one hot היתה לא מספיק אינפורמטיבית כי הוספנו המון פיצרים אך כבר לא חל שינוי גדול בפרדיקציה.
דבר נוסף שניסינו הוא מודל אחר LGB שראינו שהניב תוצאות יפות בפוסטים אחרים. מודל זה הצליח לקבל תוצאה מרשימה מאוד של 3.722 !!! שהיא התוצאה הכי טובה שלנו עד כה. יש לציין שהמודל מבצע בחירת פיצרים עצמאית והוא הורץ על 4V שלנו.

לאור הבעיה המרכזית שגילינו במידע אודות המצאות של דגימות בעייתיות בעלות ערך קרדיט מאוד נמוך (-33) רצינו לנקוט בגישה אחרת לבעיה. תחילה ננסה לזהות ולהפריד בין דגימה רגילה לדגימה בעייתית ע"י יצירת מסווג שזוהי מטרתו זיהוי האנומליות, לאחר החלוקה לאנומליות ולרגילים נוכל לשלוח כל דגימה למודל שיחזה את הערך לאנומליה ולמודל אחר שיזהה ערך לדגימות רגילות.
במהלך בניית המסווג הראשי נתקלנו בבעיה חדשה, הצלחנו להבדיל בין הדגימות ב98% אך נתון זה אינו מספיק כי המאגר אינו מאוזן ולכן קיבלנו את הconfusion matrix :

```
array([[186712,  12998],
       [   702,   1505]])
```

ניתן לראות שאומנם תפסנו 1505 מתוך K2 דגימות בעייתיות זה לא מפצה על ה12998 דגימות רגילות שכעת נחזה עבורן ערך מאוד גבוהה של אנומליה.
ניסינו כל מיני שיטות ליצירת דאטה יותר מאוזן אך במציאות המסווג עדין לא עבד טוב מספיק ולכן ויתרנו על האפשרות הנ"ל.

התחרות היתה מאוד מאתגרת רוב הבעיות שלנו נבעו מחוסר זמן עקב עומס חיצוני בהצלחה בבדיקה.