רוני מינדלין מילר 302242870
מיה קרמר 204219976

# למידה עמוקה עבודה 2- Kaggle competition

a) שם הקבוצה: BGU-DL MayaRonnie

הערה: לא הצגנו פה את ה-data exploration בצורה מפורטת כי עברנו על 2 מחברות מאוד טובות שנמצאות בתחרות שמציגות data exploration מצוין:
https://www.kaggle.com/sudalairajkumar/simple-exploration-notebook-elo
https://www.kaggle.com/chocozzz/simple-data-exploration-with-python-lb-3-764

ה-data עליו אנחנו עובדים מחולק לכמה קבצים:

i. train- קובץ המכיל 201917 רשומות של כרטיסי אשראי שונים (כל card_id מופיע בדיוק פעם אחת בקובץ זה) ולהם יש 3 פיצ'רים מספריים שונים שמשמעותם אינה ידועה. בנוסף נתון לכל כרטיס את תאריך החודש הראשון בו הכרטיס נהיה אקטיבי. העמודה האחרונה היא עמודת הtarget המציינת את ערך ה-*loyalty score של הכרטיס.*
ii. test- מבנה הקובץ זהה לקובץ הtrain מלבד זה שאין לו עמודת target. קובץ זה מכיל 123623 רשומות כך שאין חזרה בין ה-card_id בקובץ ה-train לקובץ ה-test.
iii. historical_transactions- קובץ המכיל נתונים על טרנזקציות שונות שהתבצעו באמצעות הכרטיסים השונים במקומות שונים (כלומר אצל merchants שונים). זהו הקובץ המסיבי ביותר ומכיל 29112361 רשומות. יש פיצ'רים נומריים וקטיגוריאליים ולכן עלינו לבצע preprocess דיי משמעותי.
iv. new_merchant_transactions- קובץ הזהה במבנה שלו לקובץ ה-historical_transactions ומכיל מידע על טרנזקציות חדשות.
v. merchants- קובץ המכיל מידע על הסוחרים עצמם בהם נעשו רכישות ע"י כרטיסי החברה.

הרעיון העיקרי לטיפול בנתונים- נשלוף מידע אודות כל כרטיס מהנתונים ההיסטוריים והחדשים על טרנזקציות ומקובץ ה-merchants שנעשו באמצעות הכרטיס ובכך נבנה "פרופיל" ייחודי המתאים לכל כרטיס. הפיצ'רים שנשלוף יהיו ברובם נומריים למשל ממוצע הטרנזקציות המאושרות (authorized_flag=Y), מספר ה-merchants הייחודים בהם בוצעו רכישות באמצעות הכרטיס, וכו'.
הפיצ'רים הקטגוריאליים יהיו feature_1, feature_2, feature_3 הנתונים בקבצי הtrain וה-test וגם החודש והשנה בהם הכרטיס הפך להיות אקטיבי על פי התאריך הנתון בפיצ'ר first_active_month.
לאחר שליפת נתונים אלו עבור כל כרטיס (card_id) ביצענו merge בין קבצי ה-train וה-test לערכים שקיבלנו על בסיס הנתונים בקבצים historical_transactions ו-new_merchant_transactions ומerchants ובכך קיבלנו פרופיל לכל כרטיס.
בנוסף, הוספנו לנתונים עמודה של merchant_id של הסוחר שאותו card_id ביצע עמו הכי הרבה טרנזקציות ואז מיזגנו בין נתוני ה-merchant (מקובץ ה-merchants) לבין קובץ הtrain וה-test על פי אותו ה-merchant.
כל תהליך הfeature extraction נמצא ב-kernel בשם "BGU_DL_assignmnt2_features_extraction".

קראנו את קבצי הtrain והtest לאחר תהליך ה-feature extraction.
הpreprocess שביצענו:

- מילאנו ערכים חסרים ע"י החלפת כל ערך חסר ב-1- מאחר וחיפשנו ערך שונה מהערכי הנתונים- לאחר התייעצות עם נתי הבנו שעדיף להחליף לערך שלא שכיח בנתונים ולכן החלפנו ל20- ואכן יש שיפור קטן בתוצאות (ניסינו עוד ערכים כמו 15-, 50-, 5- וגם ניסינו לקחת את הערך המינימלי בכל עמודה, להחסיר ממנו 1 ושהוא יהיה הערך שבאמצעותו נמלא ערכים חסרים אך עם 20- קיבלנו את התוצאות הטובות ביותר עבור המודל הסופי שיצרנו- הembedding עם שאר הערכים)
- נרמלנו את הערכים המספריים ע"י min-max normalization.

רוני מינדלין מילר 302242870
מיה קרמר 204219976

- לכל פיצ'ר קטיגוריאלי החלפנו כל ערך אפשרי עבור אותו פיצ'ר בערך מספרי (השתמשנו לשם כך בlabel encoder)

b) המודל שבו השתמשנו להגדרת הbenchmark הוא: LGBMRegressor והגענו באמצעותו לתוצאה:

| Fill missing values | Test RMSE | Train RMSE |
|---|---|---|
| -1 | 3.595 | 3.347 |
| -20 | 3.593 | 3.358 |

ידוע לנו שבמודל LGBM אין צורך בהשלמת ערכים חסרים\נרמול אך בכל זאת בדקנו את התוצאה על נתונים לאחר השלמת ערכים חסרים ונרמול על מנת לגבש benchmark איכותי עבור הנתונים שנכניס למודלים הבאים.

הנתונים עליהם עבדנו הם על קובץ הtrain כאשר ביצענו חלוקה רנדומלית לtrain –test כך שגודל ה-test הוא 0.2 מגודל קובץ הנתונים.

c) הגדרנו שכבת embedding עבור הפיצ'רים הקטיגוריאליים feature_1, feature_2, feature_3, year, month (הסיבה שלא עשינו embedding עבור כל הפיצ'רים הקטיגוריאליים היא שיצרנו אותם רק בהמשך עבור התחרות ולא רצינו לשנות את כל החלק הראשון של העבודה בגלל זה):

```
f1_inp = Input(shape=(1,),dtype='int64')
f2_inp = Input(shape=(1,),dtype='int64')
f3_inp = Input(shape=(1,),dtype='int64')
year_inp = Input(shape=(1,),dtype='int64')
month_inp = Input(shape=(1,),dtype='int64')

f1_emb = Embedding(len(train_set['feature_1'].unique()),3,input_length=1, embeddings_regularizer=l2(1e-6))(f1_inp)
f2_emb = Embedding(len(train_set['feature_2'].unique()),2,input_length=1, embeddings_regularizer=l2(1e-6))(f2_inp)
f3_emb = Embedding(len(train_set['feature_3'].unique()),1,input_length=1, embeddings_regularizer=l2(1e-6))(f3_inp)
year_emb = Embedding(len(train_set['year'].unique()),2,input_length=1, embeddings_regularizer=l2(1e-6))(year_inp)
month_emb = Embedding(len(train_set['month'].unique()),4,input_length=1, embeddings_regularizer=l2(1e-6))(month_inp
```

d) בנינו מודל שהקלט שלו שווה לאוסף ערכי הפיצ'רים הקטיגוריאליים: feature_1, feature_2, feature_3, year, month, והשכבה הראשונה היא בעצם שכבת ה-embedding שלהם, כך שהשכבה הבאה היא שרשור פלט ה-embedding של כל אחד מהפיצ'רים הקטיגוריאליים שהגדרנו קודם:

```
x = concatenate([f1_emb,f2_emb,f3_emb,year_emb,month_emb])
x = Flatten()(x)
x = BatchNormalization()(x)
x = Dense(10,activation='relu')(x)
x = Dense(10,activation='relu')(x)
x = Dropout(0.4)(x)
x = BatchNormalization()(x)
x = Dense(10,activation='relu')(x)
x = Dense(10,activation='relu')(x)
x = Dropout(0.7)(x)
x = Dense(1, activation='sigmoid')(x)
emb_model = Model([f1_inp,f2_inp,f3_inp,year_inp,month_inp],x)
```

רוני מינדלין מילר 302242870
מיה קרמר 204219976

על מנת להעריך את דיוק המודל באמצעות RMSE הגדרנו גם את פונקציית ההפסד שתחשב את ה-RMSE:

```
def root_mean_squared_error(y_true, y_pred):
        return K.sqrt(K.mean(K.square(y_pred - y_true)))

emb_model.compile(optimizer = "rmsprop", loss = root_mean_squared_error, metrics =["accuracy"])
```

ולסיום אימנו את המודל על 80% מהנתונים שהתקבלו לאחר עיבוד על נתוני ה-train set. קיבלנו שה-RMSE על בסיס ה-embedding של הפיצ'רים הקטיגוריאליים: feature_1, feature_2, feature_3, year, month בלבד הוא:

| Fill missing values | Test RMSE | Train RMSE |
|---|---|---|
| -1 | 3.781 | 3.892 |
| -20 | 3.781 | 3.892 |

e) הוספנו את שאר הפיצ'רים לצירוף הembedding ואימנו את המודל מחדש:

```
continuous_input = Input(shape=(len(other_col),))
```

```
x = concatenate([continuous_input,f1_emb,f2_emb,f3_emb,year_emb,month_emb])
x = BatchNormalization()(x)
x = Dense(10,activation='relu')(x)
x = Dense(10,activation='relu')(x)
x = Dropout(0.4)(x)
x = BatchNormalization()(x)
x = Dense(10,activation='relu')(x)
x = Dense(10,activation='relu')(x)
x = Dropout(0.7)(x)
x = Dense(1, activation='sigmoid')(x) #activation='linear'
emb_model = Model([continuous_input,f1_inp,f2_inp,f3_inp,year_inp,month_inp],x)
```

אימנו את המודל על 80% מהנתונים שהתקבלו לאחר עיבוד על נתוני ה-train set. קיבלנו RMSE על בסיס ה-embedding בצירוף שאר הפיצ'רים:

| Fill missing values | Test RMSE | Train RMSE |
|---|---|---|
| -1 | 3.735 | 3.840 |
| -20 | 3.723 | 3.828 |

f) ה-embedding שביצענו היה יחסית פשוט כך שהוא "אפיין" את הפיצ'רים הקטיגוריאליים ביחס למשתנה המטרה שהוא ה-loyalty score. לאחר שדיברנו על כך עם נתי בהרצאות הבנו שיש המון אפשרויות מעניינות לביצוע הembedding ביחס למשתנים אחרים (למשל amount) ובכך למצוא קשרים בין הפיצ'רים שלא קיימים בנתונים כפי שהם נראים כרגע ולכן לא בהכרח באים לידי ביטוי בתהליך הלמידה

רוני מינדלין מילר 302242870
מיה קרמר 204219976

והחיזוי ללא ה-embedding. לא ביצענו פעולות אלא מחוסר זמן אך יכול להיות מעניין מאוד לבדוק אם הוספת קשרים אלו משפרים את ה-RMSE של המודל.

g) השתמשנו בפלט של שכבת ה-flatten כתוצאת ה-embedding של המודל שהתקבל בסעיף e:

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 1)            0
__________________________________________________________________________________________________
input_2 (InputLayer)            (None, 1)            0
__________________________________________________________________________________________________
input_3 (InputLayer)            (None, 1)            0
__________________________________________________________________________________________________
input_4 (InputLayer)            (None, 1)            0
__________________________________________________________________________________________________
input_5 (InputLayer)            (None, 1)            0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 1, 2)         10          input_1[0][0]
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 1, 1)         3           input_2[0][0]
__________________________________________________________________________________________________
embedding_3 (Embedding)         (None, 1, 1)         2           input_3[0][0]
__________________________________________________________________________________________________
embedding_4 (Embedding)         (None, 1, 3)         24          input_4[0][0]
__________________________________________________________________________________________________
embedding_5 (Embedding)         (None, 1, 4)         48          input_5[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 1, 11)        0           embedding_1[0][0]
                                                                 embedding_2[0][0]
                                                                 embedding_3[0][0]
                                                                 embedding_4[0][0]
                                                                 embedding_5[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 11)           0           concatenate_1[0][0]
==================================================================================================
Total params: 87
Trainable params: 87
Non-trainable params: 0
__________________________________________________________________________________________________
```

הכנסנו את הפלט של מודל ה-features extractor שהתקבל כקלט למודל ML קלאסי LGBMRegressor.
אימנו את המודל על 80% מהנתונים שהתקבלו לאחר עיבוד על נתוני הtrain_set. קיבלנו:

| Fill missing values | Test RMSE | Train RMSE |
|---|---|---|
| -1 | 3.759 | 3.870 |
| -20 | 3.759 | 3.870 |

כלומר אין שיפור בדיוק חיזוי המודל כאשר משתמשים בפלט המודל המבוסס על ה-embedding של המשתנים הקטיגוריאליים בלבד כ-features extractor והכנסת הפלט למודל ML LGBMRegressor.

רוני מינדלין מילר 302242870
מיה קרמר 204219976

<u>תחרות:</u>

עבור התחרות השתמשנו בנתונים שיצרנו בחלק זה (כפי שתיארנו למעלה).

ללא השלמה של ערכים חסרים הצלחנו להגיע ל-RMSE=0.705.
לאחר השלמת ערכים חסרים עם 20- הגענו ל-RMSE=0.703.

הכנסנו את נתונים אלו למודל LGBMRegressor וביצענו k-fold על מנת לאמן את המודל על כל הנתונים ולא רק חלק מהם כאשר k=5 (ניסינו גם עבור k=10 אך התוצאה זהה וזמן האימון הרבה יותר ארוך).
ניסנו להסיר פיצ'רים בעלי importance נמוך על מנת לשפר את החיזוי אך זה לא עזר.

חילקנו את הקוד ל-3 kernels מאחר והקריאה והכנת הנתונים לקחה המון זמן:

חלק 1- עונה על כל סעיפי שאלה 3.

חלק 2- הכנת הנתונים- יצירת פיצ'רים חדשים שעוזרים לאפיין טוב יותר כל card_id כפי שהוסבר למעלה. בסוף שמרנו את קבצי הtrain, test שנוצרו לשימוש בkernel השלישי.

חלק 3- שימוש בקבצי הtrain והtest שהתקבלו בחלק השני, preprocess שלהם (השלמת ערכים חסרים, label encoder וכו') והזנתם למודל עבור התחרות

link to kaggle kernel part 1: https://www.kaggle.com/ronniemiller/bgu-dl-assignmnt2

link to kaggle kernel part 2: https://www.kaggle.com/ronniemiller/bgu-dl-assignmnt2-features-extraction/notebook

link to kaggle kernel part 3: https://www.kaggle.com/ronniemiller/bgu-dl-assignmnt2-competition

הערה: כרגע הkernels פרטיים ולכן לא ניראה לנו שיהיה אפשר לגשת אליהם דרך הלינק, לאחר ההגשה נהפוך אותם לפומביים. בכל אופן העלנו את ה-kernels לGitHub.

מיקום בתחרות ביום שבת 22.12.2018 בשעה 23:30:

| | | | | | | |
|---|---|---|---|---|---|---|
| 562 | ▲ 287 | BGU-DL MayaRonnie | | 3.703 | 22 | 33m |