הסברים בנוגע לתרגיל 1 - Big Data:

Amazon Fine Food Reviews

https://www.kaggle.com/qwikfix/amazon-recommendation-dataset/data

Context

This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012. Reviews include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories.

Data

Reviews.csv, 286MB

Data includes:
- Reviews from Oct 1999 - Oct 2012
- 568,454 reviews
- 256,059 users
- 74,258 products
- 260 users with > 50 reviews

Columns

Id

Row Id

ProductId

Unique identifier for the product

UserId

Unqiue identifier for the user

ProfileName

Profile name of the user

HelpfulnessNumerator

Number of users who found the review helpful

HelpfulnessDenominator

Number of users who indicated whether they found the review helpful or not

Score

Rating between 1 and 5

Time
Timestamp for the review

Summary
Brief summary of the review

Text
Text of the review

אנליזה:

משימה 1: טעינת ה- dataset:

שמירת ה- headers שמופיעים בשורה הראשונה, ופלטור ה- headers מה- dataset

ה- Dataset אמור להיות מורכב מ-10 עמודות.

אבל בפועל, הוא מכיל שורות עם מספר שונה של עמודות (Corrupted data)

[למשל בשורה 3 יש 14 עמודות (13פעמים מופיע התו: ',' שמשמש כ- delimiter בקובץ csv)

לפני הפלטור של השורות הלא רלוונטיות, היו 568,454 שורות.

לאחר פילטור של שורות אלו, נשארו 149,388 שורות ועל ה- dataset המפולטר תתבצע האנליזה.

משימה 2: המרת ה- rdd ל- pair rdd, שבו עמודה ראשונה מכילה ערך ייחודי והעמודה השנייה מכילה את שאר העמודות.

הערך הייחודי לכל שורה הוא Id

```
transformed_reviews = reviews.map(lambda x:(x.split(",")[0],x.split(",")[1:]))
```

משימה 3:

סעיף א – ספירת מספר ההופעות של 5 עמודות נבחרות:

העמודות שנבחרו הן:

ProductId, UserId, HelpfulnessNumerator, HelpfulnessDenominator, Score

לכל עמודה בוצעה אגרגציה ברמת ה-rdd ע"י reduceByKey

```
columnCountRdd = reviews.map(lambda x:(x.split(",")[columnsDict[column]],1)).reduceByKey(lambda x, y: x+y)
```

התוצאה היא: RDD Pair לכל עמודה שמורכב מהערך, ומספר ההופעות שלו.

את ה- RDD האלו מוסיפים לרשימה (בגודל 5)

סעיף ב - הצגת היסטוגרמות:

עבור discrete values , מוצג Bar graph

עבור continuous values, מוצגת היסטוגרמה

לצרכי visualization נבחרה ספרית pandas.

בוצעה המרה של ה- rdd המקורי, ל- dataframe של spark שהומר ל- pandas dataframe לצרכי visualization.

ההמרה של הנתונים אפשרית, כי כמות ה- data בכל rdd כזה היא לא גדולה מאחר וזה סיכום של ערכים, וללא כפילויות.

בוצעה אגרגציה של ערכים, בדומה לסעיף א, ע"י שימוש ב- spark sql

df.groupBy(columnName).agg(F.size(F.collect_list(columnName))).select(col(columnName).alias("count
("lias(columnAlias), col("size(collect_list(" + columnName + "))").alias("count))

מכל data frame pandas, הוצגו 10 התוצאות עם התדירות הגבוהה ביותר, ו-10 התוצאות עם התדירות הנמוכה ביותר.

סעיף ג – הסקת מסקנות מהגרפים:

מהגרפים (histogram, bar graph) הוסקו המסקנות הבאות:

1. Score column:
   106,054 reviews סומנו ע"י המשתמשים ב- 5 כוכבים.
   16,691 reviews סומנו ע"י המשתמשים ב-4 כוכבים.
   7309 reviews סומנו ע"י המשתמשים ב-3 כוכבים.
   5928 reviews סומנו ע"י המשתמשים ב-2 כוכבים.
   13,406 reviews סומנו ע"י המשתמשים בכוכב אחד.
   Total number of reviews is 106054 + 16691 + 7309 + 5928 + 13406 = 149388
2. ProductId column:
   המוצר הפופולרי ביותר הוא: B007JFMH8M עם 401 reviews
   הרוב המוחלט של המוצרים קיבלו פחות מ- 50 reviews
3. UserId column:
   ה- user הפופולרי ביותר הוא: AZV26LP92E6WU, שפרסם 161 reviews
   רוב המשתמשים פרסמו רק review אחד.
4. HelpfulnessNumerator column:
   88594 reviews לא סומנו ע"י אף משתמש כ- helpful
   מעט מאוד reviews סומנו ע"י הרבה משתמשים כ- helpful, למשל קיים review שסומן כ- helpful ע"י 406 משתמשים.
5. HelpfulnessDenominator column

79295 reviews לא סומנו ע"י אף משתמש כ- helpful או not helpful
מעט מאוד reviews סומנו ע"י הרבה משתמשים כ- helpful או not helpful, למשל קיים review שסומן ע"י 488 משתמשים.

משימה 4 - מילוי של ערכים חסרים (missing values)

אסטרטגיה למילוי ערכים חסרים:

1. עבור עמודות המכילות ערכים נומריים:
(HelpfulnessNumerator, HelpfulnessDenominator, Score)
ברצוננו "ללכלך" כמה שפחות את העמודה, ולכן נבחר את הערך הנפוץ ביותר.
לדוגמא:
בעמודת Score קיימים 5 ערכים אפשריים: 1,2,3,4,5
מההיסטוגרמות הגענו למסקנה שעבור כל score זאת טבלת התדירות שלו:

5	106054
4	16691
3	7309
2	5928
1	13406

ולכן ברור שאם ניתקל ב- score עם ערך ריק, נעדיף להחליפו בערך '5', כי זה הערך הנפוץ ביותר ב- dataset

את הערך הנפוץ ביותר מוצאים באופן הבא:
maxValue = columnCountRddList[selectedColumns.index(column)].max(lambda x: x[1])[0]

2. עבור עמודות מסוג identifier שמזהות את הרשומה: (Id)
נייתר ערך ייחודי ע"י: Hash של כל ה- record
3. עבור עמודות מסוג Timestamp: (Time)
בחירת ערך אקראי בין ה- min timestamp לבין ה- max timestamp
4. במקרה של עמודות מסוג Text (Text, Summary):
לא ניתן למלא ערכים ריקים, כי א"א להמציא סתם טקסט. ולכן נשאיר אותו ריק.
5. במקרה של עמודות מסוג identifier ספציפי (UserId, ProductId)
לא ניתן להשלים מידע חסר – כי המידע הזה לא זמין. א"א פשוט לנחש את ה- identifier המתאים. ייתכן שאת profileName, ניתן לחלץ מה- database אם UserId איננו ריק.

ולכן העמודות הרלוונטיות למילוי ערכים חסרים הן:

Id, HelpfulnessNumerator, HelpfulnessDenominator, Score

משימה 5 – המרת 2 categorical columns, לערכים נומריים:

הבעיה: שלא קיימות ב- dataset עמודות עם categorical columns

הפתרון: נמיר 2 עמודות המכילות ערכים נומריים רציפים, לערכים נומריים דיסקרטיים (קטגוריות)

נבחר בעמודות: HelpfulnessNumerator, HelpfulnessDenominator כי אלו העמודות היחידות שמכילות ערכים נומריים רציפים.

מציאת אלגוריתם לקטלוג עמודות עם ערך נומרי, לערך דיסקרטי:

אלו 10הערכים הנפוצים ביותר של העמודה helpfulness numerator:

helpfulness numerator	count
0	88594
1	30261
2	12690
3	6404
4	3487
5	2116
6	1445
7	953
8	661
9	547

אלו 10 הערכים הכי פחות נפוצים:

helpfulness numerator	count
58	1
274	1
193	1
71	1
466	1
77	1
55	1
54	1

```
|1    |446              |
|1    |64               |
+-----+---------------------+
```

מספרים דומים נמצא גם בעמודה HelpfulnessDenominator

ניתן ללמוד מכך שהרוב המוחלט של הערכים עבור עמודות אלו הוא 0, כלומר: רוב ה- reviews לא סומנו ע"י אף משתמש כ- helpful או not helpful. ומתוך אלו שסומנו באופן כלשהוא הרוב המוחלק סומן כ- not helpful, ולכן המספר 0 צריך להיות קטגוריה בפני עצמה.

הערך הפופולרי הבא בתור הוא 1

קיימת התאמה הפוכה בין רמת הפופולריות של הערך לבין ערכו.

כלומר ככל שהערך יותר נמוך הוא יותר פופולרי, ולהיפך.

ולכן האלגוריתם k-means לא מתאים למשימת הקטלוג הזאת. [גם אם נספק את k כפרמטר, חלק מהערכים 0, עלולים להיות מקוטלגים בקטגוריה כלשהיא, וערכים אחרים יקוטלגו לקטגוריה אחרת]

ולכן האלגוריתם לקטלוג הערכים הנומריים הוא:

```
getCategory(value)

if value < 2:

   return value

return len(str(value)) + 1
```

משימה 6 – המרת timestamp column, ל- year, month, day, hour

Rdd mapping

משימה 7 – נורמליזציה של 2 עמודות:

נבחר בעמודות: HelpfulnessNumerator, HelpfulnessDenominator כי אלו העמודות היחידות שמכילות ערכים נומריים רציפים.

קיימות 2 שיטות עיקריות לנורמליזציה:

1. Zscore standardization:
   כל ערך מוחלף ב- zscore המתאים, כלומר: בכמה צעדים (נמדד ביחידות של standad deviation) הערך חורג מהחציון (כלפי מעלה או כלפי מטה)
   הנוסחה לחישוב ערכי zscore היא: (value - mean) / stdev

   לדוגמא: z=1 משמעותו חריגה ב- stdev אחד מה- mean (חציון)
   הסיכוי שערך מסויים יהיה בטווח בין z=1 לבין z=-1 הוא 0.68 (68%)
   הסיכוי שערך מסויים יהיה בטווח בין z=2 לבין z=-2 הוא 0.954 (95.4%)

2. Min max normalization:
טרנספורמציה ליניארית שממירה את כל הערכים לטווח: 0-1
נוסחה לנרמול ערכים:
(value-min)/(max-min)

מתי נעדיף standardization, ומתי נעדיף normalization:
http://datareality.blogspot.com/2016/11/scaling-normalizing-standardizing-which.html

RESCALING attribute data to values to scale the range in [0, 1] or [-1, 1] is useful for the optimization algorithms, such as gradient descent, that are used within machine learning algorithms that weight inputs (e.g. **regression** and **neural networks**). Rescaling is also used for algorithms that use distance measurements for example **K-Nearest-Neighbors (KNN)**. Rescaling like this is sometimes called "normalization". MinMaxScaler class in python skikit-learn does this.

STANDARDIZING attribute data assumes a Gaussian distribution of input features and "standardizes" to a mean of 0 and a standard deviation of 1. This **works better with linear regression, logistic regression and linear discriminate analysis**. Python StandardScaler class in scikit-learn works for this.

כלומר: נעדיף להשתמש ב- standardization למודלים של regression או classification
אבל זה בתנאי שה- data מתפלג נורמאלית. אחרת ה- standardization לא תעבוד.

כדי לבדוק אם data מתפלג נורמאלית יש לבצע normality test.
אחת השיטות המומלצות לביצוע בדיקת נורמאליות במקרה של big data נקראת:
Kolmogorov-Smirnov
נבדוק אם ה- data מתפלג נורמאלית באמצעות Kolmogorov-Smirnov method
https://www.spss-tutorials.com/spss-kolmogorov-smirnov-test-for-normality
הבדיקה מתבצעת באופן הבא:
מניחים שה- data מתפלג נורמאלית. דוגמים מספר דגימות מה- data ומניחים שהוא מתפלג נורמאלית עם אותם ערכי mean, stdev כמו הדגימות.
ואז עוברים על ההיסטוגרמה של הדגימות ובודקים עבור כל דגימה האם היא סוטה מערך ה- mean ביותר מ- stdev אחד. אחוז הדגימות שסוטות ביותר מ- stdev אחד נקרא: test statistics. המספר הזה מבטא את מידת הסטיה של ה- data מה- null hypothesis (הטענה: של הנחת העבודה שלפיה ה- data מתפלג נורמאלית עם ערכי ה- mean וה- stdev שהצבנו).
אם טענת ה- null hypothesis נכונה, אז סביר להניח שאחוז הדגימות עם שסוטות מהסתברות נורמאלית יהיה נמוך. כלומר: ערכי ה- p-value המתאימים יהיו גבוהים.
ולהיפך (אם אם טענת ה- null hypothesis איננה נכונה, אז סביר להניח שאחוז הדגימות עם שסוטות מהסתברות נורמאלית יהיה גבוה. כלומר: ערכי ה- p value המתאימים יהיו נמוכים)

נשאלת השאלה, איזה ערכי p-value, נחשבים לגבוהים/נמוכים?
התשובה היא שה- threshold שמשתמשים בו בספרות המקצועית הוא: p=0.05
כלומר: נדחה את ה- null hypothesis אם $p < 0.05$ (כלומר ה- data לא מתפלג נורמאלית כאשר $p < 0.05$)

משימה 8:

סעיף א – tokenization

המרה של משפט בעמודה Text למילים

בוצע שימוש ב- RegexTokenizer מהספריה pyspark.ml.feature

סעיף ב – Stop words removal

מחיקת stop words מה- tokens מסעיף א

בוצע שימוש ב- StopWordsRemover מהספריה pyspark.ml.feature

סעיף ג – Binary vectorization

נעשה שימוש ב- CountVectorizer מהספריה sklearn.feature_extraction.text

(לא בוצע שימוש ב- CountVectorizer של spark, כי לא קיים של הפרמטר binary=true)

נעשה שימוש ב- Compressed sparse row (CSR) כדי להימנע משמירת מטריצה ענקית שרובה אפסים שחונקת את ה- memory.

ה- Count vectorizer של sklearn לא עובד עם rdd או dataframe, אלא עם list.

מאחר וה- list of words תופס הרבה מקום בזכרון בוצעה המרת ביניים של ה- dataframe ל- pandas data frame ולאחר מכן קריאה למתודה- toList של pandas data frame

ההמרה בוצעה באמצעות מנגנון שנקרא spark arrow שעושה שימוש במנגנון deserialization של ה- jvm באמצעות הפקודה:

spark.conf.set("spark.sql.execution.arrow.enabled", "true")

בדרך זו ה- data הומר ב- chunks, ע"י deserialization של רשימת המילים, ולא נטען כולו לזכרון.

למידע נוסף: https://arrow.apache.org/blog/2017/07/26/spark-arrow/

הסבר בנוגע לשמירת compressed sparse matrix (CSR) בזכרון:

הטכניקה שבה עובד csr היא באופן הבא: (https://en.wikipedia.org/wiki/Sparse_matrix)

במקום לשמור מטריצה דו-מימדית (M_{mXn}) עם הרבה אפסים. שומרים 3 מערכים:

NNZ – מספר האיברים השונים מ-0 ב-M

A – מערך בגודל NNZ שמכיל את כל האיברים שאינם 0 ב- M

IA – מערך בגודל m+1 המוגדר באופן הבא:

- $\mathrm{IA}[0] = 0$
- $\mathrm{IA}[i] = \mathrm{IA}[i-1] + (\text{number of nonzero elements on the } i\text{-1-th row in M})$

מההגדרה הזאת נובע שהאיבר האחרון של AI הוא: IA[m+1] = NNZ

את ערכי M בשורה ה- i, ניתן לשחזר מ- IA ע"י הנוסחה הבא: A[IA[i]] to A[IA[i + 1] − 1]

AJ – מערך בגודל NNZ המכיל את אינדקס העמודה של איברי A במטריצה M

תנאי לחסכון במקום: NNZ < (m (n − 1) − 1) / 2

דוגמא 1:

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{bmatrix}$$

```
A  = [ 5 8 3 6 ]
IA = [ 0 0 2 3 4 ]
JA = [ 0 1 2 1 ]
```

NNZ=4

(m (n − 1) − 1) / 2 = 5.5

חסכנו 3 תאים (13 תאים ב-3 מערכים במקום 16 תאים ב-M)

דוגמא 2:

$$M = \begin{bmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 40 & 0 & 0 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{bmatrix}$$

```
 A = [ 10 20 30 40 50 60 70 80 ]
IA = [  0  2  4  7  8 ]
JA = [  0  1  1  3  2  3  4  5 ]
```

NNZ=8

(m (n − 1) − 1) / 2 = 9.5

- IA splits the array A into rows: `(10, 20) (30, 40) (50, 60, 70) (80)`;

- JA aligns values in columns: `(10, 20, ...) (0, 30, 0, 40, ...)(0, 0, 50, 60, 70, 0) (0, 0, 0, 0, 0, 80)`.

חסכנו 3 תאים בזכרון (21 תאים ב-3 מערכים במקום 24 תאים ב-M)