

הסבר על הוספת עמודות:

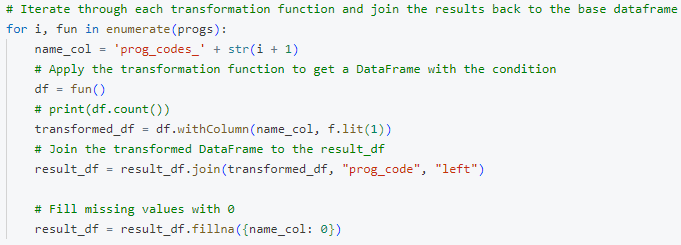
הוספנו 7 עמודות -עמודה אחת עבור כל שאילתא עם ערך True אם השאילתא מתקיימת עבור הprogram באותה שורה וFalse, אם השאילתא לא מתקיימת עבור הprogram באותה שורה.

(בנפרד, הוספנו עמודה 8, שאיכסנו בתוכה את כל הרשומות, שמקיימות לפחות 4 מתוך 7 התנאים.)

הסיבה להוספת 7 העמודות, היא שמירת המידע אודות קיום השאילתות, ללא הצורך לבצע action של מניה עבור כל שאילתא בנפרד- מכיוון שככל שה action מתבצע בשלב מאוחר יותר, כך הסיבוכיות בו יותר יעילה: בגלל lazy executionספרק מבצע אופטימיזציה באופן כזה, שכשיש לו את כל הטרנספורמציות, הוא יודע לבצע אופטימיזציה יותר מלאה, של כל הטרנספורמציות ביחד ברגע, שצריך. זה מאפשר לדחות עד כמה שאפשר את הactions, ובכך לשפר את הסיבוכיות. ובנוסף נחסוך מלהטעין מידע לדרייבר (שזה מידע- מספר שורות לא חסום שיכול לגרום לבעיות שמפעילים עליה פעולה מסויימת).

לכן הוספת העמודות, ושמירת מידע על התוצאות של השאילתות בתוכן, זו הדרך היעילה יותר מבחינת סיבוכיות, ומבחינת תפיסת מקום בזיכרון.

דוגמאה מהקוד:





A screenshot of a computer

Description automatically generated

פתרון חלופי לJoin:

בדומה לתרגיל הבית, בכל שאילתא השתמשנו רק בעמודות, שרלוונטיות לאותה שאילתא (פרגמנטציה אנכית), ובשביל לשמור את הכל באופן כולל, ללא איבוד של מידע, הוספנו 7 עמודות (פרגמנטציה אנכית), שישמרו ערכים בינאריים המתאימים לכל מספר מזהה של תוכנית, שורה בינארית באורך 7, בה הערך של האינדקס הk מסמל קיום, או אי קיום של השאילתא ה- k+1

חסרונות של Join:

1. אם נעשה inner join, יכול לגרום לאיבוד של מידע.

- אם לעשות אאוטר גוין, יהיו הרבה שורות עם Null, וזו תהיה תפיסת מקום מיותרת.

2. הפרוצדורה של ההרכבה מאוד יקרה מבחינת סיבוכיות זמן ומקום, ובנוסף Join גורר את Shuffle. ברלציות מאוד גדולות זה דבר שייקח הרבה מאוד זמן. Shuffle עושה תעבורה על כל המידע. נפרט על הפעולה זו כדי לראות איך זו חיסרון:

Shuffle: עובר על הדאטא בכל הדאטא פריים (שמחולק לpartitions), מעלה אותו לאינטרנט, ומסדר אותו מחדש בין השרתים.

את Shuffle צריך, בשביל לקחת את כל הtuples, שיש להם את אותו key, לשרת אחד פיזי.

Shuffle היא פעולה יקרה מאוד באינטרנט, בתעבורה, בקריאה וכתיבה ולא תמיד בטוחים, מתי היא קורית. לפעמים Join יכול לגרור שאפל מאחורי הקלעים.

ב Join, Shuffle עושה מעבר על כל הנתונים, תעבורה בין השרתים.

כשיש data גדול, מתבצעת תעבורה מאוד גדולה כתואה מכך, והיינו רוצים להמנע מזה.

יתרונות של Join:

1. כשיש תוצאת join מוכנה, יהיה נוח לגשת ולחלץ את כל המידע, שמופיע בשורה, בפעימה אחת. יש מידע מלא במקום אחד עבור כל תוכנית (או כל אובייקט או תכונה, שמעוניינים לבדוק). הקישור בין המידע כבר קיים, אין צורך ל

2. הכל מאוחסן ביחד. יש פחות תעבורה על פני האינטרנט (שלוקחת משאבי זמן).

**הסברים לטרנספורמציות:**

**DropDuplicates**: הdata גדול, יתכנו טעויות וכפילויות. הspark יודע לפלטר את הכפילויות הנ"ל ע"י שובר שיויון. למשל, שיוך של device\_id אחד לכמה DMAs, וכדומה. לא יתכן כמובן, ש- device\_id יהיה משוייך לכמה DMAs, אבל ב data אמיתי יתכנו כאלו טעויות. יתכנו גם כפילויות של שורות.

את כל השגיאות הנ"ל יש צורך להסיר, בשביל לקבל תוצאות, שעובדות באופן תקין.

**Dropna**: יתכנו בdata טעויות של ערכי Null באטריביוטים של key. זהו מידע שגוי, שצריך להסיר.

**Select**: בשביל לא להתעסק עם data גדול מידי, משיקולים של חיסכון בזמן, הסרנו את כל העמודות, שלא מחזיקות מידע רלוונטי לשאילתא. (עבור כל שאילתא ביצענו Select מתאים).

Filter: בכל שאילתא ביצענו סינון לפי שורות, לפי הדרישות של השאילתא.

**Join**: בשביל לקשר בין אטריביוטים, שהשאילתא אמורה להיפתר בעזרת הקישור ביניהם, ואינם נמצאים באותה הרלציה, ביצענו Join על הרלציות, שמכילות את העמודות הדרושות.

ההוספת עמודות עם ערכים בינאריים יכולים לחסוך זמן כי קודם רק צריך להפעיל פעולה שהעמודות כבר מוכנות עם המידע שצריך ובפרט חוסך מלהטעין מידע לדרייבר (RAM) לפני שצריך => חוסך זמן ריצה ומשאבים נוספים. מכיוון שאנו רק משתמשים במידע שצריך ולא רגע לפני.

רוב הפילטרים שעשינו זה היה לנקות את הדאטה כי הדאטה שקיבלנו הוא לא נקי ולכן היינו צריכים לקבל החלטות כדי לענות השאלות בצורה הנדרשת. ואם לא לנקות אז כדי לבצע את הדרישות השאילתה או על מנת לקבל את הDF שנרצה להגיע אליה.

בנוסף, הוספנו תיעוד בקוד כדי להסביר את הקוד כדי שיהיה קריא. השתדלנו לכתוב קוד נקי ולהימנע משכפול קוד איפה שניתן. יתר על כך ניסינו לכתוב קוד בצורה יעילה כדי לחסוך בזמני ריצה כפי שלמדנו בהרצאה ומשעות קבלה עם יאיר.



**פרגמנטציה**

יתרון: אם מתקיים, שככל שלעיר יש ציון יוקרה יותר גבוה, כנראה שהיא מבצעת יותר תשאולים (מה, שמניב יותר רווח מהשימוש בפרגמנט, שנוצר בדרך הזו באותה העיר), אז הפרגמנטציה המתוארת היא גם היעילה ביותר מבחינת מזעור תעבורה.

בנוסף, הפרגמנטציה יחסית מוגבלת במקרה הזה. ומבחינת סיבוכיות, יש יתרון בלהמעיט בפרגמנטציה. וזה ממקסם את הרווח של החברה.

חיסרון:

ההעדפות של הזאנרים וגם רמת היוקרה לפי איזורים, יכולות להשתנות עם הזמן.

למשל, עיר, שלא הובאה בחשבון לפי רמת היוקרה, כעבור שנה נכנס לעשיריית הערים הכי יוקרתיות, ואז הפרגמנטציה הנוכחית כבר לא תהיה רלוונטית, מה שיגרור תעבורה גדולה על פני האינטרנט, בגלל העברה מערים, שפעם היו הכי יוקריות, לערים, שכיום הכי יוקרתיות. ולעשות את החישובים שוב אחרי תקופה כלשהיא יכול להיות יקר.

**הצעת דרך אלטרנטיבית לפרגמנטיציה, שהוצעה בתרגיל**

פרגמנטציה יותר ספציפית.

עבור כל זאנר, לבדוק את הביקוש שלו בין הערים השונות.

למשל:

|  |  |  |
| --- | --- | --- |
| מתוך 3,030 צופי ה**דרמה**: | מתוך 4,280 צופי ה**בישול**: | מתוך 10,000 צופי ה**ריאליטי**: |
| |  |  |  | | --- | --- | --- | | קרית גת | רחובות | ירושלים | | 540 | 2300 | 190 | | |  |  |  | | --- | --- | --- | | טבריה | חיפה | קרית גת | | 80 | 1200 | 3000 | | |  |  |  | | --- | --- | --- | | באר שבע | רחובות | ירושלים | | 500 | 4500 | 5000 | |

עבור כל זאנר, למשל "**ריאליטי**", "**בישול**", "**דרמה**", נמקם אותו בעיר, ו הנבחר את הגאנר שהיא -“argmax” שלו כאשר אין כפיליות בגאנרים שנבחר.

בדוגמה, המיקום של הזאנרים "**ריאליטי**", "**בישול**", "**דרמה**", יהיה:

|  |  |
| --- | --- |
| ריאליטי | ירושלים |
| דרמה | רחובות |
|  | באר שבע |
| בישול | קרית גת |
|  | חיפה |
|  | טבריה |

באופן כזה, נקבע מיקום לזאנר, שבו הוא הכי נצפה, מה שימזער את התעבורה על פני האינטרנט ותהיה יעיל מבחינת סיבוכיות זמן. התשאול של כל זאנר יהיה בעיקר מהעיר, שבה הוא מוקם.

מכל עיר ניקח את ה-11 גאנרים בלי כפליות אבל לא ניקח את הargmax עבור הtop ערים שנבחר בבעיה זו, זה 10 סה"כ ערים כמו שראינו ב2.2 ובצורה דומה ניקח את 11 גאנרים לכל DMA עיר (נשלים עוד 10 גאנרים בלי כפילויות).

זה ימנע להבא תעבורה יותר גדולה ממה, שהיא יכולה להיות: שאם נמקם זאנר באיזור, בו הוא פחות ניצפה, תתבצע תעבורה גדולה יותר למקום, שבו הוא יותר ניצפה, מאשר אם נמקם את הזאנר במקום, שבו הוא לא הכי ניצפה ואז התעבורה למקומות, שפחות צופים בו, תהיה גדולה יותר מהמקרה, שאנחנו מציעים. בצורה כזו עושים העברה יעילה על הדאטה ומוצאים פתרון שניתן למימוש באופן סביר ותוביל לרווח עבור החברה.