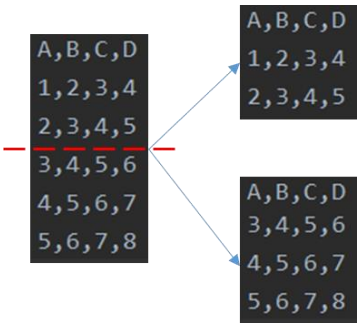
אבן דרך 3

Time Series

בפעם הקודמת יצרנו את הטיפוס TimeSeries (סדרת זמן) – שמייצג כיצד מאפיינים מסוימים משתנים לאורך זמן (בכל שורה). הפעם נרצה להוסיף לטיפוס זה את הפונקציונליות של פיצול הסדרה למספר סדרות זמן. לדוגמה הנה המחשה ויזואלית לפיצול סדרה לשתי סדרות:



לשם כך:

א. ממשו את length המחזירה את מס' שורות הנתונים בסדרה (כ Int)
ב. ממשו את split(n:Int):List[TimeSeries] -
   a. המתודה תחלק את הסדרה ל n חלקים
      i. כל אחד מהם באורך של length / n = chunkSize (חלקות int ב int) שורות
         1. פרט לאחרון שיכיל גם את השארית אם יש כזו.
   b. המתודה תחזיר אותם כרשימה של אובייקטים חדשים מסוג TimeSeries
   c. כמובן, יש לשמור על סדר השורות המקורי. כלומר
      i. האובייקט הראשון מחזיק את chunkSize השורות הראשונות
      ii. האובייקט השני מחזיק את chunkSize השורות הבאות, וכך הלאה
      iii. האובייקט האחרון ברשימה יכלול גם את שארית השורות אם ישנן כאלה.

לדוגמה, אם אורך הסדרה הוא 103 וקראנו split(10) אז נקבל רשימה של 10 אובייקטים מסוג TimeSeries. באובייקט הראשון יהיה את המידע של 10 השורות הראשונות, בשני יהיו את 10 הבאות וכך האלה. באובייקט העשירי יהיו את 13 השורות האחרונות.

למען הסר ספק:

- אין לבצע אף פעולת IO במתודה split. בפרט, אין ליצור \ לערוך \ לקרוא אף קובץ CSV.
  - אלא פשוט לחלץ את הנדרש מתוך הנתונים השמורים ב TimeSeries.
- לפעולה split אסור לפגוע באובייקט המקורי ממנו הפעלנו את split.
  - הוא עדין יכיל את כל הנתונים שנדלו מתוך קובץ ה csv שלו.
- לכל הסדרות שנוצרו יש מן הסתם את אותם שמות המאפיינים.

**פרט לבנאי שמקבל שם קובץ תצטרכו ככל הנראה להוסיף בנאי נוסף למחלקה.

- בנאי מוסיפים ע"י ... { ()def this (עם אלו פרמטרים שאתם רוצים כמובן).
- בשורה הראשונה של כל בנאי יש להפעיל את הבנאי הראשי (זה שבהצהרת ה class).
  - ע"י קריאה ל ()this (והפרמטרים הנדרשים אם יש כאלו)

גילוי חריגות מהיר במקביל

כפי שראינו בשיעור, אחת הדרכים לגרום לחישוב לרוץ מהר יותר הוא ע"י הפרד ומשול – נחלק את המידע למס' חלקים ונעבד כל חלק במקביל (על ליבה פנויה אם ישנה כזו) ולבסוף נאחד את התוצאה הלוקאלית שהשגנו על כל חלק לכדי התוצאה הגלובאלית הרצויה. לטכניקה הזו קוראים map-reduce והיא בליבה של חישוב מקבילי במערכות מבוזרות כמו Hadoop ועוד.

האפשרות לפצל את ה TimeSeries זה צעד ראשון כדי להשיג זאת. אך יש לנו גם צורך באלגוריתם לגילוי חריגות שיכול לעבוד על כל חלק לוקאלי של המידע בנפרד ומאוחר יותר לאחד את התוצאות מבלי שזה יפגע בדיוק שלו. האלגוריתמים הקודמים שמימשנו פחות מתאימים לדרישה זו ולכן נממש באבן דרך זו אלגוריתם חדש ומתאים. אך חשוב מכך, ניצור לנו את התשתית להריץ את האלגוריתם בטכניקה של map-reduce.

הביטו ב ParAnomalyDetector.scala

- הגדרנו את Report עבור דיווח אנומליה. הוא כולל את שם ה feature שבו היא קרתה, את הזמן, וכן ניקוד עבור מידת האנומליות שלה.
- עבור הנוחות יצרנו גם את Reports כסדרה של Report.

כעת נביט ב trait ParAnomalyDetector שלנו. התכונות שנדרוש מגלאי חריגות הן:

- המתודה map אשר בהינתן TimeSeries היא תחזיר Reports
  - באמצעותה נוכל מאוחר יותר לגלות חריגות בכל חלק של TimeSeries בנפרד
- המתודה reduce אשר בהינתן שני אובייקטים מסוג Reports היא תאחד אותם לאובייקט אחד
  - באמצעותה נוכל מאוחר יותר לאחד את התוצאות הלוקאליות לכדי תוצאה גלובאלית אחת

כעת אנו יכולים לממש בתוך ה trait את העיבוד המקבילי והאיחוד בצורה כללית וללא תלות באלגוריתם(!)

ממשו את המתודה detect אשר בהינתן TimeSeries, ExecutorService ו chunks המייצג את מס' החלקים, אז המתודה תגלה חריגות באמצעות map ו reduce.

בפרט, היא תחלק את ה TimeSeries ל chunks סדרות זמן, ותשתמש ב ExecutorService כדי לעבד כל אחת מהן במקביל באמצעות map. רק לאחר הזרקת כל המשימות היא תמתין לכל future ותאחד את התוצאות לכדי תוצאה אחת באמצעות reduce.

** שימו לב שאם בסדרה ה i התגלתה חריגה בזמן t אז הזמן האמתי הוא i*chunkSize+t.

כעת כל אובייקט מסוג ParAnomalyDetection מקבל את detect בירושה וכל שעליו לעשות זה לממש בפשטות את map ואת reduce.

אלגוריתם לגילוי חריגות מבוסס אנטרופיה

כידוע, מדד האנטרופיה מציין כמה "רעש" \ "אי סדר" \ "חוסר וודאות" יש במידע שלנו. לדוגמה אם כל ערך יופיע בדיוק פעם אחת, אז לכולם הסתברות שווה, מה שמייצר חוסר וודאות מקסימלי או פשוט אנטרופיה מקסימאלית. לדוגמה אם ההסתברות שירד מחר גשם או שלג או שיהיה בכלל חמסין היא שווה לכל אפשרות אז אין לנו דרך לדעת איך להיערך למחר. לעומת זאת אם ההסתברות לחמסין היא גבוהה יותר וההתפלגות היא לא אחידה אז יש לנו יותר אינפורמציה \ יותר וודאות ויהיה לנו יותר קל להיערך למחר. במילים אחרות האנטרופיה תהיה נמוכה יותר.

איך כל זה קשור לגילוי חריגות?

כאשר יש לנו מידע שכולל בתוכו חריגה, אז החריגה מגדילה את האנטרופיה שלו. אם נוציא את החריגה מהמידע אז האנטרופיה תרד. במילים אחרות, אם יש לנו וקטור X אז לכל x_i נחשב את ההפרש בין $H(X)$ (האנטרופיה של X כולו) לבין $H(X - x_i)$ (האנטרופיה של X ללא x_i). כך נוכל להצמיד לכל ערך ציון שמגלם מהי מידת האנומליות שלו - ככל שההפרש גדול יותר כך הוא היה יותר אנומלי. נוכל להחזיר למשתמש רשימה ממוינת \ עשרת הגדולים \ הערך עם הציון המקסימלי וכדומה.

אולם, האלגוריתם הזה סובל משתי בעיות עיקריות

א. החישוב שלו די כבד
ב. הוא לא רגיש מספיק לחריגות

למשל בסדרת זמן שהיא די רציפה (כל ערך מופיע מעט מאד פעמים) אז גם אם יהיה ערך בודד חריג מאד הוא עדיין יבלע בתוך "הרעש הלבן" של הסדרה והסרתו לא תשנה את האנטרופיה.

האלגוריתם הזה טוב בעיקר למקרים שיש הרבה הופעות מאותו הערך ופתאום יש ערך שונה.

אבל!

באמצעות הפרד ומשול נוכל לפתור את שתי הבעיות בבת אחת. כאשר אנו מעבדים רק חלק מהסדרה אז ההסתברות להישנות הערכים עולה – מה שמוביל לאנטרופיה נמוכה, אלא אם כן ישנו ערך חריג. בנוסף העיבוד של חלקי הסדרה (במקום זו בשלמותה) והעובדה שהוא נעשה במקביל תורמת רבות לביצועים.

עליכם לממש את האלגוריתם הבא באמצעות התשתית שיצרתם.

ממשו את EntropyAnomalyDetector כסוג של ParAnomalyDetector

- במתודה map עליכם לחשב לכל feature את הערך עם הפרש האנטרופיה המקסימאלי.
  - המתודה כזכור מחזירה Reports – שזו רשימה של Report
  - כל דיווח כזה כולל את שם ה feature, הזמן בו קרתה החריגה, ומידת האנומליות
  - בסך הכל יוצא שאורך הרשימה הוא כמס' ה features
    - לכל feature דיווח אחד בלבד – היכן שהיה ההפרש המקסימלי
- במתודה reduce - בהינתן שני Reports נרכיב Repots אחד ע"י בחירה של הציון הגבוה יותר
  - לדוגמה, נניח שיש לנו 3 עמודות A,B,C
  - ב Report הראשון יש (A,12,0.5) , (B,150,0.04) , (C,70,0.23)
  - ב Report השני יש (A,95,0.3) , (B,20,0.041) , (C,30,0.45)
  - אז נחזיר ע"פ הציון הגובה: (A,12,0.5) , (B,20,0.041) , (C,30,0.45)

- כך ניתן יהיה לחשב באמצעות detect שירשנו את החריגות עבור כל חלק מהסדרה במקביל ולאחד את התוצאות החלקיות לכדי התוצאה הסופית.

בדיקה

הביטו ב MainTrain. הבדיקה הראשונה היא בדיקה פשוטה שעוזרת לוודא שחילקתם נכון את ה TimeSeries. הבדיקה השנייה באה לוודא שלמרות החלוקה למס' chunk-ים במס' ת'רדים עדין מתקבלת התוצאה הנכונה. הבדיקה במוד ההגשה דומה.

הגשה

עליכם להגיש אך ורק את Util, ParAnomalyDetector, EntropyAnomalyDetector, TimeSeries

תאריך ההגשה הוא ה 06.06. אין הגבלות על כמות ההגשות. בהצלחה!