# מעבדה באיסוף וניהול נתונים

# פרויקט סיום


מגישים:

משה עבאדי 324658939

רועי רימר 314828732

# הקדמה – תיאור קצר של האפליקציה

בנינו אפליקציה המאפשרת ניטור בזמן אמת של נסיעות החורגות ממסלולן. ההשוואה נעשתה בהשראת מאמר ומבוססת על השוואה של מסלול הנסיעה הנוכחית אל מול דיווחי עבר של אותו הקו. האפליקציה מנסה גם להסביר מדוע התרחשה החריגה על סמך מידע חיצוני ששילבנו עם מסד הנתונים המקורי
בנוסף, האפליקציה מאפשרת להסתכל על דיווחים מן העבר על סמך תאריך, מספר קו ומספר רכב, ומאפשרת לראות פרטים על נסיעות של הקו באותו תאריך.
בשתי האפשרויות (ניטור בזמן אמת וניתוח נתוני עבר), ניתן לעבור למצב מפה, וכך לראות את הדיווחים בצורה נוחה. בנוסף, ניתן יהיה לראות פרטים נוספים כמו איכות חיזוי הממוצעת של ה-$delay$ שנתן המודל עבור הנסיעה החריגה במקרה של ניטור זמן אמת, ועבור הנסיעה החריגה הרצויה במקרה של ניתוח נסיעות עבר.

נתאר כעת בפירוט את השלבים השונים של הפרויקט:

# Part 1- Warmup

## <u>הגדרת המשימה</u>

המשימה שהגדרנו בחלק זה היא חיזוי של השדה $delay$ עבור כל דיווח, כאשר במקום להשתמש בערך המדויק של $delay$ בחרנו להגדיר טווחי ערכים. משימה זו מעט שונה ממשימת החיזוי שהגדרנו בתרגיל השני בקורס, שם ניסינו לחזות את ערך ה-$delay$ המדויק באמצעות מודל רגרסיה ליניארית.
הרעיון מאחורי השינוי במשימת החיזוי הוא שכאשר נוסעים או מפקחים מטעם חברת האוטובוסים יתעניינו ברמת הדיוק בזמנים של האוטובוסים, סביר להניח שהם ירצו לדעת את המידה בה האוטובוס הקדים או איחר, ולא יתעניינו במספר השניות המדויק.
חילקנו את ערכי ה-delay לתשע קטגוריות, הנעות בין הקדמה מעל 10 דקות (קטגוריה 0), עד לאיחור מעל 10 דקות (קטגוריה 8).
חיזוי מידת ה-delay נעשה על סמך מודל רגרסיה לוגיסטית, המשתמש במשתנים המסבירים הבאים לצורך החיזוי:

$$atStop, justLeftStop, justStopped, ellapsedTime, distanceCovered,$$
$$actualDelay, vehicleSpeed, congestion, busStop, hour, dayOfWeek, patternLine,$$
$$event\_around$$

כאשר השדות בהם השתמשנו ולא היו בסט הנתונים המקורי הם:

- $hour$ – השעה הנוכחית.
- $dayOfWeek$ – היום בשבוע.
- $patternLine$ – קו האוטובוס הנוכחי כפי שהוא רשום בארבע הספרות הראשונות בשדה $journeyPatternId$.
- $event\_around$ - האם התרחש אירוע המוני באזור, ואם כן אז מה הוא היה (פירוט על שדה זה בהמשך).

האפליקציה שבנינו תתמוך במשימת החיזוי שהגדרנו ב-Part 1. עבור דיווחים חדשות שיתקבלו במהלך ה-stream, תחזה האפליקציה את קטגוריית רמת ה-delay.

## טיפול בנתונים חסרים

כדי לנסות ולשפר את ביצועי משימת החיזוי ביחס לביצועים שלה על הנתונים המקוריים, בחרנו להסיר ערכים חריגים בשדות $vehicleSpeed$, $distanceCovered$, $actualDelay$ , כאשר ערכים חריגים הוגדרו באופן הבא:

- עבור השדה $vehicleSpeed$: ערכים מחוץ לקטע $[0,120]$.
- עבור השדה $distanceCovered$: ערכים מחוץ לקטע $[0,5]$.
- עבור השדה $actualDelay$: ערכים מחוץ לקטע $[-2200,2000]$

הערכים המדויקים לשדות נבחרו על ידי ניתוח ההיסטוגרמות וה-box-plots של שדות אלו, וזיהוי ערכי הקיצון.

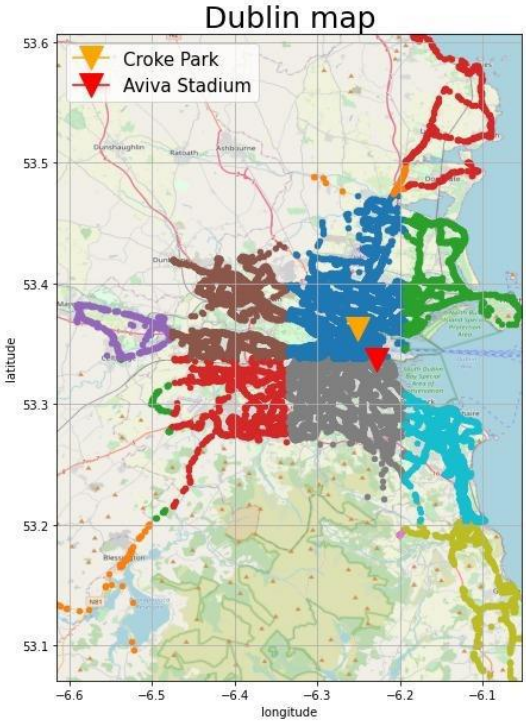
את הערכים שהסרנו בשדות אלו בחרנו להשלים באמצעות שיטת Mean Imputation.
הסיבה לבחירה בשיטה זו היא שזו השיטה עבורה קיבלנו את הביצועים הטובים ביותר מבין שיטות ההשלמה שניסינו במהלך המשימה השנייה בקורס.

# אינטגרציה של נתונים חיצוניים

במהלך העבודה, בחרנו לשלב עם מסד הנתונים המקורי שלנו את הנתונים הבאים:

## 1. נתונים אודות אירועי ספורט והופעות גדולות בדבלין

נתונים אלו הגיעו מיותר משני מקומות מידע שונים, וכללו מידע בטווח התאריכים של מסד הנתונים המקורי שלנו אודות אירועי ספורט ומוזיקה משני האצטדיונים הגדולים ביותר בדבלין (איור 1): אצטדיון **Croke Park**, המכיל 82,300 מקומות ישיבה (האצטדיון השלישי בגודלו באירופה), ואצטדיון **Aviva**, המכיל 51,700 מקומות ישיבה.




*איור 1 - האצטדיונים Aviva, Croke Park מסומנים על מפת דבלין יחד עם חלק מן הדאטה, אשר צבוע לפי השדה areaId1.*

סוגי ענפי הספורט עבורם השגנו מידע הם :

- כדורגל (בעיקר משחקים של הנבחרת הלאומית של אירלנד).
- משחקי רוגבי (בעיקר משחקים של Leinster , קבוצת הרוגבי המקומית המובילה).
- ענפי ספורט נוספים שמשוחקים תחת ארגון ה-GAA (Gaelic Athletic Association), ארגון המנהל טורנירים של משחקי ספורט איריים מסורתיים.

מרבית המשחקים היו בשלבים מתקדמים של טורנירים שונים (בעיקר משחקי רבע גמר עד גמר), שכן אצטדיונים אלו הם אצטדיונים מרכזיים מאוד בדבלין ונוהגים לארח בעיקר אירועים גדולים ובעלי חשיבות רבה.

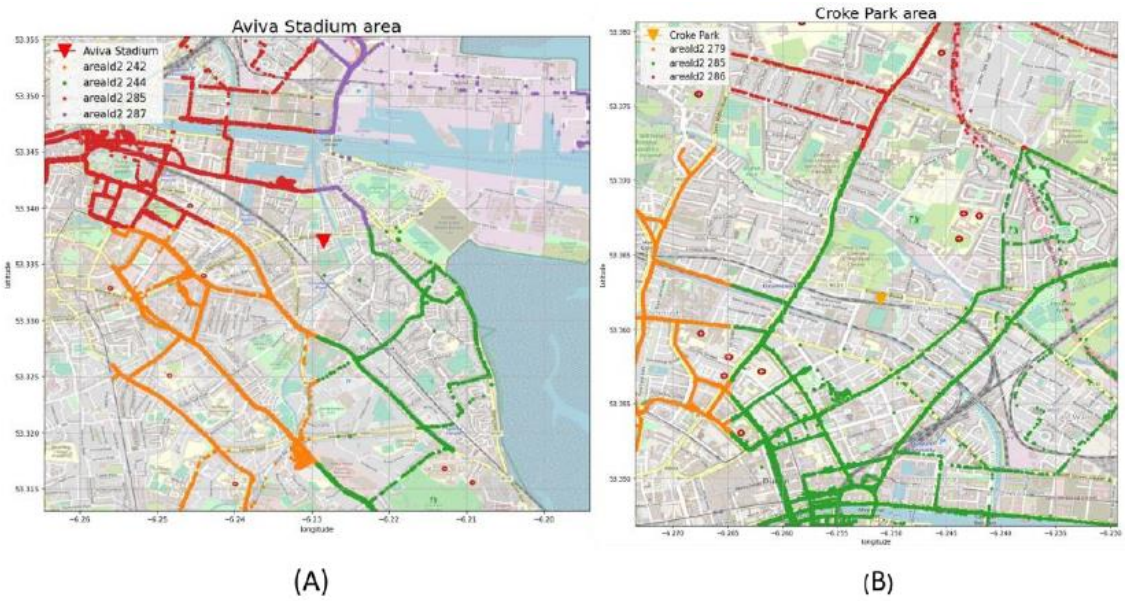
סה"כ מצאנו מידע עבור 61 אירועי ספורט ומוזיקה שאכן היו בטווח התאריכים המתאים למסד הנתונים.

את הנתונים חילצנו מן המקורות הבאים :

- האתר הרשמי של אצטדיון Croke Park – משחקים של ארגון ה- GAA בשנים 2017-2018:
  - https://crokepark.ie/matchday/2020-season-fixtures/2017-season-fixtures
  - https://crokepark.ie/matchday/2020-season-fixtures/2018-season-fixtures
- ויקיפדיה – דפי הויקיפדיה של האצטדיונים Croke Park ו-Aviva:
  - https://en.wikipedia.org/wiki/Aviva_Stadium
  - https://en.wikipedia.org/wiki/Croke_Park

**פרטי האינטגרציה של מקורות המידע החדשים עם הנתונים:**

בכדי לאחד את הנתונים על האירועים שמצאנו יחד עם מסד הנתונים הקיים, השתמשנו במידע על התאריך בו התקיים האירוע ועל האצטדיון בו הוא נערך. עבור כל אצטדיון, הגדרנו על ידי התבוננות במפת דבלין ובמסד הנתונים הקיים את רשימת ה $areaId3$ -בדאטה שלנו שסמוכים באופן יחסי לאצטדיון . ניתן לראות את האזורים שבחרנו עבור כל אחד משני האצטדיונים באיור 2:




*איור 2 - ה-$areaId3$ שנבחרו כסמוכים עבור כל אצטדיון. (A) – Aviva, (B) – Croke Park*

בנוסף למידע על המיקום, כדי לשלב את הדאטה החדש עם מסד הנתונים השתמשנו כאמור גם במידע על התאריך והשעה בה מתקיים כל אירוע. עם זאת, באתרים מהם חילצנו את הדאטה לא צוינה שעת תחילת האירוע עבור מרבית האירועים. לכן, חיפשנו את שעות תחילת המשחקים באינטרנט והשלמנו את הדאטה באופן ידני. עבור אירועי הספורט ההשלמה נעשתה בעזרת דיווחים על המשחקים באתרי ספורט שונים (בעיקר אתר ספורט אירי בשם The42), ועבור אירועי המוזיקה מצאנו את השעות הרלוונטיות בעזרת חיפוש באינטרנט אודות מידע כללי על ההופעות ועל כרטיסים להופעות.

כעת, בהינתן הגדרות מיקומים סמוכים לאצטדיון והשלמות שעות תחילת האירועים השונים, הוספנו עמודה חדשה לדאטה בשם $event\_around$. עבור כל רשומה, בדקנו קיום של שלושה תנאים:

1. ה-$areaId3$ של הרשומה היה כלול ברשימת ה-$areaId3$ שהגדרנו עבור אחד האצטדיונים.
2. תאריך הרשומה (שנה, חודש ויום בחודש) שחולץ מהשדה $datetime$ היה זהה לתאריך קיום המשחק.
3. לפחות אחד מן הבאים מתקיים:
   a. הזמן המדויק ביממה של הרשומה, שחולץ מהשדה $datetime$, היה עד שעה לפני תחילת משחק או עד 25 דקות לאחר סיום המשחק. את סיום המשחק הגדרנו כשעה ו-50 דקות אחרי תחילת המשחק, מכיוון שמצאנו כי משחקים בענפי הספורט שאספנו עבורם מידע אורכים כמות כזו של זמן בערך.
   b. הזמן המדויק ביממה של הרשומה, שחולץ מן השדה $datetime$ שב- Data Warehouse, היה בטווח הזמן שבין שעת פתיחת הדלתות (אותו מצאנו גם כן באופן ידני) של אירוע מוזיקה כלשהו ועד תחילת האירוע עצמו.

אם שלושת התנאים הללו התקיימו, אז הוזן עבור רשומה זו בשדה $event\_around$ את סוג האירוע (football, rugby, gaa או concert). אחרת, הוזן הערך no, שמסמן שרשומה זו אינה רלוונטית להתרחשות אירוע כלשהו. בסך הכול, 499,291 רשומות קיבלו ערך השונה מ-no עבור עמודה זו.

## 2. ציוצים מתוך עמוד הטוויטר dublinbusnews, שהכילו את ההאשטג #DBSvcUpdate.

עמוד הטוויטר הנ"ל מכיל עדכונים, מידע וחדשות על תנועת האוטובוסים בעיר דבלין. בפרט, הציוצים שמכילים את ההאשטג #DBSvcUpdate הם ציוצים המדווחים על הפרעות ספציפיות ואירועים חריגים שמשפיעים על תנועת קווי אוטובוסים מסוימים. דוגמא לציוץ שכזה ניתן לראות באיור 3:


*איור 3 – דוגמא לציוץ בעל ההאשטג #DBSvcUpdate מתוך עמוד הטוויטר dublinbusnews.*

אנו נשתמש בציוצים הללו באפליקציה שנכין כהסברים אפשריים לאנומליות במיקומי האוטובוסים, אותם המערכת תציג למשתמש במקרה של נסיעה חריגה. עוד על כך בתיאור של Part B.

הציוצים נלקחו מתוך טוויטר על ידי שימוש ב-API הרשמי של טוויטר.

קישור לעמוד הטוויטר dublinbusnews:

https://twitter.com/dublinbusnews?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor

**פרטי האינטגרציה של מקורות המידע החדשים עם הנתונים:**

בכדי לאחד את הנתונים על האירועים שמצאנו יחד עם מסד הנתונים הקיים, השתמשנו

במידע על התאריך של כל דיווח ושל כל ציוץ. עבור כל דיווח, אם היו ציוצים בעלי ההאשטג #DBSvcUpdate שעלו בדף הטוויטר dublinbusnews בחלון של שעה לפני או אחרי הדיווח, הוספנו אותם לדיווח תחת השדה relevant_tweet. במידה והיו כמה ציוצים רלוונטיים באותו חלון זמן, נשמר רק הציוץ האחרון.

# Part 2

המשימה החדשה שבחרנו היא זיהוי אנומליות במיקומי דיווחי קווי האוטובוס, כאשר המיקומים יימדדו ביחס לנסיעות קודמות שביצעו אוטובוסים מאותו קו.

במהלך הסמסטר למדנו והצגנו בכיתה את המאמר: DBChEx: Interactive Exploration of Data and Schema Change מאת החוקרים Bleifuß, Bonemann, Kalashnikov, Naumann, Srivastavsa. מאמר זה מתאר כלי בשם DBChEx, המאפשר לחקור ולזהות שינויים וחריגות במסדי נתונים דינמיים באמצעות ממשק נוח המאפשר השוואה פשוטה בין ערכים שקיבלו שדות מסוימים לאורך הזמן.

בהשראת DBChEx, בחרנו לנסות ולזהות אנומליות במיקומי האוטובוסים, כאשר כמו ב-DBChEx, החקירה והקביעה האם ערך מסוים הוא חריג תיעשה על סמך השוואה אל מול ערכים קודמים שהשדה קיבל בעבר, דהיינו נסיעות עבר של אותו קו אוטובוס באותו כיוון נסיעה. במקרה שלנו, תהליך ההשוואה וזיהוי הדיווחים החריגים ייעשה באופן הבא:

עבור דיווח מסוים, המערכת מחפשת את כל דיווחי העבר בטווח 200 מטרים שהם בעלי אותו מספר קו וכיוון תנועה (כלומר 5 הספרות הראשונות בשדה $JourneyPatternId$ זהות). רוב הדיווחים בעלי ערך זה הם מאוטובוסים שביצעו את המסלול הנכון עבור הקו וכיוון הנסיעה האלו. כעת, נסתכל על המרכז הגיאוגרפי של דיווחי העבר שמצאנו, ועל השונות שמתקבלת בערכי ה-$longitude$ וה-$latitude$ של קואורדינטות המיקום. הדיווח ייחשב לחריג אם:

- הדיווח הנוכחי רחוק עבור שני השדות האלו ברמה של למעלה מסטיית תקן אחת מן הממוצע (אם הדיווח הנוכחי רחוק מדי מהרוב, כנראה שהוא חריג).
- אם יש פחות מ-50 דיווחי עבר מתאימים שנמצאים בטווח 200 מטרים מן הדיווח הנוכחי (אם הדיווח הנוכחי מבודד מדי, כנראה שהוא חריג).

המערכת תתריע למשתמש על נסיעה חריגה במידה ויתקבלו שלושה דיווחים חריגים ברציפות עבור נסיעה מסוימת.

## סעיף מתוך דרישות הפרויקט:

## Explain the differences and limitations of the chosen method with respect to the Dublin dataset.

מגבלה של היישום שלנו למאמר לעומת השיטה המקורית היא ש- DBCheX מנסה להנגיש ולפשט את התהליך של חיפוש ממצאים מעניינים בדאטה ושל זיהוי חריגות בו, אך

המשתמש אמור לבצע את החיפוש באופן ידני בעזרת ההנגשות. לעומת זאת, המערכת שלנו מנסה להתריע על אנומליות על סמך נתוני העבר באופן אוטומטי. לכן ייתכן שבתהליך חיפוש ידני ניתן יהיה למצוא אנומליות נוספות בשדות אחרים במסד הנתונים, ובכך לא להגביל את החיפוש רק למיקום האוטובוס, וזה עשוי להיות יתרון לטובת השיטה המקורית שמוצגת במאמר.

עם זאת, ישנו הבדל מרכזי בין מסדי הנתונים לדוגמא שמוצגים במאמר לבין מסד הנתונים שלנו: המאמר מתאר שימוש של המערכת במסדי נתונים שאמנם משתנים עם הזמן באופן תדיר יחסית, אך באופן פחות תדיר משל מסד הנתונים של האוטובוסים בדבלין. דוגמא לכך מן המאמר היא מסד נתונים שמכיל את ה-infoboxes (טבלאות העובדות המהירות) של הדפים בויקיפדיה. אלו אמנם מתעדכנים באופן תדיר, אך סביר להניח שבאופן פחות תדיר מאשר החיישנים במסד הנתונים של דבלין (ששולחים דיווח עבור כל אוטובוס בכל 20 שניות). לכן, נראה כי חיפוש ידני של ממצאים חריגים בסט הנתונים שלנו הוא קשה מאוד גם לאחר שימוש בכלי שינסה להנגיש אותו. מכאן שלדעתנו השימוש במערכת שתתריע על אנומליות באופן אוטומטי היא נוחה יותר עבור מסד הנתונים של האוטובוסים בדבלין.

## סעיף מתוך דרישות הפרויקט:

## Expand the described approach in terms of data and methodology with respect to the Dublin dataset.

בחרנו לנסות ולשכלל את מערכת זיהוי החריגות במיקומי האוטובוסים על ידי הסברים שהמערכת תנסה לספק עבור נסיעות חריגות. לצורך כך נשתמש בשני סוגי הנתונים החיצוניים ששילבנו עם מסד הנתונים של דבלין: האירועים ההמוניים וציוצי הטוויטר של dublinbusnews.

במידה והמערכת תזהה נסיעה חריגה, היא תתריע על כך למשתמש, ובנוסף היא תציג דיווחים מתוך dublinbusnews שעלו במהלך הנסיעה. המשתמש יוכל לסקור את הדיווחים ולבדוק האם הם עשויים להיות קשורים או להוות הסבר לתנועה החריגה בקו האוטובוס המדובר.
בנוסף, אם המערכת תזהה שהתרחש אירוע המוני באותו זמן באזור סמוך, היא תודיע על כך למשתמש. ייתכן כי אירוע זה הוא הסיבה לשינוי במסלול האוטובוס, והמשתמש יוכל לבדוק זאת.

במהלך קבלת דיווחים חדשים, אנו משלבים אותם באופן אוטומטי עם הנתונים החיצוניים באופן שתואר בחלק אינטגרציה של נתונים חיצוניים במסמך זה. כאשר המערכת תזהה נסיעה חריגה, היא תבדוק את השדות $event\_around$ ו- $relevant\_tweet$ (השדות

שנוספו על סמך הנתונים החיצוניים) בדיווחים הרלוונטיים לאותה הנסיעה, ותשתמש בהם בכדי לנסות ולספק הסבר לנסיעה החריגה.

## האפליקציה ואופן השימוש בה:

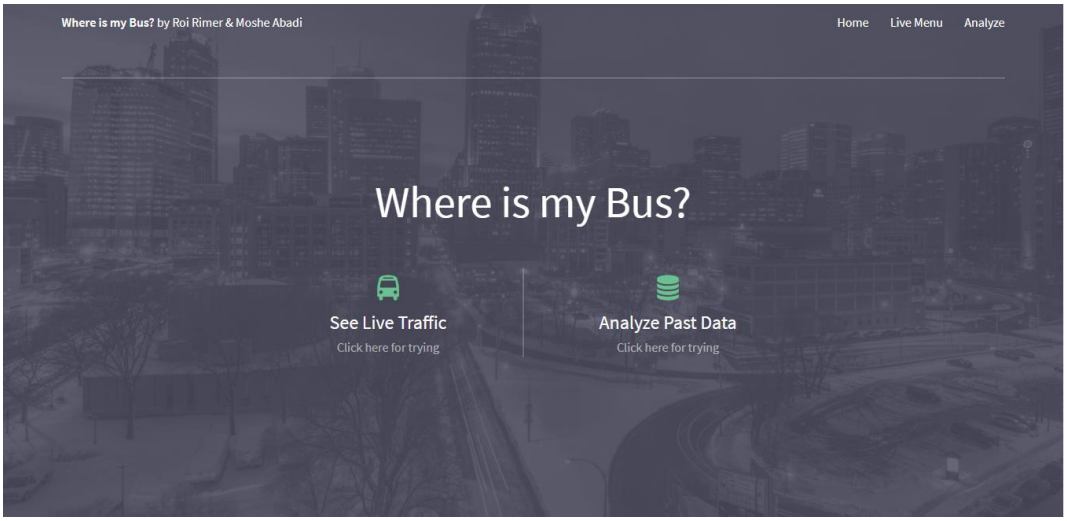
**הערה:** כאן אנו מציגים מידע על האפליקציה ועל השימוש בממשק שלה. למידע טכני מלא על אופן הרצת הפרויקט כולו יש לקרוא את קובץ README.md שנמצא ב-repository ב-GitHub.

בנינו אתר המאפשר ניטור של נסיעות חריגות בזמן אמת, וניתוח של נסיעות שהתרחשו בעבר לצורך מחקר. האתר גם יאפשר לבדוק את אמת הדיוק של המערכת בחיזוי טווחי ה-$delay$ אל מול טווחי ה-$delay$ האמיתיים עבור דיווחים הקשורים לנסיעות החריגות בזמן אמת, או לדיווחים על נסיעות חריגות מן העבר. ערכי ה-$delay$ יכולים לעניין את המשתמש, למשל הם יכולים להוות אינפורמציה משלימה עבור נסיעות שהתגלו כחריגות. השוואה בין טווח ה-$delay$ שנחזה ל-$delay$ האמיתי יכול להוות "בדיקת שפיות" עבור המערכת שחוזה את ה-$delay$ לצורך שימושם של המפקחים על המערכת.
האפליקציה תציג גם מידע נוסף שימושי עבור נסיעות חריגות (בזמן אמת או מן העבר) – האם היו אירועים המוניים קרובים בזמן הנסיעה, והאם היו ציוצים מ-dublinbusnews בזמן הנסיעה.

**לגבי המחברת מה-databricks שנגיש:** במחברת שאנו מגישים ישנה אפשרות להרצת stream של נתונים ולהעלאת batch. ניתן לציין ב-widgets ייעודיים את ה-ip של שרת ה-Kafka במקרה של stream, ואת ה-path הרצוי במקרה של batch. עם הרצת המחברת, יתאמן מודל החיזוי של טווחי ה-$delay$ על ה-batch הנתון (עלול לקחת זמן), ואז יתחיל תהליך ה-streaming ועדכון הנתונים בזמן אמת.

נתאר כעת את הדפים השונים באתר, ואת אופן השימוש בהם:

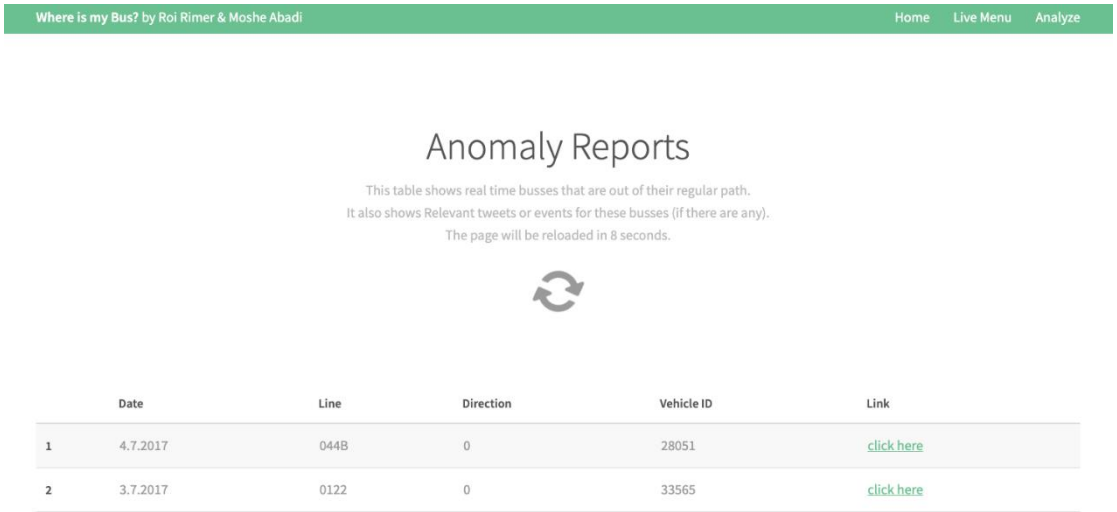
- **<u>דף הבית:</u>** צילום מסך של דף הבית ניתן לראות באיור 4:




*איור 4 – דף הבית של האתר*

בדף הבית מוצגים שני אייקונים. לחיצה על האייקון השמאלי (או על הכיתוב שמתחתיו) תוביל לדף של סקירת האנומליות בזמן אמת. לחיצה על האייקון הימני תוביל לדף של ניתוח נתוני העבר. בנוסף, בפינה הימנית העליונה ישנו תפריט המאפשר מעבר בין דפי האתר המרכזיים.

- **<u>דף סקירת אנומליות בזמן אמת:</u>** צילום מסך של דף זה ניתן לראות באיור 5:




*איור 5 – דף סקירת האנומליות בזמן אמת*

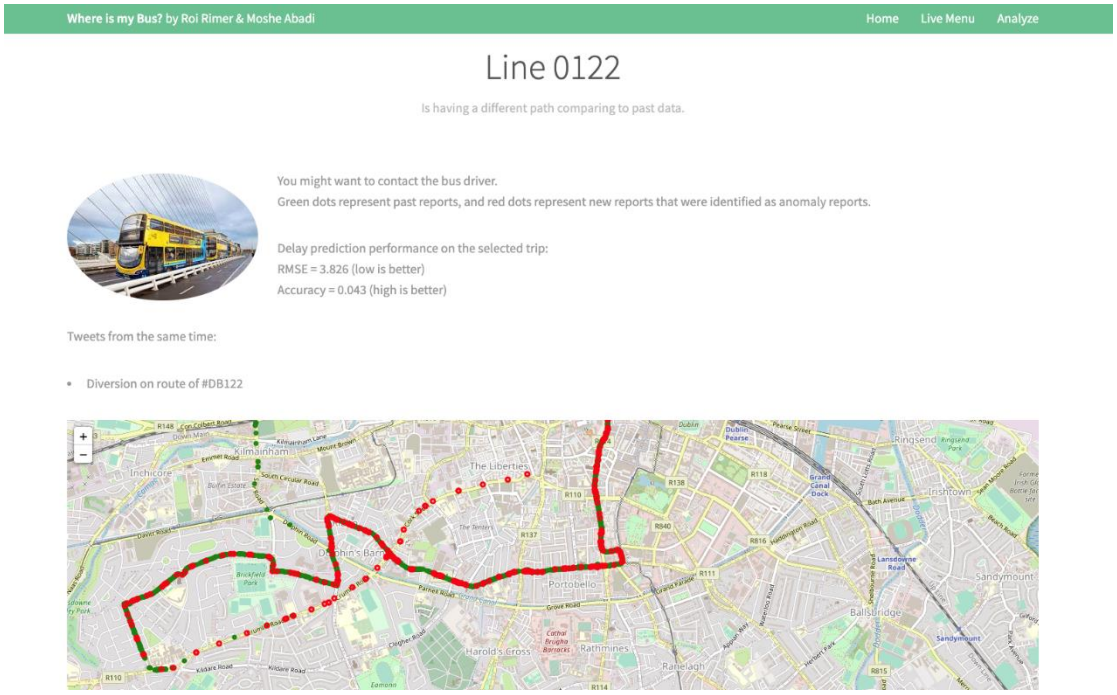
בדף זה מוצגות נסיעות שזוהו על ידי המערכת ככאלו שהכילו אנומליה, כלומר שהתקבלו בהם שלושה דיווחים רצופים שהיו בעלי מיקום חריג ביחס לנסיעות עבר. הדף מציג את כל הנסיעות החריגות שנמצאו עד כה, מן החדשה ביותר אל הישנה ביותר. בנוסף, הדף

מרענן את עצמו בכל 30 שניות בכדי שנסיעות חריגות שיימצאו בינתיים יוכלו להתווסף. הטבלה של הנסיעות החריגות מכילה נתונים על תאריך הנסיעה, מספר הקו, כיוון הנסיעה (בוליאני- הלוך או חזור), ומספר מזהה של הרכב.

בנוסף, עבור כל נסיעה ישנו לינק לדף ייעודי עבור הנסיעה. דף זה יציג על מפה עד דיווחי עבר מאותו היום של אותו הקו עבור אותו כיוון נסיעה. דיווחים של נסיעה רגילה ביום זה (אם היו) יהיו בצבע ירוק, ויעזרו למשתמש להבין את המסלול הרגיל שעובר האוטובוס. בצבע אדום יוצגו דיווחי הנסיעה שנחשבת כחריגה. המשתמש יוכל להשוות בין הנסיעה החריגה לנסיעות עבר ולזהות בצורה נוחה את ההבדלים בין המסלולים.

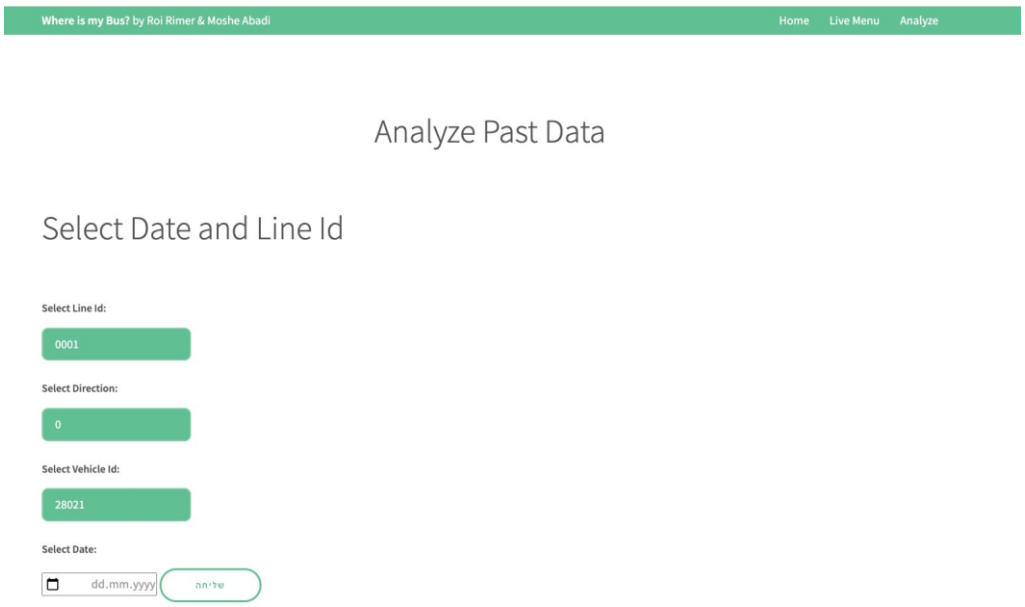
בנוסף, למשתמש יוצגו בעמוד זה מעל המפה ציוצים רלוונטיים שעלו בזמן הנסיעה, ודיווחים על אירועים סמוכים הרלוונטיים לנסיעה זו. תוצג גם רמת האיכות של חיזוי טווחי ה-$delay$ עבור הנסיעה החריגה, מבחינת שני מדדים: RMSE, Accuracy. המחשה לכך ניתן לראות באיור 6:

**<u>הערה</u>**: מומלץ לפתוח את הלינקים לדף הייעודי עבור נסיעה חריגה ב-tab חדש, זו כדי שריענון העמוד לא יפריע לטעינת הדף החדש.




*איור 6 – המחשה של הצגת מפה עבור נסיעה חריגה שאותרה. דיווחי הנסיעה החריגה צבועים באדום, ודיווחים מנסיעות עבר של אותו הקו צבועים בירוק. ניתן לראות שלנסיעה הצבועה באדום קטע הרחוק מן הדיווחים הצבועים בירוק. פרטים נוספים הקשורים לנסיעה מוצגים מעל המפה.*

- **<u>דף ניתוח נתוני העבר:</u>** צילום מסך של דף זה ניתן לראות באיור 7:




*איור 7 – דף ניתוח נתוני העבר*

בדף זה ניתן לצפות בנתוני עבר על סמך בחירת מספר קו (מתוך רשימה של אפשרויות), כיוון נסיעה (0 או 1), מספר רכב (מתוך רשימה של אפשרויות) ותאריך. לאחר לחיצה על הכפתור שליחה (submit), תוצג מפה (ראה איור 6) עם הדיווחים שהתקבלו מאותו קו אוטובוס בתאריך הנתון. דיווחים מן הנסיעה המבוקשת (המתאימה לפרטים שהזין המשתמש) יהיו צבועים על המפה בצבע אדום (רק שהפעם זו לא בהכרח נסיעה חריגה), ושאר הדיווחים המתאימים לקו הנבחר מאותו היום ייצבעו בירוק.

אם הנסיעה המבוקשת הייתה נסיעה חריגה אז תוצג למשתמש בנוסף רמת האיכות של חיזוי טווחי ה-$delay$ עבור הנסיעה המבוקשת, מבחינת שני מדדים: RMSE, Accuracy.

במקרה זה יוצגו גם ציוצים רלוונטיים שהופיעו בזמן הנסיעה המבוקשת, ואירועים המוניים שהיו בקרבת הנסיעה המבוקשת בזמן שהתקיימה.

אם עבור הקלט המבוקש אין אף דיווחים רלוונטיים, המשתמש יופנה לדף מיוחד שיודיע לו את זה, ויאפשר לו להשתמש בתפריט כדי לחזור לדפים אחרים באתר.

# סיכום

במהלך הפרויקט עסקנו בשתי משימות חיזוי, ובנינו אפליקציה המבצעת את משימות החיזוי על דאטה חדש שמגיע, ומאפשרת לקבל דיווחים בזמן אמת על נסיעות שהתגלו כחריגות מבחינת מיקום האוטובוס. האפליקציה מאפשרת גם לבחון נסיעות עבר ואת מסלולן, ולקבל פרטים נוספים במקרה שנסיעה זו הייתה חריגה.

במהלך הפרויקט התנסינו במגוון כלים, כמו עבודה עם batch data ו-stream data, שילוב של נתונים חדשים (למשל על ידי שימוש ב-API של טוויטר), השלמת נתונים חסרים, בניית אפליקציה לצורך הצגה ויזואלית של התוצאות ועוד. למרות הרבה קשיים ולמידה מטעויות בדרך, אנו מקווים שהעבודה שלנו משקפת את המאמץ הרב שהושקע בה.