Data Analytics for Business (mini 1&2) 2264-5

Dr. Gilli Shama

Big Data & Predictive AI, MBA program | IDC

08/03/2018

Bay Area Bike-share at San Francisco

Prediction model for future target destination based on rental information

Team members

Yarden Israeli, 304891047

Amitai Serphos, 200211159

Omer Klein, 200070894

אבסטרקט

שוק הרכיבה המשותפת על אופניים חווה צמיחה חדה בכל העולם. בארה"ב לבדה בוצעו מעל 88 מיליון נסיעות בעזרת מערכות שיתוף אופניים משנת 2010, כאשר בשנת 2010 בפרט בוצעו מעל 28 מיליון נסיעות השווים לכמות הנסיעות בכל מערכת הרכבות בארה"ב בשנה, וגבוה יותר ממספר המבקרים השנתי בוורלד דיסני באורלנדו. בסין קיימים כבר מעל 50 מיליון משתמשים בשירותי שיתוף האופניים.

העובדה המעניינת היא שנכון להיום, על אף הטרנד ההולך וגדל בעולם, חברות שיתוף האופניים אינן רווחיות כלל. לא נמצא מנגנון עסקי המאפשר רווח ועל כן המערכות נסמכות על תמיכה עירונית וממשלתית.

השאלה איך להפוך את עולם שיתוף האופניים לענף רווחי הפכה לשאלה קריטית ברחבי העולם, והתשובה לכך היא יצירת מערכת הודעות פרסום מבוססת יעד **"Destination Based Ad Network"**.
היכולת לנבא את מיקומם העתידי של אנשים, ובעולם שיתוף האופניים המשמעות הינה היכולת לנבא את תחנת הסיום של הרוכב עם תחילת נסיעתו, מייצרת הזדמנות עסקית להתקשר עם עסקים קטנים ובינוניים ולהציע להם פלטפורמה לפרסום מוכוון לקוח ברמה הכי גבוה ומדויקת שניתן.

פיתחנו מודל המבוסס על מערכת שיתוף האופניים של סן פרנסיסקו הכוללת 3 שנים של מעל מיליון נסיעות. המטרה הייתה לחזות את תחת הקצה של הלקוח. למרות שהגישה הקלאסית לפתרון לשאלה מסוג זה הייתה ליישם מודל קלסיפיקציה על הנתונים, אנחנו בחרנו מודל מתקדם ומדיוק יותק המבוסס על מודל פתרון בעזרת רגרסיה.

בחודשים הקרובים נשגר את רשת המודעות שלנו, המיועדת לעסקים מקומיים קטנים ובינוניים, בהתבסס על דיוק המודל שלנו ובגיבוי שוק בינלאומי של שיתוף האופניים ההולך וגדל.

תוכן עניינים

Contents

רשימת איורים

פרק 1 – Business Goal

רקע

שוק הרכיבה המשותפת על אופניים חווה צמיחה חדה בכל העולם. בארה"ב לבדה בוצעו מעל 88 מיליון נסיעות בעזרת מערכות שיתוף אופניים משנת 2010, כאשר בשנת 2010 לבדה בוצעו מעל 28 מיליון נסיעות- מספר השווה לכמות הנסיעות בכל מערכת הרכבות בארה"ב בשנה, וגבוה יותר ממספר המבקרים השנתי בוורלד דיסני באורלנדו.
בסין קיימים כבר מעל 50 מיליון משתמשים בשירותי שיתוף האופניים.

העובדה המעניינת היא שנכון להיום, על אף הטרנד ההולך וגדל בעולם, חברות שיתוף האופניים אינן רווחיות כלל. לא נמצא מנגנון עסקי המאפשר רווח ועל כן המערכות נסמכות על תמיכה עירונית וממשלתית.

אנחנו התחלנו את הדרך עם מידע על מערכת שיתוף האופניים בעיר סן פרנסיסקו בארה"ב. בניתוח ראשוני של המידע ראינו שישנן תחנות אופניים עם פעילות רבה לעומת תחנות פעילות פחות. מצאנו קשר לתחבורה ציבורית בתוך העיר ומחוץ לעיר. בהמשך הלכנו ובדקנו מסלולי נסיעה ומצאנו מסלולים פופולאריים לעומת מסלולים יחסית זניחים.

מציאת מסלולים פופולאריים היוותה נקודת מפנה עבורנו, בה הבנו מהי השאלה העסקית שלנו. למעשה, מידע זה איפשר לנו את ההשערה שישנם קשרים בין תחנות ונסיעות, אשר בעזרת אימון מאסיבי של מודל ML נוכל אולי נצליח לחזות את המסלולים העתידיים. אפשרות זו טומנת בחובה לגלות מידע על מיקומם העתידי של אנשים, אשר הינו רב שימושים ובעל פוטנציאל עסקי גבוה.

השאלה העסקית

אנו מעוניינים לחזות את תחנת היעד של הרוכב הבא שישתמש בשיתוף האופניים.
חיזוי זה יפתח לנו צוהר לפלטפורמה חדשה, בה נוכל להציע לבעלי עסקים באזור לשווק ולפרסם את מוצריהם ושירותיהם באופן ממוקד עבור לקוחות עתידיים אשר משתמשים באופניים של השיתוף. כפי שראינו, שיווק ממוקד זה יכול כבר היום לפנות למיליוני הרוכבים הקיימים ובכך קיים פוטנציאל אדיר.

מודל ML

זוהי בעיה מסוג Supervised Learning, אשר תחילה ניגשנו אליה כבעיית Classification קלאסית במטרה לייצג כל תחנת אופניים כ-Class בודד. בהמשך עברנו לאימון מודל מתוחכם יותר מסוג Regression, על ידי חיזוי המיקום הגיאוגרפי העתידי, תוך שימוש בייצוג של קואורדינטות עולם. האימון התבסס על נסיעות האופניים שהתרחשו בעבר.

פרק 2 – Data preprocessing

ה-Database שלנו היה אודות שירות שיתוף האופניים של עיריית סן-פרנסיסקו. מידע זה הינו Public dataset של Google cloud platform, כאשר קישור אליו נמצא כאן.

המידע הגולמי ביותר מבוסס על נסיעות אופניים בכל אזורי שיתוף האופניים של סן-פרנסיסקו החל מאוגוסט 2013, כאשר הוא התעדכן ברמה היומית וכלל קרוב למיליון נסיעות.
נראה זאת-

```
## 'data.frame':    983648 obs. of  12 variables:
##  $ X                 : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ trip_id           : int  944732 984595 984596 1129385 1030383 1102641 969490 1129386 947105 1011650 ...
##  $ duration_sec      : int  2618 5957 5913 6079 5780 801 255 6032 1008 60 ...
##  $ start_date        : chr  "2015-09-24 17:22:00" "2015-10-25 18:12:00" "2015-10-25 18:13:00" "2016-03-18 10:33:00" ...
##  $ start_station_name: chr  "Mezes" "Mezes" "Mezes" "Mezes" ...
##  $ start_station_id  : int  83 83 83 83 83 83 83 83 83 83 ...
##  $ end_date          : chr  "2015-09-24 18:06:00" "2015-10-25 19:51:00" "2015-10-25 19:51:00" "2016-03-18 12:14:00" ...
##  $ end_station_name  : chr  "Mezes" "Mezes" "Mezes" "Mezes" ...
##  $ end_station_id    : int  83 83 83 83 83 83 83 83 83 83 ...
##  $ bike_number       : int  653 52 121 208 44 174 650 155 157 35 ...
##  $ zip_code          : chr  "94063" "nil" "nil" "94070" ...
##  $ subscriber_type   : chr  "Customer" "Customer" "Customer" "Customer" ...
```

Figure1 : raw database on bike trips at San Francisco (taken from Big Query)

ניתן לראות כי כל שורה ב-data מייצגת נסיעת אופניים בודדת אשר מאופיינת במספר פרמטרים כגון- משך זמן הנסיעה, תחנת התחלה וסיום, תאריך, שעה ומידע אודות הרוכב (מנוי/לקוח מזדמן).

נבחין כי מרבית המידע הינו קטגורי, בפרט המידע אודות תחנת האופניים עצמה. לכן בנוסף ל-database הקיים השתמשנו ב-database משלים אשר נמצא באותו מאגר מידע ב-Google cloud (נמצא באותו הקישור מעלה) המכיל מידע ספציפי על כל תחנה ותחנה, כך שמעבר ל-ID מסוים, כעת יהיה לנו גם את מיקומם הגיאוגרפי בעולם ועוד.
נראה זאת-

```
## 'data.frame':    74 obs. of  7 variables:
##  $ station_id       : int  4 37 35 32 84 8 9 3 13 10 ...
##  $ name             : chr  "Santa Clara at Almaden" "Cowper at University" "University and Emerson" "Castro Street and El
Camino Real" ...
##  $ latitude         : num  37.3 37.4 37.4 37.4 37.3 ...
##  $ longitude        : num  -122 -122 -122 -122 -122 ...
##  $ dockcount        : int  11 11 11 11 15 15 15 15 15 15 ...
##  $ landmark         : chr  "San Jose" "Palo Alto" "Palo Alto" "Mountain View" ...
##  $ installation_date: chr  "06-08-13" "14-08-13" "15-08-13" "31-12-13" ...
```

Figure2 : raw database on bike stations at San Francisco (taken from Big Query)

כלומר טבלת נתונים זו מכילה את כל המידע המשלים על 74 התחנות הקיימות בסן פרנסיסקו.

לאחר בחינה מקדימה של הנתונים הבחנו כי כל אזור (המצוין ב- "landmark" ב-figure 2) הינו מרוחק מאוד מהשני, כך שאין כלל קשר בין נסיעות ותחנות מאזורים שונים.
מסקנה זו הינה חשובה, שכן לאמן מודל המבוסס על נסיעות מאזור אחד ולבדוק את הביצועים מנסיעות שהתרחשו בכלל באזור אחר, עם תחנות אחרות, היה מביא לשגיאה עקרונית במימוש.
לכן נאלצנו לצמצם את ה-database שלנו ובחרנו להתמקד באזור ה- Bay Area בלבד (כאשר ניתן כמובן להתאים את המודל בהמשך לאזורים אחרים).
על-פי החלטה זו סיננו את המידע המקורי ובנוסף איחדנו את שתי הטבלאות מעלה לטבלה משותפת.

קיבלנו את ה-data הבא
(עמוד הבא)-

```
## 'data.frame':    891200 obs. of  21 variables:
##  $ trip_id           : int  881807 118590 75820 133436 527919 204230 1042935 132337 167215 811717 ...
##  $ duration_sec      : int  6608 1204 4663 34426 4529 7593 88 1109 894 190 ...
##  $ start_date        : chr  "2015-08-09 08:09:00" "2013-12-08 10:52:00" "2013-10-29 14:11:00" "2013-12-22 11:18:00" ...
##  $ start_station_name: chr  "2nd at Folsom" "2nd at Folsom" "2nd at Folsom" "2nd at Folsom" ...
##  $ start_station_id  : int  62 62 62 62 62 62 62 62 62 62 ...
##  $ start_lat         : num  37.8 37.8 37.8 37.8 37.8 ...
##  $ start_lon         : num  -122 -122 -122 -122 -122 ...
##  $ start_dockcount   : int  19 19 19 19 19 19 19 19 19 19 ...
##  $ start_landmark    : chr  "San Francisco" "San Francisco" "San Francisco" "San Francisco" ...
##  $ start_insallDate  : chr  "22-08-13" "22-08-13" "22-08-13" "22-08-13" ...
##  $ end_date          : chr  "2015-08-09 09:59:00" "2013-12-08 11:12:00" "2013-10-29 15:29:00" "2013-12-22 20:52:00" ...
##  $ end_station_name  : chr  "2nd at Folsom" "2nd at Folsom" "2nd at Folsom" "2nd at Folsom" ...
##  $ end_station_id    : int  62 62 62 62 62 62 62 62 62 62 ...
##  $ end_lat           : num  37.8 37.8 37.8 37.8 37.8 ...
##  $ end_lon           : num  -122 -122 -122 -122 -122 ...
##  $ end_dockcount     : int  19 19 19 19 19 19 19 19 19 19 ...
##  $ end_landmark      : chr  "San Francisco" "San Francisco" "San Francisco" "San Francisco" ...
##  $ end_insallDate    : chr  "22-08-13" "22-08-13" "22-08-13" "22-08-13" ...
##  $ bike_number       : int  408 507 276 388 492 492 53 537 536 469 ...
##  $ zip_code          : chr  "19103" "94105" "95008" "78230" ...
##  $ subscriber_type   : chr  "Customer" "Subscriber" "Customer" "Customer" ...
```

Figure3 : San-Francisco Bay Area – trips & station

לצורך חישובי סטטיסטיקות, נדרשנו כשלב מקדים לסדר את הנתונים כך שיהיו קלים יותר לסינון ועבודה. אם כן, ביצענו מספר מניפולציות-

1. <u>סידור תאריך ושעת נסיעה</u>

במידע המקורי ישנו משתנה המתאר את זמן ההתחלה וזמן הסיום. משתנה זה הינו מסוג time stamp בפורמט של Y-m-d h-m-s. בכדי לעבוד עם מידע הנוגע לתאריכי התחלה וסיום, שעות התחלה וסיום, ימים בשבוע או אמצע שבוע אל מול סוף שבוע, היה עלינו ראשית לבצע המרה מתאימה (השתמשנו ב- POSIXct, מופיע בקוד).

2. <u>משך זמן נסיעה</u>

זמן הנסיעה הופיע בשניות, אך לצרכי נוחות העברנו את המשתנה לדקות.

לאחר מכן קיבלנו את הטבלה הבאה-

```
## 'data.frame':    891200 obs. of  27 variables:
##  $ trip_id           : int  279974 1280844 547157 1286629 268085 301428 338906 1046500 355017 983502 ...
##  $ duration_sec      : int  438 328 764 602 489 766 553 1290 775 896 ...
##  $ start_date        : chr  "2014-05-12 07:33:00" "2016-07-18 16:43:00" "2014-11-17 09:17:00" "2016-07-22 08:53:00" ...
##  $ start_station_name: chr  "Golden Gate at Polk" "Embarcadero at Sansome" "San Francisco Caltrain 2 (330 Townsend)" "Mar
ket at 10th" ...
##  $ start_station_id  : int  59 60 69 67 57 56 61 50 45 69 ...
##  $ start_lat         : num  37.8 37.8 37.8 37.8 37.8 ...
##  $ start_lon         : num  -122 -122 -122 -122 -122 ...
##  $ start_dockcount   : int  23 15 23 27 15 19 27 23 15 23 ...
##  $ start_landmark    : chr  "San Francisco" "San Francisco" "San Francisco" "San Francisco" ...
##  $ start_insallDate  : chr  "21-08-13" "21-08-13" "23-08-13" "23-08-13" ...
##  $ end_date          : chr  "2014-05-12 07:41:00" "2016-07-18 16:48:00" "2014-11-17 09:30:00" "2016-07-22 09:03:00" ...
##  $ end_station_name  : chr  "Yerba Buena Center of the Arts (3rd @ Howard)" "Steuart at Market" "Market at 10th" "Market
at 4th" ...
##  $ end_station_id    : int  68 74 67 76 65 69 77 55 61 59 ...
##  $ end_lat           : num  37.8 37.8 37.8 37.8 37.8 ...
##  $ end_lon           : num  -122 -122 -122 -122 -122 ...
##  $ end_dockcount     : int  19 23 27 19 15 23 27 23 27 23 ...
##  $ end_landmark      : chr  "San Francisco" "San Francisco" "San Francisco" "San Francisco" ...
##  $ end_insallDate    : chr  "23-08-13" "25-08-13" "23-08-13" "25-08-13" ...
##  $ bike_number       : int  542 340 267 375 313 291 483 283 542 611 ...
##  $ zip_code          : chr  "94109" "94595" "95130" "75231" ...
##  $ subscriber_type   : chr  "Subscriber" "Subscriber" "Subscriber" "Customer" ...
##  $ duration_minutes  : num  7.3 5.47 12.73 10.03 8.15 ...
##  $ year              : int  2014 2016 2014 2016 2014 2014 2014 2015 2014 2015 ...
##  $ Month             : chr  "May" "July" "November" "July" ...
##  $ Weekday           : chr  "Monday" "Monday" "Monday" "Friday" ...
##  $ dayStatus         : chr  "weekday" "weekday" "weekday" "weekday" ...
##  $ Hour              : int  7 16 9 8 13 17 12 14 16 17 ...
```

Figure4 : San-Francisco Bay Area – trips & station, after set of dates & duration set + addition of columns

להלן מספר סטטיסטיקות מעניינות

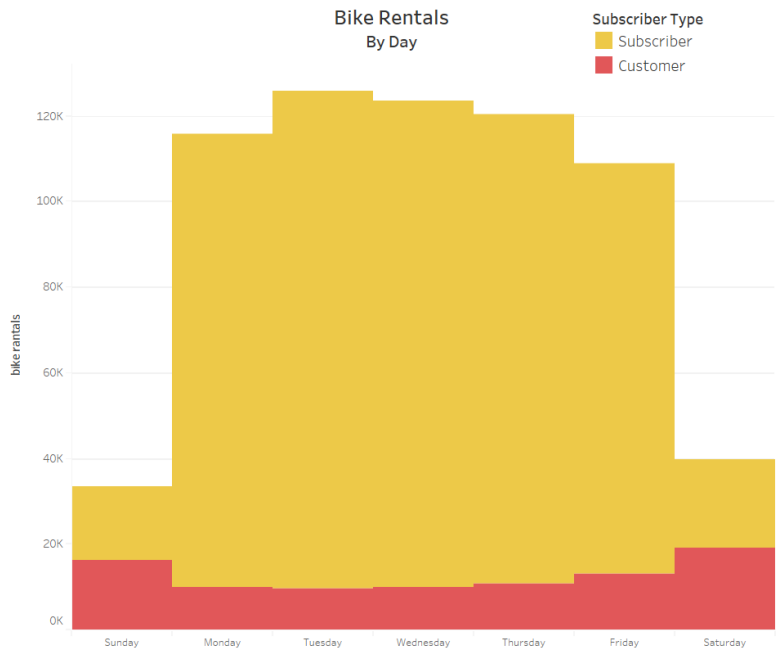


Figure5 : San-Francisco Bay Area - Histogram of bike rentals per day, divided into subscriber & customer

במהלך סופי השבוע (ימים שבת-ראשון) מספר הנסיעות מצטמצם משמעותית, בנוסף חלוקת הנסיעות בין סוגי הרוכבים הינה כזו שמרביתן מתרחשות על-ידי רוכבים מנויים. עובדה זו מעניינת אותנו מאוד, שכן בהמשך נתמקד יותר בקהל יעד זה.

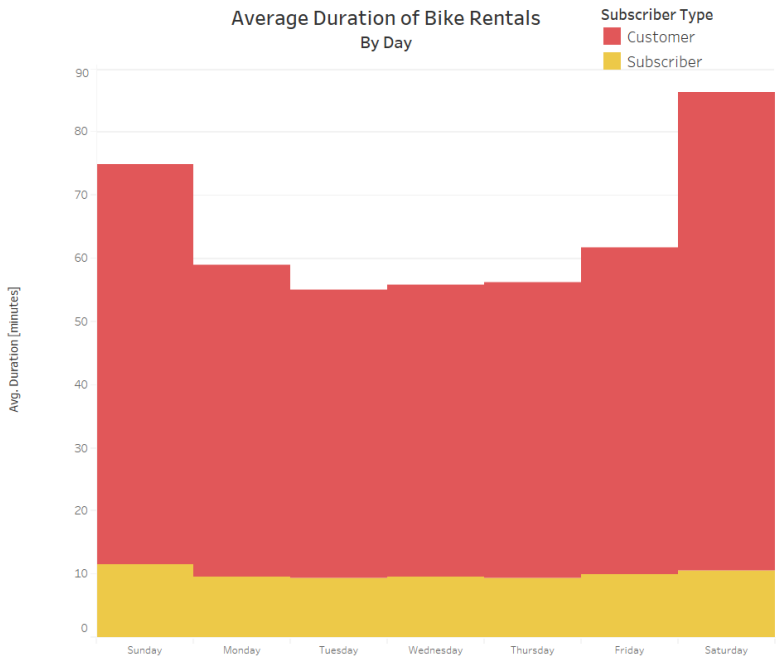


Figure6 : San-Francisco Bay Area - Histogram of Average duration per day, divided into subscriber & customer

מכאן אנו למדים כי נסיעה ממוצעת של מרבית רוכבי האופניים המנויים נמשכת כמספר דקות בודדות.

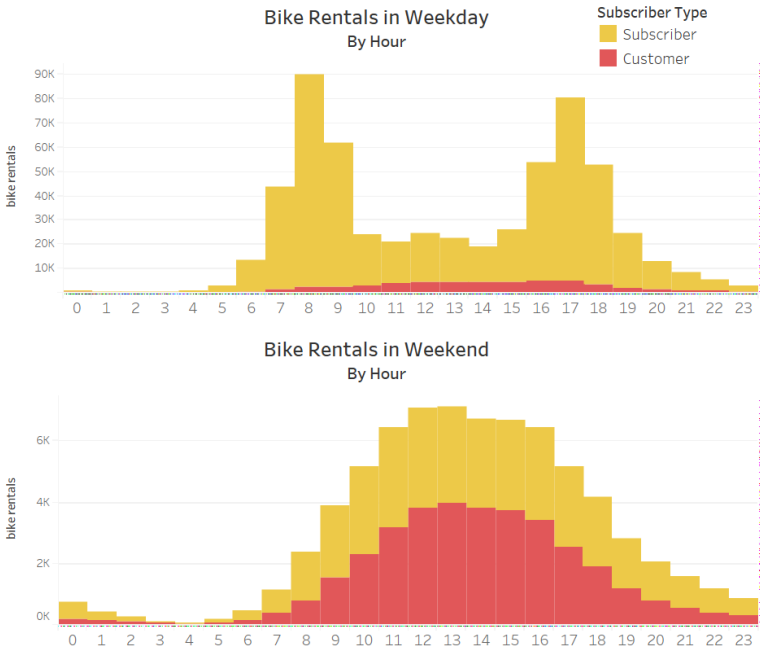


Figure7 : San-Francisco Bay Area - Histogram of Hours, divided into subscriber & customer

בנוסף ניתן לראות כי באמצע שבוע מרבית נסיעות האופניים מתרחשות בשעות הבוקר המוקדמות או בשעות הערב המודקמות, בעוד שבסופי שבוע מרבית הנסיעות מתרחשות לאורך כל היום.

אמנם הגרף המעניין ביותר קשור למסלולים השונים. למעשה אספנו את כל ה-database שלנו וחילקנו את הנסיעות לפי המסלול אליהן הן משתייכות – כאמור כל נסיעה מייצגת מסלול אחד מתחנת התחלה לסיום, וישנן כמובן נסיעות רבות בעלות אותו המסלול, לכן הדבר אפשרי.

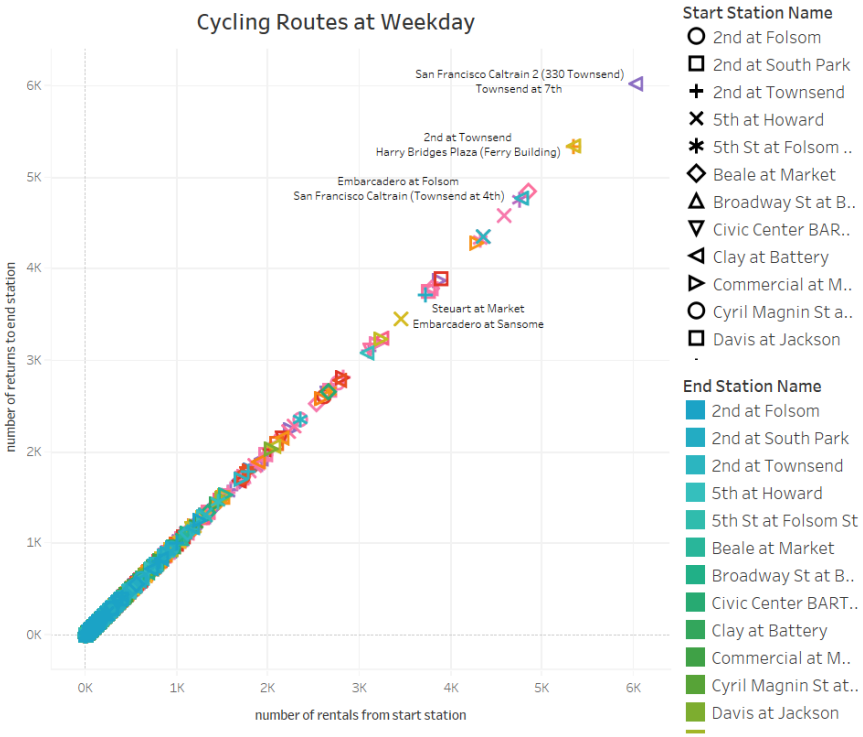


Figure8 : San-Francisco Bay Area- Cycling routes distribution

כפי שתיארנו בפרק הקודם, גרף זה למעשה אישש לנו את ההשערה שישנן מסלולים שחוזרים על עצמם ועל כן ישנו קשר בין הנסיעות השונות. אנו מצפים כי באמצעות קשר זה אלגוריתמי ML יצליחו להתכנס על פני ה-database שלנו, ולכן זוהי אבחנה חשובה ביותר.

אם כן, לאחר שלמדנו על אופי הנסיעות, נרצה כעת ללמוד על התחנות והקשר ביניהן.
בשלב הראשון הצגנו על מפה את כל התחנות, כך שהגדרנו את גודל התחנה וצבעה בהתאם לפופולאריות שלה-

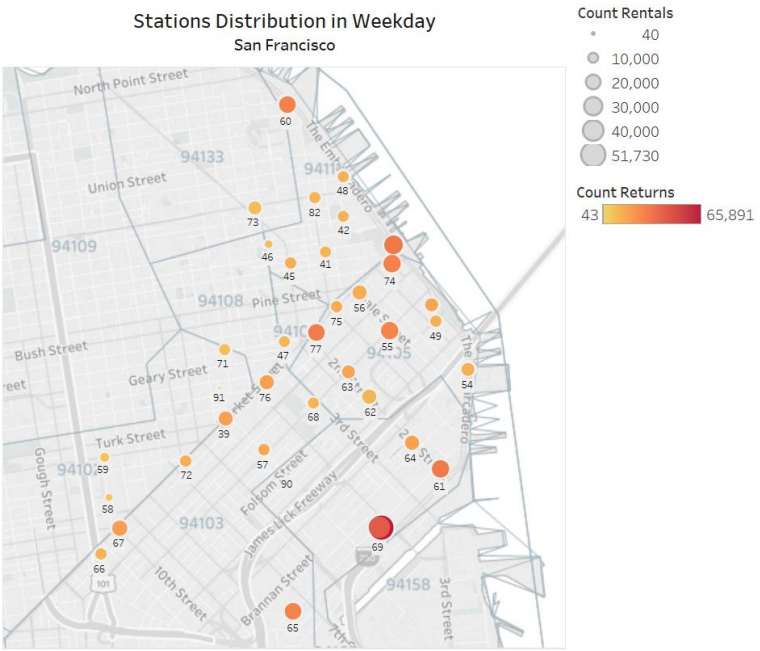


Figure9 : San-Francisco Bay Area - Station distribution & popularity placed on Map

נבחין כי ישנן מספר תחנות חזקות בעלות Traffic גבוה מאוד, לצד תחנות פעילות פחות. בנוסף הפריסה הגיאוגרפית שלהן היא על פני עשרות ק"מ בודדים. במטרה להבין מהו מקור הפופולאריות של תחנות מסוימות ואת סיבת מיקומן, הוספנו על גבי המפה מידע על מיקומי תחנות תחבורה ציבורית- כולל מידע על אוטובוסים, Muni, Bart ותחנות מרכזיות בסביבה. המידע סופק באדיבות חברת Moovit, המחזיקה את מאגר מידע התחבורה הציבורית הגדול בעולם (קישור ב-Codebook).
להלן המפה המעודכנת-

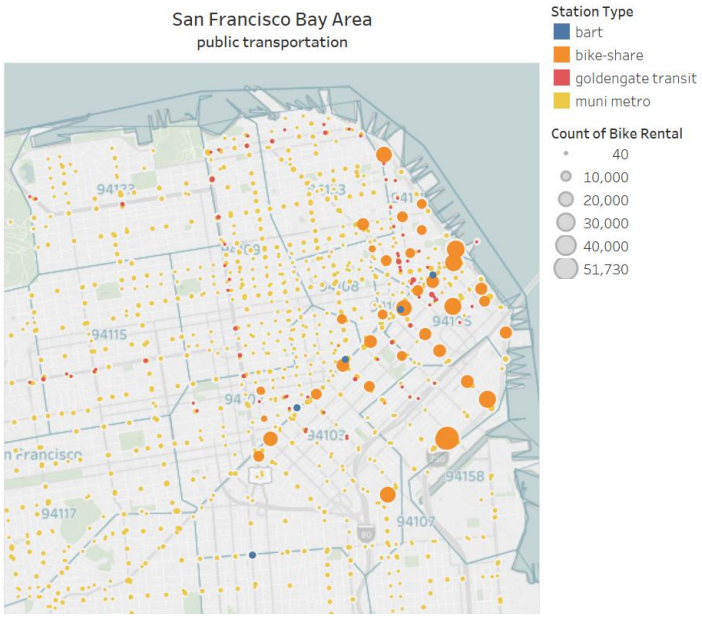


Figure10 : San-Francisco Bay Area – Station distribution & public transportation map

אחת המסקנות הבולטות ביותר הינה שמרבית תחנות האופניים פרוסות על קו פריסת תחנות ה-Bart (מסומנות בכחול), כלומר ככל שהמיקום קרוב יותר ל-Bart, כך הסיכוי שתחנת אופניים תימצא הינו גבוה יותר.
הסבר פשוט לכך נובע מ-zoom out על האזור והבנה כי ה-Bart הינה הרכבת שמחברת את ה-Bay Area אל אזורים מחוצה לו, ולכן הגיוני שמרבית האנשים שיגיעו/יעזבו את האזור ישתמשו בה (ובאופניים שבסמוך לתחנות שלה)

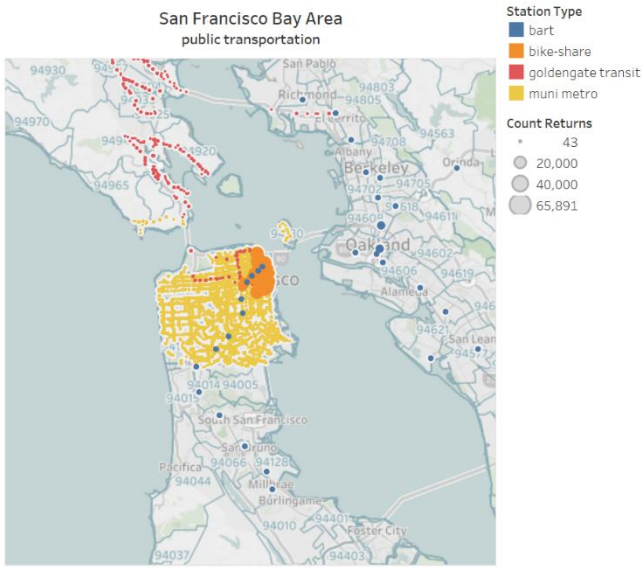


Figure11 : San-Francisco Bay Area – zoom out

בנוסף לאחר research מעמיק על אזור זה למדנו כי סיבה נוספת לפופולאריות של תחנה הינה הקרבה לתחנות Ferry וכן לתחנת רכבת caltrain.
כעת, על-מנת לאחד את כל המידע הגיאוגרפי ל-database שלנו ביצענו חישוב עבור כל תחנה בנפרד של המרחק המינימאלי ממנה אל תחנת ה-Bart הקרובה ביותר, תחנת ה-Muni, Caltrain ו-Ferry. כאשר שוב, את נתוני המיקום הגיאוגרפי של כל אחת מן תחנות התחבורה הציבוריות המעניינות לקחנו הן מ-Moovit והן מ-Google Maps, לאחר בחינה ממושכת ולמידת האזור.
קיבלנו את ה-database הבא -
(בנוסף בשלב זה גם זרקנו משתנים לא מעניינים כגון שמות של תחנות, והפכנו את המידע הקטגורי למידע נומרי)

```
## 'data.frame':    891200 obs. of  21 variables:
##  $ X              : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ trip_id        : int  153682 543180 44843 728494 469783 513185 215174 1175720 761765 246012 ...
##  $ start_station_id: int  39 39 39 39 39 39 39 39 39 39 ...
##  $ start_lat      : num  37.8 37.8 37.8 37.8 37.8 ...
##  $ start_lon      : num  -122 -122 -122 -122 -122 ...
##  $ duration_minutes: num  15.57 5.77 7.17 2.27 8.52 ...
##  $ start_dockcount : int  19 19 19 19 19 19 19 19 19 19 ...
##  $ year           : int  2014 2014 2013 2015 2014 2014 2014 2016 2015 2014 ...
##  $ hour           : int  7 14 18 13 17 7 13 13 14 14 ...
##  $ minDist_bart   : num  77.9 77.9 77.9 77.9 77.9 ...
##  $ minDist_caltrain: num  1438 1438 1438 1438 1438 ...
##  $ minDist_ferry  : num  1845 1845 1845 1845 1845 ...
##  $ minDist_muni   : num  1494 1494 1494 1494 1494 ...
##  $ zipcode        : num  94102 10025 94607 94602 94960 ...
##  $ bike_id        : int  375 529 364 484 489 448 368 505 451 469 ...
##  $ end_station_id  : int  60 68 66 72 50 70 65 67 74 70 ...
##  $ end_lat        : num  37.8 37.8 37.8 37.8 37.8 ...
##  $ end_lon        : num  -122 -122 -122 -122 -122 ...
##  $ month_num      : int  1 11 10 4 9 10 3 4 5 4 ...
##  $ day_status_num  : int  1 1 1 1 1 1 0 1 1 0 ...
##  $ user_type_num   : int  1 0 1 1 1 1 1 1 1 0 ...
```

Figure12 : San-Francisco Bay Area – trips & station, with geographic information

לסיום, לאחר מספר יישומים של אלגוריתמי ML לאימון המודל המבוקש (לחיזוי מטרת היעד), נתקלנו בקושי חישובי משמעותי, לכן נאלצנו לצמצם את ה-database שלנו. בחרנו לעשות זאת על-ידי התמקדות בנסיעות משנת 2016 בלבד (השנה האחרונה ביותר). אתגר מעניין נוסף, עליו נפרט בפרק הבא, היה הצורך לייצג בצורה שונה את מיקום תחנת הסיום של הנסיעה. לאחר מחקר בנושא המרנו את הקואורדינטות הגיאוגרפיות לייצוג UTM, וקיבלנו את ה- database הסופי, עליו אימנו את המודלים שלנו-

```
## 'data.frame':    172966 obs. of  18 variables:
##  $ trip_id         : int  1059051 1311233 1072279 1105120 1258908 1122795 1236506 1143485 1294058 1218986 ...
##  $ start_station_id: int  60 74 59 70 69 69 57 45 48 54 ...
##  $ start_lat       : num  37.8 37.8 37.8 37.8 37.8 ...
##  $ start_lon       : num  -122 -122 -122 -122 -122 ...
##  $ duration_minutes: num  8.65 27.7 13.42 9.32 6.63 ...
##  $ start_dockcount : int  15 23 23 19 23 23 15 15 15 15 ...
##  $ month           : int  1 8 1 2 6 3 6 3 7 5 ...
##  $ day_status      : int  1 1 1 1 1 1 0 1 1 0 ...
##  $ hour            : int  18 7 18 7 8 7 15 17 9 14 ...
##  $ minDist_bart    : num  1429 267 433 1419 1405 ...
##  $ minDist_caltrain: num  3196.1 1929.2 2157.6 44 62.2 ...
##  $ minDist_ferry   : num  1352 171 2717 696 715 ...
##  $ minDist_muni    : num  1891 503 2427 1466 1470 ...
##  $ zipcode         : int  4555 94538 94610 94401 94303 4561 94105 4578 94105 94105 ...
##  $ bike_id         : int  677 675 378 332 274 659 420 451 488 455 ...
##  $ end_station_id  : int  77 65 50 47 57 50 62 74 54 68 ...
##  $ X_end           : num  552757 552603 553335 552525 552383 ...
##  $ Y_end           : num  4182643 4180582 4183287 4182569 4181767 ...
```

Figure13 : San-Francisco Bay Area – final database

לסיום, הקוד המלא אשר שימש אותנו בפרק זה של data preprocessing נמצא בקישור הבא-
https://github.com/YardenIsraeli/Business-Analytics-Bikeshare-Project/tree/master/code/data_preprocessing

להלן טבלה מתוך ה-Codebook, המרכזת את המשתנים העיקריים ב-databasae המקורי שלנו-

Variable Name	Type	Description
trip_id	Integer	Numeric ID of bike trip
duration_sec	Integer	Time of trip in seconds
start_date	Character	Start date of trip with date and time, in PST
start_station_name	Character	Station name of start station
start_station_id	Integer	Numeric reference for start station
end_date	Character	End date of trip with date and time, in PST
end_station_name	Character	Station name for end station
end_station_id	Integer	Numeric reference for end station
bike_number	Integer	ID of bike used
zip_code	Integer	Home zip code of subscriber (customers can choose to manually enter zip at kiosk however data is unreliable)
subscriber_type	Integer	Subscriber = annual or 30-day member; Customer = 24-hour or 3-day member

טבלאות נוספות, הכוללות פירוט על המשתנים העיקריים של המידע החיצוני אותו שילבנו – מידע על מיקומי התחנות ומידע על תחנות ציבוריות, מופיעות ב-Codebook גם כן. ניתן למצוא את ה-Codebook בפרק ה-Appendix.

פרק 3 - פיתוח מודל ML

Classification Approach

כפי שראינו בפרק הראשון, מטרתנו הינה לחזות את תחנת הסיום של הרוכב הבא שישתמש בשיתוף. בפרק השני הצגנו את המידע עליו התבססנו, כאשר ניתן להבחין שעבור כל נסיעה (שורה בדאטה) יש שדה של תחנת הסיום, המאופיין ב-ID הספציפי של התחנה (end_station_id).
אם כן, הגישה הקלאסית הינה להתייחס לבעיה זו כבעיית קלסיפיקציה פשוטה, עם 37 classes – כמספר התחנות, כאשר פרמטר החיזוי הרצוי יהיה ה-ID של התחנה, פרמטר בעל ערך דיסקרטי.

מודל החיזוי שלנו התבסס על עצי החלטה מסוג classification trees, כאשר בחנו מודלים של עצים פשוטים לצד מודלים מורכבים יותר דוגמת Random Forest.
המגבלה העיקרית במודלים אלו הינו חסם עליון על מספר ה-classes לחיזוי לכ-32 classes בלבד, בעוד שמספר התחנות שלנו גדול יותר. בכדי להתגבר על מגבלה זו נאלצנו לצמצם את ה-database, כאשר בשלב הראשון, על מנת לבחון את הביצועים, החלטנו על צמצום באופן רנדומלי ל-10 תחנות התחלה וסיום.
נבחין בעץ החלטה פשוט שהתקבל מתוך למידת סט האימון-
(נלקח מ-Rmarkdown, לינק מצורף בסוף הפרק)

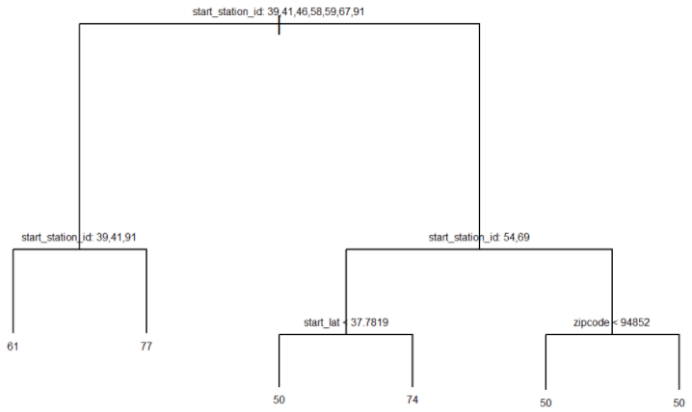


Figure14 : Simple Classification Tree

ניתן לראות כי מתוך 10 תחנות קיבלנו חיזוי ל-4 תחנות בלבד, כלומר אפשר להסיק רק מהתבוננות במודל, טרם חישוב הביצועים אל מול סט הבוחן, כי צפויות לנו החטאות רבות בחיזוי.
גם כאשר ביצענו חלוקה עדינה יותר של העץ (mindev = 0.005) ובנינו מודל מורכב יותר, קיבלנו ניבוי ל-7 תחנות בלבד-
(mindev = 0.005: the deviance at the node under consideration has to be at least 0.5% of the root-node deviance)

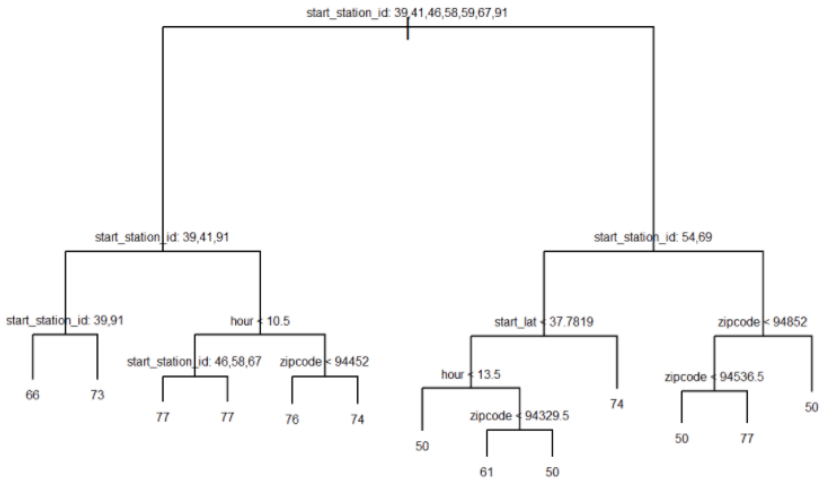


Figure15 : Complex Classification

אם כן, נראה כי על סמך מודלים של עצים פשוטים נתקשה להגיע לתוצאות טובות, לכן פנינו לאלגוריתם Random Forest, אשר מבוסס על עצי החלטה רבים, כאשר מדד ה-split נבחר כל פעם באופן רנדומלי. שיטה זו מצליחה לבנות לרוב מודל טוב יותר מעץ בודד, כיוון שהיא לומדת את סט האימון כל פעם מחדש על מספר רב של עצים ובכך מחזקת את הניבוי הסופי.
נתבונן במדדי הפיצול המשפיעים ביותר בעצים שהתקבלו-

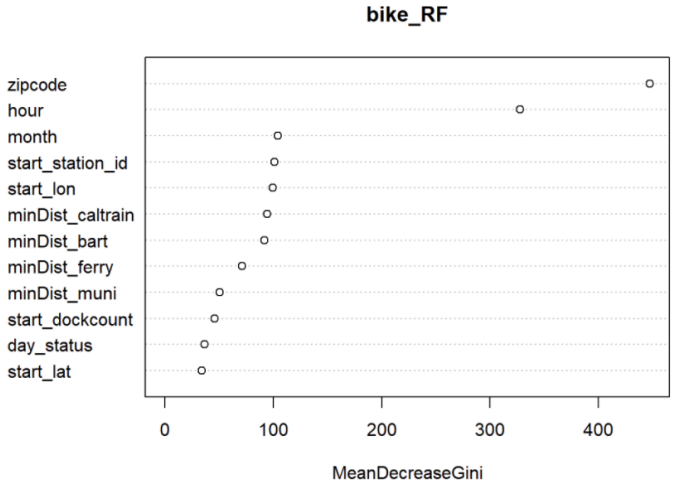


Figure16 : Variable Importance of Classification Random Forest

ניתן לראות כי בעוד שעבור עצים פשוטים מדדי הפיצול היחידים היו-
hour, zip code, start_station_id & start_lat, אלגוריתם ה-Random Forest עושה שימוש נרחב גם במדדים נוספים, המחזקים את הניבוי הסופי.
בפרק הבא נדון בהרחבה בתוצאות ובהבדלים בין המודלים.

Regression Approach

הצגנו לעיל את השיטה הקלאסית לגשת לבעיה הנתונה, אך ישנם שני חסרונות עיקריים בה-

1. אנו נדרשים לצמצם את ה-database שלנו למספר תחנות בודדות. בצמצום זה אנו מפסידים מידע רב, אשר היה יכול להוות תשתית רחבה לאימון.

2. החיזוי אינו מספיק טוב ומדויק. ישנם תחנות רבות שאינן מקבלות ניבוי כלל.

מעבר לזה, לבנות מודל פשוט של classification trees ועל-פיו לאמן את ה-data, זוהי משימה שכל מפתח המשתמש בכלים של ML מסוגל לבצע. אמנם בפרויקט זה רצינו ליצור חידוש ויתרון תחרותי, שאינו יכול לבוא לידי ביטוי באלגוריתמי classification בלבד. בנוסף, הפריסה הגיאוגרפית של סן פרנסיסקו קטנה מעט מערים גדולות כגון ניו-יורק, שנחאי ועוד, עבורן מספר תחנות האופניים יכול להגיע ל-100 ואף יותר. במצב זה, צמצום ה-data ל-10 תחנות בלבד (בשל מגבלות המודלים) יכול להפוך את כל החיזוי לחסר רלוונטיות.
אם כן, לאחר מחשבה רבה הצלחנו לגשת לבעיה מזווית שונה. אמנם תחנת הסיום מיוצגת על ידי ערך דיסקרטי ספציפי (ID), אך בנוסף מאפיין אותה מיקום גיאוגרפי בעולם. מיקום זה הינו בעל ערך רציף ולא דיסקרטי, לכן במעבר זה נוכל לחזות את הפרמטרים באמצעות עצי החלטה מסוג רגרסיה.
לשיטה זו מספר יתרונות בולטים-

1. זוהי רדוקציה מדהימה מ-37 classes ל-class בודד של מיקום. כלומר קודם תחנה יוצגה על ידי ערך אחד מתוך 37 ערכים ואילו כעת כל תחנה מיוצגת על ידי Latitude-Longitude בלבד (פרמטרים רציפים של מערכת צירים גיאוגרפית, דוגמת XY). רדוקציה זו מעניקה לנו יתרון משמעותי, שכן בכל עיר בעולם, ללא קשר למספר התחנות הקיימות בה, אנו מתכנסים לחיזוי של מיקום ולכן אין צורך לצמצם את ה-data שברשותנו.

2. בחיזוי של מיקום אנו יכולים למדוד את השגיאה מן המיקום האמיתי שלנו באמצעות מרחק (במטרים). כלומר, השגיאה מייצגת רדיוס חיפוש אמיתי במרחב, בו אנו יכולים להבטיח שהרוכב יסיים את נסיעתו. עובדה זו מעניקה למודל שלנו יתרון עצום, שכן קודם אם טעינו בהערכת התחנה ו"נפלנו" על תחנה אחרת, הביצועים

יורדים באופן מידי וכן אין לנו מידע רלוונטי על מיקום הסיום של הרוכב. כעת, יש לנו מידע ממשי על הימצאותו של הרוכב, בו אנו יכולים לעשות שימוש.

יחד עם זאת, ישנם מספר אתגרים בבואנו לממש מודל שכזה-

1. התחנות שלנו פרוסות במרחב של עשרות ק"מ בודדים. בקואורדינטות גיאוגרפיות מסוג Lat-Lon, ההבדל בין שני נ"צ הנמצאים במרחקים כאלו בלבד יכול להתבטא בספרה השנייה אחרי הנקודה ואפילו יותר. כלומר יש לנו קושי ממשי לבדל את המיקומים של התחנות. (מספיק שהמודל טעה בחיזוי בספרה כלשהי אחרי הנקודה ואנו עלולים להתרחק עשרות ק"מ מהיעד האמיתי).

2. אתגר משמעותי נוסף הינו שמיקום זהו משתנה דו ממדי, כלומר בכדי לחזות מיקום אנו נדרשים לחזות שני פרמטרים בו זמנית – Latitude ו-Longitude. בשיטות ML קלאסיות, החיזוי הוא תמיד בממד אחד, כלומר עבור כל תצפית עלינו לחזות משתנה אחד בלבד ואילו כעת סדרי הגודל משתנים ואנו צריכים להרחיב את הבעיה לדו-ממד.

בכדי לפתור שני אתגרים אלו ביצענו research רחב ולהלן המסקנות הסופיות שליוו אותנו בבניית המודל-

1. עבור קואורדינטות גיאוגרפיות ישנו ייצוג נוסף הנקרא UTM. ייצוג זה למעשה הינו ייצוג קרטזי של כדור הארץ (כאשר הוא מחולק לפלחים ו-zones) והפרמטרים המייצגים בו כל נ"צ בעולם הינם East-North (או לשם נוחות – X-Y). ייצוג זה בעל בידול ניכר יותר בין שני נ"צ אשר מרוחקים עשרות ק"מ בודדים, בנוסף כיוון שזהו ייצוג קרטזי, ניתן לחשב בקלות את המרחקים בין שתי נקודות שונות בעולם (מרחק אוקלידי).

2. עבור הרחבת הבעיה לדו-ממד, ישנם שתי גישות-
   a. הגישה הקלאסית הינה לבנות מודלים בגישת הפרד ומשול. כלומר - אנו נדרשים לחזות שני פרמטרים שונים לכן נאמן שני מודלים שונים – אחד ל-X ושני ל-Y, כך שנחשב את השגיאות בכל ציר בנפרד ובסוף נאחד אותם לכדי שגיאה כללית בדו-ממד.
   לדוגמא- בכל ציר נחשב את השגיאה בנפרד על-פי המדד הבא –

$$Err_x = \sqrt{\left(x_{prediction} - x_{real}\right)^2},\ Err_y = \sqrt{\left(y_{prediction} - y_{real}\right)^2}$$

   ולאחר מכן נקבל את השגיאה הכללית כך (הרחבה לדו ממד)-

$$Err = \sqrt{\left(x_{prediction} - x_{real}\right)^2 + \left(y_{prediction} - y_{real}\right)^2}$$

   b. מצאנו מספר מאמרים שמתעסקים בחיזוי של שני פרמטרים בו זמנית (ואף יותר) במודל אחד בלבד. מאמרים אלו אף גורסים ששיטה זו בעלת ביצועים טובים יותר מאשר חיזוי הפרמטרים בנפרד.
   ההרחבה הינה פשוטה להבנה – בעצי רגרסיה ערכו של העלה נקבע על-פי מיצוע של פרמטר החיזוי על פני כל התצפיות שהגיעו אליו, אמנם בדו ממד זה מתרחב לחישוב מרכז הכובד של שני פרמטרי החיזוי. בנוסף, שגיאת המודל מחושבת בדו ממד ע"י mean squared distance במקום חישוב מרחק וקטורי פשוט בממד אחד.
   להלן איור להמחשה -

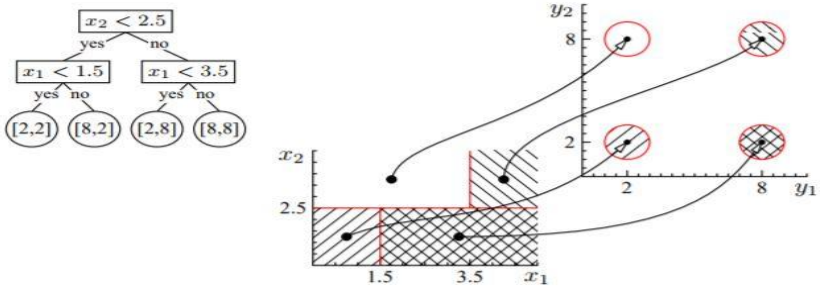
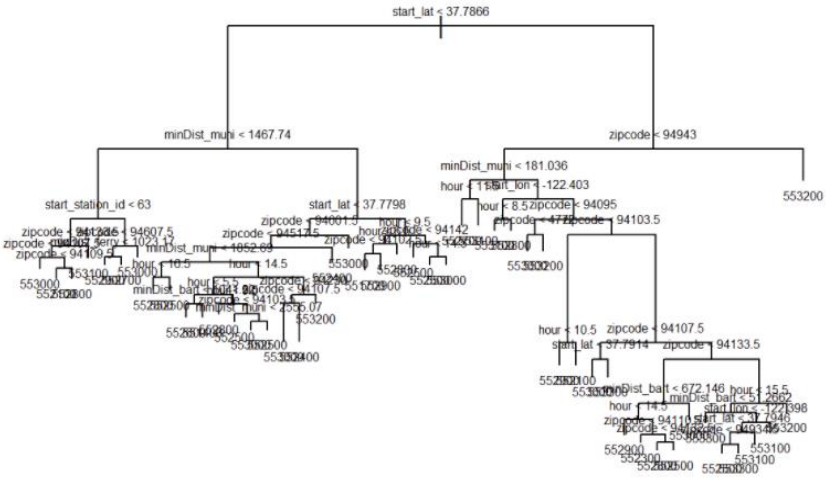


Figure17 : Extend of decision tree learning towards the case of multi-target prediction

לצורך המימוש של שיטה זו השתמשנו בספריה ייעודית ב-R לכך, הנקראת – MultivariateRandomForest

אם כן, באמצעות עצי הרגרסיה אימנו מספר מודלים.
סט המודלים הראשון היה בשיטת "הפרד ומשול", כאשר בנינו בנפרד מודל ל-X ומודל ל-Y, כאשר גם כאן בחנו מודלים של עצים פשוטים לצד מודלים מורכבים יותר דוגמת Random Forest.
נבחין למשל בעץ החלטה בעל חלוקה עדינה (שוב משחק עם mindev) לחיזוי X-
(שוב- נלקח מ-Rmarkdown, לינק מצורף בסוף הפרק)




```
## Regression tree:
## tree(formula = X_end ~ ., data = RT.trainX, mindev = 0.001)
## Variables actually used in tree construction:
## [1] "start_lat"        "minDist_muni"     "start_station_id"
## [4] "zipcode"          "minDist_ferry"    "hour"
## [7] "minDist_bart"     "start_lon"
```

Figure18 : Complex Regression Tree

שינוי משמעותי ראשון, הניכר בין עצים אלו לעצי ה-classification, הינו שימוש במדדי פיצול מגוונים ורבים, בניגוד לחד-גוניות שהייתה בגישת הקלסיפיקציה (ראינו קודם כי עץ קלסיפיקציה התבסס על 3 פרמטרים בלבד לחיזוי, גם כאשר החלוקה הוגדרה להיות עדינה יותר).
בנוסף, גם כאשר פנינו ל-Random Forest קיבלנו מודל המבוסס על מספר רב של מדדים אשר השפיעו-

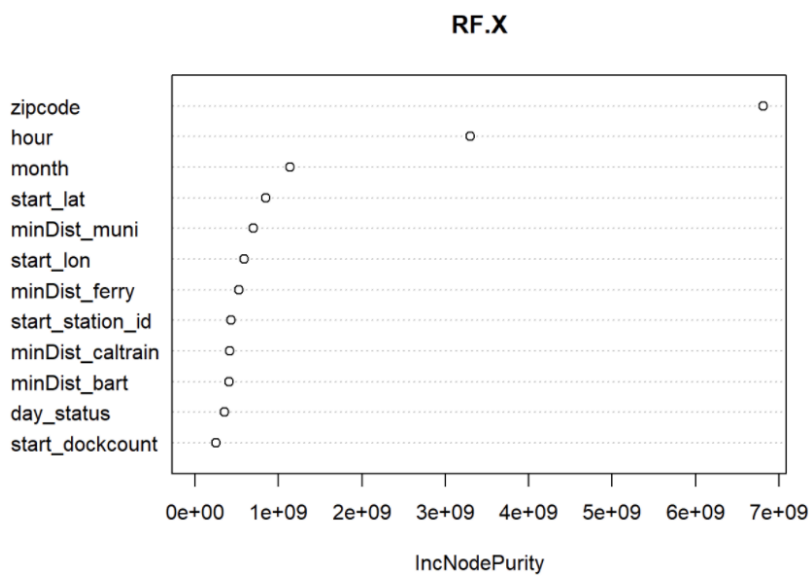
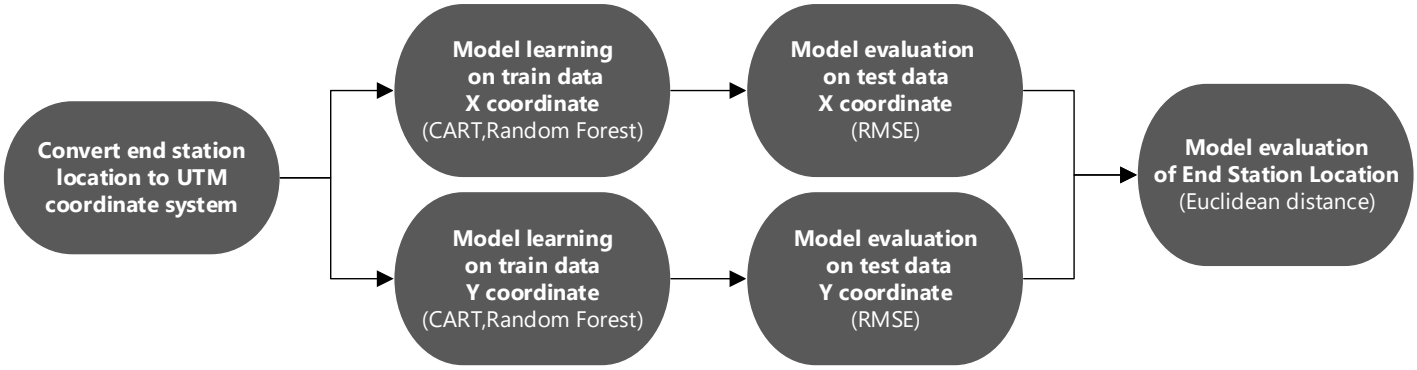


Figure19 : Variable Importance of Regression Random Forest

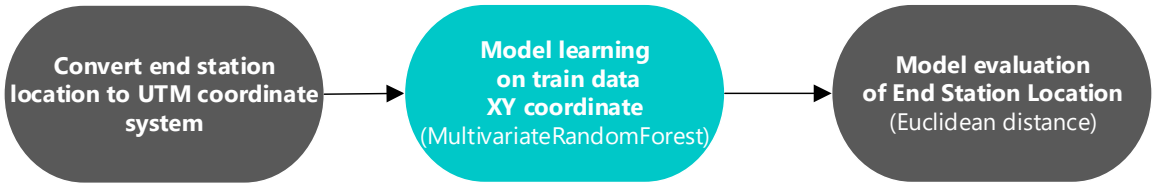
לסיום, נסכם פרק זה עם דיאגרמת בלוקים להמחשת השלבים השונים בפיתוח המודלים בשיטת עצי הרגרסיה-

עבור שיטת "הפרד ומשול"



: Flow chart for "Divide and Conquer" algorithm20 Figure

עבור שיטת Multi-Variate Target



: Flow chart for "Multivariate Target" algorithm21 Figure

Code

כל המודלים נבנו ב-R, להלן קישור ל-Rmarkdown הנמצא בתיקיית ה-Git של הפרויקט-

https://github.com/YardenIsraeli/Business-Analytics-Bikeshare-Project/blob/master/code/model_ML/treeBased_modeling1.Rmd

פרק 4 – Model Evaluation

Classification Approach

בחינת מדד להערכת המודל

להערכת הביצועים של מודלי ה-classification, אנו נשתמש במטריקות מתוך ה-confusion matrix עבור ריבוי של classes, כלומר זוהי אינה מטריצת confusion בינארית קלאסית אלא מטריצה ריבועית מסדר מספר ה-classes (במקרה שלנו 10x10, כפי שציינו בפרק הקודם צמצמנו את מספר התחנות).

כיוון שמעניין אותנו לחזות את תחנת הסיום וההתחייבות שלנו מול בתי העסק הינה על אחוז גבוה של דיוק (אחרת לא תהיה להם סיבה להשתמש בפלטפורמה שלנו לפרסום), כל החטאה בחיזוי מבחינתנו הינה פגיעה משמעותית בהתחייבות, שכן אין לנו מידע על קרבת התחנה המשוערכת לתחנת היעד האמיתית, ובכך במקרים אלו לרוב הרוכב יקבל פרסום שכלל לא מתאים לו.

אם כך, המטריקה המרכזית שתעניין אותנו הינה ה- True Positive Rate, או בשם המוכר יותר- Recall, שכן מדד זה למעשה מייצג את אחוז הפעמים שהאלגוריתם הצליח לחזות את תחנת היעד הנכונה, מתוך סך הפעמים שזו באמת הייתה תחנת היעד. יש לציין כי בעיה זו הינה multi-classes ולכן יש לחשב את המדד לכל תחנה בנפרד, כאשר התוצאה הסופית תהיה מיצוע על פני כל התחנות.

מדד זה מחושב באופן הבא – $Recall = \frac{TP}{TP + FN}$

כאשר עבור כל תחנת יעד X בנפרד-

TP (True Positive) מייצג את מספר הפעמים שחזינו את התחנה X (והיא באמת הייתה תחנת היעד הנכונה).
FN (False Negative) מייצג את מספר הפעמים שחזינו תחנה אחרת במקום.
כך שהסכום במכנה TP+FN מייצג את כלל הפעמים שתחנת היעד הייתה התחנה X ואילו המונה מייצג את כלל הפעמים שחזינו זאת נכון.

בחינת תוצאות וביצועי המודל

ראשית אימנו מודל של classification tree פשוט (figure 13) וקיבלנו את ה- prediction הבא-
(מופיע במטריצת ה-confusion כאשר העמודה הראשונה מייצגת את תחנות היעד האמתיות- Actual, ואילו השורה העליונה מייצגת את התחנות שהאלגוריתם ניבא- Prediction)

```
##     tree_test_results
##        47  50  61  66  68  73  74  76  77  90
##   47    0  78  24   0   0   0   5   0  71   0
##   50    0 687  43   0   0   0  96   0  47   0
##   61    0  69  64   0   0   0  40   0  68   0
##   66    0  83  72   0   0   0   0   0  28   0
##   68    0 102   9   0   0   0   5   0  60   0
##   73    0  12  60   0   0   0   7   0  13   0
##   74    0 433  37   0   0   0 108   0  77   0
##   76    0  87  20   0   0   0  19   0 129   0
##   77    0 405  39   0   0   0  11   0 156   0
##   90    0   5   3   0   0   0   1   0   3   0
```

Figure22 : confusion matrix for multi-classes Classification Tree

כפי שראינו בפרק הקודם, ישנם כ-6 תחנות יעד שהאלגוריתם כלל לא מנבא, כלומר מתוך 10 תחנות- תמיד רק 4 תחנות בלבד יהיו היעדים של הרוכבים – מה שכלל לא מייצג את המציאות. (קודם ראינו זאת על-ידי הניבוי בעלים של העץ, כעת אנו רואים זאת בעמודות של אפסים).

לכן בשלב זה כבר ניתן להעריך שביצועי האלגוריתם יהיו נמוכים מאוד ולא מדויקים, אך כפי שהגדרנו מעלה, המדד על-פיו נקבע זאת הינו ממד ה-Reacll (כאשר נצפה לאחוז דיוק נמוך).

עבור חישוב ה-Recall, ראשית עבור כל תחנה חישבנו את מדד ה-TP ומדד ה-FN וסיכמנו הכל בטבלה.

להלן התוצאות-

```
##       stations_id TP    FN    FP
##  [1,] "47"        "0"   "178" "0"
##  [2,] "50"        "687" "186" "1274"
##  [3,] "61"        "64"  "177" "307"
##  [4,] "66"        "0"   "183" "0"
##  [5,] "68"        "0"   "176" "0"
##  [6,] "73"        "0"   "92"  "0"
##  [7,] "74"        "108" "547" "184"
##  [8,] "76"        "0"   "255" "0"
##  [9,] "77"        "156" "455" "496"
## [10,] "90"        "0"   "12"  "0"
```

Figure23 : performance of simple classification tree

כפי שציינו, מדד ה-Recall הכללי יהיה מיצוע המדד על פני כל התחנות, ונקבל את התוצאה הבאה-

```
recall = mean(recall4each)
recall
```

```
## [1] 0.1472706
```

Figure24 : Recall for Simple Classification Tree

כלומר רק 14% מן הפעמים המודל מצליח לנבא את תחנת היעד האמיתית.

את אותו התהליך בחנו עבור מודל מורכב יותר של העץ Complex Classification Tree (figure 14) וקיבלנו את ההערכות הבאות-

```
##     ctree_test_results
##       47  50  61  66  68  73  74  76  77  90
##   47   0  73   5  13   0  11   8  21  47   0
##   50   0 613  25  19   0  24 110  26  56   0
##   61   0  35  34  28   0  36  42  31  35   0
##   66   0  71   9  70   0   2   5  15  11   0
##   68   0  93   9   2   0   7   6  13  46   0
##   73   0   6   6  14   0  46   7  10   3   0
##   74   0 362   4  34   0   3 144  18  90   0
##   76   0  71   8  10   0  10  28  72  56   0
##   77   0 269  10  29   0  10  16  41 236   0
##   90   0   5   0   3   0   0   2   0   2   0
```

```
##       stations_id TP    FN    FP
##  [1,] "47"        "0"   "178" "0"
##  [2,] "50"        "613" "260" "985"
##  [3,] "61"        "34"  "207" "76"
##  [4,] "66"        "70"  "113" "152"
##  [5,] "68"        "0"   "176" "0"
##  [6,] "73"        "46"  "46"  "103"
##  [7,] "74"        "144" "511" "224"
##  [8,] "76"        "72"  "183" "175"
##  [9,] "77"        "236" "375" "346"
## [10,] "90"        "0"   "12"  "0"
```

: performance of Complex Classification Tree25 Figure

גם כאן ישנם מספר תחנות שאינן מקבלות ניבוי כלל אך מספר זה נמוך מן המודל הקודם.

```
recall = mean(recall4each)
recall
```

```
## [1] 0.2614221
```

Figure26 : Recall for Complex Classification Tree

ואכן קיבלנו שיפור ל- 26% הצלחות בלבד.

לסיום בחנו את מודל ה-Random Forest, המבוסס על מספר גדול מאוד של עצים וכן משתמש במשתנים רבים עבור הפיצולים בעצים (figure 15), וקיבלנו את התוצאות הבאות-

```
##     RF_test_results
##       47  50  61  66  68  73  74  76  77  90
##   47  16  82   9  13   0   2   4  13  39   0
##   50   0 769  16  16   0   1  47   4  20   0
##   61   0  94  51  28   0   1  19  15  33   0
##   66   0  84   1  71   1   2   1   4  19   0
##   68   7 105   8   2   9   1   3  17  24   0
##   73   0  15   7  14   0  40   5   5   6   0
##   74   2 427   0  20   0   3 176   6  21   0
##   76   2  94  11  10   0   3  24  68  43   0
##   77   7 408  11  25   0   0  12  16 132   0
##   90   0   6   0   3   0   0   0   1   2   0
```

```
##       stations_id TP    FN    FP
##  [1,] "47"        "16"  "162" "18"
##  [2,] "50"        "769" "104" "1315'
##  [3,] "61"        "51"  "190" "63"
##  [4,] "66"        "71"  "112" "131"
##  [5,] "68"        "9"   "167" "1"
##  [6,] "73"        "40"  "52"  "13"
##  [7,] "74"        "176" "479" "115"
##  [8,] "76"        "68"  "187" "81"
##  [9,] "77"        "132" "479" "207"
## [10,] "90"        "0"   "12"  "0"
```

Figure27 : performance of Random Forest Classification model

ואכן כפי שציפינו, אלגוריתם זה הינו בעל ניבוי טוב יותר אמנם עדיין ישנה תחנת אחת שלא מקבלת ניבוי כלל. בנוסף נבחין במדד ה-FN, המייצג את מספר הפעמים שהייתה זו תחנת היעד אך הוא לא הצליח לחזות אותה, שהוא בעל ערכים גדולים, כלומר שוב לא נצפה לקבל ביצועים גבוהים. נראה זאת-

```
recall_rf = mean(recall4each_rf)
recall_rf
```

```
## [1] 0.2807682
```

Figure28 : Recall for Random Forest Classification model

ואכן כפי שציפינו הביצועים השתפרו ל- 28% הצלחות, אך עדיין הם לא טובים מספיק.

נסכם בטבלה את הערכת ביצועי מודלי ה-classification:

	Model Evaluation - Recall
Classification Tree	14%
Complex Classification Tree	26%
Random Forest	28%

במקרה הטוב ביותר יש לנו מודל שמצליח לנבא ב- 28% מן הפעמים את תחנת היעד האמיתית. תוצאה זו אינה מספקת כלל, שכן אף בית עסק לא יהיה מוכן לשלם על סיכוי נמוך שכזה.

לכן פנינו לשיטת הרגרסיה, כפי שראינו בפרק הקודם.
נדון כעת בתוצאות ובמודלים שקיבלנו אשר מנסים לנבא את מיקום היעד הגיאוגרפי של הרוכב הבא שישתמש בשיתוף.

Regression Approach

בחינת מדד להערכת המודל

שגיאה באלגוריתמי רגרסיה בין הערך המשוערך לערך האמיתי נמדדת על-פי RMSE (Root Mean Square Error), כאשר מרחק זה מתקבל ביחידות של המשתנה המשוערך. במקרה שלנו, החיזוי המתקבל הינו מיקום גיאוגרפי בעולם ולכן תוצאת המדד מתקבלת במטרים – כלומר מרחק פיסי בין המיקום המשוערך למיקום האמיתי בו יסיים הרוכב את נסיעתו.

ניזכר במדד ה- RMSE:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}RSS} = \sqrt{\frac{1}{n}\sum\left(y_{actual} - y_{estimated}\right)^2}$$

כלומר מדד זה מורכב ממד ה- RSS (Residual Sum of Squares), כאשר בעצי רגרסיה בפרט זהו המדד עליו מבצעים אופטימיזציה לקביעת החלוקה הסופית. לכן טבעי להשתמש בו גם לשערוך שגיאת המודל.

בפרק הקודם ראינו שתי גישות לבניית המודל – ראשונה בשיטת "הפרד ומשול", כאשר לכל קואורדינטה מאמנים מודל בנפרד ולבסוף מחשבים את השגיאה הכללית, וגישה שנייה לפיה ניתן לשערך במודל אחד בו זמנית את שני הקואורדינטות, כך שמבצעים הרחבה לדו-ממד.

- על-פי הגישה הראשונה, בחינת ביצועי המודל תהיה על פי החישוב הבא-

$$RMSE_x = \sqrt{\frac{1}{n}\sum\left(x_{actual} - x_{estimated}\right)^2}$$

$$RMSE_y = \sqrt{\frac{1}{n}\sum\left(y_{actual} - y_{estimated}\right)^2}$$

$$RMSE_{xy} = \sqrt{\frac{1}{n}\sum\left(\sqrt{\left(x_{actual} - x_{estimated}\right)^2 + \left(y_{actual} - y_{estimated}\right)^2}\right)}$$

כלומר לכל קואורדינטה נחשב בנפרד את שגיאת המודל, ואילו השגיאה הכללית מן הנ"צ האמיתי לנ"צ המשוערך תחושב על-פי הרחבת מטריקת המרחק לדו-ממד.

- בגישה השנייה מתקבל מודל יחיד, כך שמדד הביצועים מחושב גם כן על-פי מטריקת המרחק בדו-ממד, ללא צורך בחישוב שגיאת כל קואורדינטה בנפרד, כלומר –

$$RMSE_{Multivariate} = \sqrt{\frac{1}{n}\sum\left(\sqrt{\left(x_{actual} - x_{estimated}\right)^2 + \left(y_{actual} - y_{estimated}\right)^2}\right)}$$

בחינת תוצאות וביצועי המודל

ראשית אימנו מודל מורכב של regression tree (figure 18) בשיטת "הפרד ומשול" וקיבלנו את ה-prediction הבא- (וקטור שורה עליו ביצענו את פקודת ה-summary)

```
prediction.RTxcomplex= predict(RT.Xcomplex,RT.test)
summary(prediction.RTxcomplex)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  551369  552830  553013  552934  553050  553300
```

Figure29 : X-prediction of Complex Regression Tree model

ניתן לראות כי ערכים אלו אכן מייצגים קואורדינטת X במערכת צירי UTM בסביבת סן-פרנסיסקו. אם כן נבדוק כעת כמה הם קרובים לערכי ה-X האמתיים.

בשביל חישוב זה נשתמש במדד ה-RMSE, כפי שהגדרנו מעלה, ונקבל-

```
predictionErr.RTxcomplex = test$X_end - prediction.RTxcomplex
RSS = sum(predictionErr.RTxcomplex ^2)
MSE <- RSS/length(predictionErr.RTxcomplex)
RMSE.RTxcomplex <- sqrt(MSE)
RMSE.RTxcomplex
```

```
## [1] 572.3681
```

Figure30 : X-evaluation of Complex Regression Tree model

כלומר בציר X קיבלנו שגיאה של כ- 572 [m] מן המיקום האמיתי.

את אותו התהליך בדיוק ביצענו עבור ציר Y (תיעוד מלא ב- Rmarkdown) וקיבלנו-

```
prediction.RTycomplex = predict(RT.Ycomplex,RT.test)
predictionErr.RTycomplex = test$Y_end - prediction.RTycomplex
RSS = sum(predictionErr.RTycomplex ^2)
MSE <- RSS/length(predictionErr.RTycomplex)
RMSE.RTycomplex <- sqrt(MSE)
RMSE.RTycomplex
```

```
## [1] 806.7514
```

Figure31 : Y-evaluation of Complex Regression Tree model

כלומר בציר זה התקבלה שגיאה גדולה יותר, של כ- 806[m] מן ציר ה-Y של המיקום האמיתי.
כעת נחשב את השגיאה הכוללת-

```
dist_2D = sqrt((predictionErr.RTxcomplex)^2 +(predictionErr.RTycomplex)^2)
RSS = sum(dist_2D^2)
MSE <- RSS/length(dist_2D)
RMSE.RTxy <- sqrt(MSE)
RMSE.RTxy
```

```
## [1] 989.1679
```

Figure32 : Model evaluation of Complex Regression Tree

לסיכום, עבור מודל של עץ רגרסיה מורכב (בעל חלוקה עדינה יותר) קיבלנו כי השגיאה בין מיקום היעד המשוערך למיקום האמיתי הינה כ-989[m]. כלומר אנו יכולים להבטיח אזור חיפוש ברדיוס של 989[m] בו הרוכב יסיים את נסיעתו.
בשלב הבא אימנו מודל מסוג Random Forest, בשיטת "הפרד ומשול" גם כן (figure 17).
שוב קיבלנו וקטור prediction עבור כל ציר בנפרד וחישבנו את השגיאה.
עבור ציר ה- X קיבלנו-

```
prediction.RFx <- predict(RF.X,RT.test)
summary(prediction.RFx)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  551442  552799  552962  552943  553109  553664
```

Figure33 : X-prediction of Regression Random Forest model

ומדד ה- RMSE שהתקבל היה-

```
RMSE.RFx
```

```
## [1] 505.692
```

Figure34 : X-evaluation of Regression Random Forest model

כלומר כעת קיבלנו שגיאה נמוכה יותר בציר ה-X של כ-505[m].
ועבור ציר ה-Y קיבלנו את השגיאה הבאה-

```
RMSE.RFy
```

```
## [1] 713.5967
```

Figure35 : X-evaluation of Regression Random Forest model

שוב, שגיאה נמוכה יותר מן המודל הקודם.
אם כן, נוכל לחשב את השגיאה הכוללת ונקבל-

```
dist_2D = sqrt((predictionErr.RFx)^2 +(predictionErr.RFy)^2)
RSS = sum(dist_2D^2)
MSE <- RSS/length(dist_2D)
RMSE.RFxy <- sqrt(MSE)
RMSE.RFxy
```

```
## [1] 874.6111
```

Figure36 : Model evaluation of Regression Random Forest

לסיכום, עבור מודל בשיטת ה- Random Forest קיבלנו ביצועים טובים יותר, עם מרחק של 874[m] מן הנ"צ האמיתי ושיפור של 11% מן המודל הקודם.

כעת עבור בחינת הגישה השנייה- Multivariate prediction, השתמשנו בספרייה ייעודית ב-R אשר מרחיבה את החישוב לדו-ממד. כיוון שההרחבה משנה את סדרי הגודל והסיבוכיות של המודל, אנו נדרשים לזיכרון של מאות GB ולימים ספורים בכדי להצליח לחשב אותו. אך בכל זאת היינו רוצים לבדוק האם שיטה זו בעלת ביצועים טובים יותר, כפי שהמאמרים גורסים. לשם כך צמצמנו את ה-database שלנו בצורה משמעותית- דגמנו sample של 1000 תצפיות בלבד מתוך סט האימון ו-250 תצפיות מתוך סט הבוחן.
נראה זאת-

```
set.seed(7)
inds_train = sample(1:nrow(train),1000)
inds_test = sample(1:nrow(test),250)
train_inputs_mat = as.matrix(train[inds_train,-c(1,5,15,16,17,18)])
train_outputs_mat = as.matrix(train[inds_train,c(17,18)])
test_inputs_mat = as.matrix(test[inds_test,-c(1,5,15,16,17,18)])
```

Figure37 : sample of our database for Multivariate prediction

ולמעשה יצרנו database חדש, רק לשם השוואה בין מודל זה למודל בשיטת "הפרד ומשול".
כעת זמני הריצה הינם סבירים ואנו מקבלים את ה-prediction הבא-

```
#install.packages("MultivariateRandomForest")
library(MultivariateRandomForest)
mvrPredict = build_forest_predict(train_inputs_mat,train_outputs_mat,1,12,5,test_inputs_mat)
str(as.data.frame(mvrPredict))
```

```
## 'data.frame':    250 obs. of  2 variables:
##  $ V1: num  552961 553690 553255 551307 553387 ...
##  $ V2: num  4183494 4181639 4181203 4181191 4182952 ...
```

Figure38 : multivariate prediction

ואכן בשונה מקודם, אנו מקבלים וקטור ניבוי לשתי הקואורדינטות יחד.

נבחן את שגיאת ה- RMSE של מודל זה-

```
RMSE.MVT
```

```
## [1] 785.9917
```

Figure39 : model evaluation of Multivariate Tree

וקיבלנו תוצאה יפה של שערוך מיקום היעד כ- 786[m] מן המיקום האמיתי.
לצורך השוואה מדויקת, נרצה לבחון את מדד ה-RMSE של מודל מסוג עץ בשיטת "הפרד ומשול" על אותו sample של database בדיוק, ולראות בכמה הביצועים באמצעות Multivariate Tree משתפרים-

```
RT.X1000 = tree(X_end ~ . , data = RT.trainX[inds_train,])
prediction.RTx1000 = predict(RT.X1000,RT.test[inds_test,])
predictionErr.RTx1000 = test$X_end[inds_test] - prediction.RTx1000

RT.Y1000 = tree(Y_end ~ . , data = RT.trainY[inds_train,])
prediction.RTy1000 = predict(RT.Y1000,RT.test[inds_test,])
predictionErr.RTy1000 = test$Y_end[inds_test] - prediction.RTy1000

predictionErr.RTx1000 = na.omit(predictionErr.RTx1000)
predictionErr.RTy1000 = na.omit(predictionErr.RTy1000)
dist_2D = sqrt((predictionErr.RTx1000)^2 +(predictionErr.RTy1000)^2)
RSS = sum(dist_2D^2)
MSE <- RSS/length(dist_2D)
RMSE.RTxy1000 <- sqrt(MSE)
RMSE.RTxy1000
```

```
## [1] 1130.524
```

Figure40 : model evaluation of Regression Tree only on specific sample of data

וקיבלנו בשיטה זו כי המיקום המשוערך הינו במרחק של 1130[m] מן המיקום האמיתי, כלומר ישנו יתרון גדול לשימוש בשיטה השנייה.

לסיום, נסכם את כל התוצאות עבור מודלי הרגרסיה בטבלה-

	Model Evaluation - RMSE
Complex Regression Tree	989 [m]
Regression Random Forest	874 [m]

	Model Evaluation - RMSE
Multivariate Tree	786 [m]
Divide & Conquer Tree	1130 [m]

<u>סיכום ביצועים והצעות לשיפור</u>

אם כן, ראינו כי מודלי ה-classification אינם מצליחים לנבא את תחנת היעד באחוזים גבוהים (ואף מאוד נמוכים), לכן פנינו לגישה שונה, של ניבוי מיקום היעד הגיאוגרפי בעולם. בדרך זו עברנו לבעיית regression והיתרון הבולט בכך הוא שבאמצעות שגיאת המודל (במטרים) אנו יכולים להתחייב על אזור חיפוש ברדיוס השגיאה, בו הרוכב יסיים את נסיעתו.
אנו יכולים להבחין כי בשיטת "הפרד ומשול" מודל ה-Random Forest מוצלח יותר ומשפר את הביצועים ב- 11% ממודלים של עצים בודדים. בנוסף בשיטת ה-Multivariate אנו מקבלים שיפור של כמעט 30% לעומת עצים בודדים.
לכן אנו מאמינים כי לאחר שנצליח להתגבר על אתגרים חישוביים (זיכרון, שימוש במעבד דוגמת GPU) ונוכל לאמן בשיטת ה-Multivariate מודל מסוג Random Forest נצליח לשפר את הביצועים בצורה ניכרת ובכך להקטין את אזור החיפוש בו אנו מתחייבים שהרוכב יסיים את נסיעתו.

פרק סיום

כפי שראינו בפרק הראשון, מטרתנו הינה לחזות את תחנת הסיום של הרוכב הבא שישתמש בשיתוף. בפרק השני הצגנו את המידע עליו התבססנו, כאשר ניתן להבחין שעבור כל נסיעה (שורה בדאטה) יש שדה של תחנת הסיום, המאופיין ב-ID הספציפי של התחנה.
אם כן, הגישה הקלאסית לפתור בעיה זו הינה שימוש בכלים של Supervised learning לצורך פתרון בעיית קלסיפיקציה פשוטה, עם 37 classes – כמספר התחנות, כאשר פרמטר החיזוי הרצוי יהיה ה-ID של התחנה.
בנוסף, כפי שציינו בפרקים הקודמים, ראינו כי מיקום התחנות אינו אקראי, וכן ישנם מסלולים פופולאריים יותר ופחות, כך שניתן להבחין בדאטה שלנו בקשרים וחזרות בין הנסיעות השונות, המאפשרים לאלגוריתמי ML להתכנס.

יחד עם זאת, לאור תוצאות לא מספקות, נאלצנו לחשוב "מחוץ לקופסא", על דרך נוספת לחיזוי תחנת הסיום.
לאחר מחקר רחב שערכנו, הצלחנו להפוך את בעיית הקלסיפיקציה לבעיית רגרסיה על-ידי ייצוג התחנה כמיקום גיאוגרפי בעולם. בייצוג זה למעשה קיבלנו רדוקציה מדהימה, שכן כעת אין לנו ריבוי של classes. עובדה זו מעניקה לנו יתרון משמעותי, שכן עתה בכל עיר בעולם, ללא קשר למספר התחנות הקיימות בה, אנו מתכנסים לחיזוי של מיקום ולכן אין צורך לצמצם את ה-data שברשותנו.
בנוסף, בשיטה זו במקום להתחייב על אחוזי דיוק אנו מתחייבים על אזור חיפוש ברדיוס שגיאת המודל (במטרים), אשר נותן משמעות פיסית למיקומו של הרוכב בעולם.

התוצאות אליהן הגענו מעודדות מחד, ניתן לנבא 2 משתנים היוצרים לנו נקודת סיום משוערכת. אך מאידך אנו בטוחים שניתן לשפר את התוצאות בכמה מונים, וזאת כפי שהצגנו בסיום פרק 4.

הצעות לשלבים הבאים -

1. הגדלה משמעותית של כוח המחשוב וזאת בכדי לעבד כמות גדולה יותר של מידע לכדי ניבוי של שני משתנים בו זמנית.
2. התרחבות לניבוי בערים נוספות, גדולות יותר.
3. הוספת שכבות מידע נוספות המתארות את המשתמש עצמו וזאת בכדי לשפר עוד יותר את ביצועי המודל.

ככל שהאלגוריתם שלנו ישתפר כך גם ישתפר המוצר. כשישתפר המוצר נרחיב את מספר הערים בהם אנחנו פועלים, נגדיל את המידע שאנחנו אוספים, נמשיך לשפר את המודל וחוזר חלילה.

בהצלחה ☺

Appendix

Reference list

- All relevant work, including- code, datasets, algorithms & models, articles, statistics & previous presentations can be found at the following link- https://github.com/YardenIsraeli/Business-Analytics-Bikeshare-Project

- Our multivariate-target prediction is based on our research and can be found at the following link- https://github.com/YardenIsraeli/Business-Analytics-Bikeshare-Project/tree/master/presentationMODEL/research

- Codebook link- https://github.com/YardenIsraeli/Business-Analytics-Bikeshare-Project/blob/master/Codebook.pdf

Codebook

Data Overview

Credentials

This data set can be downloaded from BigQuery – Google's data warehouse.

- The direct link to exploring this data in the BigQuery console is: https://bigquery.cloud.google.com/table/bigquery-public-data:san_francisco.bikeshare_trips

Business goal

This data was collected to answer the question:

How can we predict the target destination/station of the next rider?

Data description

This data set is a data frame of 11 variables over 983,648 rows.
Each row represent a bike rental from Bay Area Bike-Share Company in California.

We filtered this data set to the city of San Francisco only, where most of the rentals occurred, and got updated data frame of **891,200** rows.

There are no missing values in the data.

We combined this data with complimentary tables as shown below.

Variables description

San Francisco Bay Area bike-sharing trips

Variable Name	Type	Description
trip_id	Integer	Numeric ID of bike trip
duration_sec	Integer	Time of trip in seconds
start_date	Character	Start date of trip with date and time, in PST
start_station_name	Character	Station name of start station
start_station_id	Integer	Numeric reference for start station
end_date	Character	End date of trip with date and time, in PST
end_station_name	Character	Station name for end station
end_station_id	Integer	Numeric reference for end station
bike_number	Integer	ID of bike used
zip_code	Integer	Home zip code of subscriber (customers can choose to manually enter zip at kiosk however data is unreliable)
subscriber_type	Integer	Subscriber = annual or 30-day member; Customer = 24-hour or 3-day member

Complimentary Data Overview

Links to complimentary data:

- San Francisco Bay Area bike's stations
  https://github.com/YardenIsraeli/Business-Analytics-Bikeshare-Project/blob/master/dataSets/sanFrancisco/sanFran_stations.csv

- San Francisco Bart (Bay Area Rapid Transit) stations
  https://github.com/YardenIsraeli/Business-Analytics-Bikeshare-Project/blob/master/dataSets/raw_data/Stops%20bart.csv

- San Francisco Golden Gate Transit stations
  https://github.com/YardenIsraeli/Business-Analytics-Bikeshare-Project/blob/master/dataSets/raw_data/Stops%20goldengatetransit.csv

- San Francisco Muni (Municipal Railway) stations
  https://github.com/YardenIsraeli/Business-Analytics-Bikeshare-Project/blob/master/dataSets/raw_data/Stops%20muni.csv

- Bike-share stations Availability
  https://bigquery.cloud.google.com/table/bigquery-public-data:san_francisco.bikeshare_status

Variables description

San Francisco Bay Area bike's stations

Variable Name	Type	Description
station_id	Integer	Station ID number
Name	Character	Name of station
latitude	Float	Latitude of station
longitude	Float	Longitude of station
Dockcount	Integer	Number of total docks at station
landmark	Character	City (San Francisco)
Installation_data	Character	Original date that station was installed

San Francisco Bart (Bay Area Rapid Transit) stations

Variable Name	Type	Description
stop_id	Character	Station ID
stop_name	Character	Station name
stop_lat	Numeric	Latitude of stop station
stop_lon	Numeric	Longitude of stop station

San Francisco Golden Gate Transit stations

Variable Name	Type	Description
stop_id	Character	Station ID
stop_name	Character	Station name
stop_lat	Numeric	Latitude of stop station
stop_lon	Numeric	Longitude of stop station

San Francisco Muni (Municipal Railway) stations

Variable Name	Type	Description
stop_id	Character	Station ID
stop_name	Character	Station name
stop_lat	Numeric	Latitude of stop station
stop_lon	Numeric	Longitude of stop station

Bike-share stations Availability

Variable Name	Type	Description
station_id	Integer	Station ID number
bikes_available	Integer	Number of available bikes

docks_available	Integer	Number of available docks
time	Character	Date and time, PST