פרויקט – בינה מלאכותית

236502

הפקולטה למדעי המחשב, טכניון

בנושא : חיזוי הצמתים המועדים לתאונות בסבירות גבוהה על מנת לבצע פריסת ניידות ארצית

מגישי הפרויקט :

מירית אלוש 308439306

כפיר שלו 307960146

סתיו רזניק 311448559

מנחה :

יניב נמקובסקי

הקדמה

בכל שנה תאונות דרכים בישראל הורגות 300-350 בני אדם, פוצעות כ-140 אלף בני אדם וגורמות למשק נזקים של כ-22 מיליארד ש"ח [ויקיפדיה].

אכיפת תנועה יעילה נחוצה על מנת להפחית נהיגה מסוכנת ובכך מפחיתה את תאונות הדרכים וחומרתן.

המטרה המרכזית בפרויקט היא לצמצם את מספר תאונות הדרכים במידה המרבית בעזרת תחזית לתאונות הדרכים העתידיות לקרות, ובהסתמך על מידע זה, מיקום ניידות משטרה בהתאם לאילוציה. בפרויקט הסתמכנו על תאונות הדרכים בישראל שקרו בעשר השנים האחרונות בשילוב עם מודלים הסתברותיים ואלגוריתמי בינה מלאכותית. כלומר נרצה להציע סיפוק לבעיית המשאבים, המקבלת כקלט את מספר הניידות הזמינות ונתוני מזג האוויר של אותו יום ומחזירה ניבוי של המקומות בהם נציב את הניידות. נשאף להציב את הניידות במקומות בהם ההסתברות לתאונות הוא גבוה.

המידע שאספנו הינו עבור צמתים בישראל, כך שהוא מתבסס על נתונים קבועים ונתונים משתנים לאורך עשר השנים האחרונות. במהלך איסוף המידע הבחנו כי קבוצת התכונות אינה מספיקה להכללה מספקת ולכן החלטנו בנוסף לחקור תהליך של יצירת תכונות חדשות הנקרא : Feature Generation .

ברוב המקרים לוקחים את התכונות הקיימות ומהן יוצרים תכונות חדשות. ניתן גם להשתמש בתכונות על מנת לקבל תכונות חדשות ממקורות חיצוניים. במהלך הפרויקט ביצענו מספר ניסויים ובחנו את האפשרויות השונות עבור יצירת תכונות חדשות כדי להגדיל את מאגר התכונות שלנו לכל צומת. בנוסף, בחנו את הקשרים הקיימים בין התכונות על מנת לנצל את המירב, כלומר בדיקת תלויות בין תכונות.

במסגרת הפרויקט השתמשנו בשיטות שהכרנו מקורס מבוא לבינה מלאכותית ובשיטות חדשות שלמדנו תוך כדי הפרויקט ובחנו את ההשפעה שלהן על תוצאות המסווגים שלנו – כפי שנפרט בהמשך.

מאגרי המידע בהם השתמשנו:

- http://www.ims.gov.il – השירות המטאורולוגי(עבור מאגרי מזג האוויר)
- https://old.cbs.gov.il/reader/puf/catalog_puf_main.html?id_topic=20#20 – הלמס(עבור מאגרי התאונות ונפח התנועה)
- Open street map
- משטרת ישראל
- https://www.moag.gov.il – משרד החקלאות ופיתוח הכפר(עבור מאגר הכלבים)

רקע מהקורס מבוא לבינה מלאכותית

המטרה הבסיסית של למידה היא היכולת להכליל מתוך ניסיון. במילים אחרות, היכולת לבצע חיזוי באופן מדויק ככל האפשר על מידע שעדיין לא נצפה, על בסיס צבירת ניסיון ממידע קיים.

במסגרת הפרויקט אנו השתמשנו בלמידה לניבוי הצמתים בהם ההסתברות לתאונות בחומרה מסוימת גדולה בהתאם לתנאי מזג האוויר באותו יום.

קבוצת האובייקטים שלנו היא קבוצת הצמתים בישראל. המושג מוגדר להיות הצמתים בהם ההסתברות לתאונה באותו היום גבוהה כאשר אלו מותאמים לאילוצי מספר הניידות של המשטרה לאותו היום. קבוצת האובייקטים מכילה מידע על הצמתים בהתאם לכל יום בעשר השנים האחרונות. קבוצת הדוגמאות הזו נקראת גם קבוצת האימון שעבורה נגדיר סיווג מתאים, על זה כמובן נפרט בהמשך הדו"ח. מטרתנו היא בעזרת הסיווג של הדוגמאות הנתונות, שנוכל לחזות את הסיווג עבור מושג המטרה, כלומר עבור אובייקט עבורו עוד לא קיים הסיווג.

<u>ביצענו את תהליך ההכנה לקראת הלמידה הכולל בתוכו:</u>

- הגדרת מרחב האובייקטים- הצמתים.
- הגדרת המושג אותו אנו רוצים ללמוד- צמתים בהם ההסתברות לתאונה היא גבוהה.
- הגדרת צומת מסוכן –

הגדרה זו מורכבת ומושפעת ממספר התאונות וחומרתן. בהמשך נרחיב על הגדרת צומת מסוכן וקביעת הסיווג עבורו.
לאחר תהליך של הבנת אופן פעולה של מסווגים והידברות עם משטרת ישראל החלטנו לאפשר למשתמש לבחור את הגדרת הצומת כמסוכן , כאשר מוגדר לכל צומת: 0 – לא הייתה תאונה , 1 – תאונה בחומרה קלה , 2 – תאונה בחומרה בינונית , 3 – תאונה בחומרה קשה. קיימת אפשרות למשתמש להגדיר מהי הדרך בה הוא מעוניין לקבל את תוצאת המסווג :

1. מספר תאונות דרכים כולל בצומת
2. מספר תאונות דרכים כולל בצומת בחומרה קשה
3. מספר תאונות דרכים כולל בצומת בחומרה בינונית
4. מספר תאונות דרכים כולל בצומת בחומרה קלה
5. סכום החומרות של כלל התאונות בצומת

- השגת קבוצת אובייקטים שישמשו כדוגמאות – הצמתים.
- השגת תיוגים לקבוצת אובייקטים זו – שקלול של מספר התאונות בהתאם לחומרתם/מספר התאונות.
- הגדרת אוסף פונקציות שישמשו כתכונות- תהליך איסוף הנתונים והצלבתם.
- חישוב סטטיסטיקות גולמיות על הנתונים בכדי לקבל מושג כללי על המאגר שברשותנו.

הרעיון

החזון שלנו עבור הפרויקט הוא היעילות שלו בשימוש היומי של המשטרה. כך שכל בוקר, תוכל לקבל דו"ח מפורט בהתאם למשאבי הניידות שלה באותו היום עם הצמתים הכי מסוכנים לאותו היום. לאחר התייעצות עם גורמים במשטרה, בחרנו לתת להם את היכולת להחליט ע"פ מה הם רוצים להגדיר צומת כמסוכן- האם ע"פ סכום חומרות , מספר תאונות כולל או מספר תאונות קשות, בינוניות או קלות בנפרד. כך בעצם אנחנו מאפשרים להם התאמה יותר טובה לאילוציהם ויכולת אכיפה קלה יותר. למשל, במידה ויש מעט מאוד משאבים לאותו היום, והם ירצו למקסם את היכולת לנבא נפגעים בתאונות אלה, הם יכולים לבחור לסווג צומת כמסוכן ע"פ מספר תאונות קשות בלבד. כך אנחנו מרגישים שמטרתו הגדולה של הפרויקט, לנסות להציל חיי אדם, יכול לבוא בצורה הטובה ביותר לידי ביטוי.

ראשית, התחלנו באיסוף מידע באופן עצמאי על צמתים רבים בישראל ולאחר מכן ביצענו ניסוי של יצירת תכונות חדשות באמצעות התכונות הקיימות על מנת להעשיר את מרחב התכונות לכל אובייקט.

איסוף המידע דרש עבודה עם מקורות מידע רבים כפי שצוין לעיל והתאמתם אל טבלת הדאטה שאיתה עבדנו. ראשית, יצרנו טבלה בסיסית של צמתים מרכזיים ברחבי הארץ במרחק מינימלי של 50 מטר אחד מהשני – פירוט בהמשך. לאחר מכן בחרנו רדיוס קבוע של 10 ק"מ, כך שעבור כל צומת השתמשנו בטווח הזה על מנת לאסוף ולהצליב את המידע סביב אותו הצומת. עם רדיוס זה עבדנו לאורך כל הפרויקט.

בשלב זה, עבור כל צומת מרכזי בטבלה הוספנו את התאריכים בשנים 2017-2008 על מנת שנוכל להצליב מידע על תאונות בתאריכים מוגדרים. כעת, לאחר שהכנו את הבסיס של הדאטה, עברנו על כל מקורות המידע והוספנו עבור כל צומת ועבור כל תאריך את התכונות:

מספר פאבים – (openStreetMap) לכל צומת סוכם את מספר הפאבים ברדיוס 10 ק"מ.
ההיפותזה שלנו היא שככל שברדיוס של הצומת יש יותר פאבים מספר התאונות בצומת זה תהיה גבוה יותר.
מספר מוסדות חינוך – (הלמס) לכל צומת סוכם את מספר מוסדות החינוך ברדיוס 10 ק"מ.
ההיפותזה שלנו היא שככל שברדיוס של הצומת יש יותר מוסדות חינוך , האזור מועד יותר לתאונות דרכים עם הולכי רגל ולכן החלטנו להוסיף את התכונה.
מספר תחנות דלק - (openStreetMap) לכל צומת סוכם את מספר תחנות הדלק ברדיוס 10 ק"מ.
ההיפותזה שלנו היא שבקרבת תחנות דלק יכול להיווצר מצב של האטות פתאומיות לכיוון הכניסה לתחנה וכן יציאות פתאומיות ולכן הכנסנו את התכונה.
מספר אזורי תעשייה - (openStreetMap) לכל צומת סוכם את מספר אזורי התעשיה ברדיוס 10 ק"מ **.**
ההיפותזה שלנו היא שבאזורי תעשייה יש פחות תשומת לב לחוקי התנועה וכן יש תנועה של רכבים כבדים ובכלל תנועה רבה ולכן צפי למספר תאונות רב יותר.
מספר מוזיאונים - (openStreetMap) לכל צומת סוכם את מספר המוזיאונים ברדיוס 10 ק"מ**.**

ההיפותזה שלנו, לתכונה זו לא היה לנו שום השערה והוספנו אותה כדי לבחון האם היא משפיעה.
מספר תחנות משטרה - (openStreetMap) לכל צומת סוכם את מספר תחנות המשטרה ברדיוס 10 ק"מ **.**
ההיפותזה שלנו היא שבקרבת תחנות משטרה אנשים נוהגים להירתע ולנהוג באופן זהיר יותר ולכן נצפה שמספר התאונות בקרבת תחנות משטרה יהיו מועט יותר.
מספר מסעדות -(openStreetMap) לכל צומת סוכם את מספר המסעדות ברדיוס 10 ק"מ**.**
עבור תכונה זו אין לנו היפותזה מקדימה.
מספר מרכזי קניות -(openStreetMap) לכל צומת סוכם את מספר מרכזי הקניות ברדיוס 10 ק"מ**.**
כנ"ל עבור תכונה זו אין לנו היפותזה מקדימה.
מספר מצלמות מהירות (הלמס) לכל צומת סוכם את מספר מצלמות המהירות ברדיוס 10 ק"מ**.**
ההיפותזה שלנו היא שככל שיש יותר מצלמות מהירות בסביבה כך הסבירות לתאונה תפחת בשל ההרתעה.
מספר תחנות רכבת - (openStreetMap) לכל צומת סוכם את מספר תחנות הרכבת ברדיוס 10 ק"מ**.**
עבור תכונה זו אין לנו היפותזה מקדימה.
מספר כלבים – (משרד החקלאות ופיתוח הכפר) לכל צומת סוכם את מספר הכלבים ברדיוס 10 ק"מ .
היפותזה עבור תכונה זו היא ככל שיש יותר כלבים באזור כך יש יותר הסחות דעת לנהג הממוצע ולכן יש השפעה על התאונות.
חניונים - (openStreetMap) לכל צומת סוכם את מספר החניונים ברדיוס 10 ק"מ.
ההיפותזה עבור תכונה זו היא שבאזור של חניונים יש תמיד אי סדר ביציאה ובכניסה אליו ולכן משפיעה על מצב הרוח של הנהגים הנכנסים ויוצאים ממנה ובשל כך על נסיעה הצמודה ליציאה\לכניסה.
חג\שבת\יום רגיל – שבת 1 חג 2 יום רגיל 0 יום שבת וחג באותו היום 3,
תכונה זו נועדה להציג לנו האם יש הבדל בין יום רגיל ולבין חגים או שבתות, כלומר האם בחגים ושבתות כאשר יש תנועה ערה יותר על הכביש ובעקבות כך מספר התאונות משתנה בהתאם.
מרכזיות צומת(נפח תנועה) – (הלמס) לפי נתונים של שנת 2017 : חישוב <u>ממוצע</u> של סכום ספירת התנועה לפי (מקטע,כביש) – כל צומת שבה נפח התנועה גדול מהממוצע ייחשב כצומת מרכזי (1 – מרכזי , 0 – לא מרכזי) , אחרת – ייחשב כצומת לא מרכזי).
תכונה זו מבטאת לנו את העומס באזור של הצומת.
מהירות רוח, טמפרטורה, כמות משקעים – שירות המטאורולוגי (לכל צומת מוצא מי התחנה הכי קרובה אליו ומתייחס לנתונים של תחנה זו) .
אלה הן תכונות של מזג אוויר והן בעצם משתנות באופן יומיומי ולכן ההיפותזה שלנו היא שיש להן השפעה רחבה על תאונות דרכים (יותר תאונות ביום גשום וביום עם רוח חזקה או טמפרטורות חריגות).
ממוצע KNN – כדי ליצור תכונה נוספת, השתמשנו באלגוריתם KNN והוצאנו עבור כל צומת את הממוצע של חומרת התאונה (הסיווג) ומזג האוויר בשלושת הצמתים הכי קרובים אליו.
אך את תכונה זו לא הוספנו לבסוף לDATA מכיוון שקיבלנו כי היא אינה מועילה אלא רק גורעת מהמסווג.
מספר הולכי רגל מנורמל בסך נפגעים (הלמס) – לכל יישוב מחושב היחס בין מספר נפגעי הולכי רגל לסך כל הנפגעים ביישוב. לאחר מכן לכל נ"צ נבחר את היישוב הכי קרוב אליו.
אבטלה לכל ישוב – (הלמס) חישוב ממוצע של אחוז האבטלה לכל יישוב בשנים -2008 2017 והתאמת נ"צ ליישובים ברדיוס של 10 ק"מ (ממוצע).
גיל ממוצע ביישוב – (הלמס) חישוב הגיל הממוצע בכל ישוב ועבור כל צומת בדאטה בצענו ממוצע של גילאים אלו עבור כל הישובים שנמצאים בטווח ה-10 ק"מ.

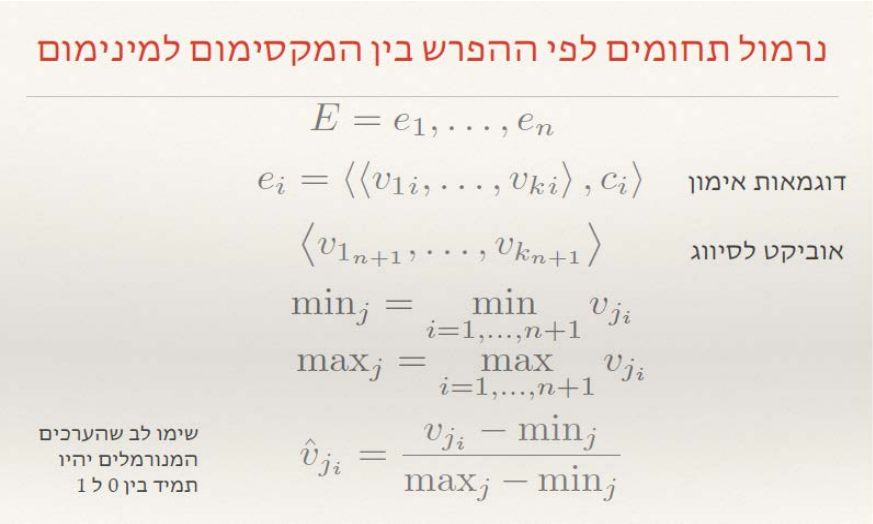
מרומזר\לא מרומזר - (openStreetMap) עבור כל צומת בדאטה יצרנו תכונה נוספת של האם באותו צומת קיים רמזור או לא.
סיווג – (משטרת ישראל והלמס) סיווג הצומת ע"פ התאונות שהתרחשו ברדיוס 10 ק"מ של הצומת הנ"ל. מפורט בהמשך.

<u>הסבר על אופן ביצוע החישובים עבור כל תכונה:</u>

- עבור התכונות שמבצעות התייחסות לרדיוס השתמשנו ב: ball_tree.query_radius על מנת להציג את אותן התכונות שנכללות בטווח הרדיוס לכל צומת. מבנה נתונים זה ball_tree הוא עץ בינארי בו כל צומת מגדיר תת-קבוצה של הנקודות שיש לחפש. בנינו עץ זה עבור הצמתים בדאטה שלנו ולאחר מכן השתמשנו בפונקציה query_radius .פונקציה זו מחזירה את השכנים של צומת ברדיוס בצורה יעילה.
  עבור תכונות אלה בצענו חלוקה לשלושה סוגים:
  <u>מנייה</u> של מקומות ברדיוס עבור: פאבים, מוסדות חינוך, תחנות דלק, אזורי תעשייה, מוזיאונים, תחנות משטרה, מסעדות, מרכזי קניות, מצלמות מהירות, תחנות רכבת וחניונים.
  <u>סכימה</u> של נתונים ברדיוס: כלבים.
  <u>ממוצע</u> של כל הנתונים ברדיוס: אחוזי אבטלה בישובים הנמצאים ברדיוס.
- עבור התכונות אשר מתאימות את עצמן לצומת הכי קרובה, דרך פעולה זו נדרשה עבור מאגרי מידע שידוע לנו מידע עבור נ"צ מסוים לדוגמא מרכז ישוב וכדומה ואנו רוצים לכל צומת להתאים את הנתונים של הישוב הקרוב ביותר. לצורך פעולה זו השתמשנו בKDTREE כדי למצוא את הצומת הקרוב ביותר. עץ זה הוא עץ חיפוש שמיועד לאחסון יעיל של נקודות במרחב וכמו כן למציאת צומת קרוב ביותר בצורה יעילה.
  התכונות הן:
  -מרכזיות של צומת שנובעת מהמאגר של נפח התנועה של הלמס.
  -ממוצע גילאים , כלומר אחוז הקשישים, המבוגרים והילדים מהאוכלוסייה בסביבה, וכן מספר הולכי הרגל שנפגעו לפי נתוני הלמס.

- גם התכונות של מזג אוויר התבצעו באותו אופן, כלומר ראשית שאבנו את הנתונים מהמרכז המטאורולוגי ולאחר מכן התאמנו את הDATA כך שתחנות אשר לא מעודכנות הוסרו(תחנות עם הרבה חוסרים, בוצע באופן ידני), וכתוצאה קיבלנו כי לכל צומת מצאנו את התחנה המטאורולוגית הקרובה ביותר אליו והתאמנו לו את הנתונים שלה.

לאחר הוספת המידע, נרמלנו כל תכונה כפי שלמדנו בקורס:

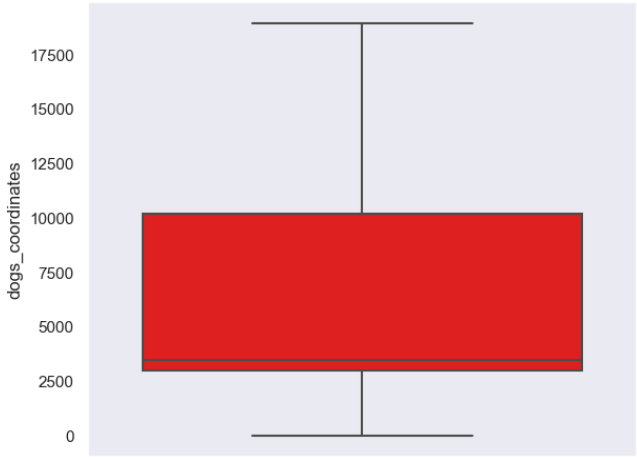


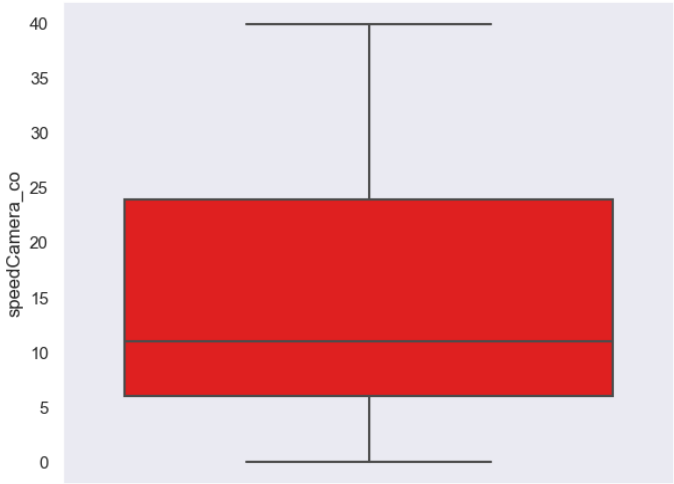
הצגת סטטיסטיקות גולמיות על הנתונים

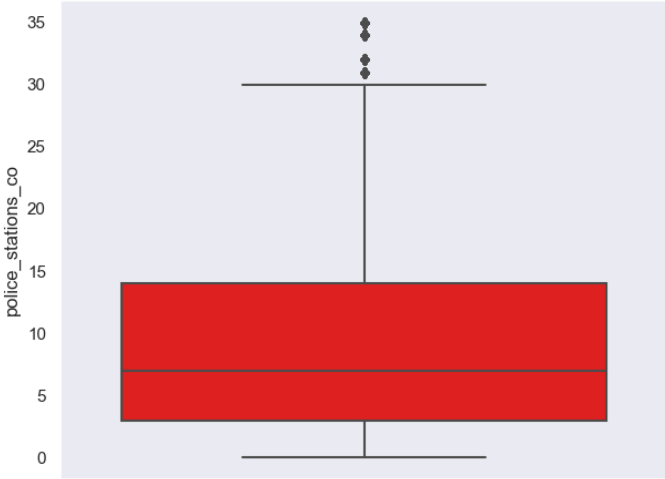
נציין שהערכים בטבלה הינם לפני נרמול כדי שנעבוד על המידע הגולמי שאספנו ונלמד ממנו. הטבלה:

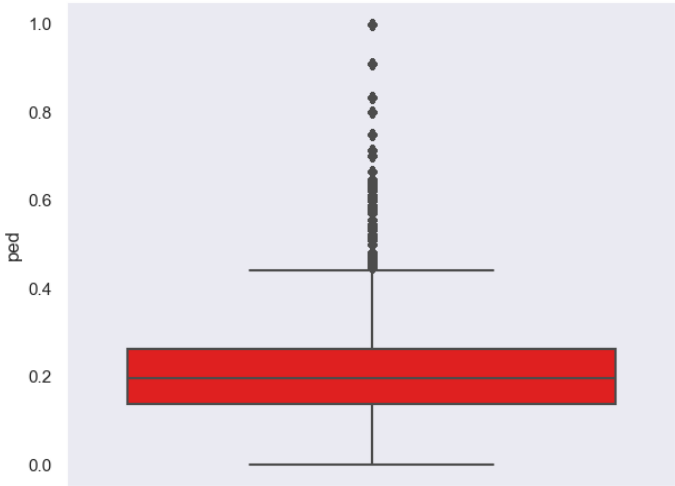
	count	mean	std	min	25%	50%	75%	max
x	13636400	195956	23613.05	136739.8	181047.8	189335.6	208483.5	629417.6
y	13636400	671317.8	49386.37	382949	644301.5	664097.5	692821.4	866621.5
year	13636400	2012.5	2.872281	2008	2010	2012.5	2015	2017
month	13636400	6.526027	3.447851	1	4	7	10	12
day	13636400	15.72055	8.796247	1	8	16	23	31
sug_yom	13636400	0.401918	0.745652	0	0	0	1	3
sum_humra	13636400	0.152212	0.477669	0	0	0	0	8
count_humra	13636400	0.136785	0.415251	0	0	0	0	8
c1	13636400	0.122891	0.391559	0	0	0	0	8
c2	13636400	0.012362	0.112571	0	0	0	0	3
c3	13636400	0.001533	0.039185	0	0	0	0	2
barandpub	13636400	51.04015	60.66088	0	4	22	82	170
education_co	13636400	1868.257	1307.901	0	760	1488.5	3262.25	4304
gas_station_only_israel	13636400	70.04069	45.9853	0	33	58	114	170
industrialarea_co	13636400	23.46494	11.78101	0	15	22	31	72
museum_co	13636400	17.29497	15.31634	0	4	10	34	46
police_stations_co	13636400	8.750535	7.423333	0	3	7	14	35
resturants_onlyCoordinats	13636400	181.2037	172.6951	0	33	96	326	501
shopcenter_co	13636400	28.42184	19.50341	0	10	26	47	70
speedCamera_co	13636400	14.90444	11.39876	0	6	11	24	40
train_co	13636400	8.466542	6.087026	0	3	7	15	21
parking	13636400	629.0158	519.1875	0	175	453	1083	1736
dogs_coordinates	13636400	5852.754	4190.013	0	2995	3476	10230	18972
precentOfAvtala	13636400	0.032433	0.006864	0	0.027566	0.03308	0.037456	0.043227
center	13636400	0.298448	0.457577	0	0	0	1	1
trafiic_signal	13636400	0.74652	0.435003	0	0	1	1	1
Elders	13636400	0.240162	0.078983	0	0.183	0.253229	0.285705	1
Adults	13636400	0.466252	0.042437	0	0.443414	0.465319	0.48652	1
Teen	13636400	0.293586	0.083438	0	0.234605	0.277291	0.333324	0.682171
ped	13636400	0.213354	0.129869	0	0.138743	0.197425	0.262195	1
rain	13636400	0.000966	0.158198	0	0	0	0	76
speed	13636400	1.975363	0.841916	0	1.5	2	2.5	13.5
MAX_TEMP	13636400	32.27971	7.349998	3.7	26	33	38.8	47.9
MIN_TEMP	13636400	19.92384	6.639285	-3.8	14.2	20.5	25.9	36.5

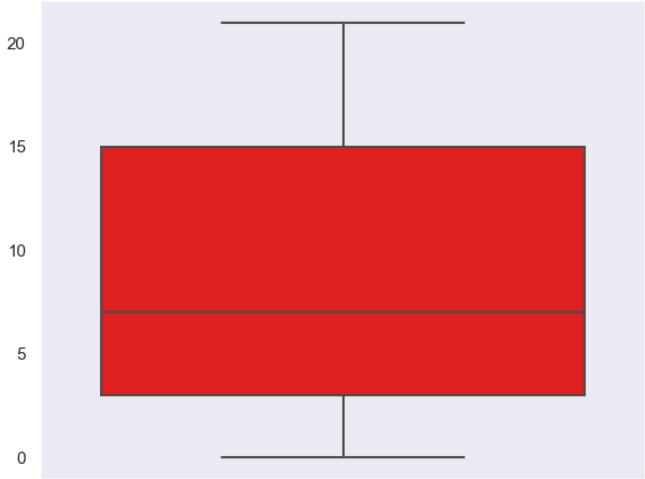
מצורפים גרפי boxplot עבור תכונות משמעותיות המתארים את הפיזור של ערכי התכונה:
הקו הרוחבי הארוך הוא החציון, הקו הרוחבי הקצר העליון הוא ערך המקסימום והקו הרוחבי התחתון הוא ערך המינימום











<u>ניתוח הנתונים:</u>

ניתן לראות שיותר מ75% מהסיוגים של הדוגמאות בדאטה שלנו הם 0.

[count_humra] זהו הסיווג הרלוונטי

נתון זה עלול להוות בעיה מבחינתנו מכיוון שהדאטה לא מאוזן – בהמשך נפרט כיצד החלטנו להתמודד עם בעיה זו.

ניתן לצפות בעקבות נתון זה שעצי החלטה שבהם נשתמש יעבדו יותר טוב עבורנו ואכן בהמשך ניתן לראות שאחוזי הדיוק הגבוהים ביותר התקבלו בעקבות שימוש במסווג של עץ ההחלטה.

בעצי החלטה נבחרת בכל פיצול התכונה שלפיה נקבל צמתים הומוגניים ביותר(בממוצע) ולכן עבור הדאטה שלנו אכן הציפייה לעבודה מדוייקת יותר עבור עצי החלטה.

כמו כן בעקבות הסטטיסטיקות הנ"ל הסקנו שנרצה להשתמש במדד שבודק עבור כל סיווג בנפרד את דיוקו באופן עצמאי מכיוון והדאטה אינו מאוזן וישנו רוב מוחלט של סיווג 0.

עבור התכונות עצמן נרצה לשאוף לסטיית תקן כמה שיותר קטנה כדי למנוע דוגמאות חריגות\"קיצוניות" , כלומר דוגמא שבה קיימת תכונה שערכה מאוד שונה משאר הערכים של תכונה זו בדאטה.

לפי הנתונים ניתן לראות שקיימות מספר תכונות שעבורן סטיית התקן קטנה יחסית אך מכיוון שאנו משתמשים בדאטה בו מספר התכונות מועט לא נרצה לוותר בשלב זה על תכונות אלו וייתכן כי נגלה שדווקא הם הועילו יותר מתכונות אחרות.

ניתן לראות כי מספר הכלבים הממוצע ברדיוס גדול יחסית ואולי נצפה לקבל השפעה גדולה של תכונה זו על הסיווג של צמתים. כמו כן מספר תחנות רכבת ממוצע ברדיוס ומספר תחנות משטרה ממוצע ברדיוס קטנים יחסית. נרצה לשים לב לתכונות אלה כאשר נבדוק מהן התכונות המשפיעות יותר או פחות על תוצאות המסווג.

בנוסף שמנו לב שנתוני כמויות הגשם הם לרוב קרובים לאפס כלומר רוב הדוגמאות מייצגות אבחנה שבה אין גשם או יש כמות מועטה של גשם.

כמו כן אחוז הולכי הרגל שנפגעו בתאונות מתוך כלל הנפגעים הוא נמוך. נתון זה גם יעניין אותנו בהמשך ונרצה לבחון את השפעתו.

ניתן לראות כי המספר המקסימלי של תאונות דרכים באותו הצומת ובאותו היום בעשר השנים האחרונות הוא 8. סכום החומרות המקסימלי הוא גם 8 ולכן נסיק שכל אחת מהתאונות בצמתים אלה היא מחומרה 1(תאונת קלות). כמו כן מספר התאונות המקסימלי מחומרה קשה באותו הצומת באותו תאריך הוא 2.

קביעת הסיווג

הצעד הבא באיסוף המידע היה להגדיר כיצד ייקבע הסיווג לכל צומת בהתאם לפרמטרים. המטרה הייתה קבלת סיווג המייצג את מספר תאונות הדרכים משוקלל בחומרתן לכל צומת ותאריך.

תחילה, חשבנו ליצור את הסיווג של צומת ע"י סכימת החומרה של כל התאונות שהיו בו. למשל, אם בצומת מסוים היו 2 תאונות באותו היום, אחת קלה ואחת קשה נקבע את הסיווג להיות 3+1=4. (נזכיר כי 1 - תאונה בחומרה קלה , 2 – בינונית, 3 – קשה) . כעת עלתה המחשבה איך בעצם צומת נקבע כמסוכן, וכמה תאונות קלות נחשבות "שוות" לתאונה אחת קשה? התלבטנו איך לתעדף צומת אם בצומת אחד היו הרבה תאונות קלות ובצומת אחר הייתה תאונה אחת קשה, איזה מהם מסוכן יותר?

אחרי מחשבה ודיונים רבים בנושא, הבנו שקביעת הסיווג באמצעות סכימת החומרות של התאונות באותו הצומת, ללא התייחסות לחומרות של התאונות, אינו מביא לידי ביטוי באופן הטוב ביותר את הגדרת צומת כמסוכן. לכן, בחרנו לנקוט בדרך פעולה אחרת שנבעה גם מקריאת חומר משטרתי והתייעצות עם גורמי הדוברות. החלטנו לתת יכולת למשטרה לבחור את ההגדרה עבורה מבין 3 אופציות:

אופציה ראשונה, היא בחירת הסיווג של צומת להיות סך כל התאונות שהיו בו, ללא התחשבות בחומרת התאונות.

אופציה שניה, היא בחירת הסיווג של צומת להיות סך התאונות הקשות, או סך כל התאונות הבינוניות, או סך כל התאונות הקלות.

אופציה שלישית, היא בחירת הסיווג של צומת להיות סכום החומרות בצומת.

עבור כל אחת מהאופציות נציין כי הסיווגים מחולקים לclasses כאשר הסיווג הוא ערך שלם בתחום סגור , כלומר סיווג נומינלי עם קבוצת ערכים קטנה וסופית.

קיבלנו שעבור סיווג הסוכם את כלל התאונות בנ"צ , החלוקה של הדוגמאות לסיווגים השונים היא:

סיווג סכום תאונות	מספר הדוגמאות
0.0	12073807
1.0	1314960
2.0	203263
3.0	35614
4.0	7179
5.0	1299
6.0	233
7.0	42
8.0	3

עבור סיווג הסוכם מספר תאונות קשות :

0.0	13615538
1.0	20826
2.0	36

עבור סיווג הסוכם מספר תאונות בינוניות:

0.0	13470973
1.0	162304
2.0	3105
3.0	18

עבור סיווג הסוכם מספר תאונות קלות :

0.0	12215688
1.0	1210570
2.0	173781
3.0	29294
4.0	5845
5.0	985
6.0	192
7.0	42
8.0	3

עבור סיווג הסוכם את כלל החומרות :

0.0	12073807
1.0	1175462
2.0	289960
3.0	75026
4.0	16963
5.0	3998
6.0	985
7.0	185
8.0	14

במהלך הניסויים נבחן את האופציה המרכזית שהיא מספר התאונות, כפי שאנו רואים ההתפלגות של הסיווג על פי מספר התאונות בעל דמיון רב לסיווג על פי תאונות קלות וסיווג על פי סכום חומרות ולכן נגיע למסווג שיהיה רלוונטי עבור שלושתם , שיפור המסווג לסיווג לתאונות קשות היא חלק מהמשך המחקר שניתן לעשות.

לאחר בניית מאגר הדוגמאות נרצה לבנות מסווג המקבל כקלט צומת , תנאי מזג אוויר לאותו היום , ומוציא כפלט את הסיווג המתאים עבורו. לאחר בניית המסווג נרצה לאפשר למשטרת ישראל להכניס כקלט את נתוני היום הנוכחי , מספר הניידות הזמינות , ומהי הדרך בה תרצה לקבוע את הסיווג (לפי האפשרויות שפורטו לעיל) ולקבל את המיקומים המדויקים בהם נחזה סיכוי גבוה לתאונות. חיזוי זה ייעשה באמצעות בחירת הצמתים בהם נקבל את הסיווג החמור ביותר(מספר תאונות מקסימלי בהתאם לבחירת סוג הסיווג\סכום חומרות מקסימלי).

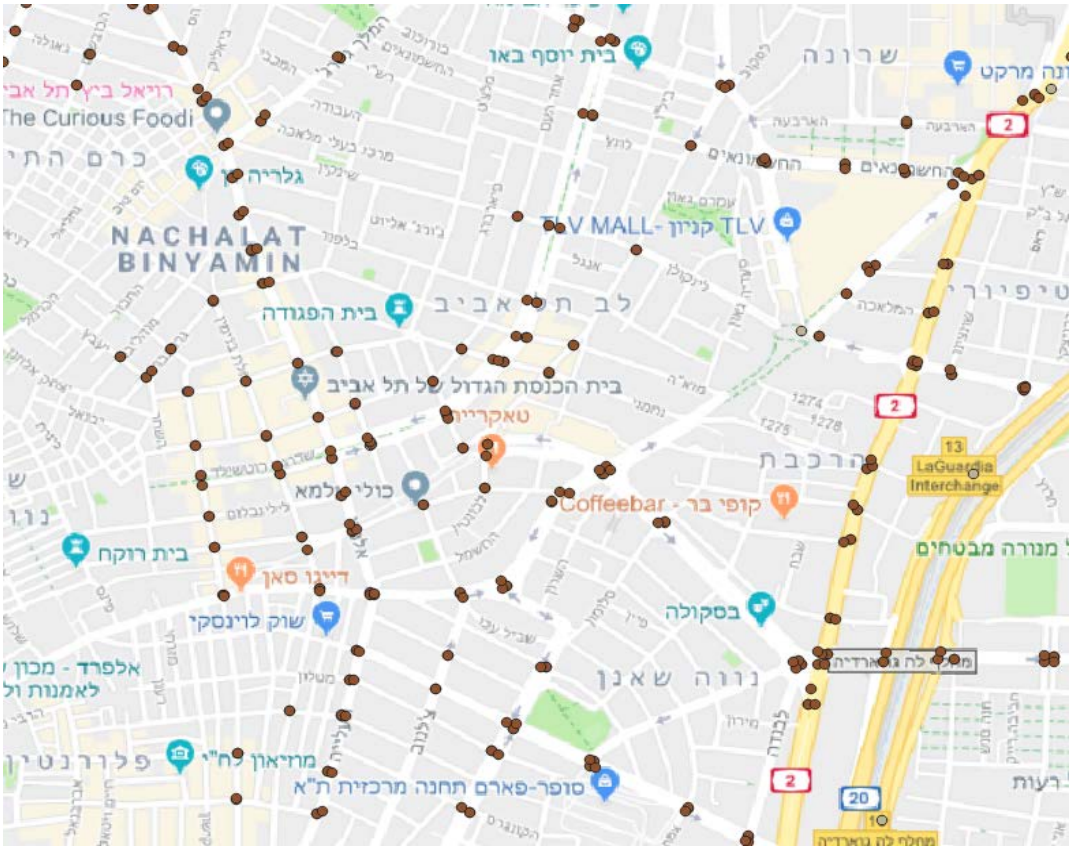
תיאור המערכת:

המערכת אותה בנינו לשם יצירת המסווג עבור חיזוי צמתים מסוכנים מורכבת מסקריפטים שכתבנו ב python 3.7 .

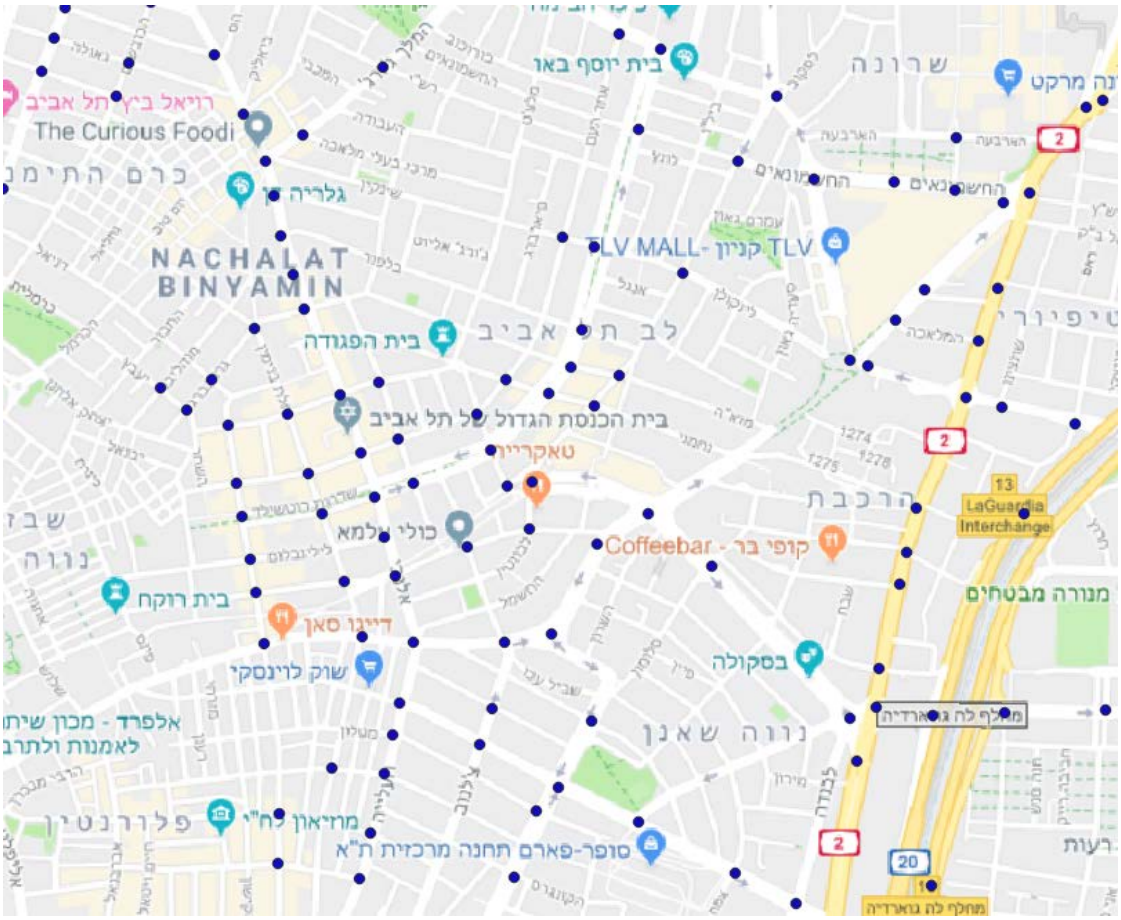
שלב העיבוד המקדים של המידע:

רשימת הצמתים הסופית שאיתה אנו עובדים התקבלה בתהליך של ניסוי וטעיה. עבדנו עם open street map ושאבנו ממנו את הצמתים המרומזרים בארץ, לאחר מכן ראינו שאין כיסוי לצמתים עירוניים ולכן פנינו שוב ל open street map והוצאנו ממנה מידע גם על צמתים שהם לאו דווקא צמתים בינעירוניים , אלא עירוניים. החלטנו להתעלם מכיכרות מכיוון שבדרך כלל כיכרות מונעות את התאונות והבעיה המרכזית היא צמתים. לאחר מכן שמנו לב כי בקבוצת הצמתים שהתקבלה יש קבוצות של נקודות ציון שנמצאות על כניסה שונה של אותה צומת וכך נוצר מצב שיש צמתים עם מספר נקודות ציון עליהם ולכן החלטנו להריץ סקריפט שמבצע סינון. כלומר עבור כל נקודה בודק אם יש נקודות ברדיוס קטן יותר מ50 מטר מהן - אם כן נסיר את הנקודות הנ"ל מהמאגר ובכך קיבלנו קבוצת צמתים שנפרסת כמו שצריך ומונעת קבוצות של מספר נקודות ביחד (כמו למשל במחלף לה גווארדיה בתרשימים מטה).

הצמצום מוצג כאן : תרשים א' מפת הצמתים בתל אביב לפני הצמצום (ניתן לראות שיש קבוצות של צמתים על אותה צומת בדיוק מה שיוצר עומס מיותר בDATA)



תרשים ב' לאחר הצמצום:



הצמצום נעשה על ידי שימוש במציאת שכנים קרובים ביותר תוך שימוש ב ball_tree לחיסכון זמן וכן לשימוש יעיל בפונקציה query_radius אשר מחזירה לנו את כל השכנים ברדיוס מסויים.

עיבוד נוסף שנעשה על המידע הוא עבור מזג האוויר, מכיוון שלא הצלחנו לקבל גישה לAPI של השירות המטאורולוגי מפאת תקלה כלשהי , ביצענו את ההורדה של כל המידע באופן ידני. לאחר מכן ביצענו איחוד של המידע לCSV מאוחד ולאחר מכן גילינו כי ישנם חוסרים בנתונים, כלומר יש ימים חסרים של מידע או שעות ספציפיות של מידע חסר. כדי להתגבר על כך הפעלנו סקריפט אשר כאשר הוא מזהה נתון ריק , הוא ניגש לנתון הקודם ולנתון הבא (מבחינת תאריך כלומר יום לפני ויום אחרי) שקיים ומבצע ביניהם ממוצע. כך ככל הנראה קיבלנו בדיוק מירבי את הנתונים הרלוונטים שהיו בשעות/ימים הנ"ל מכיוון ומזג אוויר הוא בדרך כלל זהה לאורך תקופה קרובה.

הסקריפטים הראשונים שכתבנו התמקדו ביצירת מאגר המידע והתאמת המידע שאספנו לטבלה אותה רצינו ליצור. ביניהם סקריפט המתאים לכל צומת יום בשנים 2008-2017 , התאמת מזג אוויר לכל יום ועוד. (בהמשך – פירוט הסקריפטים).

<u>שלב הלמידה והניסויים:</u>

החלטנו לבדוק את האלגוריתמים הבאים:

KNN,DecisionTree, Random Forest, Extra Trees, gussianNB

ע"י שימוש בחבילת scikit-learn , כאשר נתמקד בעיקר בעצי החלטה וknn.

במהלך הלמידה השתמשנו בתהליך הcross-validation בצירוף K-Fold כפי שלמדנו בקורס בינה מלאכותית עבור כל אחד ואחד מהמסווגים הנ"ל. כך יכולנו להעריך את טיב המסווגים ולבחור את המסווג הטוב ביותר עבורנו.

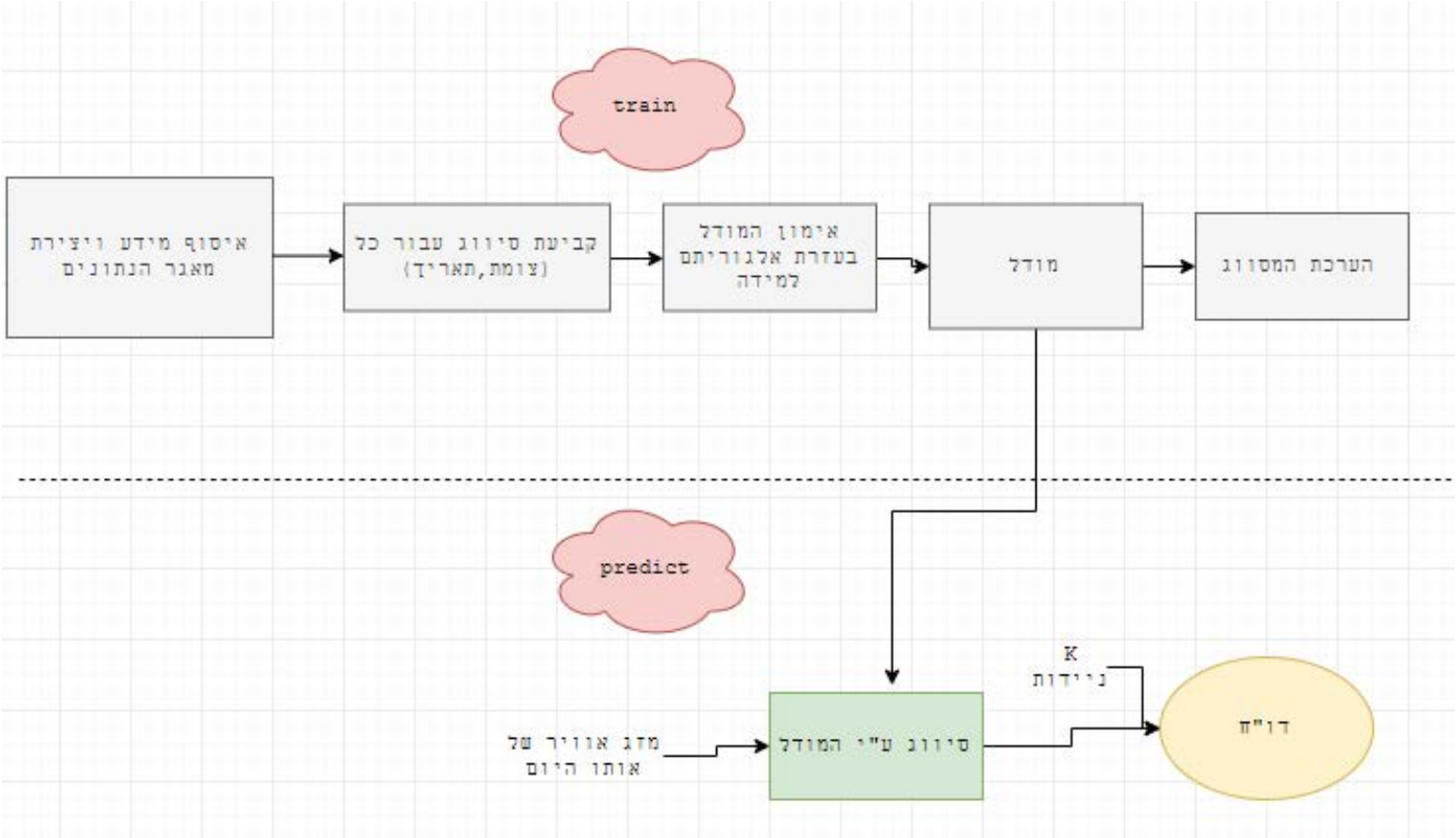
בחנו את האפשרות לשימוש בregression לעומת classification. חקרנו את הנושא והבנו שניתן לקבל סיווג רציף\נומנלי באמצעות שימוש במסווגים הנ"ל בהתאמה. רצינו לבדוק עם איזה מהסוגים נוכל להגיע לדיוק מירבי ולכן בדקנו את התוצאות עבור האלגוריתמים בשתי הגרסאות.

כמו כן ביצענו ניסויים בין תוצאות העצים השונים לעומת הknn , הרצות של knn עבור kים משתנים , הרצות של עצים עם עומקים משתנים, וניסויים של מסוגיים נוספים.

באופן כללי הפרמטרים המרכזיים שבחנו הם : שימוש בדאטה מאוזן\לא מאוזן , שימוש ברגרסור\מסווג , הגדרת הסיווג וסוגי אלגוריתמים שונים ללמידה.

בהמשך נפרט על כל אחד מהניסויים בנפרד כולל תוצאות ומסקנות.

לבסוף גם בחנו אפשרויות הורדה\הוספה של תכונות משמעותיות יותר ומשמעותיות פחות כפי שמפורט בהנדסת ובחירת הפיצ'רים.



<u>קביעת המסווג הסופי:</u>

לאחר שבחרנו את המסווג הטוב ביותר מבין המסווגים שניסינו עם התאמת הפרמטרים, כתבנו סקריפט סופי, אשר אותו תריץ המשטרה בכל יום, הכולל את אותו המסווג.

לפני הסקריפט הסופי עבור הרצה ראשונה יש צורך להריץ את הסקריפט אשר יוצא את הData ועבורו צריכים להיות קיימים הcsv הרלוונטיים.

בסקריפט הסופי, המשטרה תכניס כקלט:

- את מזג האוויר של אותו היום הכולל את כמות הגשם הצפויה, מהירות הרוח, טמפרטורה מקסימלית וטמפרטורה מינימלית.
- את סוג היום, כלומר יום רגיל, יום שבת, יום חג או יום שהוא שבת וחג.
- את מספר הניידות שהמשטרה מקצה מתוך משאביה לאותו היום.

לאחר הכנסת המידע הנ"ל, המשטרה תוכל לבחור את הדרך שבה היא מעוניינת להגדיר צומת כמסוכן, כלומר לבחור מבין האופציות:

`Severe / Medium / Slightly / Total Count / Total Sum`
כאשר `Severe` מתאר כמות תאונות חמורות, `Medium` מתאר כמות תאונות בינוניות, `Slightly` מתאר כמות תאונות קלות , `Total Count` מתאר כמות תאונות כוללת באותה צומת ו- `Total Sum` מתאר סכום חומרות כולל.

בשלב זה, נייצר את מאגר הנתונים עליו נרצה לנבא את המידע, כלומר ללא סיווג- כל הצמתים הקיימים במערכת עם הנתונים שקיבלנו כקלט בתוספת הנתונים הקבועים שאספנו לכל צומת במידע המקדים. בנוסף, טענו את הדאטה הכולל עם הסיווגים- כלומר דוגמאות ללמוד מהם, והרצנו את המסווג הנבחר.

בשלב האחרון, כדי ליצור את הדו"ח עבור המשטרה, לקחנו את n הצמתים שסווגו עם חומרה מקסימלית, כאשר n הוא מספר המשאבים שהמשטרה הכניסה כקלט, והדפסנו את התוצאות למשטרה (בהתאם לסיווג הנבחר).

כמו כן בנינו מפת חום המופעלת בצורה נפרדת המתארת באופן ויזואלי את המיקומים במפת ארץ ישראל בהם נקבעו הצמתים המסוכנות והמסוכנות פחות.

<u>פירוט הסקריפטים:</u>

- **<u>minimizeJunctions.py –</u>** סקריפט זה בעצם מבצע את הפילטור הראשוני של הצמתים. מטרתו היא לצמצם את מספר הצמתים שאספנו בקובץ traffic_signals.csv כך שנקבל צומת אחד שברדיוס של 50 מטר ממנו לא נקבל צמתים נוספים. בסקריפט זה השתמשנו בballTree כדי למצוא את הצמתים ברדיוס של 50 מטר מכל צומת ולהוציא אותם מהדאטה שלנו.
- כדי לאסוף את המידע הגולמי ביצענו עבודה מקדימה של עיבוד הנתונים וקיבוצם לקובץ דאטה אחד גדול – data.csv. חלק מהמידע נאסף ידנית עד כדי הוספה\החסרה של עמודות רלוונטיות ופעולות אריתמטיות פשוטות. עבור חלק מהמידע כתבנו סקריפטים על מנת לבצע ביעלות את עיבודם **<u>, centerlize_junctions.py.</u>**

- **dates.py –** סקריפט זה מכניס לטבלה הגדולה יום, חודש ושנה עבור כל אחד מהימים בשנים 2008-2017 המותאמים לכל צומת. כאן נוצר הקובץ dates_and_coordinates.csv כך שבהמשך בסקריפט המרכזי מתאחד עם הקובץ הראשי של הדאטה , כלומר לכל אחת מהשורות הנוצרות כאן נוסיף את התכונות הרלוונטיות.
- **make_data.py –** זהו הסקריפט המרכזי שבו אנו יוצרים class חדש המייצג את טבלת הדאטה הראשית. אובייקט זה מקבל את קובץ הקוארדינטות (junctions.csv) ואת הרדיוס שלפיו נרצה לחשב את התכונות (כדי לאפשר גמישות מבחינת הרדיוס המוגדר). בסקריפט זה אנו משתמשים בdates ומתחילים ליצור את הטבלה.
  תחילה אנו מוצאים לכל צומת בjunctions את הצמתים הנמצאים ברדיוס הנתון הנלקחים ממאגר המידע של המשטרה, ולאחר מכן מחושב הסיווג בהתאם למידע הנלקח ממאגרי המשטרה של התאונות. לבסוף מתבצע חישוב של כל אחת מהתכונות בהתאם למה שנקבע עבורה וצירוף הנתונים לטבלה המרכזית. הפלט של סקריפט זה הוא בעצם הדאטה הסופי שייצרנו.
- **dataPred.py –** כאן נקבל את הקלט מהמשתמש (מזג אוויר, סוג יום, מספר משאבים, בחירת הסיווג) ונקבל כפלט את n הצמתים בעלי הסיווג החמור ביותר – כאשר n זהו מספר ניידות המשטרה שקיבלנו כקלט מהמשטרה. סקריפט זה בעצם יוצר טבלת דאטה חדשה לכל צומת עם הנתונים הקבועים שאספנו עבורה ומצרף את הנתונים שקיבלנו כקלט. לאחר מכן בנינו את המסווג הנבחר , ביצענו fit לדאטה הראשי שאספנו ולבסוף הפעלנו פונקציית prediction כדי לקבל את תוצאות המסווג. כמו כן השתמשנו בheap על מנת לשלוף את n הצמתים שסווגו כמסוכנים ביותר מתוך כלל הצמתים בjunctions.
- **experiments.py -** מוקדש לניסויים שביצענו תוך כדי הלמידה: הרצות של המסווגים והרגרסורים שבחרנו וכמו כן תהליך הcross-validation וחישובי דיוק עבור כל אחד מהניסויים שביצענו המפורטים בהמשך.

הנדסת פיצ'רים

אחת מהאפשרויות של הוספת תכונות הייתה להריץ knn עבור kים שונים ולהוסיף כתכונה את הממוצע של הסיווגים המתקבלים מהרצת מסווג זה על מנת להגדיל את כמות התכונות שלנו. לאחר תוצאות הניסויים החלטנו שלא להוסיף תכונה זו מכיוון שראינו שהיא גורעת מביצועי המסווג ולא משפרת.

בחירת פיצ'רים

במהלך בניית מאגר הנתונים עליו הסתמכנו בפרויקט שקלנו המון את בחירת התכונות מכיוון שיש מעט תכונות ולכן כל תכונה קריטית, ואולי אפילו נגלה השפעה לא צפויה של תכונות. אחד היתרונות בבחינה של תכונות אלו, הוא להבין עבור תאונות קלות, בינוניות או קשות אילו גורמים סביבתיים עלולים להוביל לעלייה\ירידה במספרן.

ניסינו לחשוב על כל התכונות הרלוונטיות שיכולות בדרך ישירה או עקיפה להשפיע על תאונות דרכים ואותם להוסיף למאגר.

הדבר הראשון שחשבנו עליו הוא תכונת מזג אוויר באותו היום, אותה פיצלנו למספר תכונות רלוונטיות: טמפרטורה מינימלית , טמפרטורה מקסימלית, מהירות רוח וכמות משקעים.

בהמשך על תכונות הקשורות ליום עצמו כלומר האם באותו יום היה חג, שבת או יום רגיל. היה מסקרן לגלות האם התכונה על סוג היום תהיה משמעותית למסווג.

הדבר השני שחשבנו עליו הוא הצומת עצמו. הוספנו את התכונות הבאות: האם צומת מרומזר או לא, ונפח תנועה באותו הצומת (בעזרת נפח התנועה קביעה האם הצומת מרכזי או לא מרכזי).

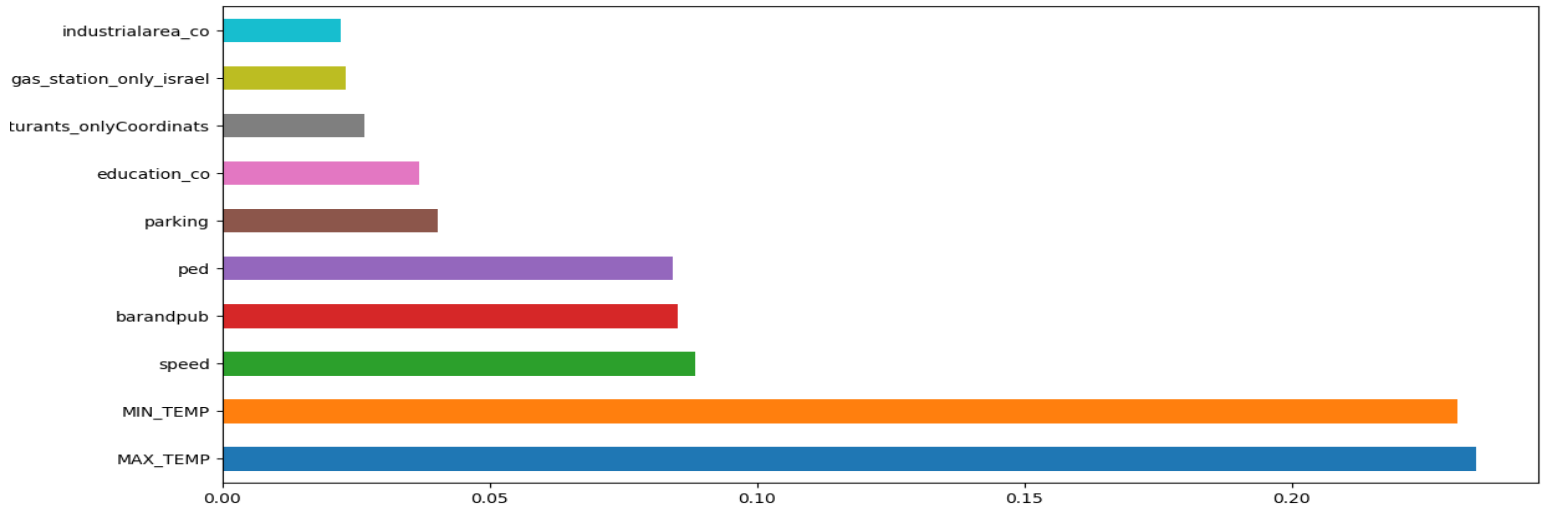
כמו כן רצינו להוסיף מידע הקשור לאזור בסביבה הכוללת של הצומת. בחרנו להגביל את האזור לרדיוס של עשרה קילומטרים. באזור הצומת אספנו מידע על התכונות הבאות: מספר פאבים, מספר מוסדות חינוך, מספר תחנות דלק , אזורי תעשייה, מספר מוזיאונים, מספר תחנות משטרה, מספר מסעדות, מרכזי קניות, מספר תחנות רכבת, מספר בעלי כלבים, מספר חניונים, מספר הולכי רגל, אחוז אבטלה בישובים מסביב לצומת וגיל ממוצע בישוב.

אחרי שבחרנו לבדוק את תכונות אלו, עלתה הסקרנות לגלות אילו מהתכונות הנ"ל השפיעו יותר על המסווג.

- ההרצות הבאות מתייחסות למסווג Extra Tree שהוא המסווג הסופי שבחרנו (לאחר הניסויים המפורטים בהמשך).

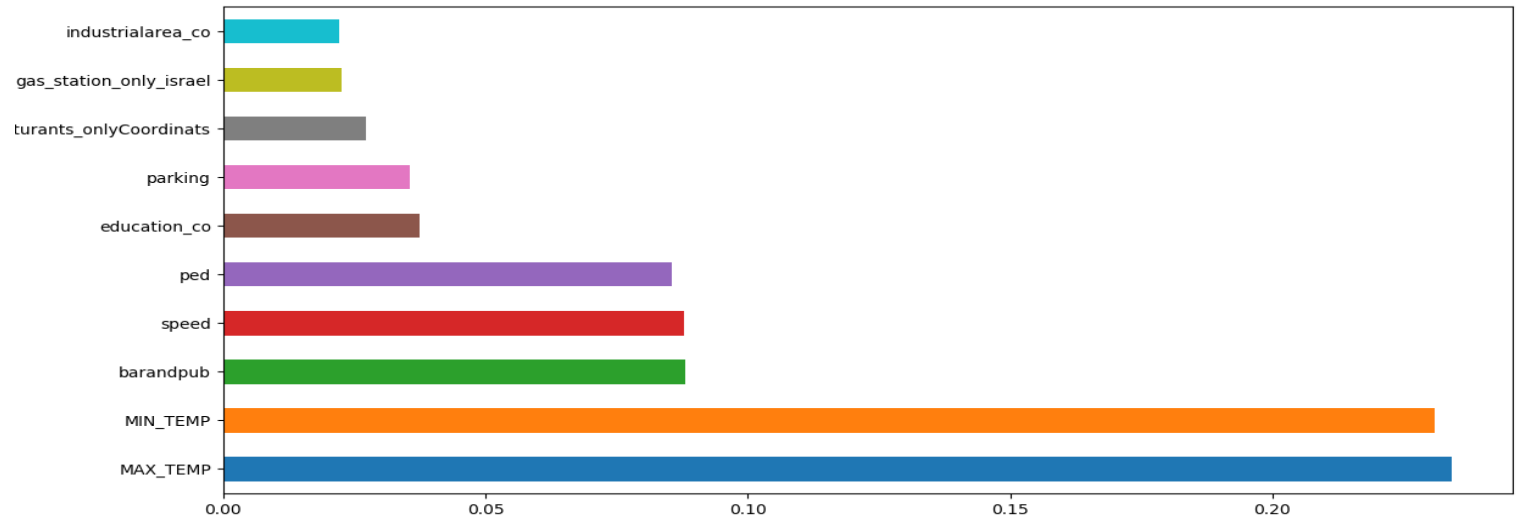
עבור כל סיווג, נציג את עשרת התכונות הכי משפיעות:

<u>עבור סכום התאונות:</u>



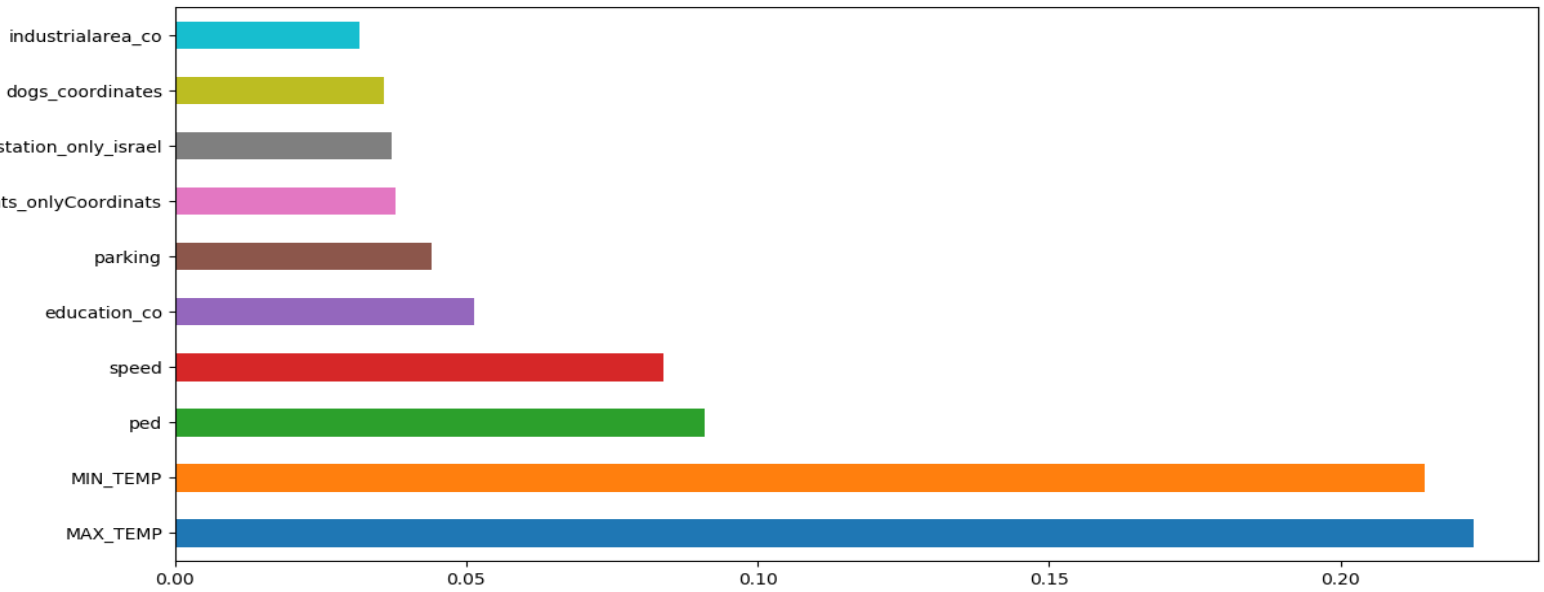
ניתן לראות כי לתכונות מזג האוויר: טמפרטורה מקסימלית וטמפרטורה מינימלית הייתה את ההשפעה הגדולה ביותר על המסווג. לאחר מכן בפער יחסית גדול: מהירות רוח, ברים ופאבים באזור ומספר הולכי רגל. לבסוף, מופיעים חניונים, בתי ספר, מסעדות, תחנות דלק, ואזורי תעשיה.
תוצאות אלה מעט מפתיעות מכיוון שגילינו למשל כי חניונים משפיעים יותר מבתי ספר, אך לא בהרבה.

<u>עבור מסווג מספר תאונות קלות:</u>



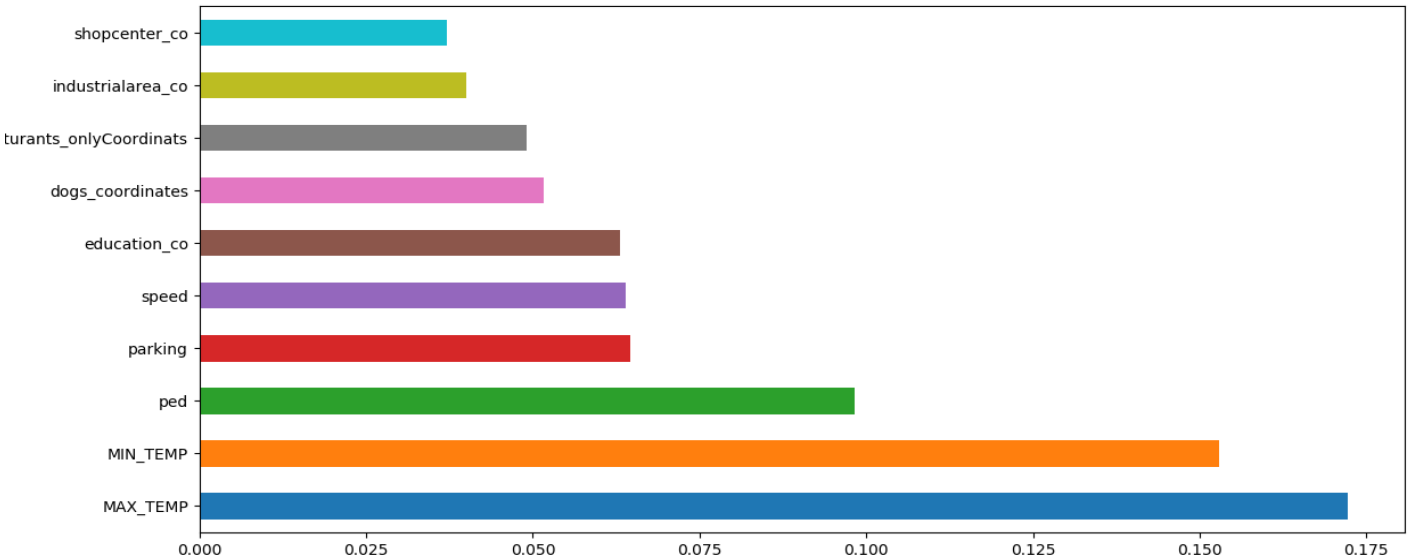
עבור מסווג זה, המספרים יצאו די דומים, אך מה שמעט הפתיע אותנו במסווג הקודם, ניתן לראות כי בתי הספר משפיעים קצת יותר מחניונים באזור הצומת.

עבור סיווג תאונות בינוניות:



עבור תאונות בינוניות בחומרתן, ניתן לראות כי הולכי רגל הינם גורם משמעותי בהרבה מאשר עבור תאונות בחומרה קלה. עוד נתון מעניין שניתן לראות כאן, הוא את בעלי הכלבים באזור שנכנסים לעשרת התכונות המשפיעות ביותר.

עבור סיווג לתאונות קשות:



עבור תאונות קשות, ניתן לראות כי מיד לאחר טווח הטמפרטורות, אחד הגורמים המאוד מרכזיים הם הולכי רגל. מיד לאחר מכן צמתים שבאזורים שלהם יש חניונים, ותכונה נוספת שמופיעה לראשונה היא מרכזי קניות.

לסיכום , מתוך כל המסווגים, ניתן לראות כי התכונות המרכזיות המשותפות אצל כולם הן טמפרטורה מקסימלית וטמפרטורה מינימלית. לאחר מכן התכונות שבאות לידי ביטוי בצורה בולטת הם הולכי רגל, חניונים, מהירות רוח – כחלק מתכונו מזג האוויר, ופאבים.

<u>ניסוי להורדת פיצ'רים</u> – בשלב זה ניסינו להוריד פיצ'רים כדי להגיע לשיפור בביצועים ולכן ניסינו ראשית להוריד את 2 הפיצ'רים שקיבלו את הניקוד הנמוך ביותר והם רמזורים ואחוז הזקנים באוכלוסייה בסביבה.

נציג את התוצאות להשוואה בין שימוש בכל התכונות לבין הורדת 2 התכונות הנ"ל עם המסווג ExtraTreestClassifier(n_estimators=10)

<u>תוצאות עבור כל התכונות:</u>

F1_score=0.917

Error=0.06

MCC=0.521

<u>תוצאות עם הורדת 2 התכונות הגרועות:</u>

F1_score=0.92

Error=0.058

MCC=0.549

בעקבות ניסוי זה החלטנו להשתמש בדאטה שלנו ללא 2 התכונות הנ"ל(אחוז קשישים באוכלסיה ורמזורים)

מתודולוגיה ניסויית:

בפרויקט זה הצבנו לנו למטרה למצוא את אלגוריתם הלמידה האופטימלי לבעיה שהצגנו, תוך כך שאנחנו מבינים כי עלינו למצוא אותו מבין ניסויים רבים שנערוך על אלגוריתמים שונים ומגוונים. חלק מהאלגוריתמים למדנו בקורס מבוא לבינה מלאכותית אך עבור חלקם חקרנו ולמדנו.

לאחר איסוף המידע התחלנו את שלב הניסויים. בשלב זה רצינו לבחון את **דיוק המסווגים השונים** הקיימים וכמו כן להגיע למסקנות עבור בחירת מסווג המתאים לבעיה שלנו (בהתאם לדאטה הקיים והסיווג).

כדי לדרג את טיב המסווגים שבחנו בניסויים השתמשנו במספר שיטות השוואה. שתי השיטות הראשונות נועדו לבחינת הרגרסורים.האחת, היא טעות ריבועית ממוצעת , MSE. מטריקה זו בוחנת את ההבדל בין החיזוי לסיווג כאשר השאיפה היא לטעות ריבועית ממוצעת עם הערך המינימלי, כלומר החיזוי הכי קרוב של ערך הפרמטר האמיתי.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

השנייה, היא R2 score המייצגת את **coefficient of determination** כלומר היא מייצגת את היחס של מספר המשתנים התלויים הניתנים להסבר מהתכונות הלא תלויות. כלומר זה בעצם שקול לscore עבור מסווג בינארי ונועד לתת "ציון" למסווג. נתון זה מספק מדד לשכפול התוצאות שנצפו היטב על ידי המודל , בהתבסס על שיעור השונות הכולל של התוצאות שהוסברו על יד המודל.

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

השיטות הבאות הן עבור מסווגי classifier בעלי multiclass .

השיטה השלישית, היא confusion matrix. בשיטה זו לרוב משתמשים כאשר הדאטה לא מאוזן כמו במקרה שלנו.

דרך מטריצה זו, מתאפשרת הדמיית ביצועי האלגוריתם. כל שורה של מטריצה מייצגת מופעים של דוגמאות תחת ה class של החיזוי, וכל טור מייצג את הדוגמאות שקיים להם סיווג ידוע. מטריצה זו לבסוף מציגה עבור כל class בסיווג שלנו, עבור כמה דוגמאות הוא צדק ועבור כמה דוגמאות המסווג טעה. לפי תוצאות המטריצה נוכל להסיק מסקנות עבור איכות המסווג. היתרון של דרך זו, היא בכך שהיא טובה יותר עם מאגר נתונים שמספר הדוגמאות עבור כל class בסיווג אינו באותו סדר גודל.

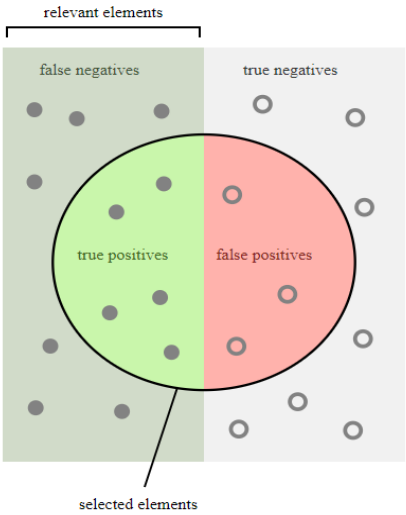
	Actual Cancer = Yes	Actual Cancer = No
Predicted Cancer = Yes	True Positive (TP)	False Positive (FP)
Predicted Cancer = No	False Negative (FN)	True Negative (TN)

<u>הערה:</u>

הקוד האחראי על הצגת המטריצה הנ"ל באופן גרפי עם צבעים נלקח מהאתר הרשמי של scikit-learn תחת confusion matrix .

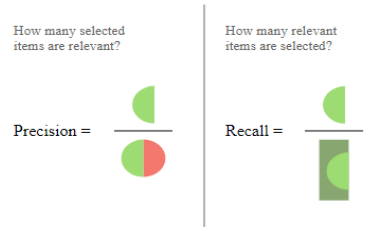
הרביעית היא f1-score – זהו מדד בין 0 ל1 אשר בעצם מהווה ממוצע בין ה **recall** לבין ה **precision** כלומר :

$$F_1 = \left(\frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} \right) = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$



כך אנו מקבלים את הממוצע בין הדוגמאות שהפרדיקט החזיר עם סיווג x והם נכונות לבין סך הדוגמאות שסווגו כx.

כמו כן ביצענו התייחסות לכל class בנפרד.



החמישית היא MCC-

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

זהו מדד המסתמך על תוצאות confusion matrix ומתאר בדיוק רב יותר את התוצאות עבור סיווגים בדאטה לא מאוזן ע"י בדיקת כל האופציות עבור סיווג נכון וסיווג לא נכון.

והשיטה האחרונה למדידה היא שיטה שאנחנו הגדרנו והיא בעצם ממוצע השגיאה – error

המחושבת ע"י ממוצע ההפרשים בין הערכים האמיתיים לערכי הpredict.

<u>פירוט הניסויים:</u>

תחילה נציין כי המאגר יצא לא מאוזן מבחינת כמות דאטה של צמתים שהיו בהם 0 תאונות או כמות תאונות כלשהי. כלומר יש הרבה יותר מידע עם 0 תאונות, מאשר עם כמות של 1 2 או 3 תאונות. בפרויקט בחרנו לנסות שתי גישות שונות:

- הראשונה, היא שימוש במאגר הנתונים שלנו ללא שינוי. למרות אי האיזון הנ"ל, רצינו לבחון את התוצאות על הדאטה הגולמי.
- השנייה, היא שינוי מאגר הנתונים. בגישה זו, רצינו להסיר את חוסר האיזון בין הסיווגים. ניתן להתמודד עם בעיה זו בשלוש דרכים שונות:

1. דרך ראשונה, הייתה להשתמש ב weights. בדרך זו, האלגוריתם בוחר "להעניש" את הסיווג שמופיע הרבה לפי משקל.
2. דרך שניה,היא under-sampling. בדרך זו, האלגוריתם בוחר דוגמאות רנדומלית מבין הclassים עם יותר הדוגמאות, משווה

את הגודל שלהם אל הclass עם מספר הדוגמאות הקטן ביותר.

3. דרך שלישית,היא צמצום class 0 (ללא תאונות) פי 3.

עבור הגישה הראשונה רצינו לבחון מסווגים לעומת רגרסורים , ושינוי פרמטרים עבור כל אחד מהם.

כמו כן, במהלך הניסוי על הדאטה ללא שינוי המאזן, קיבלנו מדדים מאוד גבוהים, כאשר השתמשנו בscore הרגיל וכן ב accuracy הרגיל. הבנו שהדבר נובע מהחוסר איזון של הדאטה. מכיוון שיש לנו רוב מוחלט של דוגמאות מקלאס 0 אז קיבלנו הטעיה מוחלטת לטובתו ולכן רוב הסיווגים היו 0 והדיוקים בקלאסים האחרים היו מאוד נמוכים, בשל כך החלטנו להוסיף את המדידה של הcunfussion matix וכן את MCC המיועד לדאטה הבנוי מסיווגי קלאסים לא מאוזנים. ואכן הגענו לדיוקים ריאליים יותר.

הערה נוספת: הניסויים בוצעו על דאטא של 5 שנים רנדומליות מתוך 10 השנים ולא כולל 2017 וזאת כדי לחסוך זמן בביצוע הניסויים וכן כדי לשמור את 2017 לבדיקה הסופית של המסווג.

<u>הניסויים הכלולים בגישה הראשונה:</u>

ניסוי 1:

במהלך הפרויקט קראנו על מסווגים רבים. מתוכם בחרנו את אלו שמתאימים לסיווג נומינלי, למספר תכונות יחסית מצומצם ,דאטה יחסית גדול וסיווגים לא מאוזנים (75% מסווג כ-0).

בניסוי הראשון בחרנו להשוות בין תוצאות רגרסורים עבור עצים שונים כאשר הסיווג הוא סכום החומרות הכולל בצומת, המוטיביציה לבחון ראשית את העצים היא מכיוון שקראנו מאמרים בנוגע לדאטה לא מאוזן אשר מציגים תוצאות ניסויים שבהם התקבל כי עצים מבצעים את הביצועים הטובים ביותר.

- ניסינו 3 סוגים שונים של עצים:

1.<u>decisionTreeRegressor –</u> עצי רגרסיה הם סוג של עצי החלטה בהם מותאם ערך רציף לכל תצפית כזו.

הרצה של cross validation עם KFold=5 הניבה את התוצאות:

R2_score= 0.364 MSE=0.106

2. <u>RandomForestRegressor-</u>

יער אקראי הוא ועדת עצים המשלב בחירת תת קבוצות של דוגמאות ובחירת תת קבוצה של תכונות. היתרון של יער הוא בכך שזו קבוצת עצים שמחליטה על סיווג במשותף ובכך מקטינה את הרעש בלמידה. ביער אקראי רגרסיה מתבצע ממוצע מבין כל ההחלטות של המסווגים בוועדה.

הרצה של cross validation עם KFold=5 הניבה את התוצאות:

R2-score=0.483 MSE=0.086

3. ExtraTreesRegressor – ה class הזה הוא סוג של יער אקראי העובד על דגימות משנה מגוונות של מערך הנתונים. בעיקרון, ExtraTreesRegressor ו- RandomForestRegressor די דומים, הראשון מהיר יותר, ומדוייק יותר כאשר יש הרבה רעש. אם כל התכונות רלוונטיות הביצועים שלהם יהיו די דומים.

הרצה של cross validation עם KFold=5 הניבה את התוצאות:

R2-score=0.538 MSE=0.077

ניתן לראות מהניסוי כי כפי שציפינו ExtraTreesRegressor הגיעו לתוצאות הטובות ביותר וזאת מכיוון שאין לנו הרבה רעש בדאטה מפני שטיפלנו בזה וכן הגיוני כי החלטה על פי מספר עצים תיהיה מושכלת יותר מהחלטה של עץ בודד.

ניסוי 2:

בניסוי זה בחרנו לחקור את הבדלי התוצאות שקיבלנו בין מסווגי עצים עבור ה-classifiers. כעת נבצע גם כיוונון פרמטרים עבור העצים (מכיוון שעל פי ניתוח הדאטה ניתן לראות שסיווג כבעיית multiclass היא האופציה הריאלית יותר).

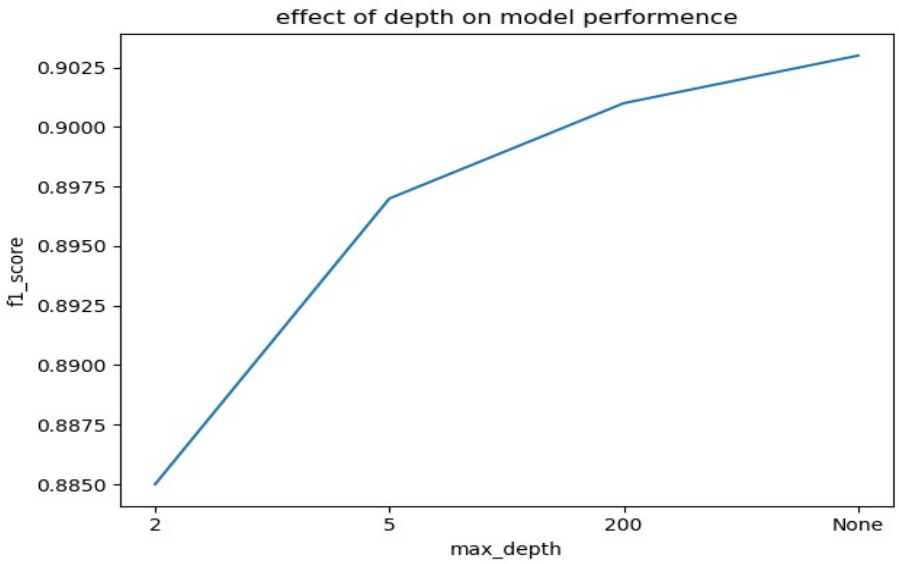
נשווה את טיב האלגוריתם בעזרת הדרכים הבאות:

DecisionTreeClassifier:

MAX_depth :

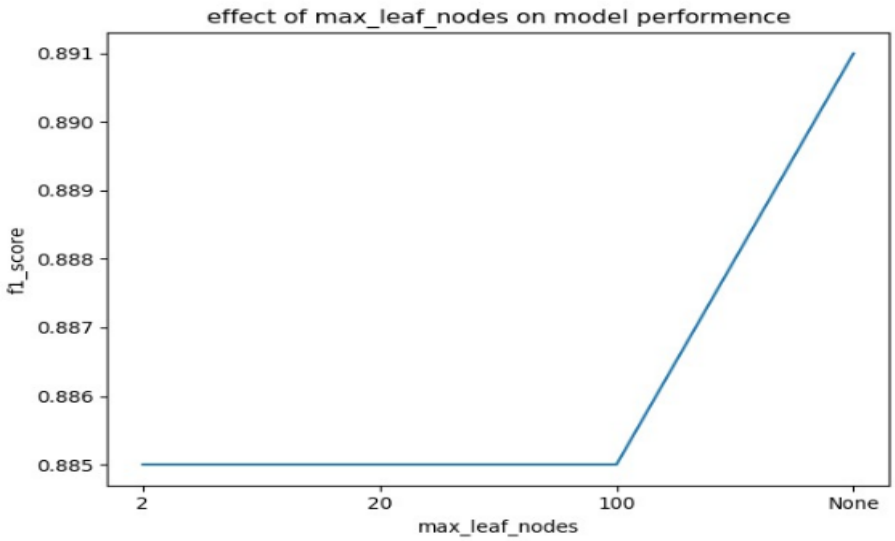
הפרמטר הראשון שבדקנו הינו עומק העץ המקסימלי:

Error average	F1_score(micro)	Matthews_corrcoef	Max_depth
0.138	0.885	0	2
0.065	0.897	0.59	50
0.041	0.901	0.595	200
0.029	0.903	0.6	None



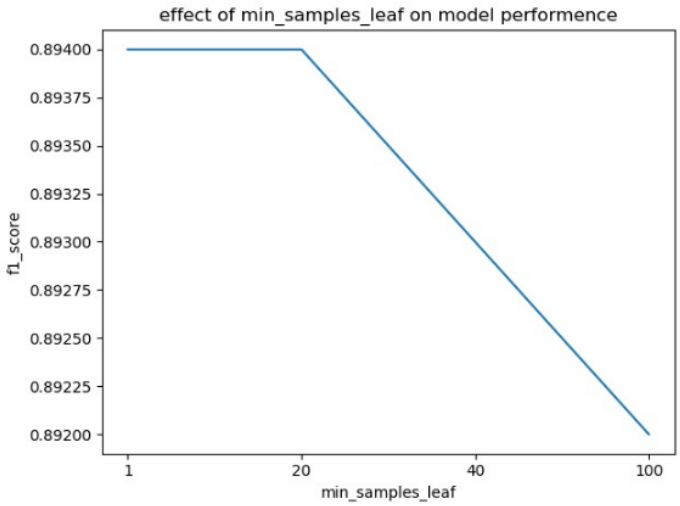
פרמטר שני שבדקנו הוא מספר עלים מקסימלי עבור כל צומת בעץ

Error average	F1_score(micro)	Matthews_corrcoef	Max_leaf_nodes
0.138	0.885	0	2
0.137	0.885	0	20
0.133	0.885	0.026	100
0.098	0.891	0.163	None



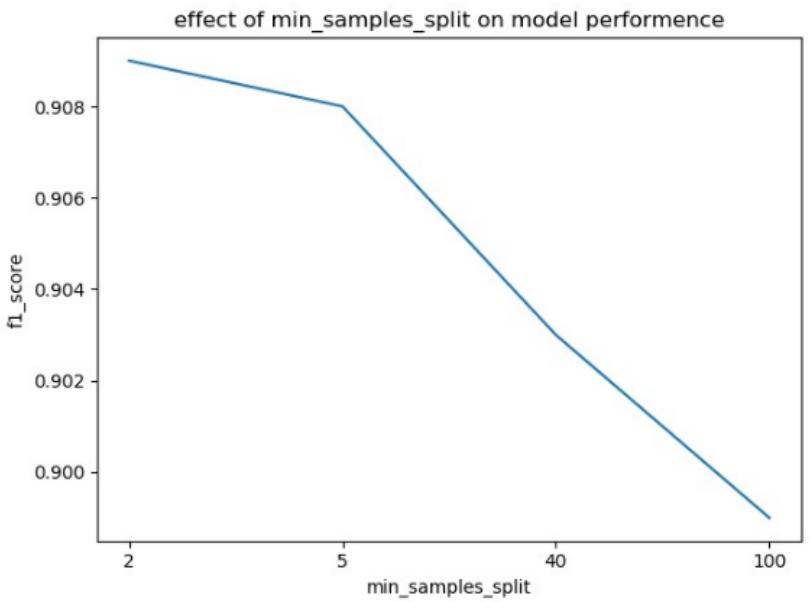
פרמטר שלישי שבדקנו הינו מספר מינימלי של דגימות הדרוש בעלה

Error average	F1_score(micro)	Matthews_corrcoef	Min_samples_leaf
0.077	0.894	0.245	1
0.075	0.894	0.264	20
0.076	0.893	0.27	40
0.079	0.892	0.265	100



פרמטר רביעי שבדקנו הינו את המספר המינימלי של הדגימות הדרוש לפיצול צומת פנימי:

Error average	F1_score(micro)	Matthews_corrcoef	Min_samples_split
0.007-	0.909	0.574	2
0.002	0.908	0.557	5
0.016	0.903	0.504	40
0.03	0.899	0.461	100



לאחר ההרצות הנ"ל הגענו למסקנה כי המודל האופטימלי עבור מסווג זה הוא עבור הפרמטרים:

ללא עומק מקסימלי, ללא מקסימום עלים, בעלה דגימה אחת מקסימלית וחמש דגימות עבור פיצול צומת פנימי:

DecisionTreeClassifier(random_state=0,max_deph=None,min_samples_leaf=1,max_leaf_nodes=None,min_samples_split=5)

-RandomForestClassifier

הפרמטר הראשון שבדקנו הוא מספר העצים בוועדה ועבור איזה מספר נקבל את התוצאות הטובות ביותר.

Error average	F1_score(micro)	Matthews_corrcoef	N_estimators
0.08	0.908	0.449	10
0.08	0.912	0.473	50
0.08	0.916	0.506	100
0.081	0.916	0.507	200

באופן די צפוי קיבלנו שככל שיש יותר עצים בוועדה, כלומר יותר דעות, קיבלנו תוצאות יציבות יותר למרות שעבור הרצה כזאת לוקח הרבה יותר זמן.

פרמטר נוסף הינו המספר המינימלי של דוגמאות הדרוש לפיצול צומת פנימי:

Min_samples_split	Matthews_corrcoef	F1_score(micro)	Error average
2	0.449	0.908	0.08
10	0.383	0.902	0.094
30	0.328	0.898	0.103

לאחר ההרצות הנ"ל הגענו למסקנה כי המודל האופטימלי עבור מסווג זה הוא לעבוד עם הפרמטרים:

מספר דגימות מינימלי של 2 ו-200 עצים בוועדה.

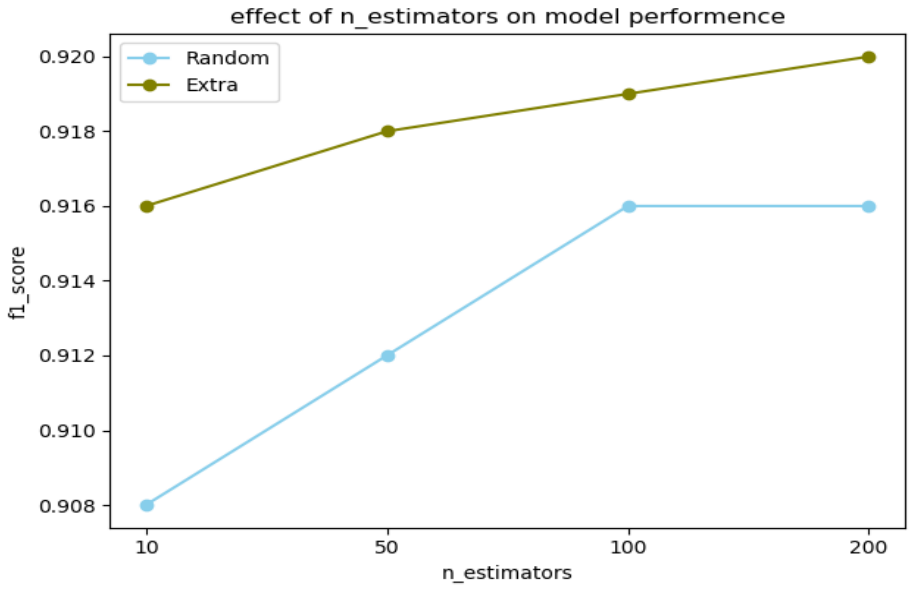
RandomForestRegressor(random_state=0,min_samples_split=2,n_estimators=200)

מודל זה תואם לשורה המסומנת באדום.

ExtraTreesClassifier:

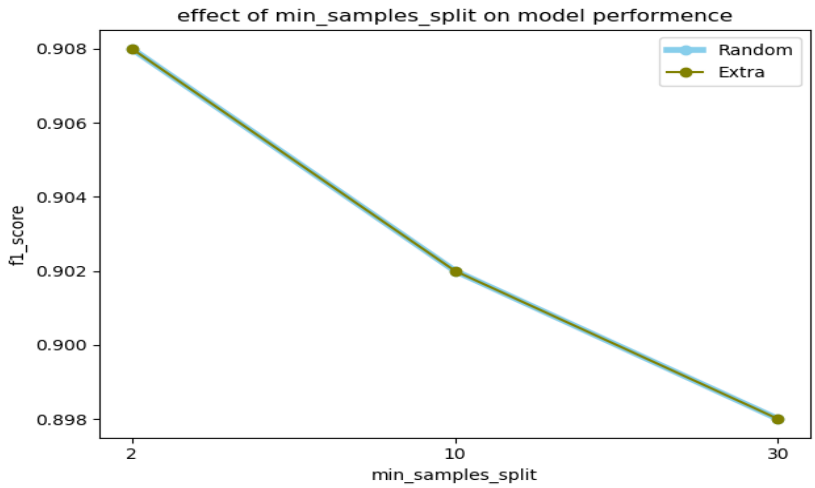
גם פה הפרמטר הראשון שבדקנו הינו מספר העצים בועדה:

N_estimators	Matthews_corrcoef	F1_score(micro)	Error average
10	0.512	0.916	0.074
50	0.523	0.918	0.07
100	0.53	0.919	0.068
200	0.536	0.92	0.067



הפרמטר השני שבדקנו הוא מספר הדוגמאות המינימלי לפיצול צומת:

Min_samples_split	Matthews_corrcoef	F1_score(micro)	Error average
2	0.449	0.908	0.08
10	0.383	0.902	0.094
30	0.328	0.898	0.103



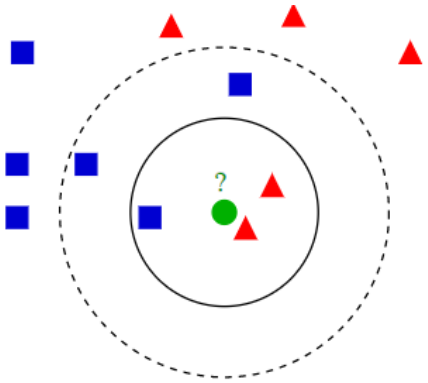
גם כאן המודל בעל התוצאות הטובות ביותר הוא בעל 200 עצים ופיצול מינימלי של שתיים התואם לשורה האחרונה בניסוי הראשון.

ניתן לראות כי ExtraTreesClassifier מקבל תוצאות טובות יותר מבין השלושה, למרות ש RandomForestClassifier מקבל תוצאות דומות, אבל ה ExtraTreesClassifier מהיר יותר ולכן נרצה להמשיך ולעבוד איתו.

ניסוי 3:

כעת רצינו לבחון את המסווג knn (כמסווג או כרגרסור) וכמו כן לבדוק את ההשפעה של שינוי הk.

- **מסווג KNN** –



KNN הוא אלגוריתם לימוד מבוסס מופעים, כלומר מסווג את הדוגמאות ע"פ K התצפיות הקרובות ביותר במרחב התכונות.

מסווג הKNN יכול לשמש לסיווג או לרגרסיה.
במהלך הפרויקט התלבטנו מאוד באיזה סוג להשתמש. הKNN לסיווג משייך את הדוגמא החדשה למחלקה הנפוצה ביותר בקרב K השכנים הקרובים ביותר בעוד ש KNN לרגרסיה מחזיר את ממוצע הערכים של ערכי K השכנים הקרובים ביותר.
תחילה חשבנו כי מסווג הרגרסיה מתאים לנו יותר מכיוון שרצינו לקבל את רמת המסוכנות הממוצעת של צומת כערך, אך לאחר מכן חשבנו שמכיוון שהסיווג הינו מחולק ל class'ים נרצה לקבל סיווג שלה לערכים קיימים.
כאשר המטרה שלנו היא לקבל את הצפי המדויק ביותר עבור התאונות, החלטנו לנסות את שתי האופציות ולבחון איזו מהן מביאה את הדיוק המקסימלי.
את המושג "קרוב ביותר" במקרה שלנו קבענו לפי תנאי מזג האוויר, כלומר, נסווג את הדוגמאות לפי טמפרטורה מינימלית ומקסימלית, מהירות רוח וכמות משקעים.

2 סוגי האלגוריתם שהרצנו קיבלו דאטה של דוגמאות כך שלכל דוגמא יש את התכונות שתיארנו מעלה – כמות משקעים, מהירות הרוח, טמפרטורה מינימלית וטמפרטורה מקסימלית. כמו כן עבור כל דוגמא כזו ניקח גם את הסיווג המתאים כפי שהגדרנו באיסוף המידע. אימנו את המסווג על קבוצת דוגמאות זו ולאחר מכן קיבלנו את חיזוי המסווג עבור השנים 2008-2017. בגלל האופן שהגדרו "קרבה" נוכל לקבל סיווג הקרוב ביותר לאזור מבחינת מזג אוויר, ולאו דווקא מבחינה קרבה פיזית.
בחרנו לבצע מספר ניסויים על k שונים כך שנקבל את התוצאה המדויקת ביותר.
ניסינו לכוונן את פרמטר הk למשימת הלמידה הספציפית ע"י cross-validation (חלוקת הדאטה לתתי-קבוצות ובדיקת אחוזי הדיוק על מנת לבחור את הk שמוביל לדיוק הטוב ביותר).

התוצאות -

<u>:knn-regression</u>

לאחר כיוונון פרמטרים מצאנו כי עבור kd_tree וכן עבור leaf_size=500 האלגוריתם רץ הכי מהר וכן מקבל את התוצאות הגבוהות ביותר.

KNeighborsRegressor(n_neighbors=K,algorithm="kd_tree",leaf_size=500,n_jobs=-1)

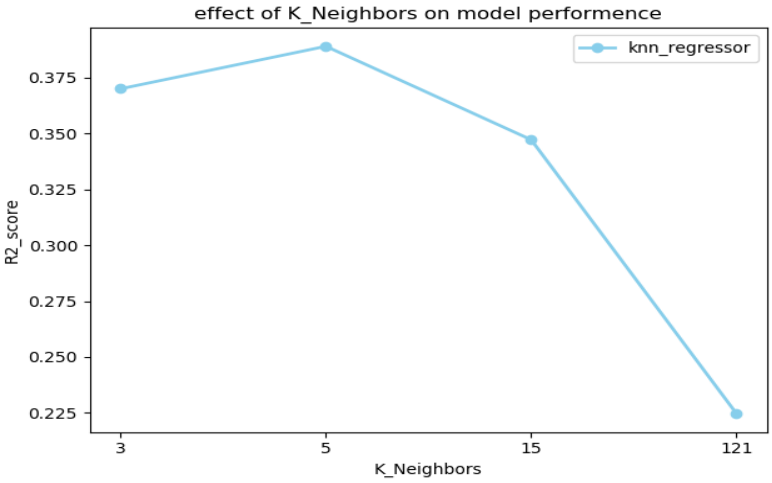
ביצענו שני ניסויים עם הKNN ניסוי אחד רק עם התכונות של המזג אוויר, וניסוי שני עם כל התכונות ולהלן התוצאות:

עבור KNN עם תכונות מזג אוויר של גשם, מהירות רוח, טמפ מקסימום וטמפ מינימום קיבלנו:

K=3	MSE= 0.2259	R2_score=-0.34
K=5	MSE= 0.1999	R2_score= =-0.184
K=15	MSE= 1.4955,	R2_score=-0.093

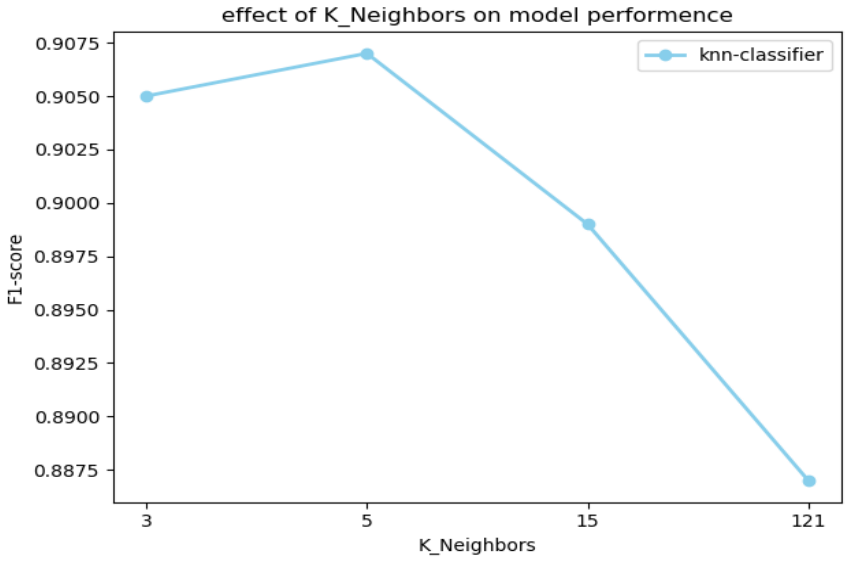
עבור KNN עם כל התכונות:

K=3	MSE= 0.1061	R2_score=0.370
K=5	MSE= 0.1028	R2_score= =0.389
K=15	MSE= 0.1100	R2_score=0.3472
K=121	MSE = 0.1306	R2_score=0.2247



:Knn-classifier

K	F1-score	error
K=3	0.905	0.041
K=5	0.907	0.06
K=15	0.899	0.093
K=121	0.887	0.124



ראשית, ניתן להבין כי עבור knn classifier והערך K=5 נקבל את התוצאות הגבוהות ביותר מבחינת מדד ה F1. בנוסף, מבחינת מדדים של שני סוגי האלגוריתם, ניתן לראות כי התייחסות אלגוריתם הKNN לסיווג כ class, משפרת את הדיוק משמעותית.

ניסוי 4:

- עוד מסווגים שניסינו לחקור:

Linear_model.Ridge():

מודל זה פותר בעיות רגרסיה כאשר פונקציית העלות (פונקציה הממפה מאורע או ערכים של משתנים למספר ממשי המייצג עלות של מאורע) היא פונקציית הריבועים הפחותים. מסווג זה והבא הם חלק מהטכניקות להפחתת המורכבות ולמניעת התאמת יתר.

עבור מסווג זה קיבלנו את התוצאות:

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}|w_j|$$

Cost function for Lasso regression

$$\text{For some } t>0,\ \sum_{j=0}^{p}|w_j|<t$$

0.156=MSE

0.059 =R2-score

נשים לב שקיבלנו MSE מאוד נמוך כלומר תוצאה טובה אך R2-Score גרוע, הדבר נובע מצורת הדאטה שלנו מכיוון שהדאטה שלנו לא מאוזנת ומוטה מאוד אל האפס אז ככל הנראה רוב הניחושים של המסווג הליניארי קרובים מאוד לאפס ולכן השגיאה היא קטנה מאוד אך בשאר הסיווגים שהם לא אפס ומהווים מיעוט השגיאה היא גדולה ולכן R2 מאוד נמוך.

AdaBoostRegressor:

המסווג הזה הוא אומדן שמתחיל בהתאמה של הרגרסור על הדאטה המקורי ואז מתאים עותקים נוספים של הרגרסור על אותו הדאטה אבל עם משקלים של הדוגמאות מותאמות לשגיאה של התחזית הנוכחית. ככזה, הרגרסור הזה מתמקד יותר במקרים קשים.

התוצאות של המסווג הזה הם:

AdaBoostRegressor(n_estimators=10)

R2-score=-0.0089,.MSE=0.177

ניתן לראות כי התוצאות של מסווג זה הם לא טובות בכך שהשגיאה של MSE מאוד גדולה והscore שלילי.

:gaussianNB()

בסיווג נאיבי גאוסיאני, בשונה מנאייב בייס, נשתמש כאשר התכונות הן לא בינאריות אלא רציפות, שיש טווח אינסופי של ערכים. בסיווג כזה, מעריכים לכל תכונה פונקציית התפלגות ומעריכים את הפרמטרים של הפונקציה בעזרת מדגם.

התוצאות שהתקבלו עבור המסווג הזה הם:

MSE=1.449 R2_score = -7.8

קל מאוד לראות שהתוצאות שקיבלנו הן מאוד לא טובות, ושהמסווג זה אינו אופטימלי לפתרון הבעיה שלנו.

השימוש באלגוריתמים אלו נעשה באמצעות שימוש בספריות sklearn קבלת תוצאות עבור score מגוונים התקבלה על ידי שימוש בMULTISCORRER.

בחרנו לעבוד עם ספריית sklearn כי היא כוללת בתוכה אלגוריתמים שונים בתחום הבינה מלאכותית, ולכן נוחה לעבודה במיוחד. בנוסף, היא חשובה כי הינה משתפת פעולה עם ספריות כמו Numpy שבהם השתמשנו לאורך הדרך.

המטרה העיקרית של שימוש במודל ה MULRISCORER הייתה הפונקציונליות שלו. מודל זה מאפשר לקבל יותר מ score אחד בcv. בפרויקט השימוש בו היה לנו שימושי כי scoer בעיקרון מחזיר מספר בודד, ומכיוון שהשתמשנו בו על מנת להשוות אובייקטים, רצינו מספר תוצאות להשוואה.

<u>הניסויים הכלולים בגישה השניה:</u>

<u>הקדמה:</u> הגישה השנייה נובעת מהסיבה שהדאטה אינו מאוזן , ולכן רצינו לבצע ניסויים שאולי יתרמו לדיוק רב יותר עבור הקלאסים הקטנים. כעת לאחר שראינו כי המסווגים טובים יותר מהרגרסורים, וכן ראינו שהאחוז דיוק עבורם גבוה מאוד חקרנו את הבעיה ע"י הוספת מטריקות נוספות אשר נועדו לבחון את הדיוקים הספציפים עבור כל סיווג.

לשם כך הוספנו מדד f1_score עבור כל סיווג באופן ספציפי אליו ונפרד לשאר הסיווגים. נשאף לכך שהממוצע של הדיוק עבור כל קלאס ביחד יהיה הגבוה ביותר. כמו כן הוספנו גם את הconfusion matrix אשר מראה לנו את FP ואת הFN בצורה רחבה יותר.

בחלק זה של הניסויים השתמשנו במודל ה IMBLEARN אשר כולל מימושים שונים לטיפול בדאטה לא מאוזן.

ניסוי 1:

המטרה שלו היא לחפש את הפרמטרים הטובים ביותר בדרך הראשונה, כלומר בשיטת המשקל. בחרנו לבחון עם המסווג ExtraTreeClassifier(n_estimators=10) מכיוון שראינו כבר מניסויים קודמים כי זהו המסווג שמביא אותנו לתוצאות הטובות ביותר עבור הדאטה כאשר הוא לא מאוזן ולכן רצינו להסתמך עליו. ניסינו למצוא את המשקלים עבור התוצאות האופטימליות.

נשים לב שחסרים לנו class7 וכן class8 מכיוון שעבור קבוצות הtrain,test שהתקבלו הם אינן מופיעים שם וזה הגיוני מפאת הנדירות שלהם בדאטה.

- המשקלים הראשונים שנוסו:

Class 0: 1
Class 1:10
Class2: 100
Class3: 1000
Class4: 10000
Class5: 100000
Class6: 1000000

הclassים הנ"ל מציינים את מספר התאונות שהיו בצומת מסוים בתאריך כלשהו.

התוצאות:

class	0	1	2	3	4	5	6
weight	1	10	100	1000	10000	100000	1000000
result	0.9605	0.5713	0.5457	0.5231	0.4002	0.4886	0.2426

F1_score=0.9238

Matthews=0.5815

Error=0.0553

- המשקלים השניים שנוסו:
  Class 0: 0.1
  Class 1:10
  Class2: 100
  Class3: 1000
  Class4: 10000
  Class5: 100000
  Class6: 1000000

class	0	1	2	3	4	5	6
weight	0.1	10	100	1000	10000	100000	1000000
result	0.906	0.5731	0.5508	0.5365	0.4071	0.3497	0.3181

F1 = 0.924

Matthews=0.5835

Error=0.0551

- המשקלים השלישיים שנוסו:
  Class 0: 0.01
  Class 1:10
  Class2: 100
  Class3: 1000
  Class4: 10000
  Class5: 100000

Class6: 1000000

class	0	1	2	3	4	5	6
weight	0.01	10	100	1000	10000	100000	1000000
result	0.9604	0.5691	0.5391	0.5074	0.4166	0.4477	0.2467

F1-score=0.9235

Matthews=0.5784

Error=0.0548

האופציה הרביעית היא על ידי הוספת הפרמטר "balanced" הנותן למערכת עצמה לקבוע את המשקלים על פי גודל הקבוצות

class	0	1	2	3	4	5	6
result	0.9611	0.579	0.5571	0.5499	0.3994	0.4099	0.3073

F1-0.9248

Matthews=0.0534

Error=0.0534

ניתן להבחין כי אין השפעה רבה מאוד בשינוי המשקלים על התוצאות וכן לבסוף התוצאה של הbalanced יצאה המדויקת ביותר ולכן נמשיך איתה.

<u>ניסוי 2:</u>

השוואה בין שלושת הדרכים: under-samples , הקטנת class 0 בשליש, והמשקלים הנבחרים מניסוי 1 תחת מסווג Extra TreeClassifier.

<u>under-samples</u> - אלגוריתם זה משווה את גודל כל קבוצות הסיווגים לקבוצת סיווג הקטנה ביותר ע"י מחיקת רנדומלית של דוגמאות מהסיווגים הגדולים.

נבחן את האלגוריתם עם 3 המסווגים הבאים:
ExtraTresesClasiifier,decisionTree,KneighborsClassifer,GaussianNB

ExtraTresesClasiifier

F1=0.4693

class0 =	class1 =	class2 =	class3 =	class4 =	class5 =	class6 =
0.6441	0.1438	0.0399	0.0131	0.0055	0.0012	0.0016

decisionTreeClassifier

f1= 0.327

class0 =	class1 =	class2 =	class3 =	class4 =	class5 =	class6 =
0.4974	0.1062	0.0275	0.013	0.0025	0.0003	0.0009

KneighborsClassifier

F1=0.4559

class6 = 0.0006	class5 = 0.0003	class4 = 0.0044	class3 = 0.0132	class2 = 0.0346	class1 = 0.1444	class0 = 0.6341

GassuianNB

f1 = 0.5487

class6 = 0.0005	class5 = 0.0	class4 = 0.0027	class3 = 0.0111	class2 = 0.0128	class1 = 0.0909	class0 = 0.7237

<u>הקטנת class 0 בשליש-</u>

ExtraTreesClassifier

F1=0.9168

class6 = 0.3432	class5 = 0.4176	class4 = 0.4351	class3 = 0.5668	class2 = 0.5531	class1 = 0.5621	class0 = 0.9566

GaussianNB

F1=0.5285

class6 = 0.0005	class5 = 0.0	class4 = 0.0037	class3 = 0.01	class2 = 0.0142	class1 = 0.1554	class0 = 0.6984

KneighborsClassifier

F1=0.8934

	class5 = 0.351	class4 = 0.3269	class3 = 0.3689	class2 = 0.3946	class1 = 0.4414	class0 = 0.9441

<u>ממושקל –"balanced"</u>

ExtraTreesClassifier

F1=0.9248

class6 = 0.3073	class5 = 0.4099	class4 = 0.3994	class3 = 0.5499	class2 = 0.5571	class1 = 0.579	class0 = 0.9611

המסקנה מהניסוי היא שהשיטה בעלת הביצועים הטובים ביותר מבין שלושת השיטות שנבדקו היא שיטת משקול כובד הטעות למסווג.

ניסוי 3:

בניסוי זה נבחר איזה מסווג אופטימלי עבור הסיווג של מספר התאונות.

הגענו עד עכשיו למסקנות הבאות:

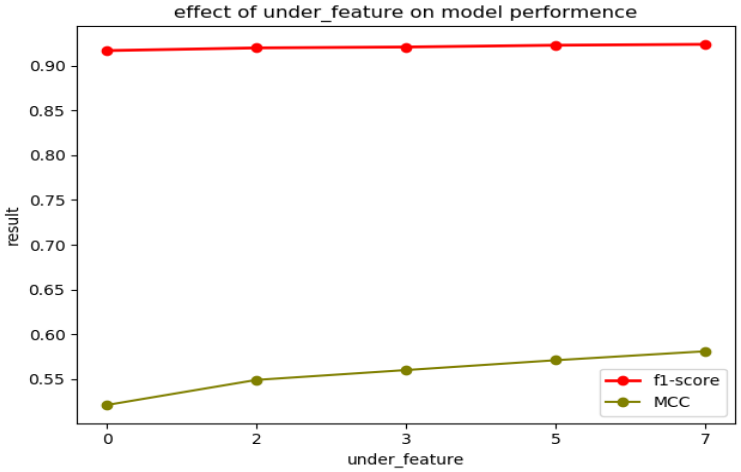
מתוך שלושת הדרכים של איזון הדאטה, הדרך שבחרנו בה היא המשקלים הנותנים "עונש" גדול יותר לטעות בקבוצות הסיווגים הקטנות יותר.

כמו כן ראינו כי המסווג שהשיג עבורנו את התוצאות הטובות ביותר היה ExtraTreesClassifier ולכן נשתמש בו ונמשיך להתאים כל מיני פרמטרים שונים שראינו שמשפיעים לטובה על הסיווג. בניסוי זה נשים דגש על ה confusion matrix וכן על MCC אשר מדרגים באופן מדוייק יותר דאטה לא מאוזן.

ראשית נציג את ביצועי המסווג ללא שינוי נוסף עם n_estimators=10:

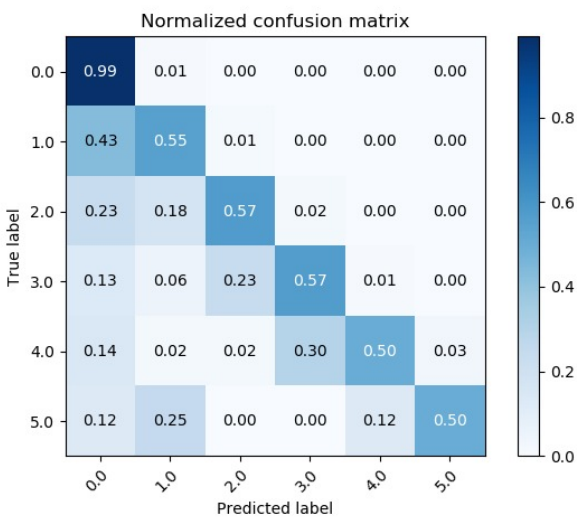
Error= 0.06 , f1_score = 0.917 , MCC=0.521.

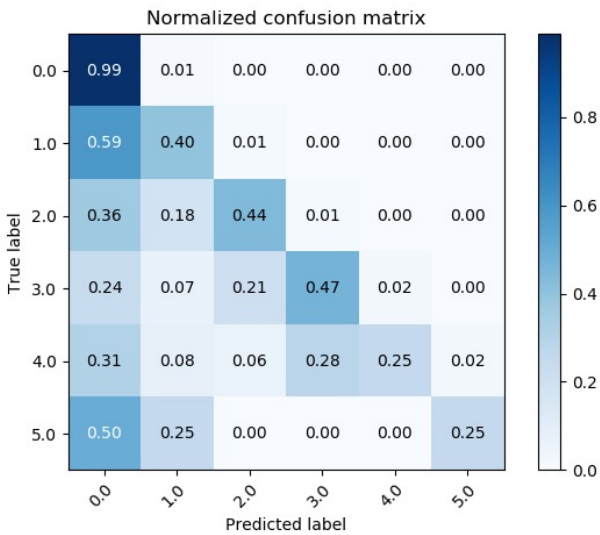
כעת נתחיל מהנדסת הפיצ'רים, כפי שראינו בחלק הרלוונטי בדוח השגנו שיפור כאשר הורדנו את 2 התכונות המדורגות הכי נמוך ולכן נמשיך כך, עם דירוג התכונות כל עוד אנו משיגים שיפור, נתאר את התוצאות באופן גרפי:



נשים לב שגם עם הורדת 7 תכונות המדורגות הכי נמוך הגענו לשיפור ולכן מעתה נשתמש רק בתכונות לאחר הורדת ה7 התכונות בעלות הניקוד הנמוך יותר.

נציג את הconfusion matrix:בימין ללא הסרת התכונות ובשמאל עם ההסרה:





ואכן ניתן לראות שהסרת 7 התכונות גרמה לclass להיות מדויקים יותר ולכך שיהיו פחות FP FN.

התוצאות שהתקבלו לאחר הסרת 7 התכונות הן : error=0.049 , f1=0.93, Matthews=0.62

כעת נוסיף את המשקול לקבוצות על ידי הפרמטר "balanced"

הגענו לתוצאות הבאות: error: 0.046 , f1_calc=0.932, Matthews = 0.638

כעת נוסיף n_estimators לextratrees מ10 ל200:

error: 0.04,f1_score=0.936,Matthews=0.66

בנוסף ניסינו להשתמש בספריית imblearn אשר בה נמצאים כל מיני טכניקות לאיזון דאטה כמו SMOTE שמבצעת הוספת דוגמאות באופן סינטטי ע"י הדוגמאות הקיימות וכן NearMiss שמנפה מהקבוצות הגדולות את הדוגמאות שממש קרובות לדוגמא קיימת אך הדבר לא צלח ולא קיבלנו תוצאות טובות יותר.

לסיכום המסווג שלנו יהיה

ExtraTressClassifier(n_estimatros=15,random_state=0,class_weight="balanced)

נראה כעת תוצאות עבור המסווג הסופי בהשוואה לבחירה רנדומלית של סיווגים.

את המסווג נפעיל על train שכולל את כל השנים מ2008 עד 2016

והtest יהיה בנוי מנתוני שנת 2017. ולהלן התוצאות:

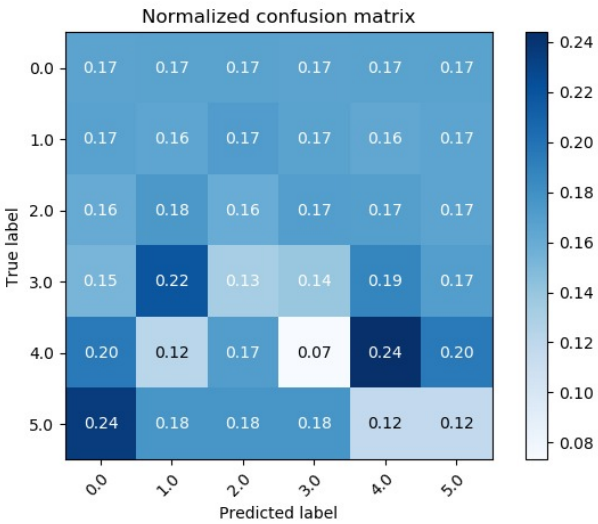
עבור y_pred שהמסווג שלנו הוציא קיבלנו כי:

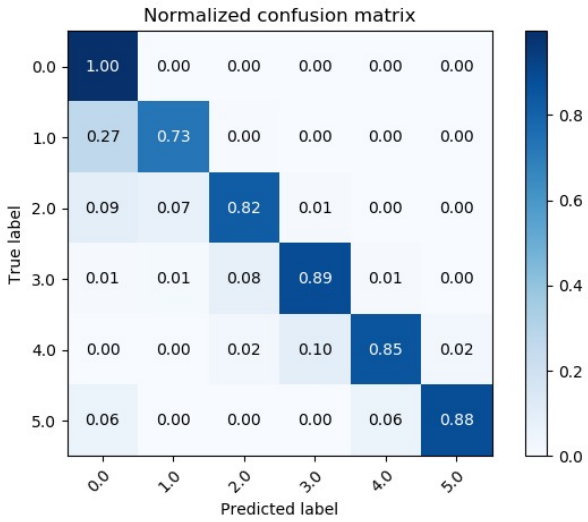
F1_score=0.9893, MCC(Matthews)=0.801

עבור הy_random כלומר בחירה רנדומלית של סיווג- קיבלנו:

F1_score=0, MCC(Matthews)=0.166

ולהלן הconfusion matrix:





נכון שראינו כי התוצאות עבור n_estimators טובות יותר ככל שהn_estimators גדול יותר אבל כאשר אנו מאמנים את המסווג על כל הדאטה מפאת הגבלת זיכרון חלק מהמחשבים נעצרו בn_estimators=10 חלק ב15 וחלק ב25 ולכן לקחת את הממוצע עבור מחשב סטנדרטי.

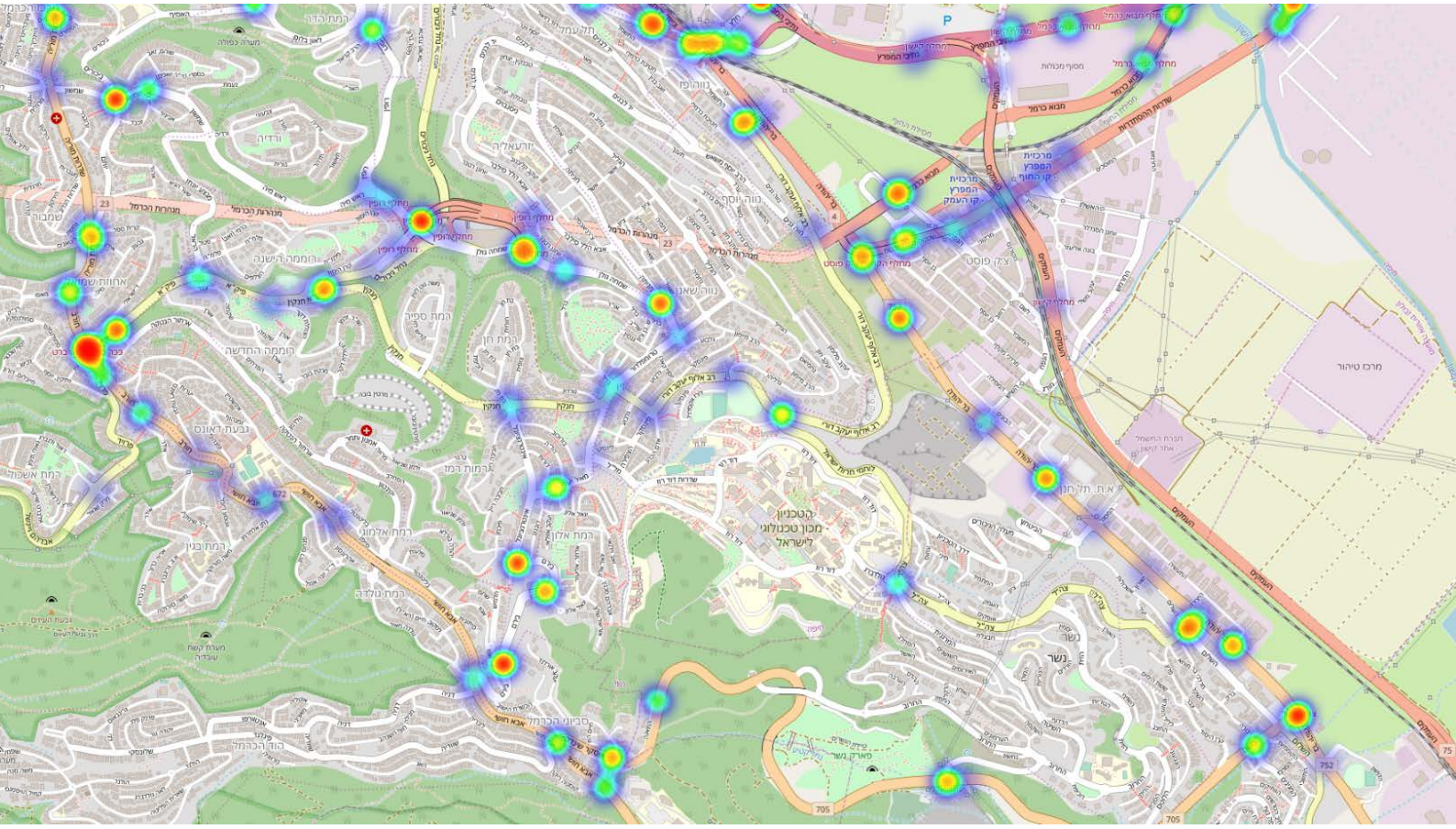
במידה ויעמוד ברשותנו מחשב חזק יותר וחדש יותר ניתן להעלות את n_estimators ובכך גם לשפר את הדיוק.

<u>תוספת–</u>

מפת חום:

החלטנו להוסיף מפת חום אשר תקבל את הפלט של סיווג הצמתים ותציג אותם על מפת ישראל כעיגולי חום על פי דרגת המסוכנות של הצומת. הscript עבור מפת החום נקרא heatmap והוא מקבל קובץ של נקודות ציון ומייצר קובץ html המייצג את מפת החום.

להלן דוגמא:



<u>סיכום הפרויקט ורעיונות המשך</u>

המטרה המרכזית שלנו בפרויקט הייתה לבחון אלגוריתמי למידה שונים וכיצד הם מתמודדים עם בעיית סיווג צומת , במטרה לבנות מסווג המסוגל לבצע דירוג צמתים ע"פ רמת מסוכנות שלהם עבור אותו היום.

במהלך הפרויקט ניסינו להתמודד עם הבעיה במספר דרכים הכוללות התאמת ההגדרה של צומת מסוכן ע"פ צורך, ניסוי של מסווגים שונים, ועבודה עם דאטה לא מאוזן לעומת מאוזן בצורה מלאכותית.

בפרויקט, במטרה להגיע לתוצאה האופטימלית, יצרנו מאגר נתונים המתבסס על צמתים בישראל במרחק מינימלי אחד מהשני, אליו הוספנו מידע שאספנו בדקדקנות והתאמנו לאותו מאגר.

רוב האלגוריתמים בתחום הינם חמדניים- הם בעיקר מחפשים היפוטזה טובה במהירות. היה חשוב לנו בפרויקט לחפש היפוטזה אופטימלית מאשר להקדיש לזמן הריצה חשיבות גדולה. במילים אחרות, רצינו לשים את הדגש על חיזוי נכון ותוצאות מדויקות מאשר על זמן למידת המסווג. אין לנו צורך בניבוי מהיר של מאית השניה כמו למשל במכוניות אוטונומיות, אלא יותר חשוב לנו לדייק בניבוי כדי להצליח להציל כמה שיותר חיי אדם וחיסכון למשק.

<u>כיוונים למחקר עתידי ושיפורים אופציונליים:</u>

1. מציאת הרדיוס האופטימלי: בפרויקט זה עקב מגבלות משאבים וזמן בחרנו לעבוד עם רדיוס של 10 ק"מ. כלומר, עבור כל תכונה מבין התכונות: ברים ופאבים, בתי ספר, בעלי כלבים, אבטלה בישובים, מסעדות, חניונים ועוד, הנתונים במאגר שבנינו הם בהתאם לרדיוס הנ"ל. לדעתנו, אם ניתן לבצע ניסויים נוספים עבור רדיוסים משתנים, נוכל לקבל תוצאות טובות יותר ובכך לממש את חזון הפרויקט בצורה אופטימלית יותר.
2. הוספת ממשק נגיש יותר למשתמש: הפלט הסופי של הפרויקט הינו דו"ח של מספר הצמתים הכי מסוכנים בהתאם למספר הניידות המוקצות מבחינת משאבי המשטרה. בנוסף, במצב הנתון כרגע, על המשטרה להכניס כקלט בכל הרצה את נתוני מזג האוויר הצפויים לאותו היום מהאינטרנט בצורה עצמאית. לראייה עתידית, עבור נוחות מקסימלית של המשתמש, ניתן לבנות ממשק נגיש יותר, היודע לגשת לבד לאתר השירות המטאורולוגי ולוקח משם את הנתונים הנדרשים עבור המסווג.
3. בחינה של שאר הסיווגים הנוספים ובדיקה האם קיימים מסווגים טובים יותר עבורם.
4. אוטומטיזציה של כל התהליך כך שבלחיצת כפתור כל הדאטה ייאסף מהמקומות הרלוונטיים ולא יהיה צורך בעריכה ידנית של המידע.