הפקולטה להנדסה ע"ש איבי ואלדר פליישמן
אוניברסיטת תל אביב

**מדעים דיגיטליים להייטק**

**פרויקט מס' 22-1-239**

# דו"ח סיום

שם הפרויקט : פיתוח מערכת לזיהוי אנומליות

ברשתות תקשורת

**מבצעים:**

שם: דניאל וולקוביץ ת.ז. 207257668
שם: שירז ישראלי ת.ז. 209126259

מקום ביצוע הפרויקט: אוניברסיטת תל אביב – עבודה מהבית

**מנחים :**
דרור יעקובי ודותן דקל

# תוכן עניינים


תקציר ..... 3

מוטיבציה ..... 5

תכולת עבודה ושינויים בדרישות המערכת ..... 6

לוח זמנים ..... 7

ספר פרויקט – תיאור ..... 8

איסוף הנתונים ..... 10

שלב ניתוח הנתונים ..... 16

מודלים ..... 22

ביצועים ..... 25

הערכת המודל ..... 30

שיפורים להמשך ..... 33

מדריך שימוש ..... 34

קישורים ..... 36

# תקציר


הפרויקט מתמקד בפיתוח ויישום של מערכות אוטומטית לזיהוי אנומליות ברשתות תקשורת. המערכת מבוססת על טכנולוגיות חדישות ושימוש בנתונים היסטוריים על מנת לאתר ולסווג תקלות והתרחשויות לא רגילות ('אנומליות') כמו עומס של broadcast packets אשר נשלחות ברשת ועלולות להעמיס עליה באופן משמעותי.

המערכת שפותחה במסגרת הפרויקט מאפשרת לגורמים המתעניינים לזהות באופן אוטומטי ומהיר תקלות ואנומליות ברשתות התקשורת. השלב הראשוני הוא בניית התשתית לקבלה, אחסון, ניתור של הנתונים במבני נתונים שמאפשרת בהמשך הפעלת אלגוריתמים לזיהוי אנומליות שונות ברשת.

בשלב השני באמצעות מסד הנתונים – הפעלת אלגוריתמים לזיהוי התקלות, הגדרת תקלות כאשר נתמקד בקבלת נתוני broadcast מהממשקים.

המערכת עשויה להיות שימושית לארגונים וחברות המבצעות ניהול רשתות תקשורת, ולסייע במניעת תקלות והפרות אבטחה על ידי איתורן בשלב מוקדם יותר.

נעשה זאת באמצעות:

(1) איסוף נתונים רלוונטיים מהמתגים וממשקי התקשורת.

(2) ניתוח המאפיינים ברשת.

(3) פיתוח מערכת לזיהוי האנומליה הספציפית בעזרת אלגוריתם Unsupervised ML

הפרויקט כולל שלושה שלבים עיקריים:

שלב א' : איסוף והקמת ממשק לאיסוף נתונים:

הכנת Data-set הכולל איסוף וארגון נתונים ומאפיינים רלוונטים מממשקים ומתגי תקשורת, כולל מבנה המאפשר חיפוש לפי בניין או מתג. נעשה באמצעות ANSIBLE וסקריפט בשפת Python . ארגון המידע שנצבר הכולל תרחישים וחריגות בבסיס נתונים לצורך ניתוח, הכשרה ובדיקה מקיפה לפיתוח מערכת זיהוי האנומליות. למידה עצמית ומחקר של מאפיינים של אירועי האנומליות.

כלים: Ansible (Playbook) , Python

שלב ב' : ניתוח נתונים והפעלת אלגוריתם ללמידת מכונה על הדאטה הנאסף

בהתבסס על המידע שנאסף בתקופה של כ-3 חודשים, ביצוע ניתוח נתונים באמצעות שפת Python למידה על פיצ'ר ה BROADCAST , באמצעות מס' מודלים מבוססים למידת מכונה, ופיתוח סקריפטים, ניסינו לקבוע היתכנות או שלילת קיום אירוע האנומליה.

השלב השני כולל פיתוח אלגוריתמים וניתוח תוצאות על ידי שיטות סטטיסטיות.

כלים: Python,Unsupervised ML/ Data Science Packages.

**כלים, שפות וסביבות עבודה :**

Python : שפת תכנות מגוונת וידידותית הידועה בפשטות ובקריאות שלה. Python נמצאת בשימוש נרחב בתחומים שונים, מפיתוח אתרים ועד לניתוח נתונים ובינה מלאכותית. Python מדגישה את קריאת הקוד ומציעה מגוון עצום של ספריות ומסגרות שהופכות אותו לעוצמתי עבור יישומים שונים.

Ansible : כלי אוטומציה יעיל עבור סביבות עבודה של תקשורת רשת. מייעל את ניהול התצורה, ממכן את הפריסה ומקנה קנה מידה יעיל עבור רשתות גדולות, באמצעות ה – Playbook המודולריים שלו. יכולות האינטגרציה שלו עם התקני רשת הופכות אותו למגוון ובעל דיווח מפורט, מה שמאפשר לאנשי מקצוע ברשת לנהל ולתחזק ביעילות תשתית רשת בדיוק ואמינות.

סביבת עבודה pycharm : היא סביבת פיתוח משולבת (IDE) שתוכננה במיוחד עבור תכנות בשפת Python. הוא מספק סט מקיף של כלים ותכונות לייעול תהליך הפיתוח, כולל השלמת קוד, איתור באגים, בדיקות וניהול פרויקטים.

Vscode : הוא עורך קוד שניתן להתאמה אישית אשר פותח על ידי מיקרוסופט. למרות שאינו ספציפי לשפה, הוא תומך במגוון רחב של שפות תכנות. הוא מציע תכונות כמו ניווט קוד, הדגשת תחביר, איתור באגים והרחבות, המאפשרות למפתחים להתאים את העורך לצרכים הספציפיים שלהם.

## מוטיבציה

המוטיבציה לביצוע פרויקט זה מבוססת על הצורך ההנדסי והממשי לזיהוי וטיפול באנומליות ברשתות תקשורת. רשתות תקשורת משמשות ארגונים רבים וחשובים בתעשייה באופן יומיומי, ולכן חשיבותן ותרומתן של טכנולוגיות שיכולות לזהות ולטפל בתקלות רשת ובאנומליות ברורה. ישנם סיכונים רבים פוטנציאליים בתקלות והתקפות רשת, לכן יש חשיבות רבה לזיהוי וסיווג בזמן ולניהול רשת יעיל.

כיום, אין כלים מפותחים לזיהוי אוטומטי וסיווג סוגי התקלות שעלולות להופיע ברשת ודרך מסודרת לסווגן. חשיבות הפרויקט טמונה בכך שהוא מאפשר זיהוי מהיר ואוטומטי של אנומליות, תקלות והפרות אבטחה ברשתות התקשורת. כך ניתן למנוע נזקים ובעיות בתפקוד הרשת ולשפר את יעילותה ובטיחותה.

כאחת מהאפשרויות למימוש הפרויקט, ניתן לשקול שימוש בכלים קיימים לניהול רשת ואבטחת מידע, אשר כוללים מערכות זיהוי חתימות ידועות של התקפות ואנומליות. אף על פי שכלים אלו מספקים פתרון מסוים, הם אינם מתבצעים על ידי מערכות לניהול רשת באופן אוטומטי, ועשויים להיות פחות מתקדמים מבחינת יכולתם לזהות אנומליות חדשות או לזהות את מקור הבעיה. בהתאם לדאטה, נכוון גישת פתרון המבוססת על מספר טכניקות ML (למידת מכונה) כגון אלגוריתמים של RL, ושימוש ברשתות NN. שיטות אלו עשויות להציע יכולת זיהוי אנומליות מתקדמת יותר בהשוואה לכלים הקיימים, אך עשויות לדרוש משאבים חישוביים גדולים יותר והשקעה משמעותית בפיתוח ויישום המודלים.

הפרויקט משלב את הטכנולוגיות המתפתחות ושימוש בכלים של AI לפיתוח ממשק אינטואיטיבי, ייעודי ואוטומטי לזיהוי וטיפול באנומליות ברשת. המערכת תעשה שימוש בטכנולוגיות למידת מכונה וכלים סטטיסטיים על מנת להגיע ליעילות העולה על הכלים הקיימים, תשפר את יכולת הארגון לזהות באופן אוטומטי ומהיר אנומליות ברשת, לזהות פתרונות פוטנציאליים, ולשפר את בטיחות ויעילות הרשת.

## תכולת עבודה

- סקירת הספרות להבנת השיטות הקיימות. מחקר רקע תיאורטי הדרוש להבנה ופיתוח הפרויקט. קריאת מאמרים ותיאורים בנושא של רשתות תקשורת וזיהוי אנומליות ברשת.
- השלמת הידע הנדרש בלמידת מכונה לצורך המודלים המבוססים דאטה .UNSUPERVISED
- למידה ושימוש ב ansible וב playbook , ממשק העבודה שממנו המידע ייאסף.
- ארגון ואיסוף הנתונים ממתגי התקשורת ומהממשקים, בצורה הנוחה לניתוח.
- הכנת סקריפט לשליפת נתונים בצורה איטרטיבית מהמתגים לפי תאריך ושעה מדוייקים, מס' המתג, שם הממשק ומאפיינים נוספים לכל ממשק, שיעזרו בהמשך לניתוח האנומליה.
- הכנת סקריפט לביצוע חיפוש של description עבור כל ממשק , יחס עם שם מתג ששייך לו.
- ניתוח הדאטה הנאסף בצורה ויזואלית וסטטיסטית ע"י כלים תיאורטיים.
- פיתוח אלגוריתם לזיהוי אנומליות – בדיקת אלגוריתמים מתאימים לניתוח האנומליה בצורה האופטימלית, בדיקת ביצועים ומדדים.

## שינויים בדרישות המערכת

השינוי העיקרי בדרישת המערכת היה בעקבות הצורך בבניית דאטה - סט מספיק יציב, על מנת שנוכל לנתח בצורה הטובה ביותר את האנומליות ברשת והתמקדות באנומליה מסוג אחד.
מאחר ולא היו נתונים קודמים לפרויקט, ולא היה ממשק קיים, הוחלט להתמקד בבניית הממשק, ובבניית מסד הנתונים, שיהיה מספיק לעיבוד ולמידה מהנתונים. לכן בבחינת האנומליות ברשת, בחרנו להתמקד באנומליית ה BROADCAST ולבצע עליה את הלמידה, ניתוח וניסיון לסווג אנומליה.
מאחר והדאטה שלנו הוא UNSUPERVISED ולא ידוע לנו מראש מה מסווג כאנומליה ומה לא, חשובה הלמידה על הנתונים שנאספו, בעזרתם ניתן לפתח יכולת, ע"י ניתוח בצורה ויזואלית וסטטיסטית, ולפתח את האלגוריתם הרצוי, וכן באמצעות התבססות על הנתונים שנאספו ניתן להעריך את המודל הנבחר.

## לוח זמנים

| אבן דרך | תאריך יעד | תאריך ביצוע בפועל | הערות |
|---|---|---|---|
| שלב מקדים- למידת החומר הנדרש והכנת תוכנית עבודה מסודרת | 30.04 | 30.04 | |
| הקמת ממשק לאיסוף הדאטה. הכנת סקריפט לאיסוף מידע רלוונטי בצורה איטרטיבית ושמירת הנתונים בדאטה בייס. | 30.05 | 18.06 | הקוד ירוץ בצורה איטרטיבית ויאסוף דאטה לבניית מסד הנתונים לעיבוד. |
| Pre Processing – ניתוח הנתונים | 30.05 | 12.09 | יצירת מסד הנתונים בתקופה של הפסקה לתקופת מבחנים של סמסטר ב' |
| מחקר על מודלים הנדרשים לבעיה | 10.06 | 12.09 | |
| הגשת דו"ח מעקב | 1.7 | 12.09 | |
| הפעלת האלגוריתם | 30.06 | 12.09 | |
| ניתוח ובדיקת תוצאות | 10.08 | 20.09 | |
| הנגשת הכלים והממשק ל הפרויקט לעבודה עתידית | 10.10 | 10.10 | |
| הגשת הפוסטר | 10.10 | 10.10 | |
| סיום הפרויקט | 10.10 | 10.10 | נדחה עקב המצב הביטחוני |

# ספר פרויקט

## תיאור

הפרויקט שלנו מתמקד בפיתוח והטמעה של מערכת אוטומטית לזיהוי אנומליות עבור רשתות תקשורת.

מערכת זו משתמשת בנתונים היסטוריים כדי לזהות ולסווג תקלות והתרחשויות חריגות, המכונות 'אנומליות'. חריגות אלו יכולות לכלול מצבים כמו עומסי Broadcast packets קיצוניים שעלולים להשפיע באופן משמעותי על ביצועי הרשת.

המטרה העיקרית של פרויקט זה הייתה ליצור תשתית המסוגלת לקבל, לאחסן ולנטר נתוני רשת ביעילות. תשתית זו תספק את הבסיס לביצוע אלגוריתמים שנועדו לזהות חריגות רשת שונות. אלגוריתמים אלו יסייעו לבעלי עניין לזהות במהירות ובאופן אוטומטי תקלות וחריגות ברשתות תקשורת, ויסייעו לארגונים ולחברות לנהל את הרשתות שלהם בצורה יעילה יותר ובאופן יזום למנוע תקלות ופריצות אבטחה.

באמצעות איסוף נתונים והגדרת ממשק ולאחר מכן ניתוח הנתונים שנאספו והפעלת אלגוריתמים מתאימים, בפרויקט שלנו יצרנו מערכת אפקטיבית ואוטומטית לזיהוי חריגות שיכולה להועיל לארגונים וחברות המנהלות רשתות תקשורת. על ידי איסוף וניתוח נתוני רשת והטמעת טכניקות למידת מכונה מתקדמות, אנו מספקים כלי רב ערך לזיהוי מוקדם של תקלות והפרות אבטחה, התורם לאמינות הכללית והאבטחה של רשתות התקשורת.

**<u>אנומליית Broadcast ברשתות תקשורת :</u>**

זיהוי אנומליות הוא בעצם שלב איתור התצפיות שאינן תואמות ליתר התצפיות בבסיס הנתונים.
לרוב, התצפיות החריגות מעידות על בעיה.

"Broadcast " מתייחס לסוג של שידור נתונים שבו חבילת מידע בודדת נשלחת מהתקן רשת אחד ומיועדת להתקבל על ידי כל המכשירים באותו קטע רשת או תחום שידור.
Broadcast הוא חסר הבחנה ונשלח לכל המכשירים בתוך מקטע הרשת המיועד לו.

השפעת Broadcast קיצוני :

לתנועה קיצונית של Broadcast בתוך רשת יכולות להיות מספר השפעות שליליות:

1. עומס ברשת: כאשר מספר רב של חבילות שידור נשלחות בתוך תקופה קצרה, זה עלול להוביל לעומס ברשת. עומס זה יכול להאט את הרשת כולה, ולהשפיע על הביצועים של שירותי רשת ויישומים קריטיים אחרים.

2. ניצול משאבים: מכשירים בתוך תחום השידור צריכים לעבד ולבחון כל חבילת Broadcast כדי לקבוע אם היא רלוונטית עבורם. זה צורך כוח עיבוד ועלול להוביל למיצוי משאבים במכשירים, במיוחד במקרים של תעבורת Broadcast גבוהה במיוחד.

3. חששות אבטחה: תעבורת Broadcast עלולה לשאת תוכן זדוני, ואם לא נשלטת כראוי, עלול לגרום להתקפות רשת או פרצות אבטחה.

4. חוסר יעילות בניצול הרשת - עלול להפריע ליכולתה של הרשת לשאת נתונים ביעילות.

*כדי לטפל בבעיות אלה, עבור מנהלי רשת אשר שואפים לנטר ולנהל את תעבורת Broadcast ביעילות, מערכת זיהוי אנומליות, כמו זו בפרויקט שלנו, מקיימות תפקיד חיוני בזיהוי תנועות חריגות בתעבורת Broadcast ובשלב הבא תוכל להתריע בפני מנהלי רשת לחקור ולצמצם בעיות פוטנציאליות.
כמו כן, מערכת זו תסייע לשמור על הביצועים והאבטחה של רשתות תקשורת.

# איסוף הנתונים

**איסוף הדאטה והקמת הממשק לפרויקט :**

בשלב ראשוני זה התמקדנו באיסוף וארגון של נתונים ומאפייני רשת רלוונטיים מממשקי תקשורת ומתגים. זה כולל יצירת מערך נתונים מקיף, הכולל נתונים ממרכיבי רשת שונים.

כדי להשיג זאת, השתמשנו בPlaybook של Ansible ובסקריפטים של Python כדי ליצור אינטראקציה עם מתגי רשת, לאסוף נתונים ולאחסן אותם במסד נתונים מרכזי.

שלב זה כולל גם יצירת שאילתה לשליפת נתונים בצורה מהירה מהממשקים ומהמתגים.

בנוסף, שלב זה כולל למידה עצמית ומחקר מעמיק להבנת המאפיינים של אירועי אנומליה ברשת תקשורת.

חשוב לציין ששלב איסוף הדאטה בצורה טובה ועל פני זמן הוא חיוני, מאחר והדאטה שלנו לא מסווג, כלומר בנתונים הנאספים לא נאמר עבור כל נתון האם הוא חריג או לא, אלא האלגוריתם ילמד מהנתונים הנאספים כיצד לסווג אנומליה ברשת, על בסיס נתונים היסטוריים שאספנו.

לכן, על מנת להגיע למסקנות הנדרשות, חשוב לעבוד עם מידע רב.

- **שליפת נתונים בצורה איטרטיבית מהמתגים לדו"ח שאותו נעבד בהמשך :**

הנתונים הנכללים:

מידע עבור כל ממשק משני מתגי תקשורת מרשת האוניברסיטה שעבדנו עליהם ( בפרויקט עתידי ניתן לצרף עוד מתגים לעיבוד )

כל שורה בדו"ח כוללת את המאפיינים הבאים :

1. Date and time
2. Swich number
3. Interface name
4. Output_queue(size/max)
5. Packets_input
6. Bytes
7. Buffer
8. Broadcast
9. Multicast

**הקוד: נכתב בשפת פייתון באנסיבל : PY CODES : run and save playbook.py**

```
  GNU nano 6.2                                                    /etc/ansible/py_codes/run_and_save_playbook.py
import pandas as pd
import re
import subprocess
import os
import time

def parse_report_data(report_data, df):
    report_data = report_data.splitlines()
    date = report_data[3].strip().strip(",").strip('"')
    # Iterate over the lines and extract the relevant information
    for line in report_data:
        line = line.strip().strip('"').strip()  # Remove leading/trailing whitespace and "

        if line.startswith("ok: [switch-"):
            switch_pattern = r'(\d+)'
            switch = re.search(switch_pattern, line, re.MULTILINE).group(0)

        if line.startswith(('Vlan', 'FastEthernet', 'TenGigabitEthernet', 'GigabitEthernet')):
            if " is down," in line:
                flag = False
            else:
                flag = True
            interface = line.split()[0]

        elif line.startswith('Output queue:'):
            regex = r'Output queue: (\d+/\d+)'
            output_queue = re.search(regex, line).group(1)

        elif "packets input" in line:
            packets_input = line.split()[0]

            regex = r'(\d+) bytes'
            num_bytes = re.search(regex, line).group(1)

            regex = r'(\d+)\s+no buffer'
            buffer = re.search(regex, line).group(1)

        elif line.startswith("Received "):
            regex = r'Received (\d+) broadcasts'
            broadcast = re.search(regex, line).group(1)

            regex = r'Received \d+ broadcasts \((\d+) (?:IP )?multicasts\)'
            multicast = re.search(regex, line).group(1)
```

```
            if flag:
                record = {"date": date, "switch_number": switch, "interface_name": interface,
                          "output_queue(size/max)": output_queue, "packets_input": packets_input, "bytes": num_bytes,
                          "buffer": buffer, "broadcast": broadcast, "multicast": multicast}

                # Convert the record to a DataFrame and concatenate with the existing DataFrame
                record_df = pd.DataFrame(record, index=[0])
                df = pd.concat([df, record_df], ignore_index=True)

    return df

def create_or_append_to_csv(csv_path, report_df):
    if not report_df.empty:
        # Check if the CSV file exists
        if os.path.isfile(csv_path):
            # Append the DataFrame to the existing CSV file
            report_df.to_csv(csv_path, mode='a', index=False, header=False)
            print(f"Data appended to CSV file: {csv_path}")
        else:
            # Create a new CSV file and write the DataFrame
            report_df.to_csv(csv_path, index=False)
            print(f"CSV file created: {csv_path}")
    else:
        print("No report data found.")

def run_playbook_and_process_output(csv_path):
    # Column names for the DataFrame
    column_names = ["date", "switch_number", "interface_name", "output_queue(size/max)", "packets_input", "bytes", "buffer",
                    "broadcast", "multicast"]

    # Change the working directory to /etc/ansible/
    os.chdir('/etc/ansible/')

    # Run the Ansible playbook and capture the output
    playbook_path = "playbooks/collect_interfaces2.yml"
    playbook_command = ["ansible-playbook", "-i", "inventory", playbook_path]

    # Start the playbook execution and capture the output
    playbook_process = subprocess.Popen(playbook_command, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    playbook_output, _ = playbook_process.communicate()
```

המשך קוד:

```
    # Wait for the playbook process to finish
    while True:
        if "PLAY RECAP" in playbook_output.decode():
            break

        # Sleep for a short duration before checking again
        time.sleep(1)

    # Check if the playbook execution was successful
    if playbook_process.returncode == 0:
        print("Playbook execution completed.")
    else:
        raise RuntimeError("Playbook execution failed.")

    # Check if "PLAY RECAP" is present in the output
    if "PLAY RECAP" in playbook_output.decode():
        # Extract the report data from the playbook output
        report_start_index = playbook_output.decode().find('"report_data": ')
        if report_start_index != -1:
            report_start_index += len('"report_data": ') - 40
            report_end_index = playbook_output.decode().find('PLAY RECAP') - 4
            report_data_str = playbook_output.decode()[report_start_index:report_end_index]

            # Process the report data and obtain the DataFrame
            report_df = pd.DataFrame(columns=column_names)
            report_df = parse_report_data(report_data_str, report_df)

            # Create or append to the CSV file
            create_or_append_to_csv(csv_path, report_df)
        else:
            print("Report data not found in playbook output.")
    else:
        print("No 'PLAY RECAP' found in playbook output.")

# Path to the existing CSV file
csv_path = "/etc/ansible/DB_cisco_ios_logs/interfaces_full_report.csv"

# Run the playbook and process the output
run_playbook_and_process_output(csv_path)
```

הסקריפט הנ"ל מקושר ל Playbook : Collect interafces2.yml ומסד הנתונים נשמר בקובץ CSV

שנמצא בנתיב : /etc/ansible/DB_cisco_ios_logs/interfaces_full_report.csv

הפלייבוק ליצירת הדו"ח :

```
GNU nano 6.2                                    /etc/ansible/playbooks/collect_interfaces2.yml
- name: Generate Interface Report
  hosts: all
  gather_facts: false
  vars:
    ansible_python_interpreter: /usr/bin/python3

  tasks:
    - name: Execute show interfaces command
      ios_command:
        commands:
        - show interfaces | include (GigabitEthernet|FastEthernet|TenGigabitEthernet|Vlan1|Vlan48)|packets input|total output drops|Received|Output
      register: show_interfaces_output

    - name: Get current time
      shell: date +"%Y-%m-%d %H:%M:%S"
      register: current_time

    - name: Generate JSON report
      set_fact:
        report_data:
          - "{{ current_time.stdout }}"
          - "{{ show_interfaces_output.stdout_lines }}"

    - name: Debug report_data
      debug:
        var: report_data
```

חלק ממסד הנתונים הנצבר במשך 3 חודשים :

```
2023-08-22 20:20:06,2,GigabitEthernet0/0,0/40,0,0,0,0,0
2023-08-22 20:20:06,2,GigabitEthernet1/0/1,0/40,1369336,241851556,0,946,919
2023-08-22 20:20:06,2,GigabitEthernet1/0/34,0/40,52217491,23792093516,0,628312,235313
2023-08-22 20:20:06,2,GigabitEthernet1/0/36,0/40,1601060,199043383,0,22513,22367
2023-08-22 20:20:06,2,GigabitEthernet1/0/37,0/40,1686972,217218358,0,31789,31651
2023-08-22 20:20:06,2,GigabitEthernet1/0/38,0/40,311721608,232989264370,0,1263077,1260691
2023-08-22 20:20:06,2,GigabitEthernet1/0/39,0/40,318357530,451689265289,0,1393215,1008486
2023-08-22 20:20:06,2,GigabitEthernet1/0/40,0/40,240478078,210529734411,0,1777456,1525400
2023-08-22 20:20:06,2,GigabitEthernet1/0/44,0/40,6396228,726294833,0,62120,54160
2023-08-22 20:20:06,2,GigabitEthernet1/1/1,0/40,0,0,0,0,0
2023-08-22 20:20:06,2,GigabitEthernet1/1/2,0/40,0,0,0,0,0
2023-08-22 20:20:06,2,GigabitEthernet1/1/3,0/40,0,0,0,0,0
2023-08-22 20:20:06,2,TenGigabitEthernet1/1/1,0/40,0,0,0,0,0
2023-08-22 20:20:06,2,TenGigabitEthernet1/1/2,0/40,0,0,0,0,0
2023-08-22 20:20:06,2,TenGigabitEthernet1/1/3,0/40,0,0,0,0,0
2023-08-22 20:20:06,2,TenGigabitEthernet1/1/4,0/40,5787145427,2821484070829,7165,4487945248,3283101940
2023-08-22 20:20:06,1,Vlan1,0/40,4706080,996804357,0,0,0
2023-08-22 20:20:06,1,Vlan48,0/40,2623471874,794970027694,0,0,0
2023-08-22 20:20:06,1,FastEthernet0/2,0/40,222839940,103736116377,0,601388,584146
2023-08-22 20:20:06,1,FastEthernet0/10,0/40,28667193,5402893564,0,3089564,3075652
2023-08-22 20:20:06,1,FastEthernet0/14,0/40,132045714,36886866496,0,7828868,7810291
2023-08-22 20:20:06,1,FastEthernet0/17,0/40,54856098,5729936847,0,219706,177616
2023-08-22 20:20:06,1,FastEthernet0/22,0/40,116632121,29153242165,0,1193585,1190821
2023-08-22 20:20:06,1,FastEthernet0/23,0/40,10138203,2790236800,0,1247487,1244529
2023-08-22 20:20:06,1,FastEthernet0/25,0/40,42741951,9730186175,0,628835,604542
2023-08-22 20:20:06,1,FastEthernet0/29,0/40,2121740,298589790,0,27571,27423
2023-08-22 20:20:06,1,FastEthernet0/30,0/40,13460777,2613620082,0,28363,28185
2023-08-22 20:20:06,1,FastEthernet0/36,0/40,100539707,11075254683,0,446081,443451
2023-08-22 20:20:06,1,FastEthernet0/37,0/40,21372026,4393680647,0,225539,211489
2023-08-22 20:20:06,1,FastEthernet0/43,0/40,12802140,1402943640,0,1494468,1478801
2023-08-22 20:20:06,1,FastEthernet0/45,0/40,1410090230,286547748026,0,1155683,1152493
2023-08-22 20:20:06,1,FastEthernet0/48,0/40,992065,230916699,0,513000,170256
2023-08-22 20:20:06,1,GigabitEthernet0/1,0/40,2256354395,1758138929757,0,2009927,1445842
2023-08-22 20:20:06,1,GigabitEthernet0/2,0/40,16756724472,15462407395491,0,4471858894,3269824163
2023-08-22 20:25:06,2,Vlan1,0/40,809622,69271898,0,0,0
2023-08-22 20:25:06,2,Vlan48,0/40,31602776,2916731122,56,0,0
2023-08-22 20:25:06,2,GigabitEthernet0/0,0/40,0,0,0,0,0
2023-08-22 20:25:06,2,GigabitEthernet1/0/1,0/40,1369376,241854116,0,946,919
2023-08-22 20:25:06,2,GigabitEthernet1/0/34,0/40,52219712,23793108248,0,628343,235314
2023-08-22 20:25:06,2,GigabitEthernet1/0/36,0/40,1601100,199045943,0,22513,22367
2023-08-22 20:25:06,2,GigabitEthernet1/0/37,0/40,1687014,217225000,0,31789,31651
2023-08-22 20:25:06,2,GigabitEthernet1/0/38,0/40,311722970,232989547982,0,1263105,1260719
2023-08-22 20:25:06,2,GigabitEthernet1/0/39,0/40,318357530,451689265289,0,1393215,1008486
2023-08-22 20:25:06,2,GigabitEthernet1/0/40,0/40,240478592,210529818817,0,1777499,1525443
2023-08-22 20:25:06,2,GigabitEthernet1/0/44,0/40,6396440,726331386,0,62144,54183
2023-08-22 20:25:06,2,GigabitEthernet1/1/1,0/40,0,0,0,0,0
2023-08-22 20:25:06,2,GigabitEthernet1/1/2,0/40,0,0,0,0,0
```

- **<u>טבלת חיפוש נתונים :</u>**

**<u>יצירת קוד לביצוע חיפוש description עבור כל ממשק , עם שם המתג ששייך לו:</u>**

נכתב בשפת פייתון באנסיבל : תחת PY CODES: extract_description2

בהינתן description שמשתמש מחפש, הקוד יבצע פעולת חיפוש בכל הממשקים ובכל המתגים, והתוצאה תהיה הממשק והמתג אליו הוא שייך.

נעשה באמצעות פעולת show run על המתגים.

כאשר מפעילים את פקודת show run על כל מתג, מוצגת הקונפיגורציה של המתג ולכן משם נשלוף את הנתונים הנדרשים למציאת ה description .

```
import pandas as pd
import re
import csv
import subprocess
import os
import time
from datetime import datetime
import json

def extract_report_data(playbook_output):
     # Get the current datetime
    current_datetime = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    # Extracting the lines containing the data
    start_index = playbook_output.find("ok: [switch-")
    end_index = playbook_output.rfind("\"")
    nested_data = playbook_output[start_index:end_index].split("\\n")

    # Debugging statement
#    print("Debugging nested_data", nested_data)

    # Extracting descriptions
    descriptions = []
    search_description = input("Please enter the description to search for: ")
    switch_number = None

    for index, line in enumerate(nested_data):
        line = line.strip()
#        print("Debugging the line", line)
        # Extract switch number
        if "ok: [switch-" in line:
            matches = re.search(r'(\d+)', line)
            if matches:
                switch_number = matches.group(0)
```

```
        # Extract interface and description
        if line.startswith('interface'):
            interface_name = line.split()[1]

            if index + 1 < len(nested_data) and 'description' in nested_data[index + 1]:
                description = nested_data[index + 1].split('description')[1].strip()

                # Check if the description matches the search_description
                if description == search_description:
                    descriptions.append({
                        'date': current_datetime,
                        'switch_number': switch_number,
                        'interface_name': interface_name,
                        'description': description
                    })

    return descriptions

def create_or_append_to_csv(csv_path, descriptions):
    # If there's data in the descriptions list
    if descriptions:
        # Check if the CSV file exists
        if os.path.isfile(csv_path):
            # Append to the existing CSV file
            with open(csv_path, 'a', newline='') as csvfile:
                writer = csv.writer(csvfile)
                for desc in descriptions:
                    writer.writerow([desc['date'], desc['switch_number'], desc['interface_name'], desc['description']])
                    print(f"Appended: Date: {desc['date']}, Switch: {desc['switch_number']}, Interface: {desc['interface_name']}, Description: {desc['description']}")
            print(f"Data appended to CSV file: {csv_path}")
        else:
            # Create a new CSV file and write the data
            with open(csv_path, 'w', newline='') as csvfile:
                writer = csv.writer(csvfile)
                writer.writerow(["Date", "Switch Number", "Interface", "Description"])  # Writing header
                for desc in descriptions:
                    writer.writerow([desc['date'], desc['switch_number'], desc['interface_name'], desc['description']])
                    print(f"Added: Date: {desc['date']}, Switch: {desc['switch_number']}, Interface: {desc['interface_name']}, Description: {desc['description']}")
            print(f"CSV file created: {csv_path}")
    else:
        print("No report data found.")
```

המשך הקוד :

```
def run_playbook_and_process_output(csv_path):
    # Change the working directory to /etc/ansible/
    os.chdir('/etc/ansible/')

    # Run the Ansible playbook and capture the output
    playbook_path = "playbooks/description_finder.yml"
    playbook_command = ["ansible-playbook", "-i", "inventory", playbook_path]

    # Start the playbook execution and capture the output
    playbook_process = subprocess.Popen(playbook_command, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    playbook_output, _ = playbook_process.communicate()

    # Wait for the playbook process to finish
    while True:
        if "PLAY RECAP" in playbook_output.decode():
            break
        # Sleep for a short duration before checking again
        time.sleep(1)

    # Check if the playbook execution was successful
    if playbook_process.returncode == 0:
        print("Playbook execution completed.")
    else:
        raise RuntimeError("Playbook execution failed.")

    # Extract the descriptions from the playbook output
    descriptions = extract_report_data(playbook_output.decode()) # Assuming the extract_descriptions function is the one we discussed before

    # Create or append to the CSV file
    create_or_append_to_csv(csv_path, descriptions)

# Path to the existing CSV file
csv_path = "/etc/ansible/DB_cisco_ios_logs/description_report2.csv"

# Run the playbook and process the output
run_playbook_and_process_output(csv_path)
```

מקושר ל PLAYBOOK : **description_finder.yml** (ב ANSIBLE)

התוצאה שמתקבלת לצורך המחשה :

באשר ה Input שנשלח לחיפוש הוא 4A-49

וה output עבור ה description הנ"ל הוא : מס' מתג : 1 , שם מתג : FastEthernet0/1

ונשמר לתוך קובץ CSV ששומר אליו תוצאות חיפוש שהמשתמש ביצע, בנתיב :

/etc/ansible/DB_cisco_ios_logs/description_report2.csv

```
study@dotan-spar2-pc:/etc/ansible$ sudo nano /etc/ansible/py_codes/extract_description2.py
study@dotan-spar2-pc:/etc/ansible$ sudo python3 py_codes/extract_description2.py
Playbook execution completed.
Please enter the description to search for: 4A-49
Appended: Date: 2023-09-24 11:43:23, Switch: 1, Interface: FastEthernet0/1, Description: 4A-49
Data appended to CSV file: /etc/ansible/DB_cisco_ios_logs/description_report2.csv
study@dotan-spar2-pc:/etc/ansible$
```

בהתבסס על מערך הנתונים שנאסף על פני תקופה של כשלושה חודשים, השלב השני כולל ניתוח נתונים יסודי באמצעות שפת התכנות Python , שלב זה מתועד במחברת Jupyter Notebook , אפשרי לצפות באמצעות סביבת העבודה של VSCode או Pycharm .

בחרנו להתמקד בפיצ'ר 'BROADCAST' בתוך נתוני הרשת.

(בפרויקט עתידי ניתן לבחון גם פיצ'רים אחרים על מנת לאבחן אנומליות נוספות ברשת).

תוך שימוש במודלים שונים של למידת מכונה וטכניקות של מדעי נתונים, נפתח אלגוריתם שינסה להעריך את ההיתכנות של זיהוי אירועים חריגים.

שלב זה כולל פיתוח אלגוריתמים וניתוח תוצאות בשיטות סטטיסטיות.

**PRE-PROCCESSING :**

עיבוד מקדים של נתונים הוא שלב ראשוני חשוב בתהליך, באמצעותו נבנה הבסיס לניתוח למידול יעיל, מסייע להבנה טובה יודע של הנתונים שנאספו.

בשלב ניתוח הנתונים הראשוני, עם פיצ'ר ה BROADCAST , עם תאריך, ושם ממשק מאוחד עם מס' מתג, ראינו את התפלגות הדאטה וכלים סטטיסטיים ראשוניים, על מנת להבין את התפלגות הנתונים ובהתאם לדעת כיצד לפעול עם האלגוריתם לסיווג.

*שינוי Feature : פיצ'ר ה- Broadcast מייצג ערך מצטבר , לכן שינינו את הערכים להפרש :
broadcast_delta

**ערכים חסרים :**

בשלב העיבוד המקדים של ניתוח הנתונים, טיפול בערכים חסרים הוא קריטי. זה כרוך באסטרטגיות לטיפול בנקודות נתונים עם מידע חלקי או חסר, על מנת להבטיח שלמות במערך הנתונים, תורם לניתוח שלאחר מכן.

ראשית, יצרנו קוד שישלוף את הערכים החסרים, מאחר והנתונים שנאספו מייצגים ערכים על פני זמן, הערכים החסרים לא מופיעים באופן נגיש.

האסטרטגיות לטיפול בנתונים חסרים:

- ממשקים אשר לא היה בהם מידע במשך יום שלם – נמחקו מהמסד נתונים לעיבוד.
- עבור ערכים חסרים באופן נקודתי, השתמשנו באינטרפולציה לינארית, והשלמנו את הערך החסר ע"י ממוצע בין הערך הקודם לערך הבא.

**חקר הנתונים :**

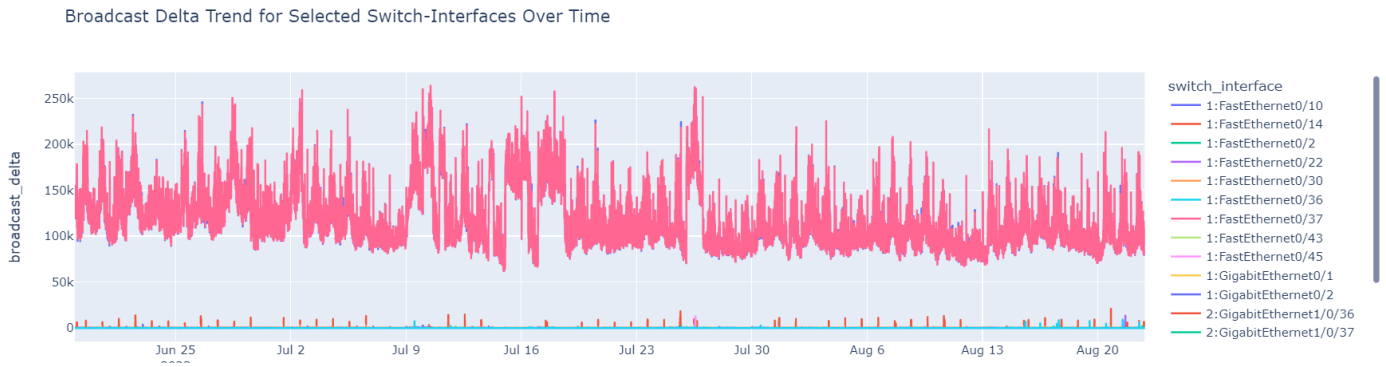
בדיקה ביסודיות את הנתונים על מנת לחשוף דפוסים ומגמות ולקבלת הבנה מקיפה של מאפייני הנתונים.

**התפלגות הדאטה במשך הזמן :**

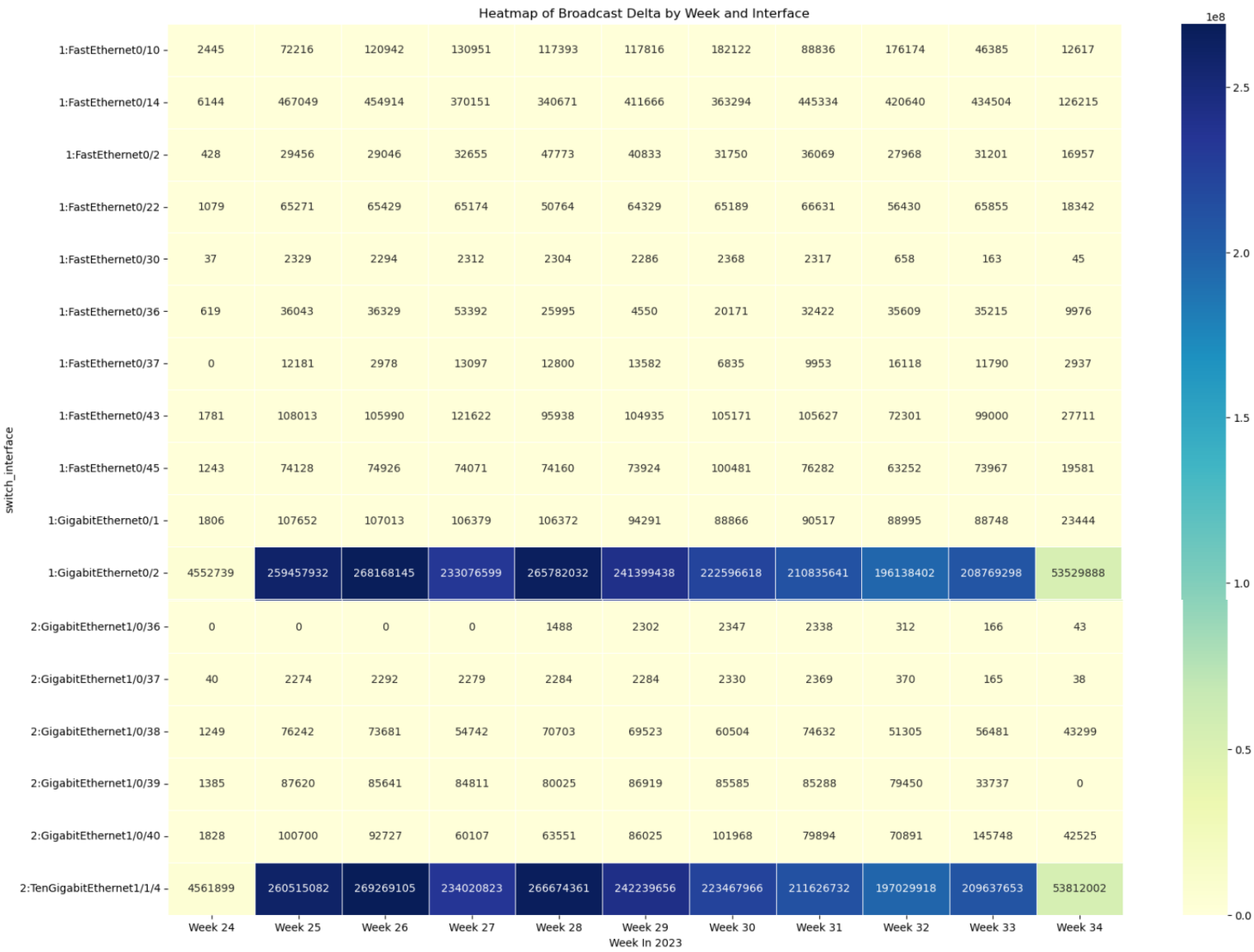
# data exploration

```python
selected_switch_interfaces = df['switch_interface'].unique()

subset = df[df['switch_interface'].isin(selected_switch_interfaces)]

# Use broadcast_delta instead of broadcast for the y-axis
fig = px.line(subset, x='date', y='broadcast_delta', color='switch_interface', title='Broadcast Delta Trend for Selected Switch-Interfaces Over Time')

fig.show()
```

[6] ✓ 7.4s Python

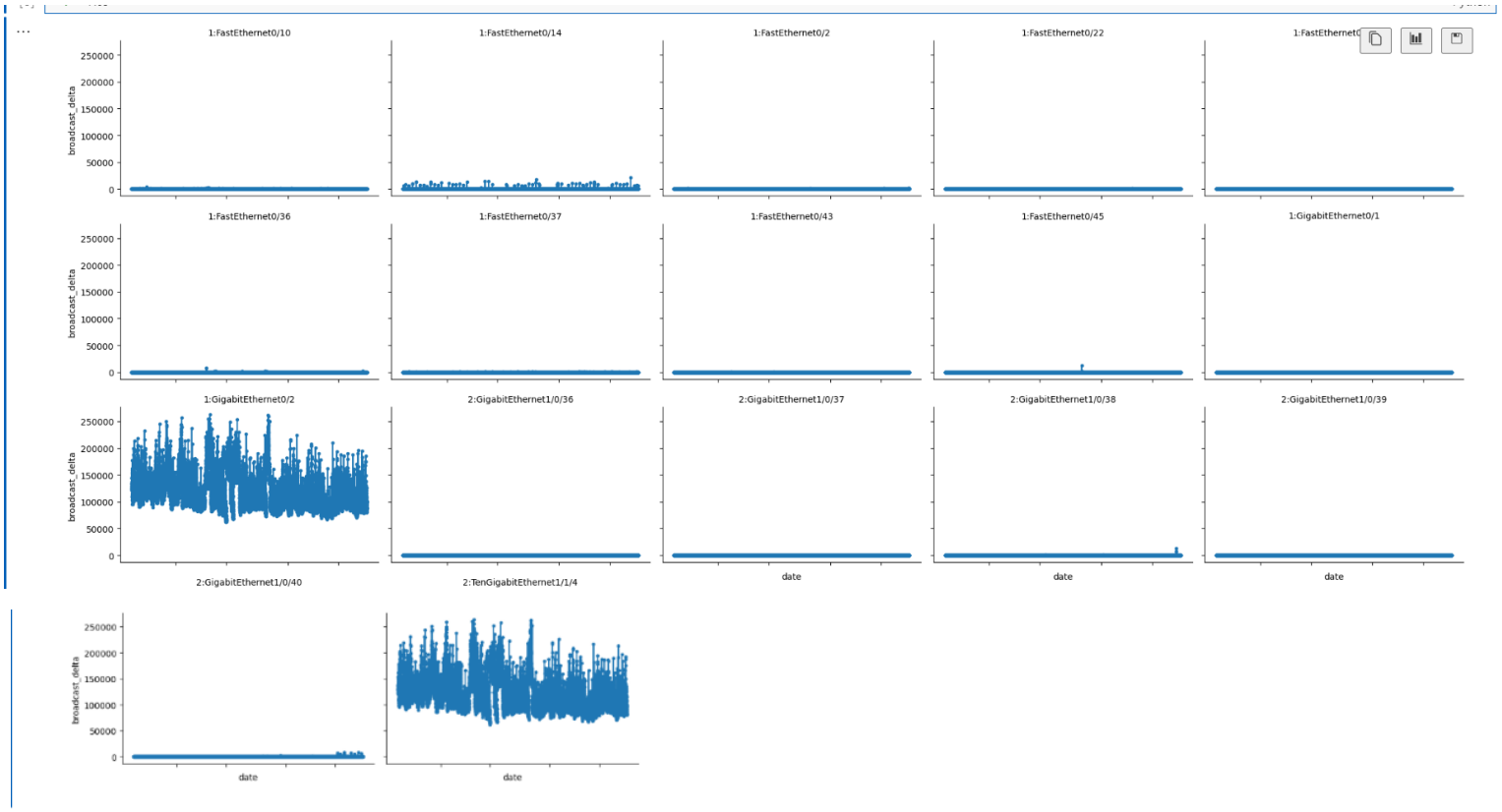



**התפלגות שבועית עבור כל ממשק :**




| switch_interface | Week 24 | Week 25 | Week 26 | Week 27 | Week 28 | Week 29 | Week 30 | Week 31 | Week 32 | Week 33 | Week 34 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1:FastEthernet0/10 | 2445 | 72216 | 120942 | 130951 | 117393 | 117816 | 182122 | 88836 | 176174 | 46385 | 12617 |
| 1:FastEthernet0/14 | 6144 | 467049 | 454914 | 370151 | 340671 | 411666 | 363294 | 445334 | 420640 | 434504 | 126215 |
| 1:FastEthernet0/2 | 428 | 29456 | 29046 | 32655 | 47773 | 40833 | 31750 | 36069 | 27968 | 31201 | 16957 |
| 1:FastEthernet0/22 | 1079 | 65271 | 65429 | 65174 | 50764 | 64329 | 65189 | 66631 | 56430 | 65855 | 18342 |
| 1:FastEthernet0/30 | 37 | 2329 | 2294 | 2312 | 2304 | 2286 | 2368 | 2317 | 658 | 163 | 45 |
| 1:FastEthernet0/36 | 619 | 36043 | 36329 | 53392 | 25995 | 4550 | 20171 | 32422 | 35609 | 35215 | 9976 |
| 1:FastEthernet0/37 | 0 | 12181 | 2978 | 13097 | 12800 | 13582 | 6835 | 9953 | 16118 | 11790 | 2937 |
| 1:FastEthernet0/43 | 1781 | 108013 | 105990 | 121622 | 95938 | 104935 | 105171 | 105627 | 72301 | 99000 | 27711 |
| 1:FastEthernet0/45 | 1243 | 74128 | 74926 | 74071 | 74160 | 73924 | 100481 | 76282 | 63252 | 73967 | 19581 |
| 1:GigabitEthernet0/1 | 1806 | 107652 | 107013 | 106379 | 106372 | 94291 | 88866 | 90517 | 88995 | 88748 | 23444 |
| 1:GigabitEthernet0/2 | 4552739 | 259457932 | 268168145 | 233076599 | 265782032 | 241399438 | 222596618 | 210835641 | 196138402 | 208769298 | 53529888 |
| 2:GigabitEthernet1/0/36 | 0 | 0 | 0 | 0 | 1488 | 2302 | 2347 | 2338 | 312 | 166 | 43 |
| 2:GigabitEthernet1/0/37 | 40 | 2274 | 2292 | 2279 | 2284 | 2284 | 2330 | 2369 | 370 | 165 | 38 |
| 2:GigabitEthernet1/0/38 | 1249 | 76242 | 73681 | 54742 | 70703 | 69523 | 60504 | 74632 | 51305 | 56481 | 43299 |
| 2:GigabitEthernet1/0/39 | 1385 | 87620 | 85641 | 84811 | 80025 | 86919 | 85585 | 85288 | 79450 | 33737 | 0 |
| 2:GigabitEthernet1/0/40 | 1828 | 100700 | 92727 | 60107 | 63551 | 86025 | 101968 | 79894 | 70891 | 145748 | 42525 |
| 2:TenGigabitEthernet1/1/4 | 4561899 | 260515082 | 269269105 | 234020823 | 266674361 | 242239656 | 223467966 | 211626732 | 197029918 | 209637653 | 53812002 |

Plot נוסף להמחשת המגמה של משתנה ה - "broadcast_delta" לאורך זמן עבור כל ממשק :

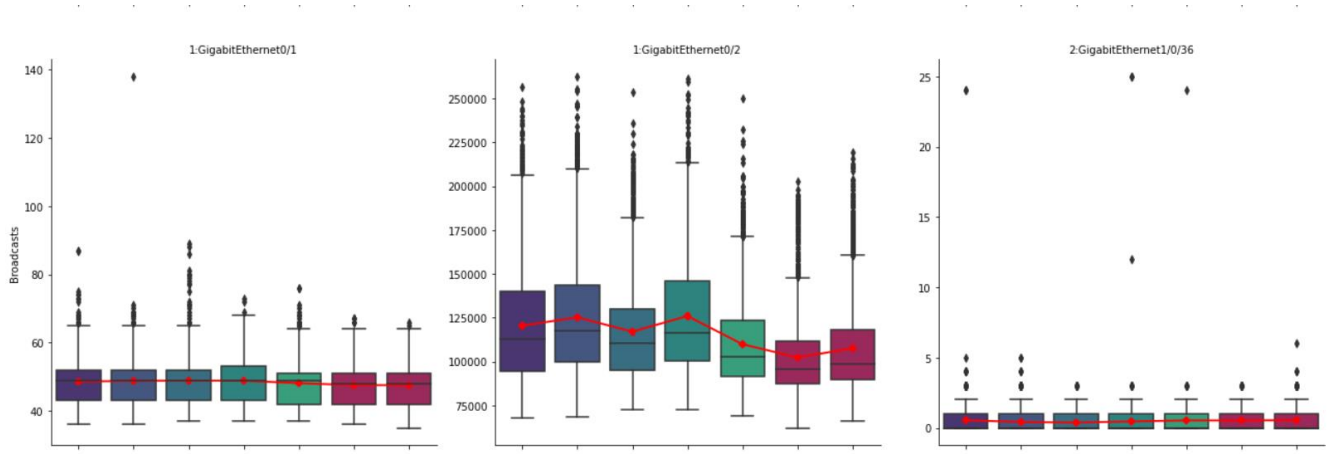



*לפי שני התוצאות האחרונות, ניתן לראות שיש שני ממשקים עמוסים יותר, בעלי ערכים גבוהים של Broadcast . ובנוסף, ניתן לראות שיש התנהגות שונה עבור כל ממשק,

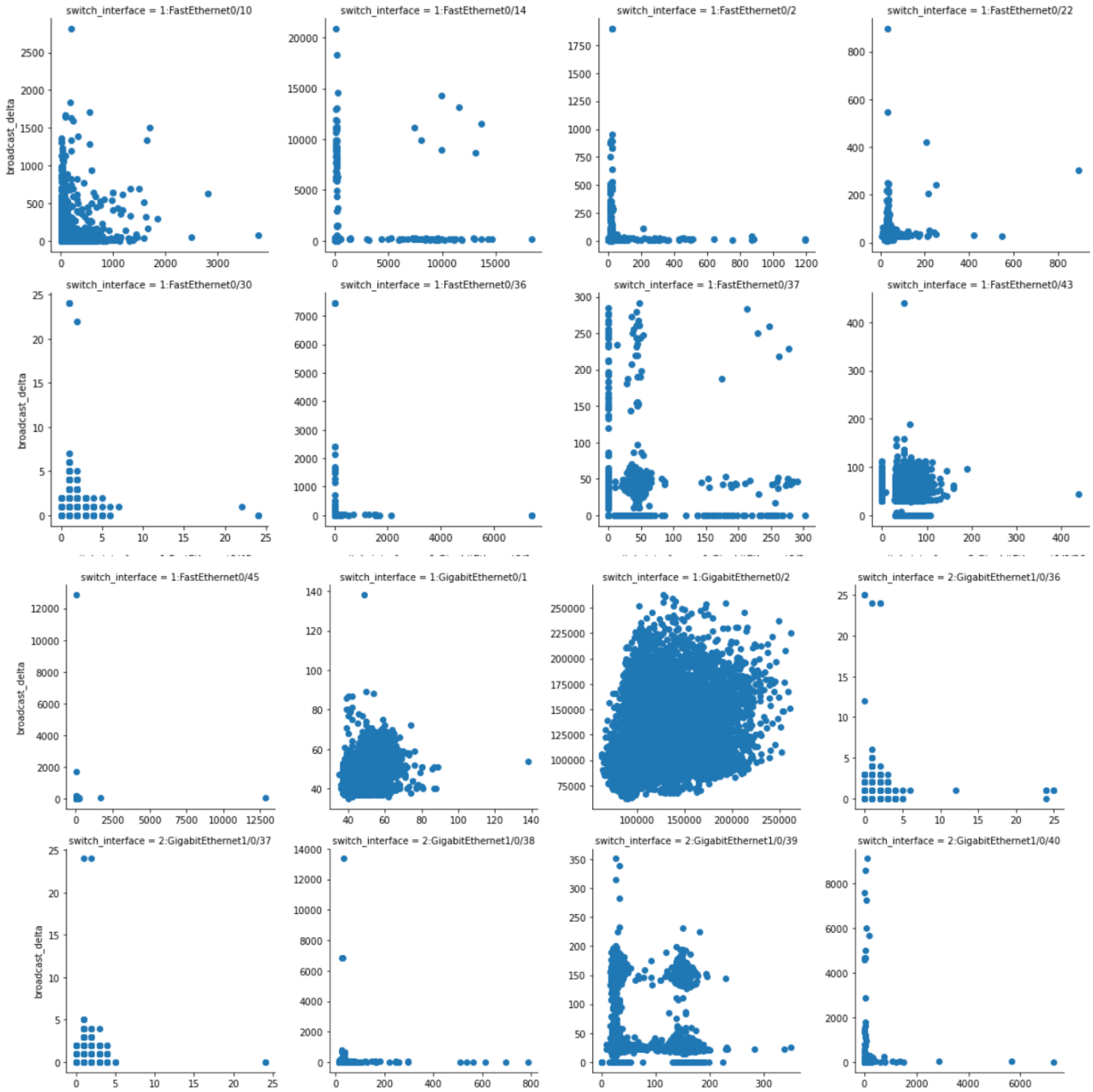
לכן סביר ששלב המידול ייעשה עבור כל ממשק בנפרד.

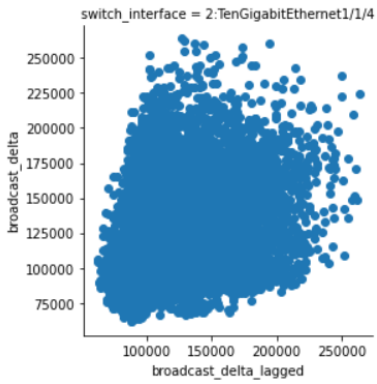
*בדיקה באמצעות גרף מתאים, האם יש תבנית כלשהי או הבדל כלשהו בין ימים באמצע השבוע לבין ימי הסופ"ש. לפי התוצאות - ניתן לראות שאין הבדל ושהמידע הוא די אחיד.

דוגמה עבור מס' ממשקים בודדים מהנתונים :

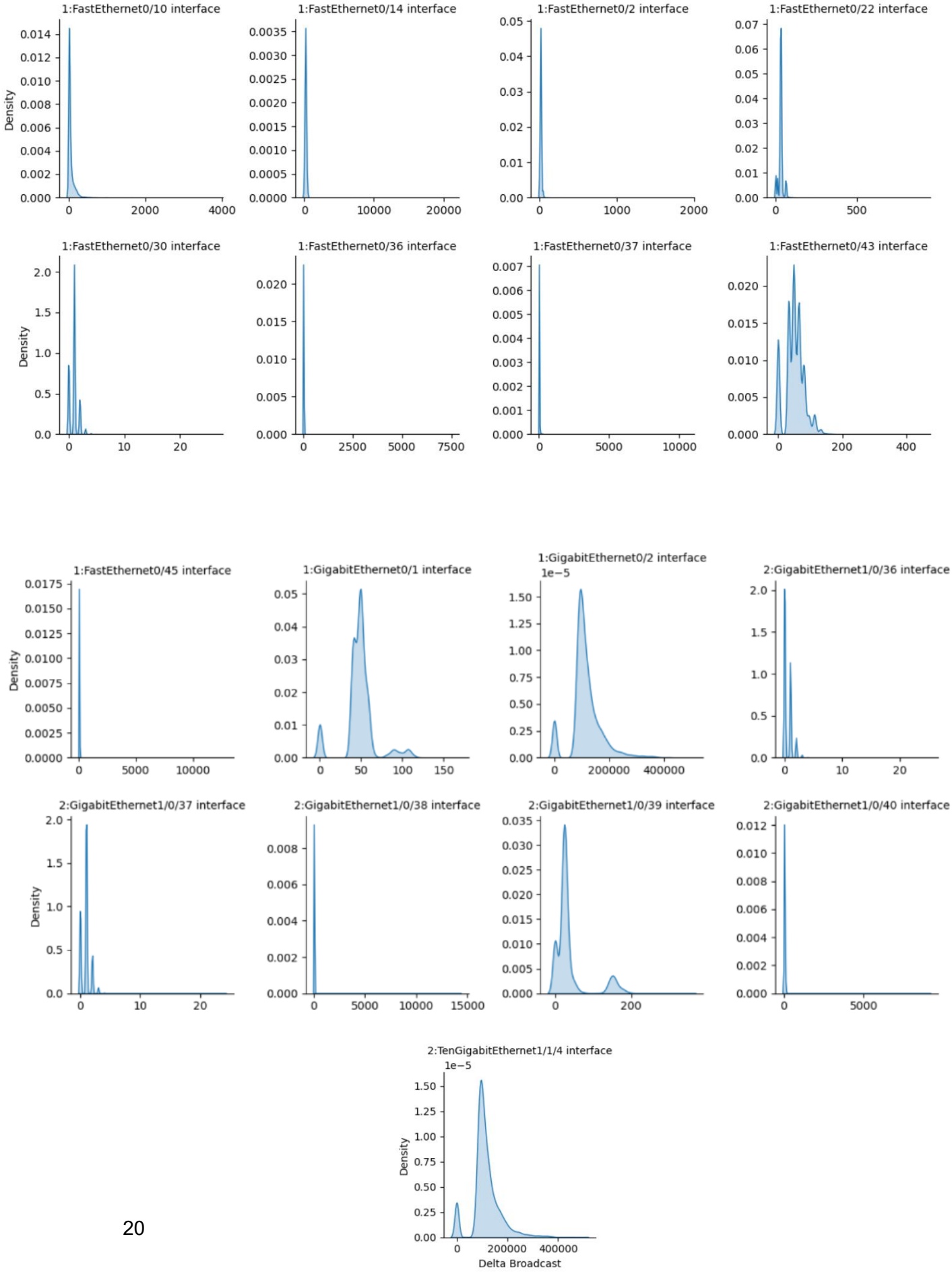


Plot הבא מציג ביצד ערכי Broadcast מלפני 7 ימים ('broadcast_delta_lagged') מתייחסים לערכי Broadcast נוכחיים ('broadcast_delta') עבור ממשקי מתגים שונים. תורם לנתח מגמות ומתאמים בנתונים עבור כל ממשק לאורך זמן :



**היסטוגרמה של כל ממשק :**



**טבלת מינימום ומקסימום :**

| | Mean | Median | Min | Max | Mode | Skewness | Standard Deviation | Kurtosis | 25th Percentile | 75th Percentile |
|---|---|---|---|---|---|---|---|---|---|---|
| **switch_interface** | | | | | | | | | | |
| 1:FastEthernet0/10 | 57.06 | 22.00 | 0.00 | 3773.00 | 0.00 | 8.95 | 109.90 | 158.97 | 12.00 | 58.00 |
| 1:FastEthernet0/14 | 205.32 | 165.00 | 0.00 | 20847.00 | 0.00 | 18.53 | 609.66 | 388.73 | 129.00 | 212.00 |
| 1:FastEthernet0/2 | 17.10 | 14.00 | 0.00 | 1895.00 | 14.00 | 28.89 | 30.78 | 1184.15 | 10.00 | 21.00 |
| 1:FastEthernet0/22 | 31.26 | 31.00 | 0.00 | 895.00 | 31.00 | 13.83 | 18.06 | 545.71 | 28.00 | 35.00 |
| 1:FastEthernet0/30 | 0.92 | 1.00 | 0.00 | 26.00 | 1.00 | 4.96 | 0.78 | 127.75 | 0.00 | 1.00 |
| 1:FastEthernet0/36 | 15.31 | 17.00 | 0.00 | 7441.00 | 17.00 | 60.75 | 79.32 | 4804.45 | 1.00 | 17.00 |
| 1:FastEthernet0/37 | 5.95 | 0.00 | 0.00 | 10529.00 | 0.00 | 123.72 | 79.57 | 16346.91 | 0.00 | 0.00 |
| 1:FastEthernet0/43 | 50.75 | 49.00 | 0.00 | 439.00 | 0.00 | 0.33 | 28.84 | 2.30 | 33.00 | 65.00 |
| 1:FastEthernet0/45 | 37.79 | 36.00 | 0.00 | 12874.00 | 36.00 | 126.64 | 96.43 | 16794.08 | 33.00 | 40.00 |
| 1:GigabitEthernet0/1 | 48.34 | 49.00 | 0.00 | 163.00 | 49.00 | 0.26 | 19.37 | 3.56 | 41.00 | 53.00 |
| 1:GigabitEthernet0/2 | 115869.09 | 105936.00 | 0.00 | 492404.00 | 0.00 | 1.15 | 56719.30 | 4.62 | 90833.00 | 134255.25 |
| 2:GigabitEthernet1/0/36 | 0.48 | 0.00 | 0.00 | 25.00 | 0.00 | 6.50 | 0.75 | 167.99 | 0.00 | 1.00 |
| 2:GigabitEthernet1/0/37 | 0.89 | 1.00 | 0.00 | 24.00 | 1.00 | 2.42 | 0.72 | 58.46 | 0.00 | 1.00 |
| 2:GigabitEthernet1/0/38 | 34.50 | 32.00 | 0.00 | 14306.00 | 36.00 | 77.93 | 158.31 | 6487.26 | 25.00 | 37.00 |
| 2:GigabitEthernet1/0/39 | 38.02 | 24.00 | 0.00 | 351.00 | 0.00 | 2.09 | 45.98 | 3.26 | 21.00 | 30.00 |
| 2:GigabitEthernet1/0/40 | 45.71 | 36.00 | 0.00 | 9126.00 | 35.00 | 39.84 | 163.86 | 1771.47 | 24.00 | 50.00 |
| 2:TenGigabitEthernet1/1/4 | 116239.27 | 106320.00 | 0.00 | 496437.00 | 0.00 | 1.15 | 56930.84 | 4.62 | 91098.50 | 134861.25 |

## מודלים

לכל אחד מהאלגוריתמים שהשתמשנו יש גישה שונה לזיהוי חריגות בדאטה לא מסווג :

1. FUZZY - KMEANS :
מקבץ את הנתונים ל - Clusters ע"י הקצאת נקודות בדאטה עם דרגות שונות של קשר ביניהן, והאנומליות מזוהות כנקודות עם ציון נמוך של קשר לclusters . יכול להועיל במציאת חריגויות כאשר נקודה לא שייכת באופן מובהק לאשכול מסוים. בפרוייקט שלנו בחרנו במודל זה עם 2-3 אשכולות בלבד.
2. HDBSCAN :
אלגוריתם מבוסס צפיפות. מזהה אשכולות בצורות וגדלים משתנים ומציין נקודות בדאטה שאינן שייכות לאף אשכול כאנומליות. היתרון שלו על פני K-MEANS שהוא קובע אוטומטית את מספר האשכולות ומטפל באשכולות בצפיפויות שונות. גם פה בחרנו שהאלגוריתם יפעל על 2-3 אשכולות בלבד.
3. ISOLATION-TREE / ISOLATION-FOREST
על ידי חלוקה רקורסיבית של הנתונים לתת-קבוצות עד להשגת בידוד (מספר נקודות נתונים בכל תת-קבוצה). אנומליות ממוקמות קרוב יותר לשורש העץ.
Isolation Forest פועל עם מספר עצי בידוד וע"י חישוב ממוצעים של התוצאות שלהם מזהה חריגות. נקודות בדאטה שדורשות פחות פיצולים לבידוד נחשבות לאנומליות.
4. ONE CLASS SVM :
אלגוריתם One-Class Support Vector Machine (SVM) הוא כלי עוצמתי לזיהוי חריגות. על ידי אימון מודל על מערך נתונים, הוא מגדיר גבול המקיף את רוב נקודות הנתונים "הנורמליות" במרחב בעל ממדים גבוהים. גבול זה משמש לסיווג נקודות נתונים חדשות כנורמליות או חריגות על סמך מיקומן ביחס אליה (גבול החלטה).
5. LOF :
אלגוריתם LOF פועל על ידי מדידת הצפיפות המקומית של נקודות בתוך מערך נתונים, זיהוי חריגות כנקודות בעלות צפיפות מקומית נמוכה משמעותית בהשוואה לשכנותיהן.

לכל אחד מהאלגוריתמים יש יתרונות וחסרונות בזיהוי אנומליות על דאטה שאינו מסווג, רוב החסרונות תלויים בפרמטרים שכל מודל דורש, ומאחר ואין לנו מידע מקדים על האנומליה הנבחנת, אלא אנו רוצים לקבוע מהי אנומליה, יש לבדוק כל אחד מהם באופן ספציפי על כל אחד מהממשקים שדגמנו, וכך על ידי ניתוח התוצאות והביצועים, נוכל להבין באיזה מבין המודלים נשתמש.

## ROLLING WINDOWS :

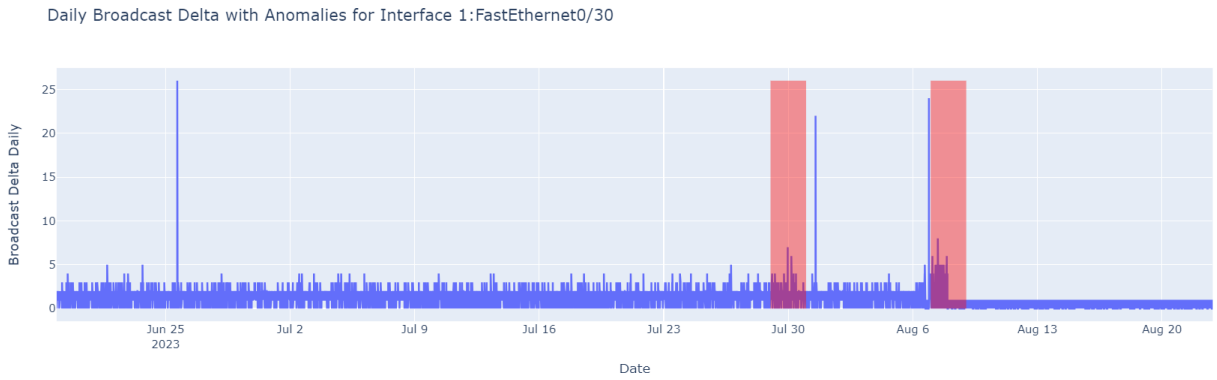
שיטה שתורמת למודלים ללמוד על הנתונים, באמצעות חלוקת מערך נתונים מסדרת זמן למקטעים של מרווחי זמן קבועים. מאפשרת לאלגוריתמים לנתח ולזהות אנומליות בתוך חלונות זמן מוגדרים. מספקת תצוגה דינמית של דפוסי נתונים ומגמות לאורך זמן.

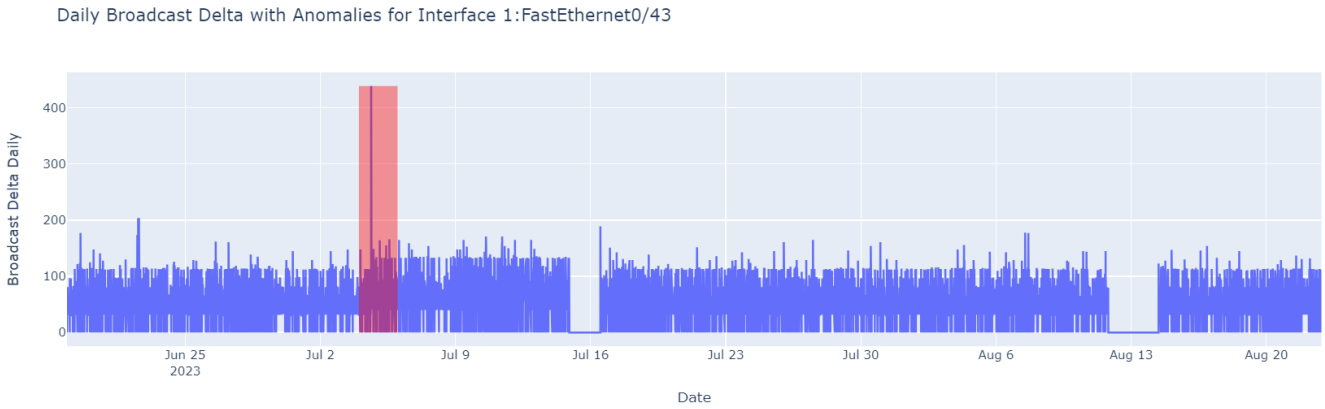
בחרנו להשתמש בשיטה זו על מנת לזהות אנומליות בחלונות זמן של 5 דקות, שעתי ויומי.

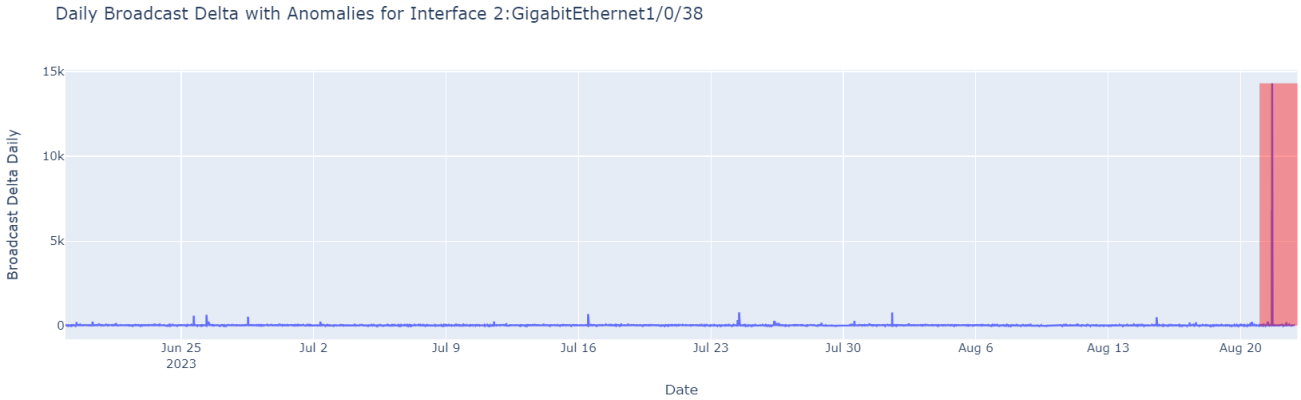
**הצגת חלון זמן יומי :**

ע"י חישוב סטטיסטיקה יומית ניתן לזהות חריגות בפעילות כל ממשק ברשת. עבור כל ממשק, נוצר קו של ציר זמן של ה – broadcast_delta היומי. מזהה חלונות חריגים ע"י זיהוי חיפוש ימים רצופים עם אנומליות, שמסומנות באדום.

נציג כאן עבור מספר בודד של ממשקים, עם בחירה רנדומלית לצורך המחשה :

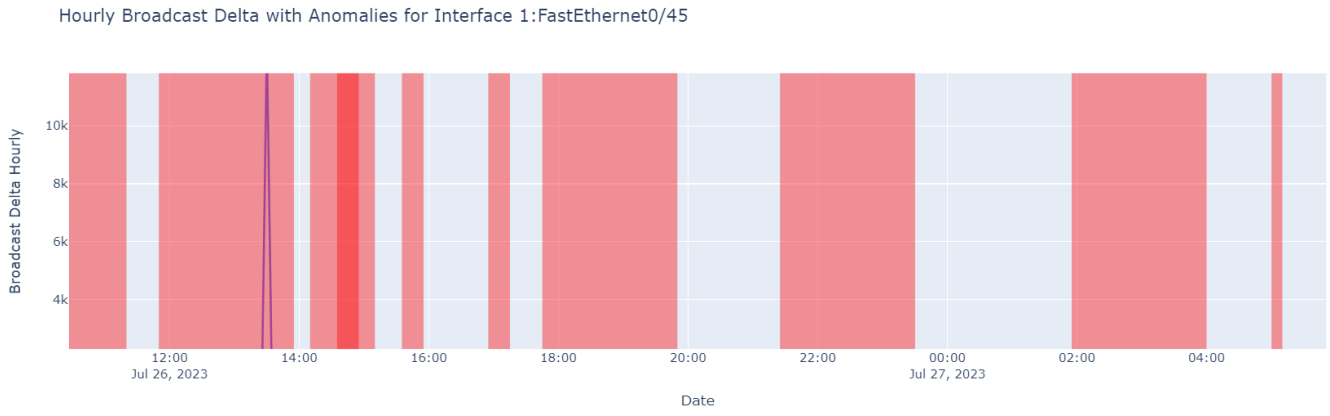


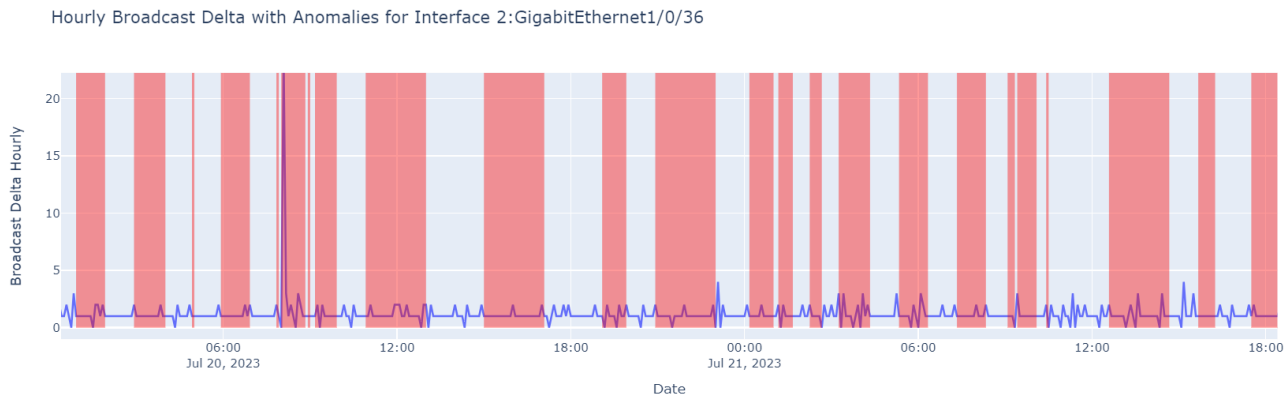



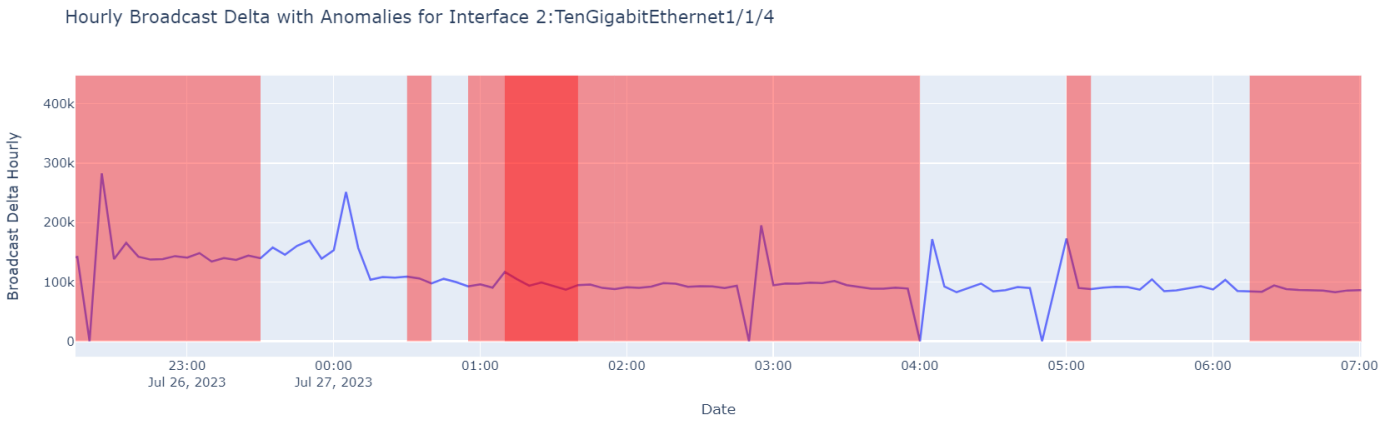



**הצגת חלון זמן שעתי :**

נעשה באופן דומה לחלון זמן יומי, נציג חלונות זמן של ממשקים בודדים בבחירה רנדומלית :



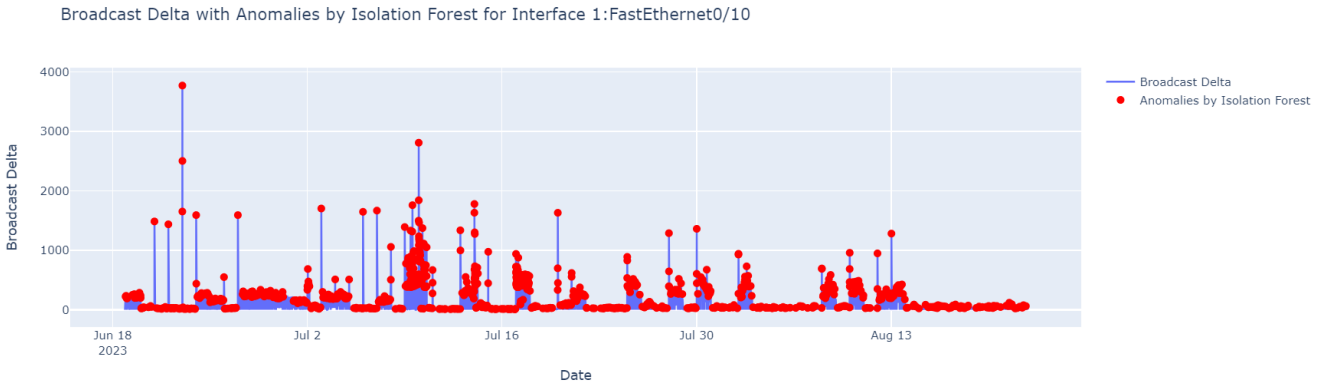



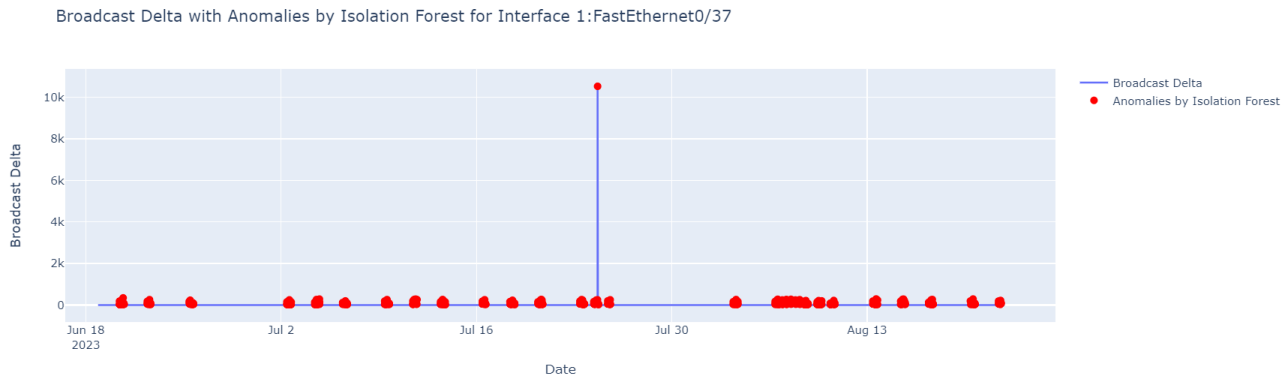
# ביצועים

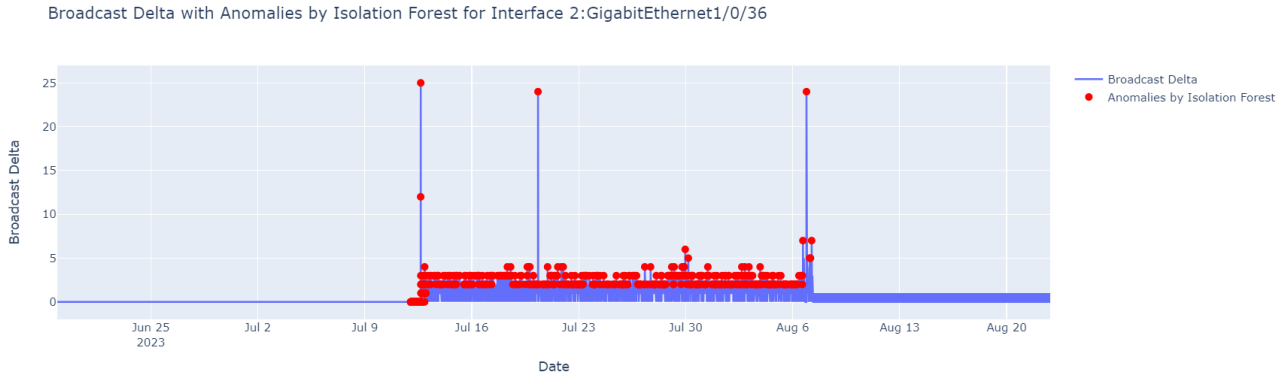
להלן התוצאות עבור המודלים הנבחרים : IsolationForest , One-class VM , and LOF

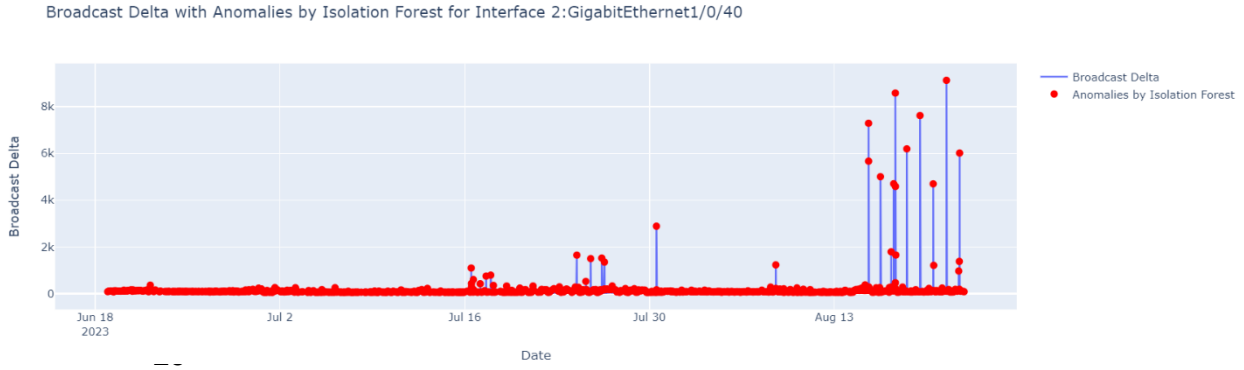
נציג עבור מספר ממשקים בודד, במחברת המקושרת לספר הפרויקט יש מהצגה ננרחבת יותר.

**1) מודל ה – IsolationForest :**

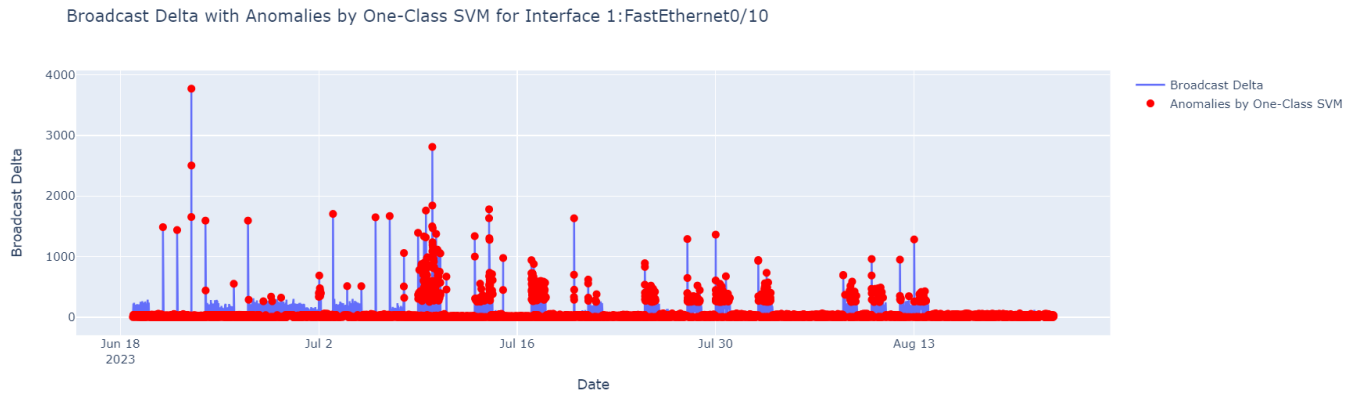


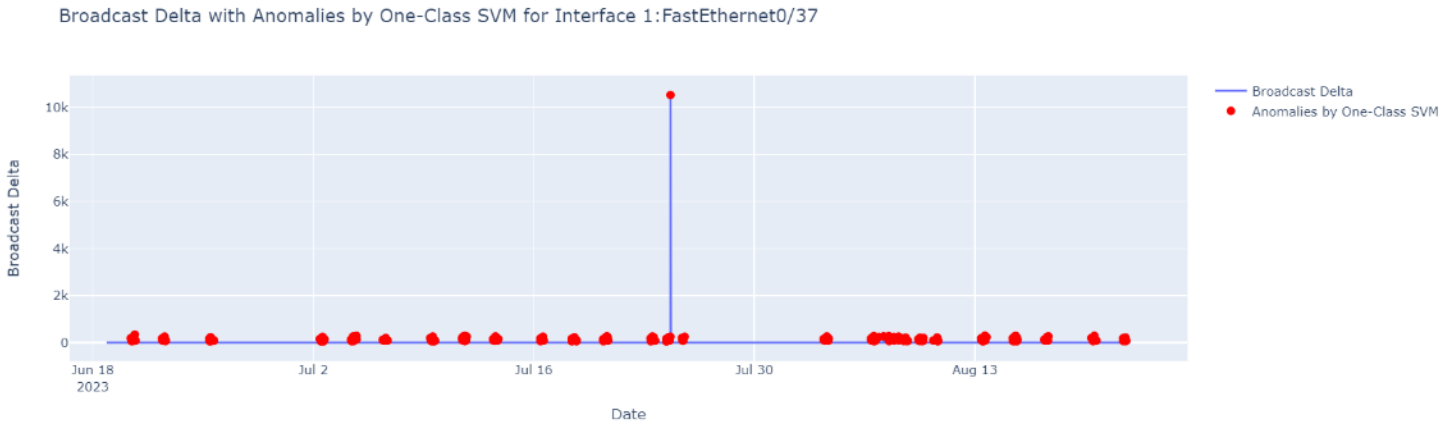



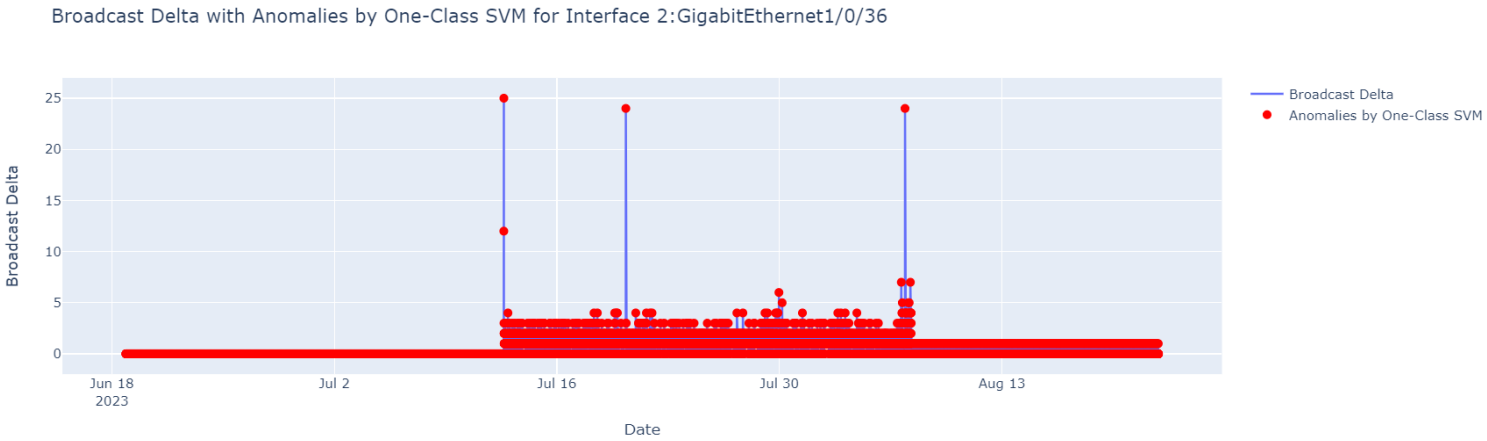



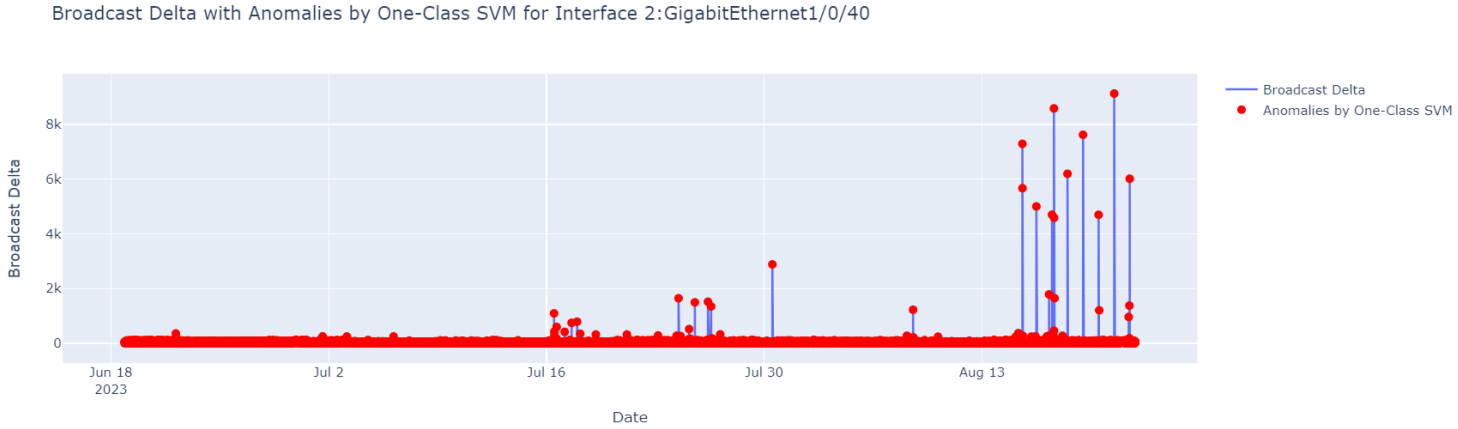



**One Class SVM (2**

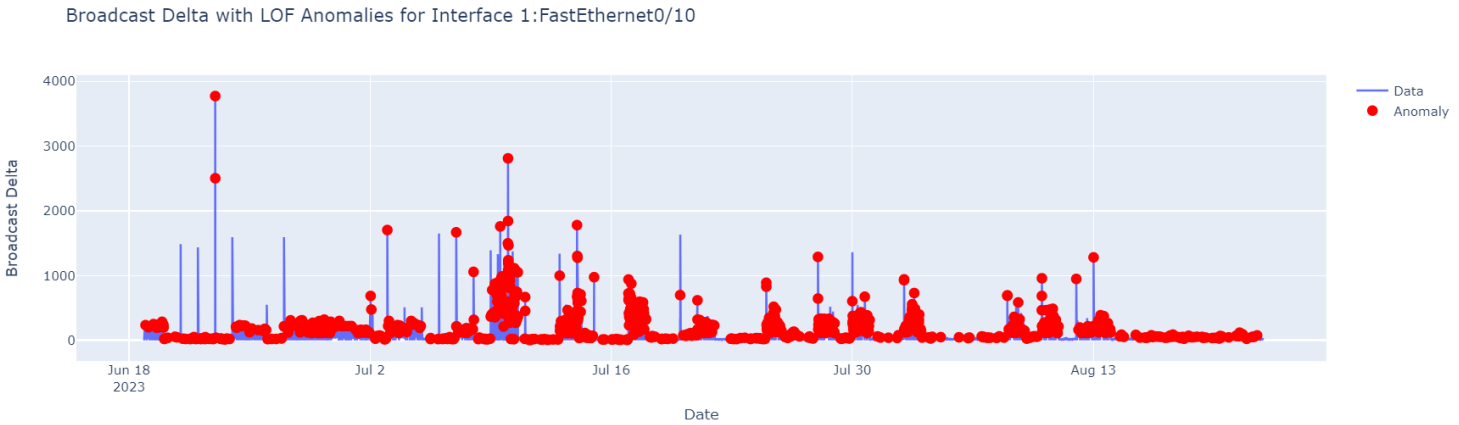


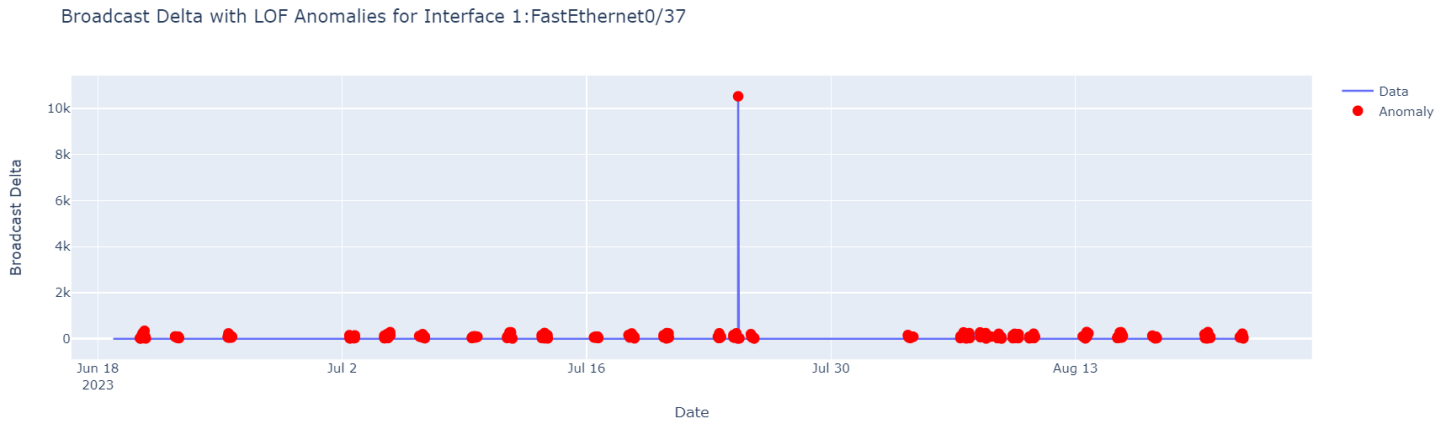



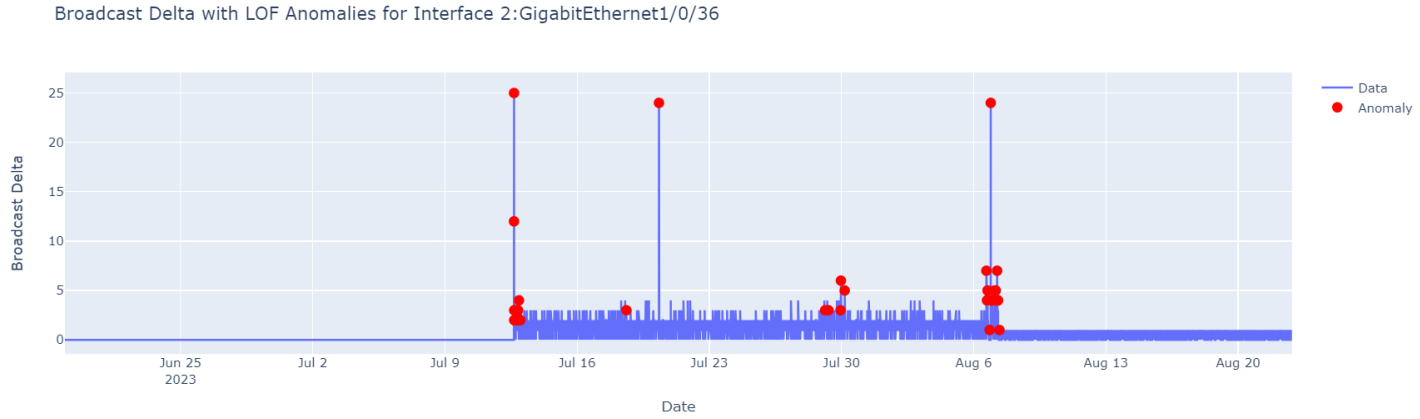



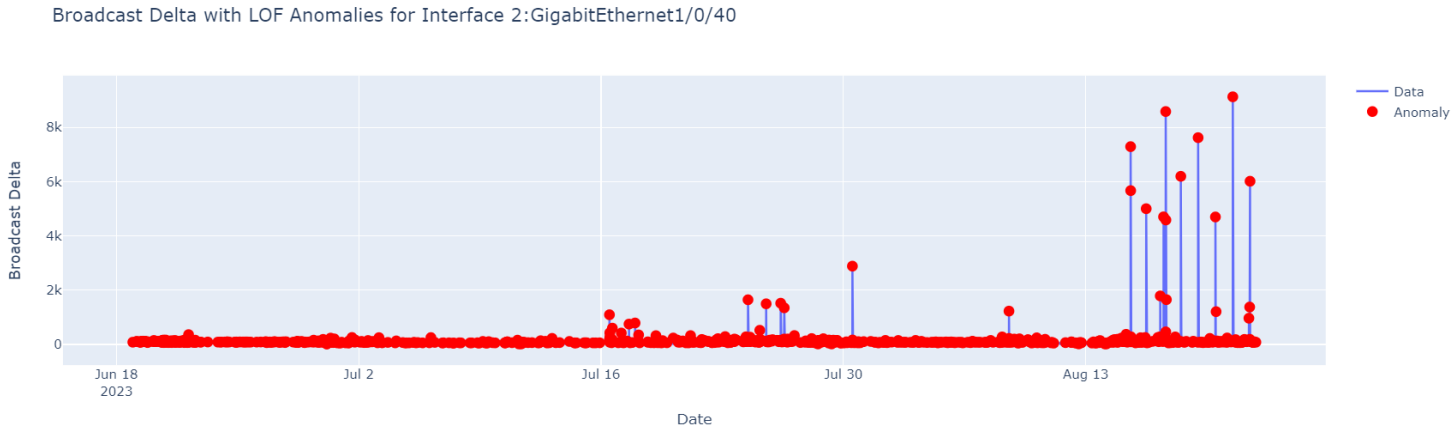



**: LOF (3**


Broadcast Delta with LOF Anomalies for Interface 1:FastEthernet0/10
Broadcast Delta
4000
3000
2000
1000
0
Jun 18
2023
Jul 2
Jul 16
Jul 30
Aug 13
Date
Data
Anomaly


Broadcast Delta with LOF Anomalies for Interface 1:FastEthernet0/37
Broadcast Delta
10k
8k
6k
4k
2k
0
Jun 18
2023
Jul 2
Jul 16
Jul 30
Aug 13
Date
Data
Anomaly


Broadcast Delta with LOF Anomalies for Interface 2:GigabitEthernet1/0/36
Broadcast Delta
25
20
15
10
5
0
Jun 25
2023
Jul 2
Jul 9
Jul 16
Jul 23
Jul 30
Aug 6
Aug 13
Aug 20
Date
Data
Anomaly


Broadcast Delta with LOF Anomalies for Interface 2:GigabitEthernet1/0/40
Broadcast Delta
8k
6k
4k
2k
0
Jun 18
2023
Jul 2
Jul 16
Jul 30
Aug 13
Date
Data
Anomaly

## שיפור המודלים :

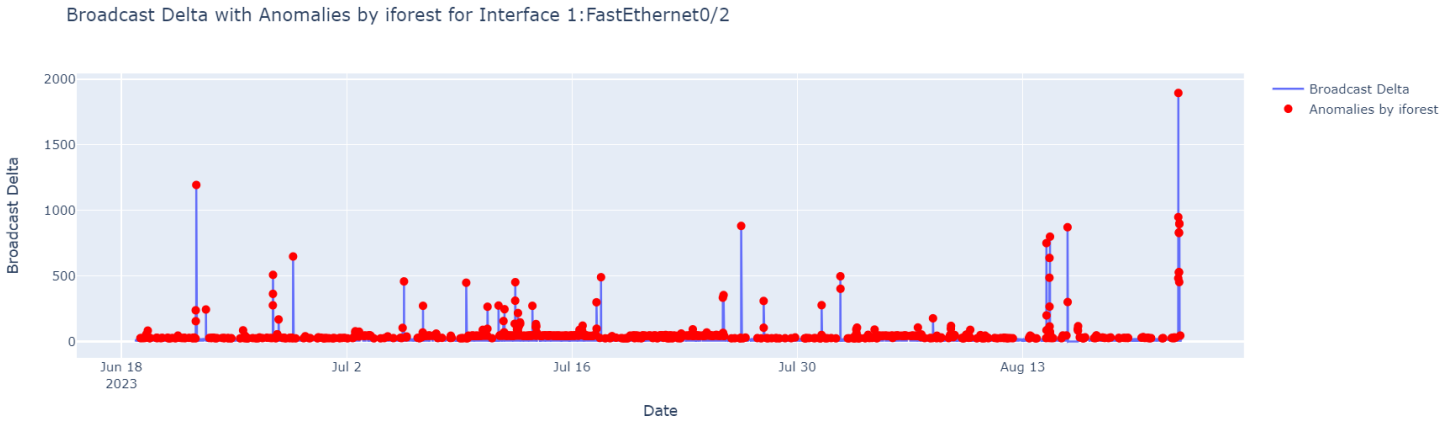
בשלב זה ביצענו תיקון לאנומליות :

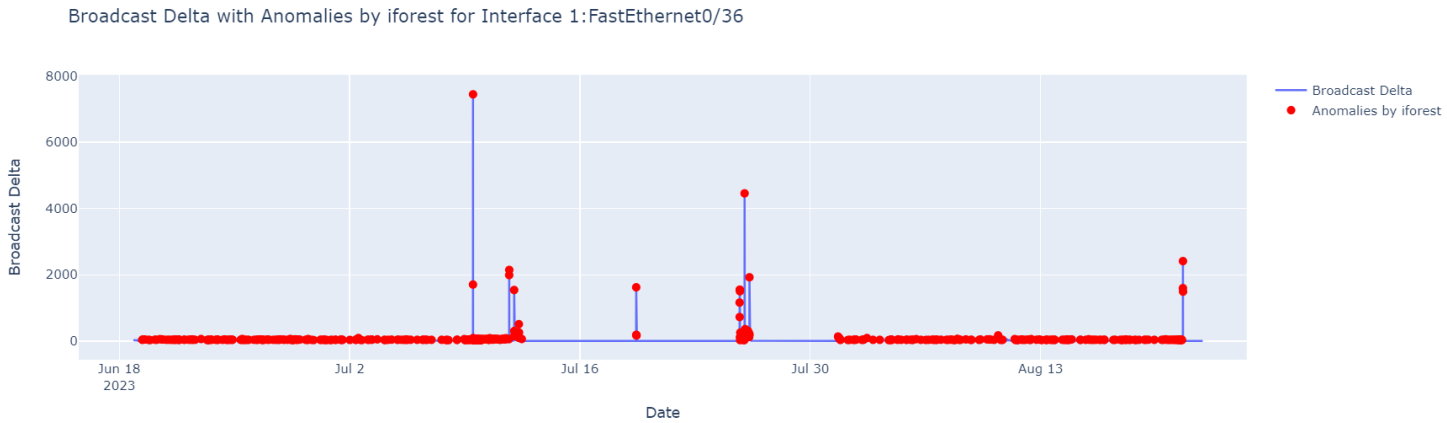
- נתון המסווג כאנומליה עם הבדל זניח של עד 5 Broadcasts מהערך הקודם – לא הוחשב כנתון חריג, ולכן לא יסווג כאנומליה.
- התעלמנו מאנומליות המסווגות מלמטה – הגדרנו נתונים אלו כנתונים לא חריגים, מאחר והנתונים מתייחסים לעומס ברשת, לכן נתונים עם ערכים גבוהים הם הנתונים שאליהם נרצה להתייחס.
- מאחר והמודלים מתייחסים לכל ממשק בנפרד, הסתכלנו על כל ממשק בנפרד וסימנו threshold בצורה ידנית שמעליו הנתון יסווג כאנומליה.

**לאחר התיקונים, ביצוע אבחנה בין שלושת המודלים, בין היתר ע"י יצירת מודל המחבר בין שלושתם.**

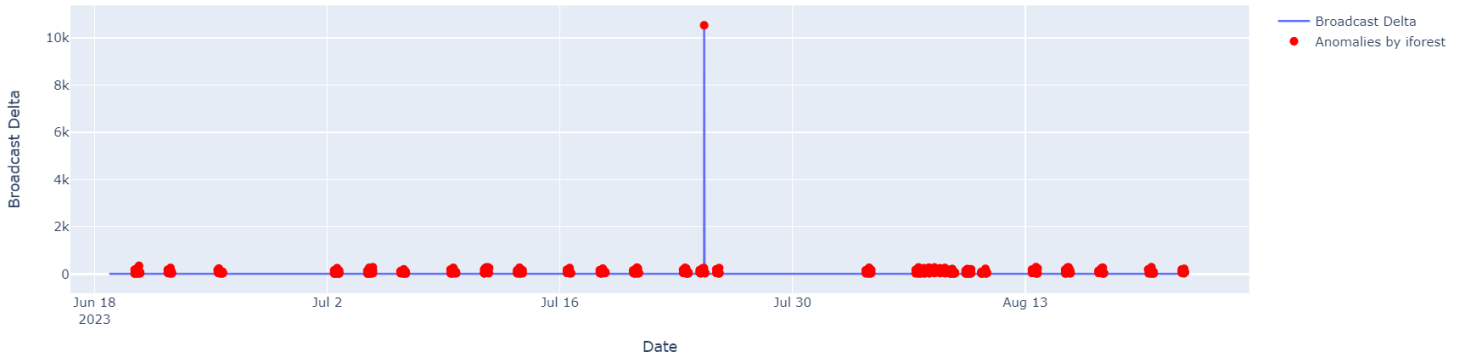
**IsolationForest שונה מהשאר ועל בסיס מערך הנתונים גילינו שבזיהוי האנומליות הוא מסווג בצורה הטובה ביותר. בנוסף, החלטנו לבחור באלגוריתם ה- IForest מאחר והוא מסוגל לעבוד עם כמות נתונים גדולה יותר משאר המודלים. (בפרויקט שלנו, מאחר ודאטה אינו מסווג, האלגוריתם יהיה חכם יותר כאשר מקבל יותר דוגמאות להתנהגות הממשקים ברשת.**

**חלק מביצועי האלגוריתם הסופי לאחר השיפורים : IForest**

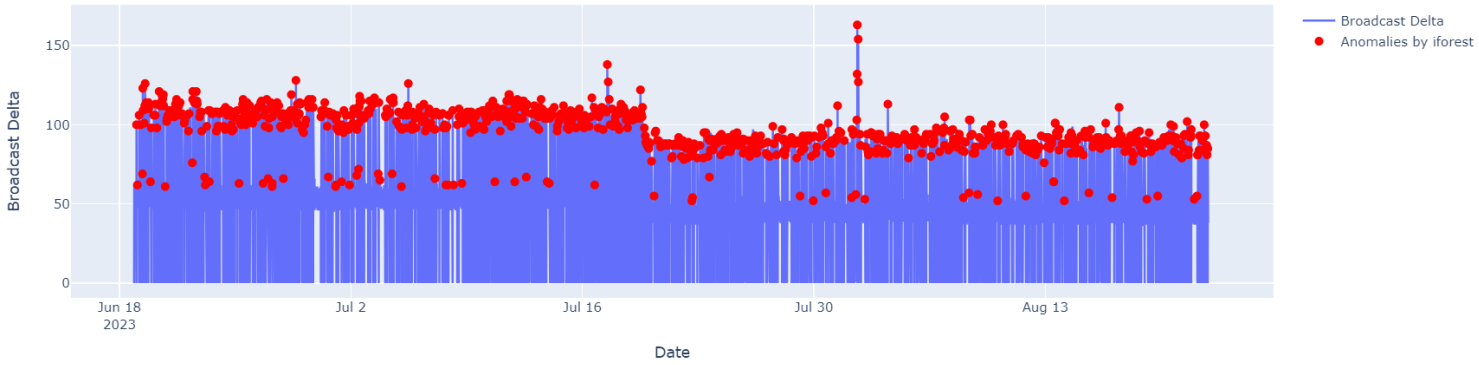


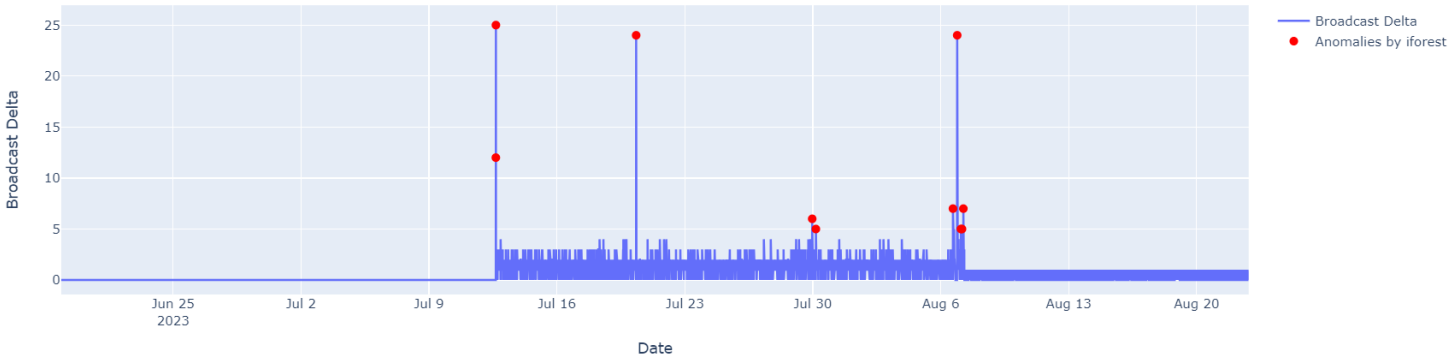
Broadcast Delta with Anomalies by iforest for Interface 1:FastEthernet0/37



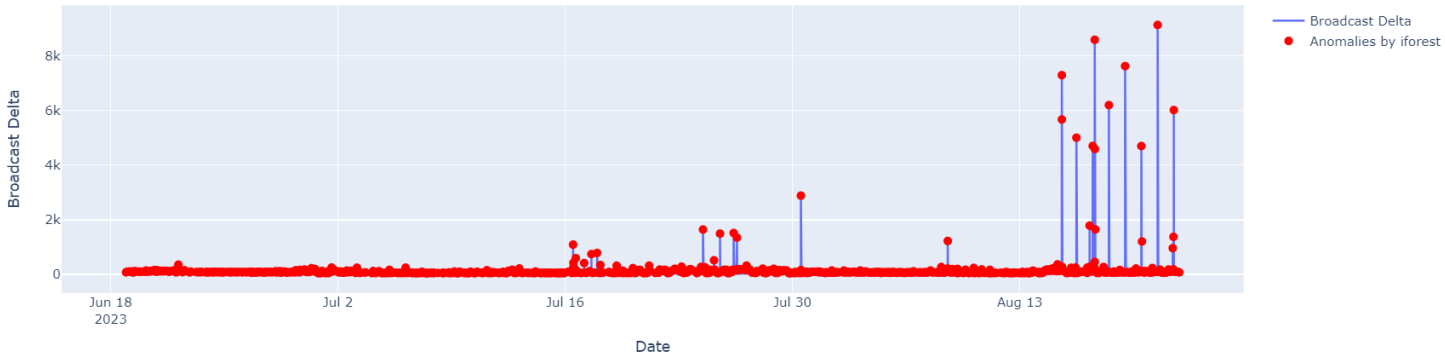

Broadcast Delta with Anomalies by iforest for Interface 1:GigabitEthernet0/1




Broadcast Delta with Anomalies by iforest for Interface 2:GigabitEthernet1/0/36




Broadcast Delta with Anomalies by iforest for Interface 2:GigabitEthernet1/0/40

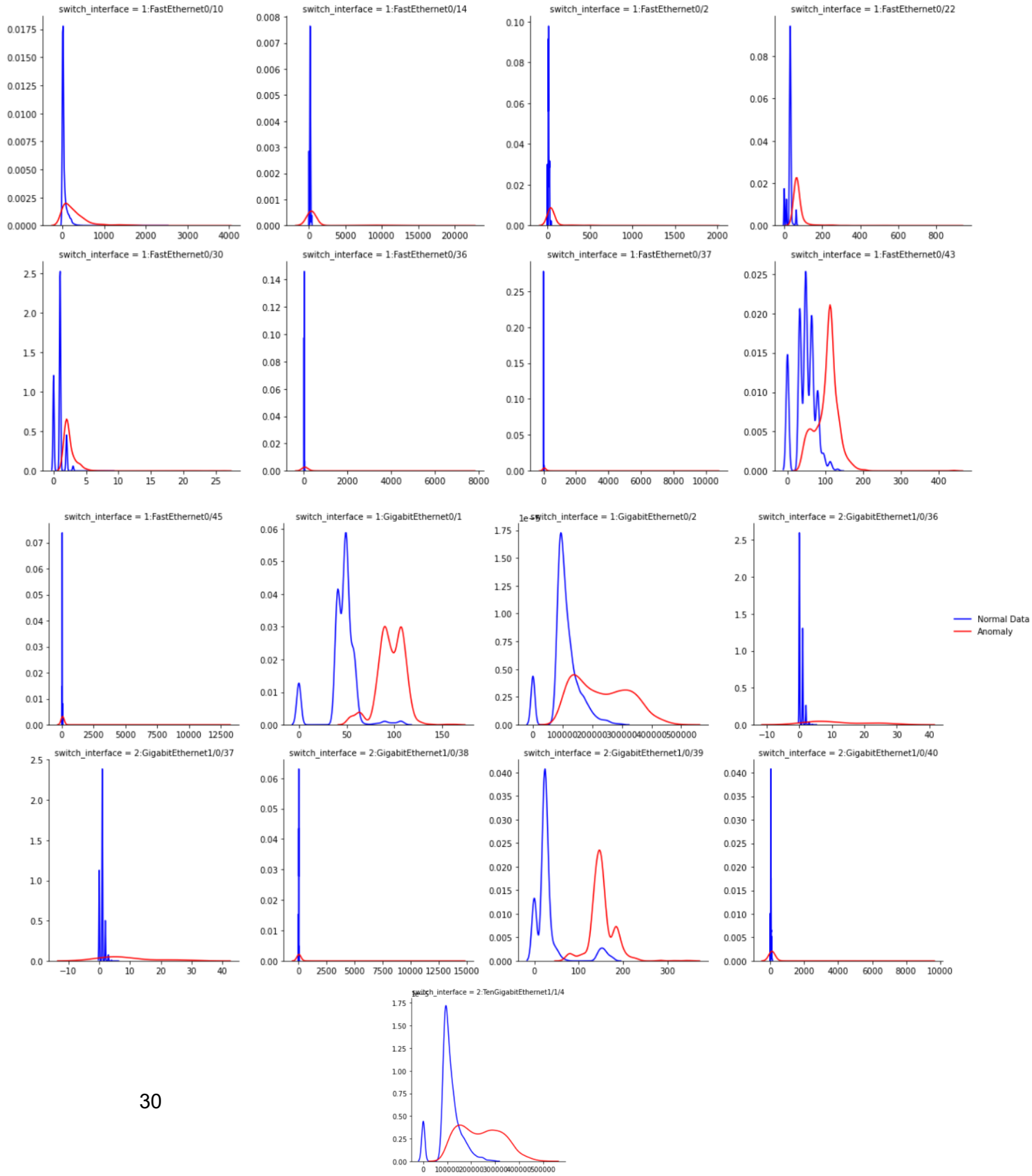


## הערכת המודל

הגרפים הבאים מציגים את התפלגות הנתונים עם הבחנה בין הדפוסים של פעילות רגילה ברשת לבין אנומלית בכל ממשק. מאפשר לראות את ההבדלים ביניהם.

ניתן לראות שלרוב, ההתפלגות של הערכים החריגים קטנה יותר ובטווח ערכים דומה -

וזה בדיוק מה שהיינו מצפים.



## מטריצת קורולציות בין המודלים : Correlation Heatmap

**מתאם בין הפלטים השונים של אלגוריתמי זיהוי האנומליות :**

תת-קבוצה של ה-DataFrame המקורי המכילה רק את העמודות הקשורות לאנומליות.

עמודות אלו כוללות את האנומליות משלושת המודלים הנבחרים : 'anomaly_iforest' , 'anomaly_ocsvm' 'anomaly_lof' ובנוסף 'early_ist_anomaly' (אנומליות ממודל אנסבל של חיבור שלושת המודלים) .

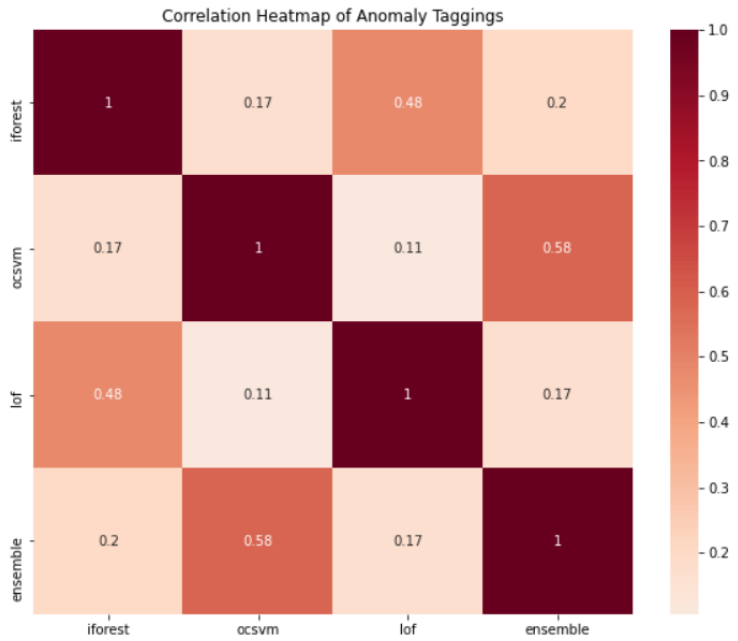
חישוב מטריצת מתאם להבנת הקשר בין המודלים. מטריצת המתאם מודדת עד כמה סיווג האנומליות מהמודלים הללו קשורים זה לזה באופן ליניארי.

מתאמים חיוביים גבוהים (ערכים קרובים ל-1) מצביעים על כך שכאשר אלגוריתם אחד מתייג נקודת נתונים כאנומליה, אחרים נוטים לעשות את אותו הדבר, מה שמצביע על הסכמה.

לעומת זאת, מתאמים שליליים (ערכים קרובים ל-1) מצביעים על אי הסכמה, כאשר אלגוריתם אחד מסמן נקודת נתונים כאנומליה בעוד אחרים לא.

המפה מסייעת בהבנת העקביות וההבדלים בתוצאות זיהוי חריגות על פני האלגוריתמים המופעלים, ומספקת תובנות לגבי יעילותם וגישות האנסמבל הפוטנציאליות לשיפור ביצועי זיהוי החריגות.

...



## מדדי Silhouette Score and Davies-Bouldin Score :

הערכת הביצועים של מודל זיהוי חריגות של Isolation Forest בממשקי רשת שונים על ידי חישוב שני מדדי הערכה: Silhouette Score ו- Davies- Bouldin Score :

- Silhouette Score : מודד עד כמה אובייקט דומה לאשכול שלו בהשוואה לאשכולות אחרים. ציון גבוה יותר מציין שנקודות חריגות מופרדות היטב מהנקודות הנורמליות. כלומר, אם הציון גבוה, זה מרמז שהמודל יצרו בהצלחה אשכולות או קבוצות נפרדות עבור חריגות ונקודות נתונים רגילות.

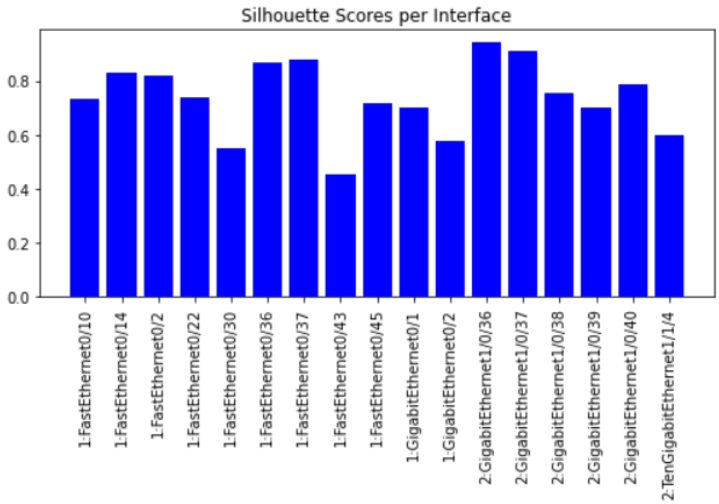
- Davies-Bouldin Score : מודד את הדמיון הממוצע בין כל אשכול לאשכול הדומה ביותר שלו. ציונים נמוכים יותר מצביעים על הפרדה טובה יותר בין האנומליות לנקודות הנורמליות.

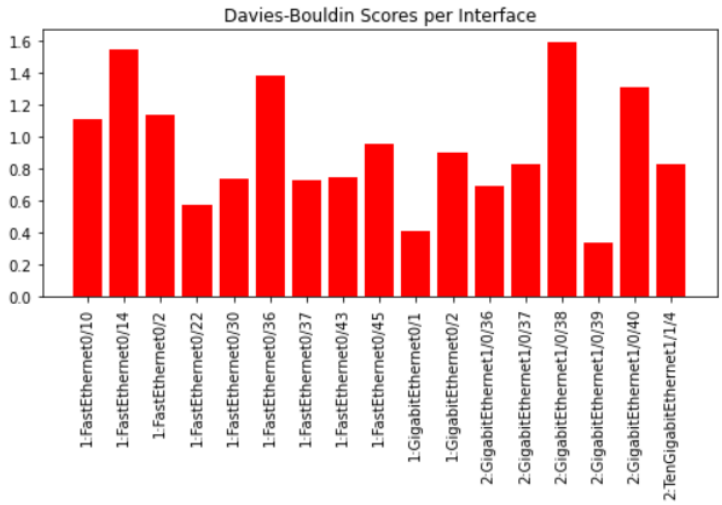
מדדים אלה עוזרים להעריך את הביצועים של מודל זיהוי החריגות של iForest עבור כל ממשק, ומספקים תובנות לגבי איכות זיהוי החריגות עבור ממשקי רשת שונים.

בשני המדדים, קיבלנו הערכה די גבוהה למודל ה- iForest .

- במדד ה – Silhouette : הערכים הם בין (-1) ל(+1) , לכן ניתן לראות שקיבלנו ערכים יחסית גבוהים עבור כל ממשק.
- במדד ה – Davies-Bouldin : הערכים הם מ-0 עד אינסוף, ומצפים לקבל ערכים נמוכים, לכן ניתן לראות שקיבלנו הערכה טובה על המודל, בעוד הערכים שהתקבלו נעים בין 0 ל- 1.6 .

אולם, יש לשים לב, כי בעוד שציון Silhouette גבוה יכול להוות אינדיקטור חיובי לזיהוי חריגות יעיל, לא כל החריגות ניתנות להפרדה בקלות מהנתונים הרגילים, במיוחד במקרים בהם חריגות תלויות הקשר או כרוכות בסטיות עדינות מהתנהגות רגילה.



## שיפורים להמשך

- זיהוי נק' חשודה על פני זמן :
המערכת תזהה Broadcasts חריגים מכמה ממשקים במקביל על פני זמן, ואם יש נקודה בזמן שמסומנת כאנומליה בכל אחד מהממשקים באופן גורף – סימון כנק' חשודה וניתן לבדוק ולחקור עם מנהלי הרשת האם קרה אירוע חריג שבעקבותיו עלתה חריגות בכל הממשקים.

- שימוש בפיצ'רים הנוספים שנאספו במסד הנתונים בשלב הראשוני, יכולים לתרום לזיהוי אנומליות בצורה טובה יותר. מאחר ויש קשר ביניהם. להלן הפיצ'רים :

Output_queue(size/max)

Packets_input

Bytes

Buffer

Multicast

- MAC Finder :
שליפת נתונים מכתובת ה Mac :
ביצוע פעולת חיפוש בכל הממשקים והמתגים של כתובת MAC
התוצאה תהיה המתג והממשק בו נמצאת כתובת הטקסט.
הפקודה לשליפת הנתונים בכל המתג היא : show mac address
ממשק שהוא trunck אומר שהוא מצביע על המתג הבא לחיפוש, לכן יש להחזיק רשימה שמפרטת עבור כל מתג את המתגים הבאים שמחוברים אליו.

# מדריך שימוש

**הסבר התחברות למתגים לצורך הוצאת נתונים בצורה ידנית ( לפרויקט עתידי) :**

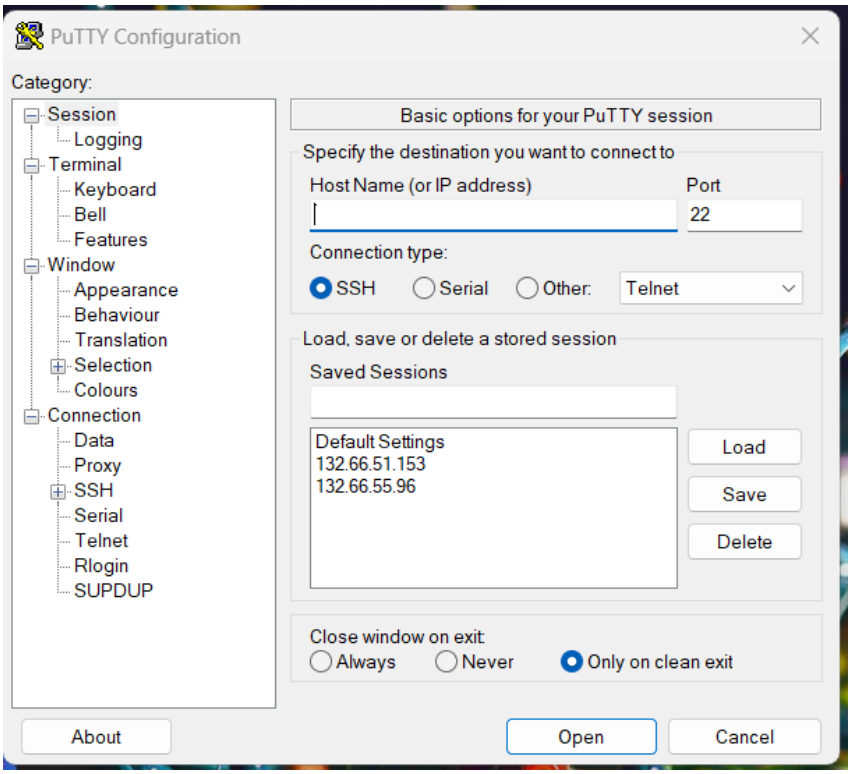
**ראשית, בחיבור מרחוק, יש להתחבר קודם לרשת האוניברסיטה ע"י אפליקציית GlobalProtect**

**שנית, באמצעות PuTTY מתחברים למחשב ולמתגים:**

המתגים שאנו מתחברים אליהם הם :

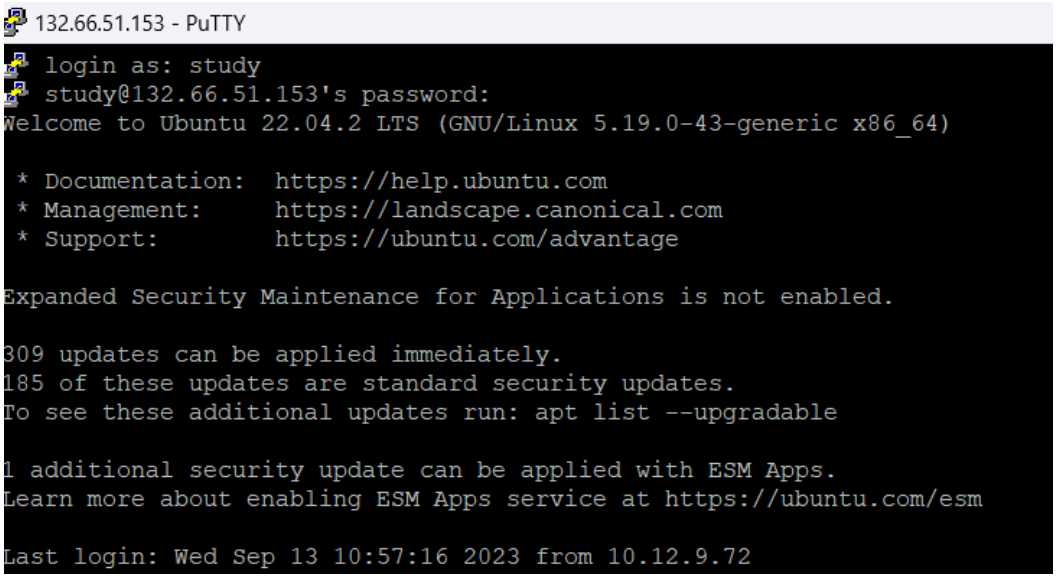
1. 132.66.55.96
2. 132.66.55.97

תחת פורט SSH 22




מתוך מחשב : 132.66.51.153 – שם נמצאת ספריית ANSIBLE עם PLAYBOOKS, הסקריפטים והדו"חות שיצרנו.

שם משתמש : study , סיסמה : Study



הקוד לאיסוף הנתונים נמצא בריצה תמידית, על מנת שיתקבל מערך נתונים גדול יותר לעיבוד בפרויקט עתידי.

שלב ניתוח הנתונים והאלגוריתמים נמצא במחברת של Jupyter Notebook

שניתן להתחבר אליה דרך סביבות עבודה שונות, כגון : VSCode ,

קישור למחברת המעודכנת נמצא בפרויקט בגיט האב. (כלל הקישורים בעמוד הבא)

# קישורים

**לינק לפרויקט שלנו בגיט – האב :**

TAU_Anomalies_Detection_Project_GitHub

**לינק ללמידה של סביבת ה - Ansible : תיאוריה, מבנה ה- playbook וכו' :**

https://docs.ansible.com/

**לינק לפוסטר הפרויקט :**

https://www.canva.com/design/DAF0pO0QbxY/C_l-rU1krMP_RR8hXFUVQg/edit?utm_content=DAF0pO0QbxY&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton