

קורס שיטות לגילוי התקפות סייבר - סיכום**תוכן**

1	קורס שיטות לגילוי התקפות סייבר - סיכום
3	שיעור 1
3	הקדמה
3	אתגרי ביטחון מידע
3	סוגי פתרונות
4	דגשים
6	שיעור 2
6	תהליך הלמידה
7	מודלים בסיסיים
8	Decision tree
9	ANN – artificial neural network
10	שימוש בפונקציית kernel
10	Ensamble learning
10	clustering
11	שיעור 3
11	Anomaly detection
12	אלגוריתמים למציאת אנומליות
13	מעבדה –התנסות hands on
14	למידת מכונה
16	מטריקות דיוק
17	שיעור 4
17	נושא
17	נושא
17	נושא
17	שיעור 5
17	נושא
17	נושא
17	נושא
17	שיעור 6
17	נושא
17	נושא
17	נושא
17	שיעור 7

17	נושא
17	נושא
17	נושא
18	שיעור 8
18	נושא
18	נושא
18	נושא
18	שיעור 9
18	נושא
18	נושא
18	נושא
18	שיעור 10
18	נושא
18	נושא
18	נושא
18	קריאה מומלצת

שיעור 1**הקדמה**

1. הקורס הוא קורס מחקרי, ולא תכנותי פשוט, ולכן דורש השקעה ומוטיבציה לחקור ולהעמיק בעניין. הקורס נועד להיות קורס ראשוני בכל הקשור למערכות למידה ולכן לא מצופה ידע בתחום הסייבר או ידע בספריות מיוחדות. אנו נתנסה בתרגילים hands-on מהבסיס. עולם מערכות הלמידה הוא רחב מאוד, ובקורס נתנסה בצד המעשי, לא התיאורטי, של חלק ממנו, ולכן נדרש מהסטודנטים להתנסות ולהרחיב אופקים לבד, ולהתנסות ולחקור מעבר, לכן המלצתי היא שלא להשתמש ב-ChatGPT כי כאמור מטרת הקורס היא לא רק לסמן וי על ביצוע משימה תכנותית.
2. מעבר לכך שנלמד מערכות למידה חשוב שנבין גם את המגבלות שלהן. התרגלנו לכך שיש למשל את ChatGPT או כל מערכת למידה אחרת שהיא מין נוסחת קסם שאתה זורק אליה משהו והוא עושה הכל. בפועל מערכות למידה הן לא מושלמות, וחלק מהפרויקטים שנעשה זה ללמוד איך לעקוף את מערכות הלמידה האלו.
3. כל דבר הוא ידע שניתן לנצל אותו כדי להגן/ לתקוף מערכות.

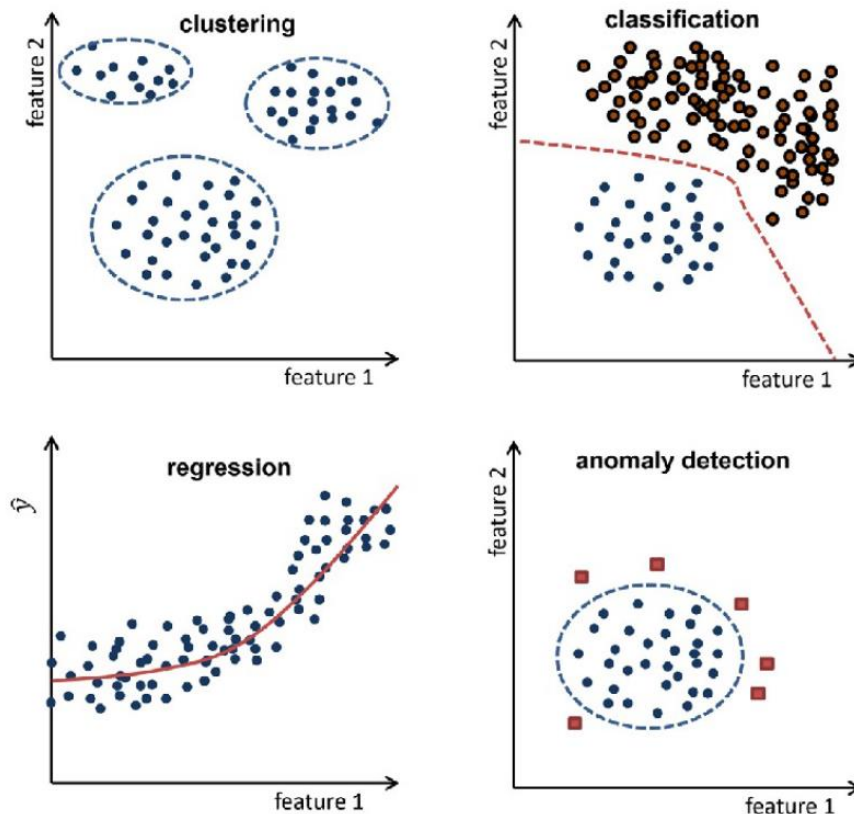
אתגרי ביטחון מידע

4. כשאנו מסתכלים על רשת האינטרנט יש לנו מספר אלמנטים ממנה היא מורכבת. יש את מערכת הדאטה של הארגון, שרתים, משתמשים, לקוחות, שותפים, עובדים וכו'. כל עובד/ שותף מתוך הארגון המחובר החוצה מהווה סיכון לארגון, שדרכו ניתן לחדור למערכות. 92% מהתקפות על ארגונים הם התקפות פשינג. הונאה על ידי דבר שנראה תמים: "החבילה שלך הגיעה, יש לשלם על המכס, אנא הכנס פרטיי אשראי...". ההתקפות האלו נראות תמימות לחלוטין. עובד יכול לקבל מייל תמים, של חוזה למשל, המבקש לסרוק ברקוד על מנת לראות את החוזה, ושם יש מקום להכניס את השם והסיסמה של הארגון/ חשבון הגוגל שלך, ולמעשה גרמת לפריצה.
5. כיום כמעט כל דבר מחובר לרשת. יש את המושג IOT – internet of things, שהוא למעשה מכשירי החשמל המחוברים לאינטרנט, כגון מצלמות אבטחה, דוד חכם, מצלמות בקודן של בית וכו'. והאמצעים האלו חשופים לפריצה גם כן, כך שניתן להיכנס דרכם ולראות למשל את הבית שלנו בלי שנדע בכך. למעשה כל ממשק שיש לנו עם גורם אחר זו סכנה של חדירת גורם זר שעלול לפגוע בנו.
6. ישנן עוד אין ספור בעיות וסכנות, למשל data leakage, זליגת מידע, למשל פרויקט סודי שעובד רוצה לעבוד עליו בבית אז הוא מעביר אותו לדיסק און קי וחושף את המידע בדרך זו או אחרת. או סכנות אחרות על ידי משתמשים פיקטיביים ברשתות חברתיות וכו'. הבעיה המרכזית היא שאין דרך שפותרת את כל הבעיות. אך מה שאנו עושים בעולם מערכות הלמידה זה לנסות לפשט את דרך האבטחה ולעלות על הסכנות בקלות ויעילות.

סוגי פתרונות

7. ניתן לחלק את סוגי הפתרונות בעולם מערכות הלמידה ל4 סוגים:
 - א. זיהוי אנומליות – זיהוי של מידע חריג שלא קשור, זיהוי של נקודות רחוקות ושונות מהנורמה, וננפה אותן, ובכך נבין מה המאפיין של דבר מסוים ומה כבר מוגדר חריג.
 - ב. קלסיפיקציה (סיווג למחלקה) – זו הבעיה הכי נפוצה במערכות למידה, והיא חלק מנושא שנקרא supervised learning, למידה עם תיוגים, שמטרתה לבנות מסווג המקבל נתונים עם

- תיוג, ולומד את המאפיינים. למשל גובה משקל ומידת נעליים, ולייבל האם זה גבר או אישה. ואז בהינתן נתונים לא מתויגים יחליט האם זה גבר או אישה.
- ג. גרסיה (חיזוי מספרי) – שיטה שמתבססת גם על למידה עם תיוגים, בו ננסה לחזות נתון מספרי מסוים. למשל מחירי בתים באריאל לאורך השנים, לחזות מה יהיו מחירי הדירות בשנה הבאה. או למשל מחירים של מניה מסוימת – חיזוי מה יהיה מחיר המניה מחר.
- ד. קלאסטרנינג (חלוקה לאשכולות) – ניתן לומר שזה סוג הלמידה הכי קשה, בו למעשה אנו מקבלים דאטה שלא בהכרח קשור באופן ישיר לדבר שאני מעוניין בו, ועל פיו לבצע חלוקה. למשל בהינתן הגילאים והציון פסיכומטרי של הסטודנטים בכיתה – להצליח לחלק מי במסלול מדעי המחשב רגיל, מי מדעי נתונים ומי סייבר. זו למעשה בעולם האמיתי חלק מנושא שנקראת unsupervised learning, למידה ללא תיוגים, ללא ידיעה מה התיוג של סטודנט מסוים. זהו נסיון להסיק מסקנות מתוך ההנחה שדברים דומים נמצאים אחד ליד השני, כלומר הם בעלי נתונים דומים.



דגשים








8. עולם data science הוא עולם של המון דאטה, ונדרש ללמוד שיטות איך להוציא את המידע הרלוונטי מתוך כמות המידע העצומה שיש. כמו כן נדרש לתת את הדעת על מצבים בהם יש דאטה חסר, או דאטה שלא מתויג עד הסוף, או שחלק מהמידע חסר בחלק מהשורות, צריך להבין איך משלימים את החוסרים בדאטה.
9. כאשר נפתח מודל נצטרך להתחשב בהמון פרמטרים – האם הנתונים באים בreal time או שהם נתונים יבשים, offline. או למשל בעיה שנקראת concept drift, בו קרה אירוע מסוים ששינה

לגמרי את כל האופן או כל הנחות היסוד שהתבססנו עליהם בבניית המודל, כלומר נדרש להבין מתי המודל כבר לא מייצג נאמנה את המציאות.

- בעיה אחרת – דאטה לא מאוזן, למשל דאטה המכיל 3 קלאסים, אבל קלאס אחד מהווה 80 אחוז מהדאטה, למשל מידע על סטודנטים, אבל ש80% מתוכם זה סטודנטים לעבודה סוציאלית, ונרצה לחזות משהו עליהם. מערכת הלמידה תרצה להצליח כמה שיותר, והיא תצליח אם היא תיתן דגש כביכול על הקלאס הכי גדול, כי שם יש אחוז גבוה יותר להצליח. ולעיתים זה עלול להרוס הכל – רוב המיילים הם לא פשינג, ועל זה המערכת לומדת, אבל דווקא בחלק היחסי הפחות מיוצג – אלו שהן באמת התקפות פשינג – הוא פחות למד חלק זה, ולכן חשוף לטעויות בו יותר.
10. איך מודדים הצלחה? חייב להגדיר ולייצר מדדים איך אני מודד מערכת/ מודל כטוב ומוצלח.
11. איך נגן על הארגון? דבר ראשון מחקר – להבין מה יש בארגון, כמה משתמשים יש? כמה מתחברים לשרת? מול מי עובדים? וכו'. לאחר מכן ננסה לחקור את רמת הסיכון של כל שירות ושירות, במי אני שולט ברמת האבטחה שלו ובמי פחות? איך אני שומר על שרתים? השלב הבא הוא לבצע תיעדוף, התעסקות באבטחת מידע הוא עסק יקר, ובהתאם לכך להבין איפה נדרש להשקיע את עיקר המאמץ. ולבסוף להטמיע את מימד האבטחה בתוך הארגון.
12. נוכל להגדיר מערכת לומדת על ידי הסכימה הנ"ל:

שיפור בביצוע במשימה T, בביצועים הנמדדים על ידי P, בהסתמך על הניסיון הנלמד E. לדוגמא:

אנו רוצים לשפר את T – זיהוי מיילים שהם ספאם, המטריקה P היא אחוז הספאמים שאכן

Input Attributes				Target Attribute
Number of new Recipients	Email Length (K)	Country (IP)	Customer Type	Email Type
 0	2	Germany	Gold	Ham
 1	4	Germany	Silver	Ham
 5	2	Nigeria	Bronze	Spam
 2	4	Russia	Bronze	Spam
 3	4	Germany	Bronze	Ham
 0	1	USA	Silver	Ham
 4	2	USA	Silver	Spam

Instances

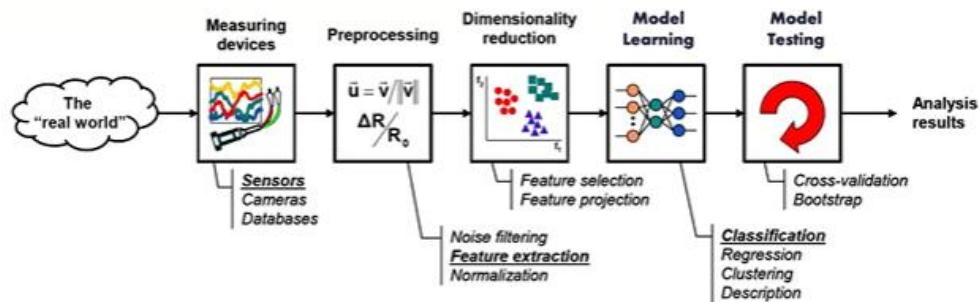
זיהיתי, והE,
הניסיון – זה
הדאטה שט
שקיבלתי של
מיילים.
הדאטה שט
הזה למשל
מכיל את
העמודות –
פיצ'רים –
שזה התכונות
של המייל,
ובסוף לייבל –

תיוג – האם זה מייל ספאם או לא. זוהי דוגמא לבעיה מסוג supervised.

שיעור 2

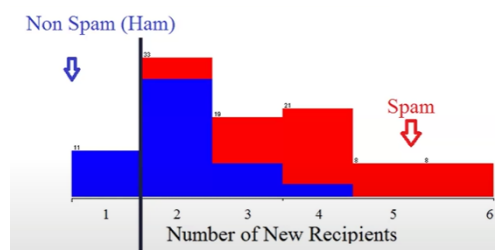
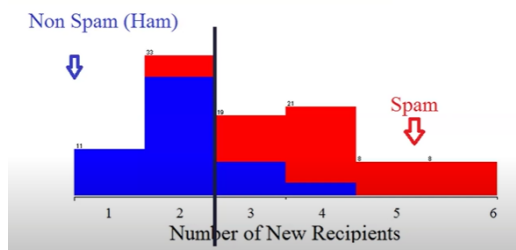
תהליך הלמידה

1. העולם האמיתי והנתונים בו אינם ניתנים לפיענוח ישיר במחשב, על מנת להגיע למסקנות וניתוח



של נתונים נדרשים מספר שלבים :

- א. מדידות והעברה לנתוני מחשב, למשל תמונה המומרת לביטים, נתונים מספריים ומדידות.
- ב. Pre-processing – לרוב לא כל הנתונים על אובייקט מסוים מספקות מידע משמעותי, ולא רק שלא מקדם להסקת מסקנות אלא אף מפריע ומעמיס על המודל ללמוד עוד מידע ולאחסן עוד זיכרון. לשם כל נדרש ניתוח מקדים – למשל נורמליזציה – העברת הנתונים המספריים להיות תחת אותה סקאלה (למשל שחתך הגילאים מתפרס על 20 עד 80, לעומת מספר ילדים שינוע בין 1 לבין מספר חד ספרתי אחר). או למשל ניקוי רעש, של נתונים שנכנסו "בטעות". ניתוח מקדים נוסף שנדרש הוא לבדוק שהדאטה מאוזן, או שלא - imbalanced dataset (למשל). ואם כן אז צריך לאזן אותו, אפילו בצורה מלאכותית
- ג. איך אני יודע אם פיצ'ר הוא טוב? נדרש לבדוק, לחקור – data exploration, למשל על ידי הסתכלות על פיצ'ר (עמודה, פרמטר) אחד – ונראה אם ניתן ללמוד רק ממנו משהו. אולי על העמודה על המיילים new recipients – על פה שפחות מ2 זה ספאם? לפי הדוגמה בתמונה זה נכון אבל לא מדויק. אז נגדיר פחות מ3? זה שיפר אבל עדיין לא מדויק. לכן צריך עוד פיצ'רים.



מצד שני זה לא נכון להוסיף עוד ועוד פיצ'רים, נדרש לעבוד גם על צמצום הבעיה לממדים נמוכים יותר כך שיהיה קל יותר להתמודד איתה, למשל על ידי feature selection - הבנה אילו נתונים רלוונטיים ואילו לא. (אין סיבה להתחשב בהמון פרמטרים כשניתן לבצע זאת עם הרבה פחות מא_מץ – עמודות).

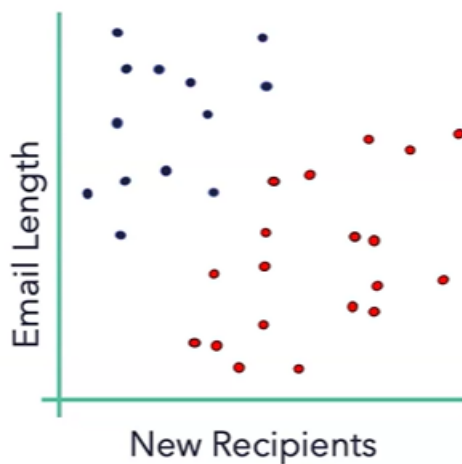
- ד. בחירת המודל - מודלים יכולים טובים מאוד עבור דאטה סט אחד, וגרועים מאוד על דאטה סט אחר. לכן יש פה 2 אלמנטים חשובים – הראשון, זה שלא ניתן לנתק בין המודל לבין הדאטה סט עליו הוא עובד, זה המודל שפותר את הנתונים הללו. והשני, זה שנדרש כחלק מהמודל לנתח את הנתונים ולייצר פיצ'רים חדשים בהתאם לצורך שלי, וכאלו שהמודל יתמודד איתן טוב יותר.

ה. לימוד של המודל על מידע מבחן שהמודל לא ראה ולא התאמן עליו, ולבסוף אוולואציה, מימוש ודירוג – scoring – של מדד ההצלחה (יכול להימדד על פי כמה פרמטרים – accuracy, precision, recall ...)

מודלים בסיסיים

2. Linear classifier – יש לי נתונים הפרוסים על מרחב צירים. המודל הנ"ל מבצע קלסיפיקציה

(סיווג) על ידי חלוקה של הנתונים על ידי



משוואת קו ישר שמחלקת את הדאטה סט ל2,

ובהינתן נקודה חדשה אני בודק באיזה צד הוא

ולפיו מסווג. הדבר נובע מתוך הנחה שדברים

דומים הם בעלי מאפיינים דומים. למשל דירות

עם כמות חדרים גדולה יתומחרו גבוה יותר

מאשר דירות עם מספר מועט של חדרים.

3. השיפור של המודל (האימון) באופן רעיוני הוא

על ידי הזזת והטיית הקו עד למצב האידיאלי

שמחלק בצורה הכי תואמת למציאות.

4. מה הבעיה בשיטה הזו? נניח והיינו מצליחים

לבצע את ההפרדה המושלמת. אבל המודל הזה

נקבע לפי הנתונים בדאטה סט שקיבל. יכולים להיות מצבים ונתונים שהוא לא ראה ולכן לא יסווג

בצורה לא נכונה. בין אם כי הדאטה סט מצומצם, או במידה ויש data drift – השתנות של המידע,

ואז צריך לבנות מודל חדש, או לאמן אותו מחדש.

5. דרך טובה לבדוק את טיב המודל היא על ידי confusion

matrix – מה שסיווגתי מול מה שבפועל בכל אחת

מהקטגוריות. כמובן שהאלכסון הוא האידיאלי שיכיל את

הערכים הגבוהים ביותר למול מה שלא באלכסון שאלו

הטעויות לכאן או לכאן. עם זאת מודל שצודק ב100% הוא

לא בהכרח מודל טוב. או שהבעיה קלה מדי, או שהשתמשת

בלייבלים בתור הפיצ'רים, זה כמו לקבל מבחן כשכבר קיבלת את התשובות מראש.

6. הסיבה שקשה מאוד מאוד להגיע לדיוק של 100% היא שלא בהכרח שהדאטה ניתן לחלוקה על

ידי קו אחד למשל (אם מדברים על המודל של סיווג לינארי), לכן אולי נדרש להוסיף עוד קווים, או

אפילו בעולם אידיאלי אני אשרטט קו שיעבור ויתפתל בדיוק בהפרדה מושלמת. אבל מה הבעיה

בה? כמו שאמרנו - שהיא מושלמת רק על הדאטה שאני יודע. לכן נדרש להשתמש במודל אחר או

קונפיגורציה אחרת.

7. נראה דוגמא לאיך לממש בקוד:

הכנסנו בציר הX זוגות סדורים

של ערכי נקודות בצירים x,y.

ציר הY הוא הלייבל, התיג

האם הנקודה הזו היא 1 או

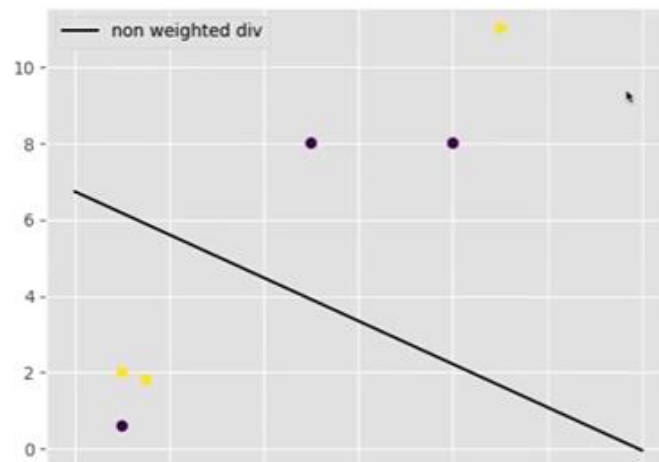
שהיא 0.

```
1 X = np.array([[1,2], [5,8], [1.5,1.8], [8,8], [1,0.6], [9,11]])
2 y = [0,1,0,1,0,1]
3 #y = [1,0,1,0,0,1]
4
5 clf = svm.SVC(kernel='linear', C = 1.0)
6 clf.fit(X,y)
7 t1 = [[0.58,0.76]]
8 print(clf.predict(t1)) #will predict 0
9 print(clf.predict([[10.58,10.76]])) #predict 1
10
[1]
[0]
```

מייצרים אובייקט בשם `clf` שיוגדר כמסווג לינארי, ומאמנים אותו (`fit`) על ערכי X, Y . לבסוף נוסיף נקודה חדשה t_1 , ונבצע חיזוי על הנקודה לפי המודל ונדפיס אותה, או שנוכל לבצע חיזוי ישיר על נקודה מסוימת (שורה 9).
ניתן להדפיס את הנקודות ולשרטט את הקו הלינארי של המודל המאומן ולראות את הטעויות.

```
1 w = clf.coef_[0]
2 print(w)
3 a = -w[0] / w[1]
4 xx = np.linspace(0,12)
5 yy = a * xx - clf.intercept_[0] / w[1]
6 h0 = plt.plot(xx, yy, 'k-', label="non weighted div")
7 plt.scatter(X[:, 0], X[:, 1], c = y)
8 plt.legend()
9 plt.show()
10
```

[0.1380943 0.24462418]

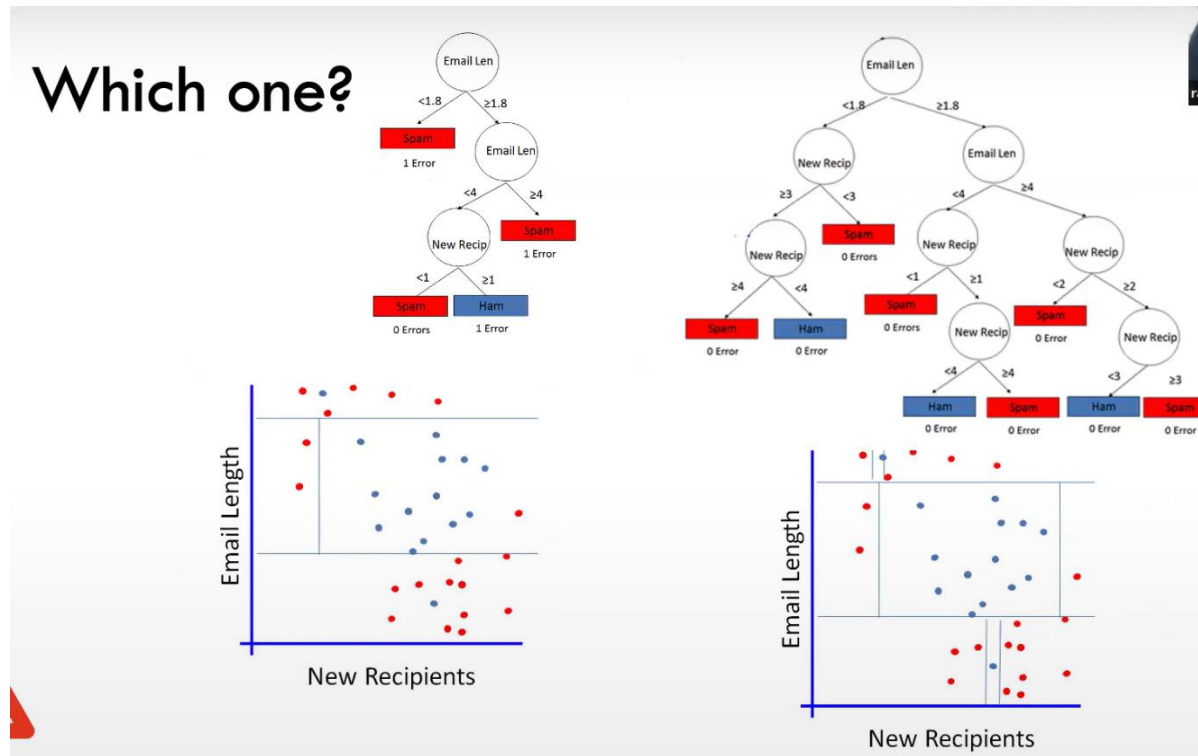


8. מודל סיווג אחר הוא על ידי `k nearest neighbors`, כלומר בהינתן הנקודה נבחר את הסיווג לפי הסיווג של הרוב מבין k השכנים הקרובים ביותר לנקודה. שוב מתוך ההנחה שאלמנטים ששייכים לאותו סוג – הערכים שלהם דומים. בהינתן נקודה אקראית – נמקם אותה על הצירים, ונוכל לחזות מה הסיווג שלה לפי פונקציית המרחק של k השכנים הכי קרובים. כדי לעבוד עם האלגוריתם הזה נדרש לעבוד עם k אי זוגי.

Decision tree

9. מערכות למידה כאמור עובדות לפי מטריקה של טעויות. כמה שפחות טעויות יותר טוב. דרך אחרת לפתור בעיות מורכבות זה על ידי עץ החלטה. הרעיון בגדול – יצירה אוטומטית של המון `if` ו-`else`. הוא מייצר קו, או מסלול, לפי הפרמטרים, עד שיווצר מסלול שיביא לתוצאה הכי טובה, לפיו המודל יחליט כל פעם לפי מה לחתוך – לסנן, למשל בהתחלה את כל המיילים עם כמות מילים של יותר מ-100, כבר אני מוריד כמות משמעותית של מידע להתעסק בו, ולאחר מכן לפי נתון אחר וכו'. מבצעים זאת בצורה איטרטיבית עד שלבסוף יוצא שנפרש לנו עץ.

10. גם פה אגב יש מקום לשאול עד כמה מעמיקים ומסבכים את העץ – המודל – שיהיה יותר מסובך אבל יותר מדויק אך מצד שני גם יותר ממוקד רק למה שהתאמן עליו? ועלול לא לייצג את המידע אמיתי בטבע (`over fitting`) – המודל כל כך הדוק על הדאטה עד שיטעה משמעותית בכל מה שלא ראה) או לבחור מודל פשוט יותר אבל כזה שמפספס וטועה בחלק משמעותי מהמידע שקיבל (`under fitting`) – היינו רפוסים מידי באופן הלימוד כך שלא ניתן לומר שלמדנו)?



ANN – artificial neural network

11. זהו אלגוריתם, או יותר נכון צורת הסתכלות, שלוקח השראה ממערכות הנוירונים בגוף, נוירונים

במוח מקבלים פקודות, מתחילים חשמליים, ואם הוא עובר סף מסוים הוא מעביר את המתח הלאה, ואם המתח נמוך מידי מהסף – הנוירון לא "יורה" החוצה כלום (זרם חשמלי), ואם הוא עובר סף מסוים הוא מעביר את הזרם (מידע) הלאה. כך דברים נקשרים במוח, שהזרם החשמלי

ביניהם חזק. הנוירון באלגוריתם מקבל

את הקלט, סוכם אותם, מפעיל

פונקציית אקטיבציה, והיא זו שקובעת

– אם עבר את הסף – מעביר ערך

מסוים לכל הנוירונים אליהם הוא

מחובר. זה תהליך שנקרא forward

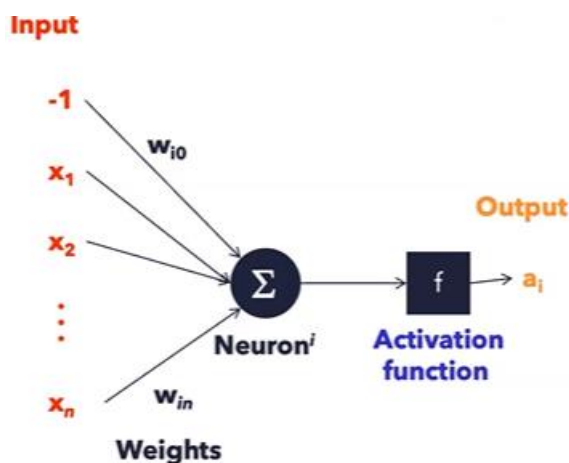
propagation – התפשטות קדימה,

ולאחר מכן נבצע back propagation –

היזון אחורה לפי הטעויות ולפי מידע

חדש שלומד ולמעשה מכוון מחדש את

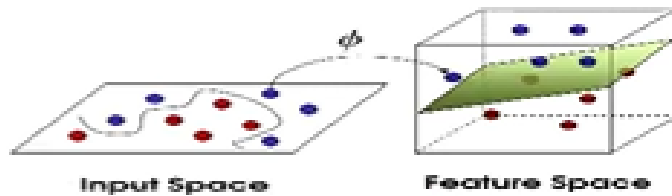
המשקלים שאני נותן למאפיינים



בקלט, כלומר החשיבות שאני נותן להם. ניתן להוסיף כמה שכבות, וכל שכבה במודל משנה את האינפוט ובעצם מסתכלת על מימד או על כמות נתונים שונה כך שלמעשה לומדת את המידע יותר לעומק.

שימוש בפונקציית kernel

12. מה עושים אם הבעיה לא ניתנת להפרדה לינארית כזו או אחרת? הייתי רוצה לשנות את הבעיה לבעיה אחרת, כך שתבטא את אותה בעיה עם ערכים אחרים. הרעיון של kernel זה להעביר פונקציה מסויימת על כל האיברים בדאטה. למשל כשמדובר בתמונה – שכל פיקסל יכול ערך

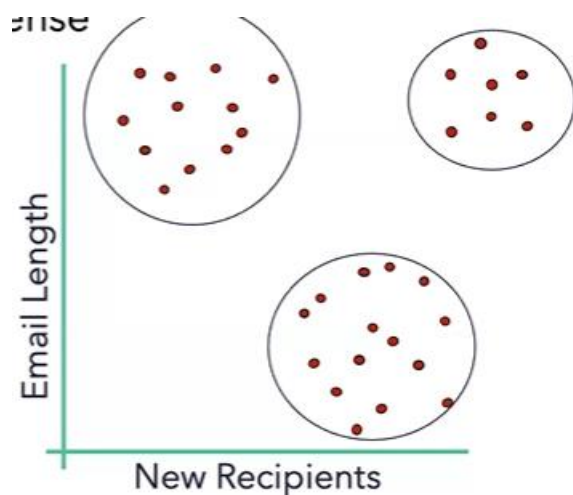


חדש, למשל - ממוצע כל הפיקסלים הצמודים לו, או בחישוב – להכפיל את הכל ב*i* ולהציג כמספרים מרוכבים וכד'. הערכים השונים מאפשרים לי מרחב חדש של

הדאטה שבו אני יכול להפריד את הדאטה בצורה ברורה יותר.

Ensamble learning

13. זהו לא אלגוריתם אחד אלא קונספט של שימוש במספר מודלים שונים, על אותו דאטה, כל אחד עובד בשיטה אחרת, וכל פעם לוקחים את זה שהצליח יותר. כמו למשל 3 תלמידים שפותרים תרגילים יחד, הם פותרים תרגיל מסוים, כל אחד פותר בדרך והשיטה שלו, ואז הם בודקים מי פתר הכי מהר והכי נכון, ומעכשיו ישתמשו בשיטה שלו.

clustering

14. זהו אלגוריתם, או יותר קונספט תחת unsupervised learning, בו הדאטה לא מתוייג. למשל פרטים על מיילים בלי לדעת מהם תקין ומה ספאם. הקונספט הוא למיין אותם לאשכולות של נתונים הדומים אחד לשני, בלי בהכרח לדעת מהי הקבוצה. כך שלאחר מכן נחקור ונבין מה המשמעות של כל קבוצה ואיך ניתן לסווג את אותו אשכול. קלאסטרינג הוא טוב בעיקר לזיהוי של אנומליות, של נקודות המרוחקות מאוד מכל יתר הדאטה למשל ולא מעידות באמת, ולכן נוכל אולי גם להסיר אותן מאחר והן מסיטות את הדאטה לכיוון אחר.

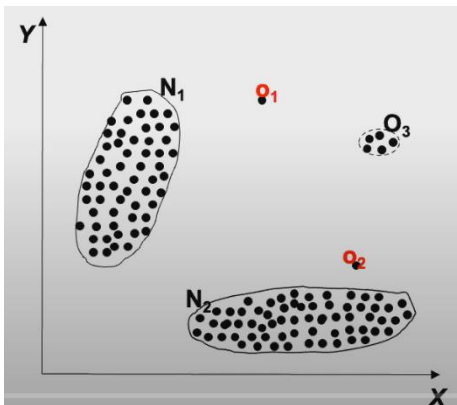
שיעור 3**Anomaly detection**

1. לא תמיד הדאטה שלנו יהיה מסווג, (supervised learning), וכדי להתמודד עם מצבים כאלו יש כמה שיטות, למשל קלאטרנינג, שנגענו בו מעט, וזיהוי אנומליות. אנומליה פירושה זרות, סטייה מן הנורמלי והמקובל, ובהיבטי ניתוח נתונים זיהוי אנומליות (או גם outlier detection) הוא דבר חשוב. הבעיה היא שבהגדרתה יכול להיות שהאנומליה דווקא היא זו שתעיד לי על בעיה שאני נדרש לתפוס, למשל כאשר משהו פרץ לי לרשת וכו'. ולחילופין אנומליה יכולה לא להעיד כביכול באמת, למשל בניתוח של פקקים ועומסי תנועה, יכול להיות שמופיע רחוב שהיה בו עומס תנועה חריג - אבל הדבר נבע מתאונה שהייתה שם, וב-99.9% מהזמן הרחוב שקט לחלוטין.
2. עם זאת – אנו נשתמש באנומליות כי אין לנו דרך אחרת להבין את ההתקפה, אין לנו שום דרך לדעת ולצפות איך התוקף ינסה להתקיף אותנו, ולכן גם בהינתן שהתרחש אירוע שנראה לא קשור ואנומלי לנתונים – אולי הוא כן חלק מהמתקפה.
3. כשמדברים על זיהוי התערבות (Intrusion detection) אנו למעשה מדברים על תהליך שמבצע מוניטורינג על מחשב או על רשת במטרה לעקוף את האבטחה. דוגמא ליכולת שמשתמשים בה הרבה פעמים DDOS – רשת של מחשבים שגולשת ופותחת המון בקשות אינטרנט לשרת מסוים. בהתחלה השרת חושב שמדובר במשתמשים רגילים, אבל הם בעצם מגיעים כדי להפריע לשרת לבצע את תפקידו. איך ניתן לזהות התקפות כאלו? אלגוריתם לזיהוי אנומליות, לזיהוי לפי כמות ולפי אזור וכו' מתי ככל הנראה מתבצעת התקפת DDOS ולמעשה להשבית את האפשרות לקבל את הבקשות
4. באופן דומה חברות אשראי מצליחות לעלות על כך שמישהו גנב את כרטיס האשראי של לקוח, הן יודעות את הרגלי הקניה או את תחומי הקניה של הלקוח וכשמתבצעות קניות בתחומים/מחירים/מדינות שהלקוח לא נוהג לקנות בהן – הן מזהות שיש פה אנומליה, משהו זר.
5. אנו מחלקים את האנומליות ל-2 סוגים: univariate (חד משתנית) ו-multivariate (רב משתנית). univariate - פיצ'ר או אירועים בודדים או פיצ'ר של זמן מול אירוע. למשל כמות מידע בשנייה לאורך היום ועל זה אנו מחפשים אנומליות. באנומליה חד משתנית ניתן למשל לעלות על אנומליה על ידי סטיית תקן, מונח מקובל הוא 3σ , כאשר סיגמא מסמלת את סטיית התקן, אנומליה היא חריגה מ-3 יחידות סטיית התקן.
- ב-multivariate הבעיה היא יותר קשה ודורשת אלגוריתם הרבה יותר מורכב, וזה בזיהוי אנומליות לא באחד מהם אלא ביחסים ביניהם, שלשם כך צריך להבין את הקורלציה ביניהם, ואז כשאחד מהערכים עולה מאוד למול אחר שיוורד בניגוד למה שהייתי מצפה מהקורלציה לדרוש – יכולה להתריע לי שיש פה אנומליה.
6. למדנו שבקלסיפיקציה אנו נרצה דאטה סט שהוא מאוזן. בזיהוי אנומליות הנחת העבודה היא שאין אנומליות. לכן למשל בפיתוח אלגוריתם לזיהוי אנומליות ברשת היא שאין התקפות בדאטה, כי אנו למעשה נקבל את האנומליה כמידע תקין.
7. טעות בזיהוי יכולה להיות ל-2 כיוונים: false positive – שסיווגתי בתור "כן" (למשל כן אנומליה) אבל זו טעות (כלומר לא אנומליה), וfalse negative – שסיווגתי בתור "לא" (למשל שלא אנומליה) אבל זו טעות (כלומר כן אנומליה).
- כשבונים מודל חייבים להחליט על אילו מבין האופציות האלו אנו מוכנים פחות לוותר. למשל באלגוריתם של נטפליקס שמציע סרט שאולי תאהב – הטעות היא לא נוראית (זה מקרה של

false positive – שהמערכת סיווגה כ-כן, תוכן שאני עשוי לאהוב, אך אני לא אוהב אותו וזו טעות). אך כאשר מדברים על התקפות סייבר – אני יכול לסבור מצבים של false positive, למשל של סינון תוכן מהימן שסיווג בטעות כחשוד, למרות שגם מהם אנסה להימנע. לעומת זאת אני ממש לא יכול להשלים עם טעות של false negative – שסיווג בתור לא חשוד, אבל זו טעות, ולמעשה פרצו לי למחשב.

אלגוריתמים למציאת אנומליות

8. Isolation forest algorithm



כמו שהסברנו לא תמיד זה קל למצוא outliers, יכולים להיות מופעים רבים של הדאטה המכונסים במספר מקומות בודדים, ואז קל לזהות את החריגים (למשל o1 ו-o2). אבל מה קורה עם הקבוצה של o3? האם נסווג כאנומליה מאחר והיא מרוחקת מאוד מהדאטה המרכזית? או שלא מאחר ויש מספר של מופעים המקובצים יחד וזה יכול להעיד על אשכול (cluster) תקין נוסף?

אחת השיטות היא אלגוריתם Isolation forest,

שמטרתו לזהות אוטליירים. הקונספט

הוא שנקודה שמאוד מרוחקת מנקודות

אחרות – קל להפריד אותה מאחרים.

באופן אקראי נמתח קווים – פעם

המקבילים לציר ה-X ופעם לציר ה-Y,

ונשאף שתוך כמות קטנה יחסית של

צעדים נצליח לייצר נתיב בו הנקודה היא

נקודה יחידה. למשל בתמונה הנ"ל הצלחנו

לבודד את הנקודה המסומנת בכוכב תוך 8

בידודים (קווים). לכן ניתן לסווגה

כאנומליה.

9. DBSCAN

בניגוד לאלגוריתם clustering רגיל, באלגוריתם זה אין הנחות מקדימות על כמות האשכולות.

(למשל knn או מגדירים עבור איזה k לחפש וכו').

אלגוריתם DBSCAN מסתכל על 2 הגדרות – מרחק של

נקודה אחת מנקודה אחרת, וכמה מהנקודות יכולות

להיחשב כמרכז. לכן נגדיר לפי מרחק מסוים כרדיוס, ואז

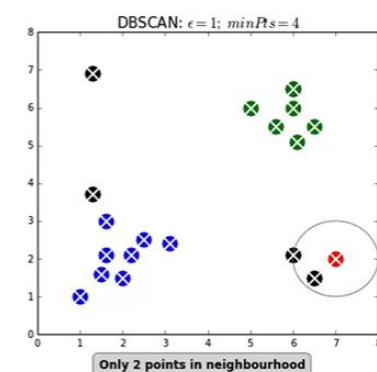
נקודות שלא ישוייכו לאף אשכול יחשבו כאנומליות. הבעיה

באלגוריתם זה הוא שנדרש לחוש / לראות את פריסת

הדאטה קודם כדי להבין אם זה הדאטה מתאים לאלגוריתם

10. ישנם עוד אלגוריתמים בנושא למשל MAD – median

absolute deviation וכו'.



מעבדה – התנסות hands on

11. נטען את הדאטה השמור כקובץ CSV באמצעות ספריית pandas, ונגדיר לו את שמות העמודות.

```
df = pd.read_csv(f_path, names=["record ID", "duration_", "src_bytes", "dst_bytes"], header=None)
```

```
# we could have nan values in the dataset (issue in the data) lets dropna()
df
```

	record ID	duration_	src_bytes	dst_bytes
0	1	0	236	1228
1	2	0	239	486
2	3	0	234	1364
3	4	0	239	1295
4	5	0	181	5450

נשים לב שאין בדאטה לייבלים. ננסה למצוא התקפה בדאטה.

כבר אנו יכולים לשים לב שהעמודה הראשונה היא לא פיצ'ר אלא סתם אינדקס, ולכן אפשר יהיה להוריד את העמודה על ידי `df.drop('record ID')`. אבל לא נעשה זאת כרגע.

A. Handling missing data - detect if we have this case:

```
1 # determin the missing data precentage
2 df.apply(lambda x: sum(x.isna()) / len(df))
3
```

```
record ID    0.0
duration_    0.0
src_bytes    0.0
dst_bytes    0.0
dtype: float64
```

– `df = df.dropna()` נותן היה לרשום NAN, במידה והיו לנו ערכים של NAN.

מחיקת של כל השורות שיש בהם ערך NAN. לחילופין ניתן היה להשלים ערך כלשהו (0) ממוצע העמודה וכו') על ידי פונקציית `fillna`.

```
1 number_range = range(0,200)
2 print(number_range)
```

```
range(0, 200)
```

```
1 subset_loc = df.loc[number_range]
2 subset_loc#.head()
```

	record ID	duration_	src_bytes	dst_bytes
0	1	0	236	1228
1	2	0	239	486
2	3	0	234	1364
3	4	0	239	1295
4	5	0	181	5450
...
195	196	0	264	1356

12. הדאטה שלנו יכול להכיל הרבה

פיצ'רים (פרמטרים, עמודות), יכול להיות מצב שפיצ'ר מסוים בתצפית (שורה) מסוימת לא זמין/ לא קיים – NaN. לכן אני ארצה לראות כמה פיצ'רים NAN לנו. אם כן אנו

13. ניתן גם לבחור ערכי אינדקסים מסוימים, למשל

דוגמה בתמונה – טווח של 200 ונבקש לקחת את כל השורות שהם במיקום (loc) של הטווח. באופן דומה היינו יכולים להכניס טווח בקפיצות של כל כמה שורות למשל וכד'.

14. כמו כן ניתן לסנן חלק מהשורות לפי ערך ספציפי

עם `where`, למשל כל השורות שיש בהם NAN בהם ערך `src_bytes` גדול מ-240.

```
df = df.where(df['src_bytes'] > 240).dropna()
```

לכן נוכל להגדיר מחדש את הדאטה סט שלנו להיות כל השורות לפי התנאי הנ"ל:

```
1 ran_df = df[(df['src_bytes'] > 240) & (df['dst_bytes'] > 1000)]
```

```
1 ran_df.shape
2
```

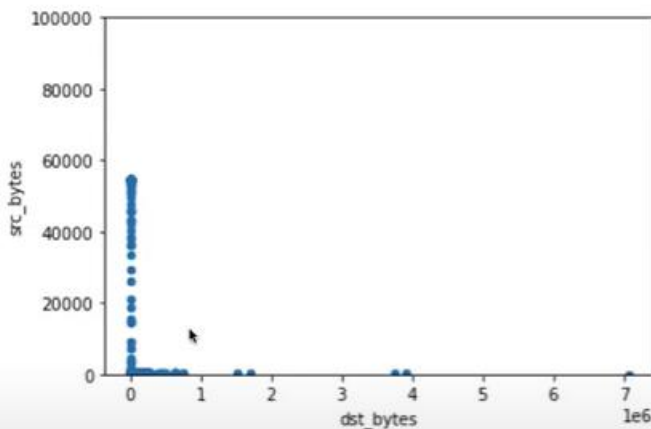
```
(90108, 4)
```

ונרצה לראות מה המימדים של הדאטה סט – 90,108 שורות על 4 עמודות.

```

1 #Relationship with numerical variables
2 var = 'dst_bytes'
3 data = pd.concat([df['src_bytes'], df[var]], axis=1)
4 data.plot.scatter(x=var, y='src_bytes', ylim=(0,100000));

```



```
1 df.corr()
```

	record ID	duration_	src_bytes	dst_bytes
record ID	1.000000	0.027244	0.019209	-0.002970
duration_	0.027244	1.000000	-0.001714	-0.005791
src_bytes	0.019209	-0.001714	1.000000	0.010770
dst_bytes	-0.002970	-0.005791	0.010770	1.000000

15. ניתן להשתמש גם

בספריית matplotlib כדי להציג את הדאטה. למשל אנו רואים שיש פה מספר מופעים המרוחקים מאוד מכל היתר, ואנו יכולים לחשוד בהם כoutliers, כמובן בהתאם למה שהצגנו לפיו.

16. הקורלציה בין הנתונים היא גם

משמעותית ומספקת מידע רב בשלב המחקר המקדים של הדאטה. למשל לפי טבלה זו אנו מבינים שהעמודות כמעט ולא תלויות אחת בשניה- הכל קרוב מאוד ל0.

למידת מכונה

1. אופן העבודה בלמידת מכונה הוא שאנו לוקחים את כל הדאטה, ומחלקים אותו לחלק אימון, חלק של אימות, מעין המבחן עבור עצמי בשלב האימון אבל לא המבחן עצמו, וחלק של המבחן, כלומר דאטה שלא ראה ולא למד באף שלב ואנו בוחנים עד כמה המודל טוב והאם מצליח לעבוד כשורה גם עליו. ניתן להשתמש בספריות של scikit learn שמקצרות את התהליך.

```

1 import numpy as np
2 from sklearn.model_selection import train_test_split
3 from sklearn import datasets
4 from sklearn import svm
5 # Load Dataset
6 X, y = datasets.load_iris(return_X_y=True) # note this is a known dataset with dedicated loader. In your work
7 X.shape, y.shape

```

: ((150, 4), (150,))

Dataset:

Train and Validation are used in the training while Test is left out for model verification



2. לרוב חלק המבחן יהיה כ-10% מהדאטה הכללי, אבל זה משתנה בהתאם לדאטה סט. דגש משמעותי – בעת החלוקה לtrain וtest חשוב לוודא שהדאטה מעורבב, ולא ממויין למשל, כי אז יהיו תצפיות (שורות) רבות שלא התאמן עליהן ולכן גם לא יידע לחזות לגביהן.



3. כלי משמעותי שניתן להשתמש בו הוא שיטה שנקראת cross validation, (או גם k-fold), בו למעשה ניתן להשתמש בחלק האימון בצורה הרבה יותר רחבה, מעין לייצר עוד דאטה – וזה על ידי ביצוע חלוקה של חלק האימון, וכל פעם לקחת חלק אחד (מחלק האימון) ומשתמש בו כvalidation, מעין חלק של test.

```

1 X_train, X_test, y_train, y_test = train_test_split(
2     X, y, test_size=0.2, random_state=0)
3
4 print("Train:", X_train.shape, y_train.shape)
5
6 print("Test", X_test.shape, y_test.shape)
7
8 # Validation can be taken from the Train:
9 # Further split the training set into 75% training and 25% validation
10 # This results in 60% training, 20% validation, and 20% testing of the original dataset
11 X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=42)

```

Train: (120, 4) (120,)
Test (30, 4) (30,)

4. ניקח לדוגמה את הדאטה סט המפורסם של האירוסים (3 סוגי פרחים של אירוס שנמדדים לפי אורכי עלי הכותרת שלהם, ונדרש לסווג כל אחד לסוג הספציפי שלו). אופן המימוש כפי שמתואר בקוד הוא על ידי הפונקציה train_test_split שמחזירה 4 איברים לפי סדר האיברים שפורט לפי סימן ה=. לאחר מכן אנו רוצים לייצר את חלק הולידציה ולכן נחלק גם אותו לפי אותה שיטה אלא שנעביר כאמצעים לחלוקה את x_train ואת y_train.
5. אם כן נכניס את הנתונים למודל הלינארי שלנו, נבצע אימון של המודל – fit, ולאחר מכן חיזוי על חלק המבחן – predict, ונעריך את הציון שלנו – score, ונרצה גם לראות את התוצאות שלנו לפי confusion matrix. נוכל לראות שיש לנו 2 טעויות (מה שלא באלכסון הראשי).

```

1 X_train, X_test, y_train, y_test = train_test_split(
2     X, y, test_size=0.4, random_state=2)
3
4 X_train.shape, y_train.shape
5
6 X_test.shape, y_test.shape
7 '''
8 When evaluating different settings ("hyperparameters") for estimators, such as the C
9 setting that must be manually set for an SVM, there is still a risk of overfitting
10 on the test set because the parameters can be tweaked until the estimator performs optimally.
11 You can optimize "C" with GridSearch.
12 '''
13
14 clf = svm.SVC(kernel='linear', C=1).fit(X_train, y_train)
15 print('the accuracy here is :', clf.score(X_test, y_test))
16 #let's predict the data
17 y_predicted = clf.predict(X_test)
18 # lets review the labels
19 labels = np.unique(y)
20 print("unique labels", labels)
21 # The confusion matrix is:
22 from sklearn.metrics import confusion_matrix
23 confusion_matrix(y_test, y_predicted, labels=labels)

```

the accuracy here is : 0.9666666666666667
unique labels [0 1 2]
array([[23, 0, 0],
 [0, 16, 0],
 [0, 2, 19]])

6. ניתן גם לעשות אל הקרוס ולידציה על ידי פונקציה ולבחור $cv=5$, כלומר חלוקה ל-5 חלקי אימון.

```
1 from sklearn.model_selection import cross_val_score
2 clf = svm.SVC(kernel='linear', C=1, random_state=42)
3 scores = cross_val_score(clf, X, y, cv=5)
4 print("we are getting in scores the result of each iteration. Len:", len(scores), "values:", scores)
5 print("%.2f accuracy with a standard deviation of %.2f" % (scores.mean(), scores.std()))
6 print("The Max set was: ", scores.max())
7 print("NOTE: the result here are different! why? - OVERFITTING")
```

we are getting in scores the result of each iteration. Len: 5 values: [0.96666667 1. 0.96666667 0.96666667 1. 0.98 accuracy with a standard deviation of 0.02
The Max set was: 1.0
NOTE: the result here are different! why? - OVERFITTING
Now let's see how can we use data normalization with this example

7. חשוב לשים לב שלעיתים נדרש לנרמל את המידע:

```
from sklearn import preprocessing
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.4, random_state=0)
print("changed")
#standardScaler normalization
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_transformed = scaler.transform(X_train)
clf = svm.SVC(C=1).fit(X_train_transformed, y_train)
X_test_transformed = scaler.transform(X_test)
print("result score: ", clf.score(X_test_transformed, y_test))
clf.predict(X_test_transformed)
confusion_matrix(y_test, y_predicted, labels=labels)
```

מטריקות דיוק

8. איך מודדים דיוק? יש מספר שיטות לפי מה אני מעוניין למדוד. (נבין לדוגמא, אי אפשר לבדוק רק

לפי כל מקרה לגופו האם צדק או לא, למשל גלאי פצצות שאומר **תמיד** "אין פצצה" – יעברו 999 אנשים בלי פצצה ואחד עם – האם נוכל לומר שהציון שלו הוא 99.9% ברור שלא). נסתכל על 3 מטריקות נפוצות:

א. Accuracy – כמות הפרדיקציות הנכונות לחלק לסך כל הניסויים. למשל – סיווג 150 פרחים

$$\text{לפי 3 זנים, למשל } \frac{78}{150}$$

ב. Precision – כמות הפרדיקציות שסיווגתי כחיובי וצדקתי מתוך כל מה שניחשתי כחיובי,

$$\left(\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \right), \text{ למשל כמות האימיילים שסיווגתי כספאם ואכן היו ספאם,}$$

מתוך כמות המיילים שניחשתי שהם ספאם (כלומר כאלו שצדקתי בהם וכאלו שטעיתי בהם).

לדוגמא הגיעו 100 מיילים, מתוכם באמת יש 20 ספאמים. ואני ניחשתי 16 מהספאמים

כספאם, 41 ספאמים סיווגתי כלא ספאם, ומתוך הלא ספאם סיווגתי אחד מהם כספאם. ציון

$$\text{ה precision יהיה } \frac{16}{17}. \text{ כלומר מתוך 17 הספאמים שהגדרתי – צדקתי ב-16 מהם.}$$

ג. Recall – כמות הפרדיקציות שסיווגתי כחיובי וצדקתי מתוך כל הניסויים שהייתי נדרש

$$\left(\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \right), \text{ למשל כמות האימיילים שסיווגתי כספאם}$$

ואכן היו ספאם, מתוך כמה באמת ספאם הגיעו. לדוגמא 100 מיילים, מתוכם 20 ספאמים.

וניחשתי 16 מהספאמים כספאם, 41 ספאמים כלא ספאם, ומתוך הלא ספאם סיווגתי אחד

$$\text{מהם כספאם. ציון ה recall יהיה } \frac{16}{20}. \text{ כלומר מתוך 20 ספאמים שבאמת היו – עליתי על 16.}$$

ישנן עוד מטריקות שלא נגענו בהן כמו R , R^2 וכו'.

שיעור 4

נושא

1. .

נושא

2. ש

נושא

.

שיעור 5

נושא

.

נושא

1. .

נושא

.

שיעור 6

נושא

.

נושא

1. .

נושא

.

שיעור 7

נושא

.

נושא

1. .

נושא

.

שיעור 8

נושא

.

נושא

.1 .

נושא

.

שיעור 9

נושא

.

נושא

.1 .

נושא

.

שיעור 10

נושא

.

נושא

.1 .

נושא

.

קריאה מומלצת

.1 .