מסווג בייס - גישה הסתברותית לסיווג

Bayes Classifier - A Probabilistic Approach to Classification

ד"ר יורם סגל

November 2025

גרסה 1.0

תוכן עניינים

רשימת איורים

רשימת טבלאות

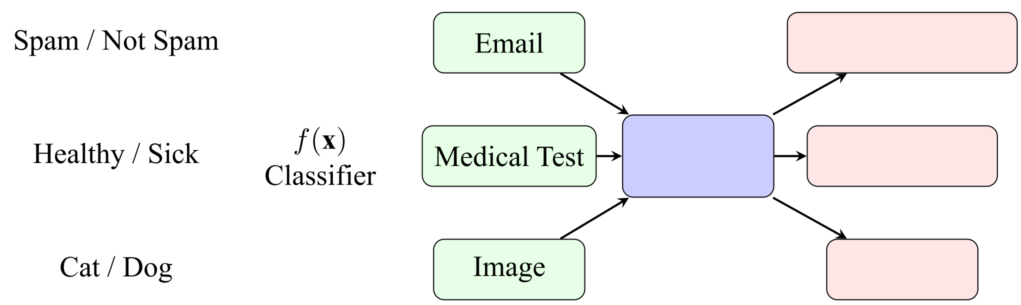
1 פתיחה ומבוא

1.1 מהו מסווג?

בכל יום, מאות פעמים ביום, אנו מבצעים פעולות סיווג מבלי לשים לב. כשאנו קוראים הודעת דואר אלקטרוני חדשה, המוח שלנו מחליט תוך שברירי שנייה אם מדובר בהודעה חשובה או בדואר זבל. כשרופא בוחן תוצאות בדיקת דם, הוא מסווג את המטופל כבריא או כחולה. כשמערכת בנקאית בוחנת בקשה להלוואה, היא מסווגת את המבקש כבעל סיכון נמוך או גבוה. כל אלה הן בעיות classification – מיפוי של תצפיות לקטגוריות מוגדרות מראש.

מבחינה פורמלית, מסווג הוא פונקציה מתמטית המקבלת כקלט וקטור של מאפיינים $\mathbf{x}$ ומחזירה תווית מחלקה y. נסמן את פעולת הסיווג כ$\hat{y} = f(\mathbf{x})$, כאשר $\hat{y}$ היא התחזית שלנו למחלקה הנכונה. אך מאחורי הפשטות הנראית לעין של נוסחה זו מסתתר אתגר מתמטי עמוק: כיצד נלמד את הפונקציה f מתוך נתוני אימון? איזו אסטרטגיה תבטיח שהמסווג שלנו יצליח לא רק על הדוגמאות שראה בעבר, אלא גם על מקרים חדשים שטרם פגש?

איור 1 ממחיש את מושג הסיווג הבסיסי. בתרשים ניתן לראות כיצד תצפיות שונות – הודעות דואר, תוצאות בדיקות רפואיות, נתוני משתמשים – עוברות דרך פונקציית הסיווג f ומקבלות תווית מחלקה. התרשים מדגיש נקודה חשובה: אותו מסווג צריך לעבוד על מגוון רחב של קלטים, ולהפיק החלטות עקביות גם כשהנתונים רועשים או לא מלאים.



איור 1: תרשים המדגים את מושג הסיווג: מיפוי מתצפיות (מאפיינים) לתחזיות (מחלקות)

[1] מציע לראות בבעיית הסיווג מקרה פרטי של בעיית קבלת החלטות תחת אי-ודאות. במקום לחפש פונקציה דטרמיניסטית קשיחה, נוכל לאמץ גישה הסתברותית: לכל מחלקה אפשרית c, נחשב את ההסתברות שהתצפית $\mathbf{x}$ שייכת למחלקה זו, $P(C = c|\mathbf{X} = \mathbf{x})$. לאחר מכן, נבחר את המחלקה עם ההסתברות הגבוהה ביותר:

$$\hat{y} = \arg\max_c P(C = c|\mathbf{X} = \mathbf{x}) \tag{1}$$

נוסחה 1 מגדירה את כלל הMaximum A Posteriori (MAP), שהוא הבסיס למסווג בייס. כאן טמון היופי של הגישה ההסתברותית: במקום להחליט בצורה חד-משמעית "כן" או "לא", אנו מכמתים את מידת הוודאות שלנו. זה מאפשר לנו לא רק לסווג, אלא גם להעריך

עד כמה אנו בטוחים בהחלטה שלנו.

1.2 תומס בייס והתפתחות התיאוריה

בשנת 1763, שנתיים לאחר מותו של הכומר האנגלי Thomas Bayes, פורסם מאמר שהיה אמור לשנות את פני תורת ההסתברות לנצח [2]. המאמר, שנושאו היה "מסה לקראת פתרון בעיה בדוקטרינת הסיכויים" (An Essay towards solving a Problem in the Doctrine of Chances), הציג רעיון מהפכני: כיצד ניתן לעדכן את אמונותינו לאור ראיות חדשות.

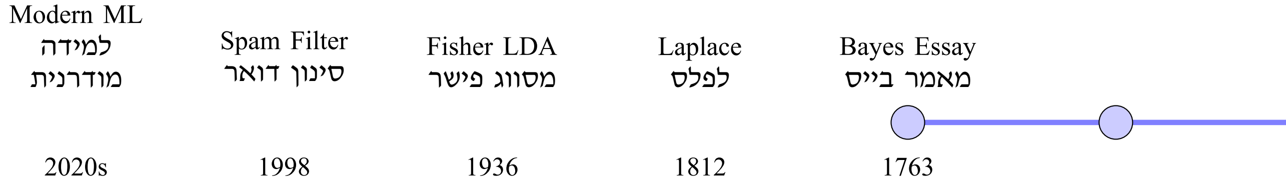
בייס התמודד עם מה שנקרא היום "בעיית ההסתברות ההפוכה" (inverse probability). השאלה שהעסיקה אותו הייתה פשוטה לכאורה אך עמוקה במשמעותה: אם אנו יודעים את ההסתברות לתוצאה מסוימת בהינתן סיבה, האם נוכל להסיק את ההסתברות לסיבה בהינתן התוצאה? לדוגמה, אם ידוע לנו שבחולי שפעת קיימת הסתברות גבוהה לחום, האם נוכל להסיק מהימצאות חום על הסתברות לשפעת?

התשובה של בייס, שזוקקה והורחבה מאוחר יותר על ידי Pierre-Simon Laplace, הייתה נוסחה פשוטה לכאורה אך בעלת השלכות עצומות:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)} \tag{2}$$

נוסחה 2 היא משפט בייס, אבן היסוד של כל מה שנלמד בספר זה. היא אומרת שההסתברות להשערה H בהינתן ראיה E שווה למכפלת שני גורמים: הראשון הוא ה-likelihood – ההסתברות לראיה אם ההשערה נכונה, $P(E|H)$; השני הוא הprior – ההסתברות להשערה לפני שראינו את הראיה, $P(H)$. את המכפלה מנרמלים על ידי חלוקה בהסתברות הכוללת לראיה, $P(E)$.

איור 2 משרטט את המסע ההיסטורי של תיאוריית בייס. מהרעיון המקורי שגובש בשנת 1763, דרך הרחבותיו של לפלס ופיתוחיו של פישר, ועד ליישומים המודרניים בסינון דואר זבל ובלמידת מכונה. ציר הזמן מראה שלעיתים רעיון מתמטי עמוק זקוק למאות שנים כדי למצוא את השימוש המעשי שלו – ושכשהוא מוצא אותו, ההשפעה יכולה להיות מהפכנית.



איור 2: ציר זמן המתעד את התפתחות תיאוריית בייס ויישומיה מ־1763 ועד ימינו

למעלה ממאתיים שנה לאחר פרסום המאמר, משפט בייס הפך למנוע המרכזי של למידת מכונה מודרנית. הסיבה פשוטה: בעולם של נתונים חלקיים, רועשים ומשתנים, אנו זקוקים לשיטה עקבית לעדכן את ההערכות שלנו. משפט בייס מספק בדיוק את זה – מסגרת מתמטית קוהרנטית לשילוב ידע קודם עם תצפיות חדשות.

1.3 יישומים מודרניים

ביום בהיר אחד בשנת 1998, צוות חוקרים מMicrosoft Research פרסם מאמר על שיטה חדשה לסינון דואר זבל באמצעות מסווג בייס [3]. הרעיון היה פשוט להחרידה: לכל מילה בהודעת דואר, חשבו את ההסתברות שהיא מופיעה בדואר זבל לעומת דואר רגיל. לאחר מכן, שילבו את ההסתברויות הללו באמצעות משפט בייס כדי לחשב את ההסתברות שההודעה כולה היא דואר זבל. התוצאות היו מרשימות – דיוק (accuracy) של מעל 95 אחוז בזיהוי דואר זבל.

מה שהפך את הסיפור הזה למרתק במיוחד לא היה רק הצלחת השיטה, אלא העובדה שהיא הסתמכה על הנחת עצמאות פשטנית בעליל: ההנחה שהסתברות המילה "ויאגרה" בהודעה אינה תלויה בהסתברות המילה "זול". כל מי שקרא אי פעם דואר זבל יודע שהנחה זו רחוקה מהמציאות – מילים בדואר זבל מופיעות במקבצים צפויים למדי. ובכל זאת, למרות "נאיביות" זו (מכאן השם Naive Bayes), המסווג עבד בצורה מצוינת.

תופעה זו, שנחקרה לעומק על ידי [4], מלמדת אותנו משהו עמוק על טבען של בעיות למידה: לעיתים, מודל פשוט עם הנחות לא מדויקות עדיף על מודל מורכב שמתאים יתר על המידה לנתוני האימון. הפשטות של מסווג בייס היא לא חולשה אלא כוח – היא מונעת התאמת יתר ומאפשרת למידה מהירה גם מכמויות קטנות של נתונים.

כיום, מסווגי בייס משמשים בשלל תחומים. בתחום הרפואה, הם עוזרים לאבחן מחלות על סמך תסמינים ותוצאות בדיקות [5]. במערכות המלצה, הם מנבאים אילו מוצרים יעניינו משתמש. בעיבוד שפה טבעית, הם מסווגים טקסטים לקטגוריות, מזהים סנטימנט, ואף מתרגמים בין שפות. כפי שמציין [6], גם בישראל מסווגי בייס נמצאים בשימוש נרחב, מסינון דואר זבל בעברית ועד ניתוח רשתות חברתיות.

הקוד הפשוט הבא ממחיש את הרעיון הבסיסי של סיווג:

דוגמה זו, פשטנית ככל שתהיה, מדגימה את הלב של הרעיון: אנו מיפים תצפיות (הכנסה והיסטוריית אשראי) להחלטות (רמת סיכון) על בסיס הסתברויות שנלמדו מעבר. מסווג בייס אמיתי יבצע את החישובים הללו בצורה שיטתית ומתמטית, אך העקרון זהה.

1.4 למי מיועד ספר זה?

ספר זה נכתב עבור שלושה קהלים עיקריים, שכולם חולקים עניין באיזון בין תיאוריה לפרקטיקה. הקהל הראשון הוא סטודנטים לתארים מתקדמים במדעי המחשב, הנדסת חשמל או סטטיסטיקה, המחפשים הבנה עמוקה של יסודות הסיווג ההסתברותי. עבורם, ספר זה מציע מסע שמתחיל מהבסיס המתמטי – משפט בייס, תורת ההסתברות, ותורת ההחלטות – ומתקדם אל יישומים מעשיים עם קוד Python ממשי.

הקהל השני הוא חוקרים ומפתחים בתעשייה, שנתקלים בבעיות סיווג בעבודתם היומיומית. עבורם, הספר מספק לא רק כלים מעשיים מוכנים לשימוש, אלא גם הבנה מעמיקה של מתי וכיצד להשתמש בהם. מתי עדיף מסווג בייס נאיבי על פני Logistic Regression? איך מתמודדים עם מאפיינים מתמשכים? מה עושים כשהנתונים לא מאוזנים? על שאלות אלה ואחרות מקדיש הספר פרקים שלמים.

הקהל השלישי, ואולי המעניין ביותר, הוא קוראים שמחפשים לא רק לדעת "איך" אלא

מסווג פשוט לסיכון אשראי

```python
def credit_risk_classifier(income_level, credit_history):
    """
␣␣␣␣Simple␣credit␣risk␣classifier␣based␣on␣historical␣training␣data.
␣␣␣␣Demonstrates␣basic␣classification␣concept␣using␣probability␣
thresholds.

␣␣␣␣Args:
␣␣␣␣␣␣␣␣income_level:␣str,␣either␣"low",␣"medium",␣or␣"high"
␣␣␣␣␣␣␣␣credit_history:␣str,␣either␣"poor",␣"fair",␣or␣"good"

␣␣␣␣Returns:
␣␣␣␣␣␣␣␣str:␣Risk␣level␣-␣"high_risk",␣"medium_risk",␣or␣"low_risk"
␣␣␣␣"""
    # Based on learned probabilities from training data
    # If P(default|low,poor) > 0.7, classify as high risk
    if income_level == "low" and credit_history == "poor":
        return "high_risk"

    # If P(default|high,good) < 0.1, classify as low risk
    elif income_level == "high" and credit_history == "good":
        return "low_risk"

    # All other combinations get medium risk
    else:
        return "medium_risk"

# Usage example
applicant_risk = credit_risk_classifier("low", "poor")
print(f"Risk␣level:␣{applicant_risk}")  # Output: Risk level: high_risk
```

גם "למה". אלה שרוצים להבין לא רק כיצד פועל אלגוריתם, אלא מה ההיסטוריה שלו, מהן המשמעויות הפילוסופיות של הבחירות המתמטיות שאנו עושים, וכיצד כל זה משתלב בתמונה הגדולה יותר של למידת מכונה ובינה מלאכותית. בהשראת סגנון הכתיבה של Yuval Noah Harari, שילבתי בספר לא רק נוסחאות וקוד, אלא גם סיפורים, הקשר היסטורי, ומבט ביקורתי על ההנחות שאנו לוקחים כמובנות מאליהן.

הספר מניח ידע בסיסי בהסתברות וסטטיסטיקה, הכרות עם תכנות Python, ויכולת קריאת נוסחאות מתמטיות. אך מעבר לכך, הספר בנוי בצורה מודולרית: ניתן לדלג על הוכחות מתמטיות מפורטות ולהתמקד בקוד והיישומים, או להיפך – להעמיק בתיאוריה ולדלג על פרטי המימוש.

1.5 מבנה הספר

המסע שלנו מתחיל, כמובן, בבסיס המתמטי. פרק 2 מקדיש תשומת לב מיוחדת למשפט בייס ולתורת ההסתברות, כשהוא מדגיש את הקשר בין הסתברות, תדירות, ומידע. אנו נראה כיצד משפט בייס אינו רק נוסחה מתמטית, אלא שיטת חשיבה – דרך לעדכן אמונות לאור ראיות חדשות בצורה עקבית ורציונלית.

לאחר שהנחנו את התשתית התיאורטית, נעבור בפרק 3 אל ליבת הספר: מסווג בייס הנאיבי (Naive Bayes Classifier). נראה מדוע "הנאיביות" של ההנחה על עצמאות מאפיינים אינה מונעת מהמסווג להצליח בפועל, ונממש אותו מאפס בשפת Python. נתרגל על בעיות קלאסיות כמו סינון דואר זבל וסיווג טקסטים.

פרק 4 יוביל אותנו אל עולם המאפיינים המתמשכים, שם נפגוש את מסווג בייס הגאוסי (Gaussian Bayes Classifier) ואת Linear Discriminant Analysis (LDA). נראה כיצד הנחות על התפלגויות גאוסיות מובילות למסווגים לינאריים פשוטים אך יעילים, ומתי הנחות אלה סבירות.

בפרק 5 נטפל בשאלות המעשיות שכל מיישם נתקל בהן: כיצד מתמודדים עם ערכים חסרים? איך מטפלים במאפיינים קטגוריאליים? מה עושים כשקטגוריה מסוימת לא הופיעה בנתוני האימון? נלמד טכניקות החלקה (smoothing) כמו Laplace smoothing, ונראה כיצד הן מונעות הסתברויות אפסיות שעלולות לקרוס את המסווג.

פרק 6 עוסק בהערכת ביצועים – כיצד יודעים אם המסווג שבנינו טוב? נלמד מדדים כמו דיוק (accuracy), דיוק מדויק (precision), ריקול (recall), ומדד F1. נכיר את עקומי ROC ונבין מתי כל מדד רלוונטי. חשוב מכך, נלמד כיצד להימנע ממלכודת ההתאמת-יתר (overfitting) באמצעות אימות צולב (cross-validation).

בפרק 7 נחקור התרחבויות ווריאציות על מסווגי בייס הבסיסיים. נכיר את Semi-Naive Bayes שמאפשר תלות חלקית בין מאפיינים, את רשתות בייס (Bayesian Networks) שמדגמות תלויות מורכבות, ואת Quadratic Discriminant Analysis (QDA) שמתאים מטריצות שונות לכל מחלקה.

לבסוף, פרק 8 יציב את מסווגי בייס בהקשר רחב יותר של למידת מכונה. נשווה אותם לאלגוריתמים אחרים כמו Logistic Regression, Decision Trees, SVM, וNeural Networks. נדון במתי עדיף להשתמש במסווג בייס ומתי באלטרנטיבות, ונסיים בהשקפה ביקורתית על

יתרונות וחסרונות של הגישה ההסתברותית בכלל.

לאורך כל הספר, שילבתי דוגמאות קוד מלאות בPython, תרגילים מעשיים, והצעות למחקר נוסף. הקוד כולו זמין במאגר GitHub הנלווה לספר, ומאורגן בצורה מודולרית שמאפשרת לקוראים להריץ, לשנות ולהתנסות.

המטרה של ספר זה איננה רק ללמד אלגוריתם ספציפי, אלא לטפח דרך חשיבה. מסווג בייס הוא חלון להבנת הגישה ההסתברותית למידה ממכונה – גישה שמכירה באי-ודאות, מכמתת אותה, ומשתמשת בה לקבלת החלטות טובות יותר. זוהי פילוסופיה שמתאימה לא רק לבעיות סיווג טכניות, אלא לכל תחום בו אנו צריכים להסיק מסקנות מתוך מידע חלקי ורועש – כלומר, כמעט לכל תחום בחיים.

2 יסודות הסתברות

תארו לעצמכם שקמתם בבוקר עם כאב ראש. בעוד אתם מחזיקים את הראש בידיים, עולה בכם השאלה המטרידה: האם אני חולה בשפעת? כאב הראש יכול להיות סימפטום לשפעת, אך הוא יכול גם להיות תוצאה של מאות סיבות אחרות - מתח, חוסר שינה, התייבשות, או פשוט יום קשה במיוחד. איך נוכל לקבל החלטה מושכלת על סמך המידע הזה?

זו בדיוק השאלה שתורת ההסתברות באה לענות עליה. במשך מאות שנים, מאז המאמר המכונן של תומאס בייס Thomas Bayes בשנת 1763 [2], פיתח האנושות כלים מתמטיים מדויקים לכימות אי-הוודאות ולעדכון אמונותינו לאור ראיות חדשות. כפי שהראה בייס במאמרו, אנו יכולים לענות על שאלות מהסוג "מה ההסתברות שאני חולה בשפעת, בהינתן שיש לי כאב ראש?" באופן מתמטי מדויק.

בפרק זה נבנה את היסודות המתמטיים הדרושים להבנת מסווג בייס Bayes Classifier. נתחיל ממושגי הסתברות בסיסיים, נמשיך לכלל השרשרת Chain Rule, ונגיע לנוסחת בייס המפורסמת. לאורך כל הדרך נשתמש בסיפור כאב הראש והשפעת כדוגמה מרכזית המלווה אותנו.

2.1 מושגי יסוד בהסתברות

כדי להבין את הקשר בין כאב ראש לשפעת, עלינו קודם כל להגדיר מה זו הסתברות. בעולם המתמטי, הסתברות היא מספר בין 0 לבין 1 המתאר את מידת הוודאות שלנו שאירוע מסוים יתרחש. הסתברות של 0 פירושה שהאירוע בלתי אפשרי, הסתברות של 1 פירושה שהאירוע ודאי, וכל מספר בינהם מתאר דרגת ודאות ביניים.

בואו נגדיר את האירועים שלנו. נסמן את האירוע "יש לי כאב ראש" באות H (מהמילה Headache), ואת האירוע "אני חולה בשפעת" באות F (מהמילה Flu). כעת נוכל לשאול שאלות הסתברותיות בסיסיות.

השאלה הראשונה היא: מה ההסתברות שבאדם כלשהו, שנבחר באופן אקראי מהאוכלוסייה, יש כאב ראש? זוהי **הסתברות לא-מותנית** או **הסתברות שולית** Marginal Probability, ואנו מסמנים אותה $P(H)$. על סמך מחקרים רפואיים, נניח שבכל יום נתון כ-10% מהאנשים מתעוררים עם כאב ראש, כלומר $P(H) = 0.1$.

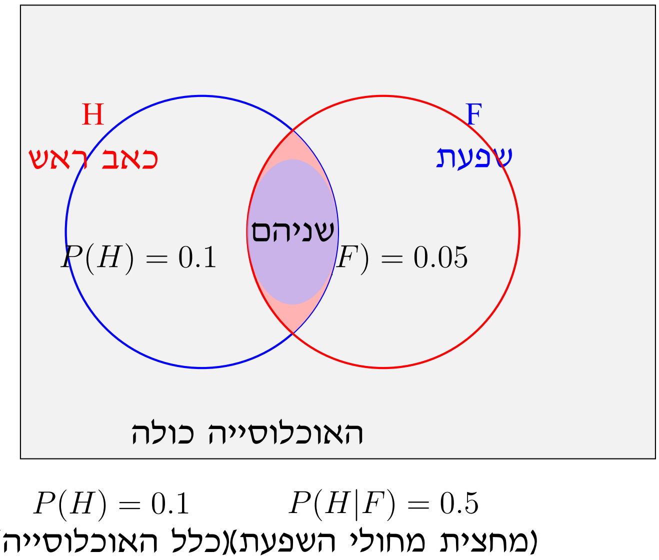
באותו אופן, ההסתברות שאדם אקראי חולה בשפעת ביום נתון היא $P(F)$. שפעת היא מחלה עונתית שמשתנה לפי העונה, אך נניח שבממוצע כ-2.5% מהאוכלוסייה חולים בשפעת בכל זמן נתון, כלומר $P(F) = 0.025 = \frac{1}{40}$.

אך מה אם אנו רוצים לשאול שאלה מורכבת יותר? מה ההסתברות שיש לאדם כאב ראש **בהינתן** שהוא חולה בשפעת? זוהי **הסתברות מותנית** Conditional Probability, ואנו מסמנים אותה $P(H|F)$. הסימון $H|F$ נקרא "H given F" או בעברית "H בהינתן F". זוהי הסתברות שונה לחלוטין מ-$P(H)$, כי אנו כבר יודעים משהו - אנו יודעים שהאדם חולה בשפעת.

מהניסיון הרפואי אנו יודעים ששפעת גורמת לכאב ראש בכ-50% מהמקרים [7]. לכן $P(H|F) = 0.5$. שימו לב: זה הרבה יותר גבוה מ-$P(H) = 0.1$! העובדה שאנו יודעים

שהאדם חולה בשפעת העלתה משמעותית את ההסתברות שיש לו כאב ראש.

כפי שמציינים Bishop [1] ו-Murphy [7] בספריהם הקלאסיים, הסתברות מותנית היא אבן היסוד של למידת מכונה הסתברותית. היא מאפשרת לנו לכמת את השפעת הידע החדש על אמונותינו. איור 3 ממחיש את ההבדל בין הסתברות לא-מותנית להסתברות מותנית.



איור 3: הסתברות מותנית לעומת הסתברות לא-מותנית. השטח של המעגל H מייצג את $P(H)$, בעוד שחלק החפיפה עם F מייצג את $P(H|F) \cdot P(F)$.

איור 3 מראה שהסתברות מותנית היא למעשה "זום פנימה" לתוך קבוצת המקרים שבהם התנאי מתקיים. כאשר אנו שואלים על $P(H|F)$, אנו מצמצמים את מרחב האפשרויות רק לאנשים שחולים בשפעת, ושואלים: מתוך אלה, כמה אחוזים סובלים מכאב ראש?

2.2 כלל השרשרת

כעת נגיע לכלל מתמטי יסודי שמקשר בין הסתברות מותנית להסתברות משותפת: **כלל השרשרת** Chain Rule או **כלל המכפלה** Product Rule. כלל זה הוא אחד מאבני היסוד החשובות ביותר בתורת ההסתברות.

בואו נשאל שאלה חדשה: מה ההסתברות שאדם אקראי **גם** חולה בשפעת **וגם** סובל מכאב ראש? זוהי **הסתברות משותפת** Joint Probability, ואנו מסמנים אותה $P(H \cap F)$ או בקיצור $P(H, F)$.

כלל השרשרת אומר לנו שאנו יכולים לחשב הסתברות משותפת על ידי מכפלה של הסתברות מותנית בהסתברות לא-מותנית:

$$P(H, F) = P(H|F) \cdot P(F) \tag{3}$$

הנוסחה 3 אומרת משהו אינטואיטיבי למדי: ההסתברות ששני אירועים יקרו ביחד שווה להסתברות שהראשון יקרה, כפול ההסתברות שהשני יקרה בהינתן שהראשון כבר

התרחש. במקרה שלנו: ההסתברות לשפעת וכאב ראש ביחד שווה להסתברות לשפעת, כפול ההסתברות לכאב ראש בהינתן שיש שפעת.

בואו נציב את המספרים שלנו. אנו יודעים ש-$P(F) = 0.025$ ו-$P(H|F) = 0.5$, לכן:

$$P(H,F) = 0.5 \times 0.025 = 0.0125 \tag{4}$$

התוצאה $P(H,F) = 0.0125$ אומרת שכ-1.25% מהאוכלוסייה סובלים גם משפעת וגם מכאב ראש בו-זמנית. זה הגיוני: שפעת נדירה יחסית (2.5%), וגם אם יש לך שפעת, רק חצי מהחולים מקבלים כאב ראש, אז התוצאה המשותפת נדירה עוד יותר.

כלל השרשרת הוא סימטרי: אנו יכולים גם לכתוב

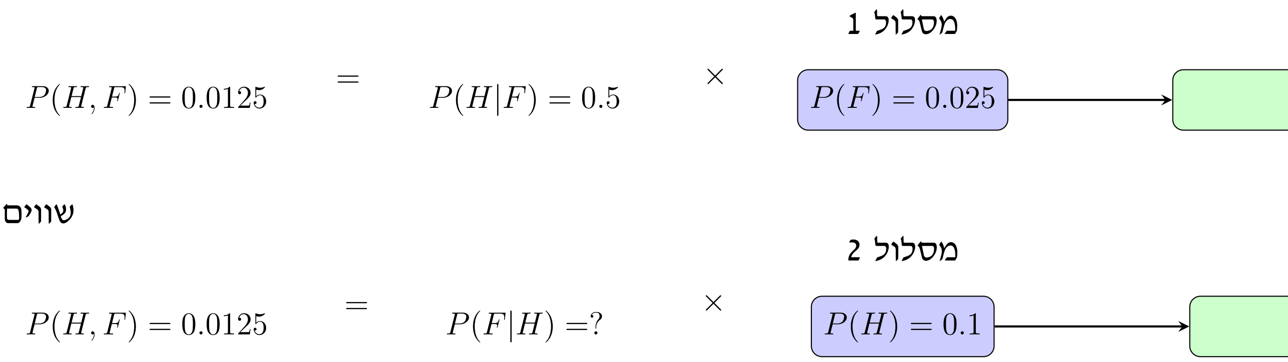
$$P(H,F) = P(F|H) \cdot P(H) \tag{5}$$

הנוסחה 5 היא פשוט אותו כלל, אך עם סדר הפוך של האירועים. זה אומר שההסתברות המשותפת שווה גם להסתברות לכאב ראש, כפול ההסתברות לשפעת בהינתן כאב ראש. שתי הנוסחאות 3 ו-5 חייבות לתת את אותה תוצאה, כי שתיהן מחשבות את אותו דבר: ההסתברות שהאירועים H ו-F מתרחשים ביחד.

דווקא הסימטריה הזו היא המפתח להבנת כלל בייס. אם שני הביטויים שווים, אנו יכולים להשוות ביניהם:

$$P(H|F) \cdot P(F) = P(F|H) \cdot P(H) \tag{6}$$

נוסחה 6 מראה את הקשר העמוק בין שני כיווני ההסתברות המותנית. היא אומרת שהסתברות לכאב-ראש-בהינתן-שפעת כפול הסתברות-לשפעת שווה להסתברות לשפעת-בהינתן-כאב-ראש כפול הסתברות-לכאב-ראש. איור 4 ממחיש את הקשר הזה בצורה ויזואלית.



איור 4: כלל השרשרת פועל בשני כיוונים. שני המסלולים מובילים לאותה הסתברות משותפת $P(H,F)$.

כפי שמראה איור 4, ישנן שתי דרכים שונות להגיע לאותה נקודה - ההסתברות המשותפת. זוהי התובנה המפתח שתוביל אותנו לכלל בייס.

2.3 כלל בייס

כעת אנו מוכנים לענות על השאלה שפתחנו בה את הפרק: האם אני חולה בשפעת, בהינתן שיש לי כאב ראש? במילים מתמטיות: מהו $P(F|H)$?

זוהי השאלה שכל אחד מאיתנו שואל כשהוא מרגיש תסמין כלשהו. אנו רואים את הראיה (כאב הראש), ורוצים להסיק על הסיבה הנסתרת (האם יש שפעת?). בשפה מקצועית, זוהי שאלת **הסקה הפוכה** Inverse Inference או **הסקה אחורה** Backward Inference.

הבעיה היא שאין לנו את $P(F|H)$ ישירות. אבל יש לנו את הכיוון ההפוך: $P(H|F)$ - ההסתברות לכאב ראש בהינתן שפעת. זה המידע שרופאים אוספים ממחקרים קליניים: "מתוך חולי שפעת, כמה אחוזים מקבלים כאב ראש?" אך מה שאנו צריכים הוא הכיוון ההפוך: "מתוך אנשים עם כאב ראש, כמה אחוזים חולים בשפעת?"

כאן נכנס לפעולה **כלל בייס** Bayes' Rule, שנקרא על שם תומאס בייס שגילה אותו במאה ה-18 [2]. אנו לוקחים את המשוואה 6 שגזרנו מכלל השרשרת, ומחלקים את שני האגפים ב-$P(H)$:

$$P(F|H) = \frac{P(H|F) \cdot P(F)}{P(H)} \tag{7}$$

זוהי נוסחת בייס המפורסמת! נוסחה 7 נראית פשוטה, אך השלכותיה הן מהפכניות. היא אומרת לנו שאנו יכולים "להפוך" את כיוון ההסתברות המותנית, אם רק יש לנו שלושה פיסות מידע: את ההסתברות בכיוון ההפוך $P(H|F)$, את ההסתברות הבסיסית למחלה $P(F)$, ואת ההסתברות הבסיסית לתסמין $P(H)$.

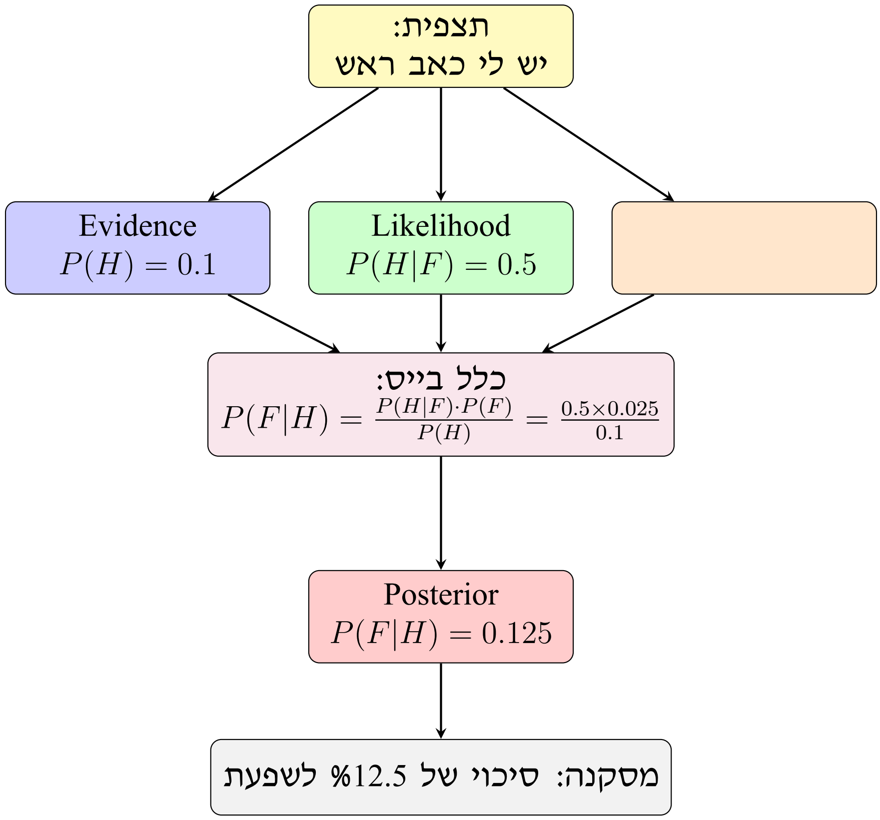
בואו נציב את המספרים שלנו בנוסחה 7:

$$\begin{aligned} P(F|H) &= \frac{P(H|F) \cdot P(F)}{P(H)} \\ &= \frac{0.5 \times 0.025}{0.1} \\ &= \frac{0.0125}{0.1} \\ &= 0.125 \end{aligned} \tag{8}$$

החישוב 8 מגלה תוצאה מעניינת: אם יש לנו כאב ראש, ההסתברות שיש לנו שפעת היא רק 12.5%! זה הרבה יותר נמוך ממה שרבים היו מצפים. למרות שכאב ראש הוא תסמין שכיח של שפעת (מופיע ב-50% מהמקרים), השפעת עצמה נדירה מספיק (2.5% מהאוכלוסייה), וכאב ראש שכיח מספיק מסיבות אחרות (10% מהאוכלוסייה), עד שהסתברות השפעת עדיין נמוכה למדי.

זוהי אחת התובנות המרכזיות של חשיבה בייסיאנית: ראיה כשלעצמה אינה מספיקה. עלינו לשקול גם את **ההסתברות הבסיסית** Base Rate של המחלה באוכלוסייה. מחלה נדירה נשארת נדירה גם לאחר שמצאנו תסמין, אם התסמין עצמו שכיח [1], [8].

איור 5 ממחיש את תהליך ההסקה הבייסיאנית מכאב ראש לשפעת.



איור 5: תהליך ההסקה הבייסיאנית מתסמין (כאב ראש) למחלה (שפעת) באמצעות כלל בייס.

איור 5 מראה כיצד כלל בייס מאפשר לנו לעבור מתצפית (התסמין) לאבחנה (המחלה), תוך שימוש במידע הרפואי הקיים.

2.4 מושגי מפתח: א-פריורי, נראות, א-פוסטריורי

נוסחת בייס שגזרנו בנוסחה 7 מורכבת משלושה רכיבים מרכזיים, שלכל אחד מהם יש שם ותפקיד חשוב בהסקה בייסיאנית. הבנת המושגים הללו היא קריטית להבנת מסווג בייס ואלגוריתמים בייסיאניים בכלל.

2.4.2.4.1 הסתברות א-פריורי (Prior Probability)

המונח **א-פריורי** מגיע מלטינית ופירושו "מראש" או "לפני". ההסתברות הא-פריורית $P(F)$ מייצגת את האמונה שלנו בהתרחשות האירוע **לפני** שראינו כל ראיה חדשה. במקרה שלנו, $P(F) = 0.025$ היא ההסתברות שאדם אקראי חולה בשפעת, בלי כל מידע נוסף.

הא-פריורי משקף את הידע הקודם שלנו על העולם. זה מה שאנו יודעים מניסיון עבר, ממחקרים סטטיסטיים, או מהשכל הישר. כפי שמדגישים Murphy [7] ו-Duda וחב' [8], הבחירה בהסתברות א-פריורית היא אחד ההיבטים החשובים ביותר בניתוח בייסיאני.

בהקשר של אבחון רפואי [5], הא-פריורי מייצג את השכיחות של המחלה באוכלוסייה הכללית. רופא מנוסה לוקח בחשבון את השכיחות הבסיסית של מחלות לפני שהוא מאבחן.

אם מחלה נדירה מאוד, הרופא ידרוש ראיות חזקות יותר לפני שיאבחן אותה, גם אם יש תסמינים שמתאימים.

2.4.2.4.2 נראות (Likelihood)

הנראות $P(H|F)$ מודדת כמה "טוב" התצפית שלנו מסבירה את ההשערה. במילים אחרות: בהנחה שההשערה נכונה (יש לי שפעת), מה הסיכוי שאראה את הראיה (כאב ראש)?

שימו לב להבדל העדין אך קריטי: הנראות אינה ההסתברות להשערה! היא ההסתברות לראיה בהינתן ההשערה. במקרה שלנו, $P(H|F) = 0.5$ אומר שאם אנו יודעים בוודאות שיש שפעת, אז יש סיכוי של 50% שנראה כאב ראש.

המונח "נראות" Likelihood מתייחס לכך שאנו שואלים: עד כמה נראה סביר (likely) שנקבל את התצפית הזו, בהינתן המודל שלנו? נראות גבוהה פירושה שהתצפית שלנו מתאימה היטב למודל, ולכן היא מחזקת את האמון שלנו במודל. נראות נמוכה פירושה שהתצפית מפתיעה או לא צפויה לפי המודל, ולכן היא מחלישה את האמון שלנו.

2.4.2.4.3 ראיה או נורמליזציה (Evidence / Normalization)

המכנה של נוסחת בייס, $P(H)$, נקרא **הראיה** Evidence או **קבוע הנורמליזציה** Normalization Constant. זהו ההסתברות הכוללת לתצפית שראינו, ללא תלות בהשערה.

מבחינה טכנית, $P(H)$ מבטיח שהתוצאה תהיה הסתברות תקפה (בין 0 ל-1). הוא "מנרמל" את המניין כך שכל ההסתברויות יסתכמו ל-1. לעיתים קרובות, במיוחד כאשר מעניינים אותנו רק השוואות יחסיות בין השערות שונות, אנו יכולים להתעלם ממנו ולכתוב:

$$P(F|H) \propto P(H|F) \cdot P(F) \tag{9}$$

הסימון $\propto$ פירושו "פרופורציונלי ל-" או "יחסי ל-". נוסחה 9 אומרת שההסתברות א-פוסטריורית פרופורציונלית למכפלת הנראות והא-פריורי, אך אנו משמיטים את קבוע הנורמליזציה.

2.4.2.4.4 הסתברות א-פוסטריורי (Posterior Probability)

המונח **א-פוסטריורי** מגיע מלטינית ופירושו "לאחר". ההסתברות הא-פוסטריורית $P(F|H)$ היא האמונה המעודכנת שלנו **לאחר** שראינו את הראיה. במקרה שלנו, $P(F|H) = 0.125$ היא ההסתברות שאדם חולה בשפעת, לאחר שגילינו שיש לו כאב ראש.

הא-פוסטריורי הוא המטרה של כל הסקה בייסיאנית. זוהי התשובה לשאלה שלנו. הוא משקלל את הידע הקודם שלנו (א-פריורי) עם הראיה החדשה שאספנו (נראות), כדי להגיע למסקנה מעודכנת.

טבלה 1 מסכמת את המושגים המרכזיים ואת הערכים שלהם בדוגמת השפעת-כאב-ראש.

כפי שמראה טבלה 1, המעבר מהא-פריורי (0.025 או 2.5%) לא-פוסטריורי (0.125 או 12.5%) מייצג עדכון האמונה שלנו. הראיה (כאב הראש) העלתה את הסיכוי לשפעת פי חמשה! אך עדיין, למרות העלייה, הסיכוי נשאר נמוך יחסית, כי המחלה נדירה והתסמין שכיח.

טבלה 1: סיכום מושגי הסתברות מרכזיים בדוגמת השפעת וכאב הראש

מושג	ייצוג	ערך בדוגמה	פירוש
א-פריורי Prior	$P(F)$	0.025	הסתברות לשפעת לפני הראיה
נראות Likelihood	$P(H\|F)$	0.5	הסתברות לכאב ראש בהינתן שפעת
ראיה Evidence	$P(H)$	0.1	הסתברות לכאב ראש בכלל
א-פוסטריורי Posterior	$P(F\|H)$	0.125	הסתברות לשפעת לאחר הראיה

טבלה 2: pbth

זוהי הפילוסופיה הבייסיאנית בתמצית: אנו מתחילים עם אמונה קודמת, פוגשים ראיה חדשה, ומעדכנים את אמונתנו באופן רציונלי ומתמטי. כפי שכתב תומאס בייס במאמרו המכונן [2], זהו התהליך שבאמצעותו האנושות לומדת מניסיון - עדכון מתמיד של אמונות לאור ראיות חדשות.

2.5 מימוש חישובי בפייתון

כדי להפוך את המושגים התיאורטיים למוחשיים, נראה כיצד ניתן לחשב את כלל בייס באמצעות קוד פייתון פשוט. הקוד הבא ממש את דוגמת השפעת-כאב-ראש שדנו בה:

הקוד לעיל מחשב את ההסתברות הא-פוסטריורית לשפעת בהינתן כאב ראש. הוא ממחיש את שלושת הרכיבים של כלל בייס: הא-פריורי $P(F)$, הנראות $P(H|F)$, והראיה $P(H)$. תוצאת ההרצה תהיה:

כפי שאנו רואים, הקוד מאשר את החישוב שביצענו ידנית בנוסחה 8. זהו מימוש בסיסי של הסקה בייסיאנית, שישמש כבסיס למסווג בייס המלא שנבנה בפרקים הבאים.

2.6 סיכום הפרק

בפרק זה הנחנו את היסודות המתמטיים הדרושים להבנת מסווג בייס. התחלנו מהשאלה הפשוטה: "האם אני חולה בשפעת אם יש לי כאב ראש?" ובנינו את כל המנגנון המתמטי הדרוש לענות עליה.

למדנו את ההבדל בין הסתברות לא-מותנית $P(H)$ להסתברות מותנית $P(H|F)$. ראינו כיצד כלל השרשרת מקשר בין הסתברות משותפת להסתברות מותנית: $P(H,F) = P(H|F) \cdot P(F)$. גילינו שהסימטריה של כלל השרשרת מובילה באופן טבעי לנוסחת בייס.

נוסחת בייס, $P(F|H) = \frac{P(H|F) \cdot P(F)}{P(H)}$, היא הלב של הגישה הבייסיאנית. היא מאפשרת לנו "להפוך" את כיוון ההסתברות המותנית, לעבור מהידוע (ההסתברות לתסמין בהינתן

חישוב כלל בייס: דוגמת שפעת וכאב ראש

```python
# Bayes Rule Calculation: Flu and Headache Example
# Based on the probability foundations discussed in Chapter 2

# Define prior probabilities (what we know before evidence)
P_headache = 0.1      # P(H): 10% of people have headaches
P_flu = 0.025         # P(F): 2.5% of people have flu (1/40)

# Define likelihood (probability of evidence given hypothesis)
P_headache_given_flu = 0.5  # P(H|F): 50% of flu patients get headaches

# Calculate posterior using Bayes' Rule
# P(F|H) = P(H|F) * P(F) / P(H)
P_flu_given_headache = (P_headache_given_flu * P_flu) / P_headache

# Display results
print("===␣Bayesian␣Inference:␣Flu␣Given␣Headache␣===")
print(f"Prior␣P(Flu):␣{P_flu:.3f}␣=␣{P_flu*100:.1f}%")
print(f"Likelihood␣P(Headache|Flu):␣{P_headache_given_flu:.3f}␣=␣"
      f"{P_headache_given_flu*100:.1f}%")
print(f"Evidence␣P(Headache):␣{P_headache:.3f}␣=␣{P_headache*100:.1f}%")
print(f"Posterior␣P(Flu|Headache):␣{P_flu_given_headache:.3f}␣=␣"
      f"{P_flu_given_headache*100:.1f}%")
print(f"\nConclusion:␣If␣you␣have␣a␣headache,␣"
      f"there␣is␣a␣{P_flu_given_headache*100:.1f}%␣chance␣of␣flu.")
```

פלט הריצה

```
=== Bayesian Inference: Flu Given Headache ===
Prior P(Flu): 0.025 = 2.5%
Likelihood P(Headache|Flu): 0.500 = 50.0%
Evidence P(Headache): 0.100 = 10.0%
Posterior P(Flu|Headache): 0.125 = 12.5%

Conclusion: If you have a headache, there is a 12.5% chance of flu.
```

מחלה) אל הלא-ידוע (ההסתברות למחלה בהינתן תסמין). זוהי בדיוק הבעיה שאנו פוגשים באבחון רפואי, בזיהוי דואר זבל, בסיווג תמונות, ובאינספור יישומים אחרים.

הבנו את שלושת המושגים המרכזיים של הסקה בייסיאנית: הא-פריורי (מה אנו יודעים לפני הראיה), הנראות (כמה טוב הראיה מסבירה את ההשערה), והא-פוסטריורי (מה אנו יודעים אחרי הראיה). שלושת המושגים הללו ילוו אותנו לאורך כל הספר.

דוגמת השפעת-כאב-ראש לימדה אותנו לקח חשוב: ראיה בודדת אינה מספיקה לאבחנה ודאית. עלינו תמיד לשקול את ההסתברות הבסיסית של המחלה באוכלוסייה. מחלה נדירה תישאר נדירה גם לאחר שמצאנו תסמין, במיוחד אם התסמין עצמו שכיח מסיבות אחרות.

בפרק הבא נרחיב את הרעיונות הללו למקרה הכללי של סיווג: במקום שתי אפשרויות בלבד (שפעת או לא-שפעת), נתמודד עם מספר שרירותי של קטגוריות. נראה כיצד מסווג בייס משתמש בעקרונות שלמדנו כדי לבחור את הקטגוריה הסבירה ביותר לכל תצפית. אך כל הבסיס המתמטי כבר הונח כאן, בפרק זה, דרך הסיפור הפשוט של בוקר אחד עם כאב ראש.

3 מסווג בייס

כל יום, בכל רגע נתון, אנו מקבלים החלטות תחת אי־וודאות. בנקים מחליטים האם לאשר הלוואה, רופאים מאבחנים מחלות, ומערכות אבטחה קובעות האם פעילות מסוימת חשודה. בכל אחת מההחלטות הללו טמון אתגר משותף: כיצד ניתן להפוך תצפיות לא מושלמות להחלטות רציונליות? התשובה המתמטית לשאלה זו מובילה אותנו אל אחד הכלים החזקים ביותר בלמידת מכונה — Bayes Classifier, או בעברית: מסווג בייס.

הפרק הקודם הציג את היסודות ההסתברותיים של כלל בייס. כעת נראה כיצד ניתן ליישם כלל זה כדי לבנות מסווג — אלגוריתם שמסוגל לקבל החלטות על בסיס נתונים. נתחיל בשאלה פשוטה: בהינתן תצפית חדשה, לאיזו קטגוריה היא שייכת? זוהי שאלת ליבה בתחום הסיווג, ומסווג בייס מספק תשובה מתמטית מדויקת המבוססת על תורת ההסתברות.

בפרק זה נלמד כיצד להפוך את כלל בייס ממשפט תיאורטי לכלי מעשי לסיווג. נציג דוגמה קונקרטית — בעיית דירוג אשראי — ונראה כיצד להמיר ספירות אמפיריות להסתברויות, וכיצד להשתמש בהסתברויות אלו כדי לקבל החלטות אופטימליות. נבין את תהליך הלמידה הדו־שלבי: תחילה נלמד את ההסתברויות הקדמיות של המחלקות, ולאחר מכן את ההסתברויות המותנות של התכונות. בסופו של דבר, נראה כיצד כלל ההחלטה argmax מאפשר לנו לבחור את המחלקה הסבירה ביותר.

3.1 יישום כלל בייס לסיווג

כאשר אנו מדברים על **סיווג** (classification), אנו מתכוונים למשימה של הקצאת תווית מחלקה לתצפית חדשה. למשל: האם אשראי של לקוח הוא טוב או רע? האם אימייל הוא ספאם או לגיטימי? האם תמונה מציגה חתול או כלב? בכל מקרה, יש לנו קבוצה סופית של מחלקות $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$, ועלינו לבחור את המחלקה המתאימה ביותר לתצפית הנתונה x.

השאלה המרכזית היא: על פי איזה קריטריון נבחר את המחלקה? מסווג בייס מציע תשובה ברורה ואלגנטית. הוא מבוסס על העיקרון הפשוט ש**נבחר את המחלקה בעלת ההסתברות הגבוהה ביותר בהינתן התצפית**. במילים אחרות, אנו מחפשים את המחלקה $\hat{y}$ שמקסימלית את ההסתברות האחורית $P(Y = c|X = x)$. זהו כלל ההחלטה שלנו[1], [8]:

$$\hat{y} = \arg\max_{c \in \mathcal{C}} P(Y = c|X = x) \tag{10}$$

נוסחה 01 היא לב ליבו של מסווג בייס. הפונקציה argmax מחזירה את הערך c שממקסם את הביטוי שאחריו. במילים פשוטות: אנו בוחרים את המחלקה שיש לה את ההסתברות הגבוהה ביותר להיות נכונה, בהינתן התצפית שלנו.

אך כיצד נחשב את ההסתברות האחורית $P(Y|X)$? כאן נכנס לתמונה כלל בייס, שלמדנו עליו בפרק הקודם. כלל בייס מאפשר לנו לחשב את ההסתברות האחורית באמצעות ההסתברות המותנית $P(X|Y)$ וההסתברות הקדמית $P(Y)$[7]:

$$P(Y=c|X=x) = \frac{P(X=x|Y=c)P(Y=c)}{P(X=x)} \tag{11}$$

נוסחה 11 היא יישום ישיר של כלל בייס לבעיית הסיווג. המונה מכיל שני איברים: $P(X=x|Y=c)$, שנקראת **הנראות** (likelihood), מודדת עד כמה סביר לראות את התצפית x בהינתן שהמחלקה היא c. האיבר השני, $P(Y=c)$, הוא **ההסתברות הקדמית** (prior probability) של המחלקה c — ההסתברות שלה לפני שראינו כלל נתונים. המכנה, $P(X=x)$, הוא ההסתברות של התצפית עצמה.

עתה מגיע שלב קריטי בהבנת מסווג בייס. כאשר אנו משווים בין מחלקות שונות עבור אותה תצפית x, המכנה $P(X=x)$ הוא **קבוע** — הוא זהה לכל המחלקות. זה אומר שבעת חישוב הargmax, נוכל להתעלם ממנו לחלוטין! נקבל אפוא ביטוי פשוט יותר:

$$\hat{y} = \arg\max_{c \in \mathcal{C}} P(X=x|Y=c)P(Y=c) \tag{12}$$

נוסחה 21 היא הצורה המעשית של מסווג בייס. היא אומרת לנו שכדי לסווג תצפית חדשה, עלינו לחשב עבור כל מחלקה את המכפלה של הנראות וההסתברות הקדמית, ולבחור את המחלקה שמניבה את הערך המקסימלי. זהו כלל ההחלטה המלא, והוא מהווה את הבסיס לכל הישומים של מסווג בייס.

מדוע כלל זה נחשב אופטימלי? מסתבר שמסווג בייס הוא **המסווג הטוב ביותר האפשרי** במובן מסוים. אם ההסתברויות שאנו משתמשים בהן נכונות, אז מסווג בייס ממזער את הסיכוי לטעות בסיווג[8]. זהו תוצאה תיאורטית חשובה: לא ניתן לעשות טוב יותר מזה, לפחות לא אם מודדים ביצועים לפי הסתברות השגיאה. כמובן, בפועל איננו יודעים את ההסתברויות האמיתיות, ועלינו לאמוד אותן מנתונים — וזה בדיוק התהליך שנראה בהמשך הפרק.

3.2 דוגמה: בעיית דירוג אשראי

כדי להבין כיצד מסווג בייס עובד בפועל, נבחן דוגמה קונקרטית: בעיית דירוג אשראי. נניח שאנו עובדים בבנק, ועלינו להחליט האם לאשר הלוואה ללקוח חדש. יש לנו מידע פשוט על הלקוח: רמת ההכנסה שלו (income), שיכולה להיות נמוכה, בינונית או גבוהה. המטרה שלנו היא לחזות את איכות האשראי (credit quality) של הלקוח: האם הוא בעל אשראי טוב או אשראי רע.

זהו מקרה פשוט אך מייצג של בעיית סיווג. יש לנו תכונה בודדת X (רמת הכנסה) עם שלושה ערכים אפשריים, ותווית מחלקה Y (איכות אשראי) עם שני ערכים אפשריים. נניח שיש לנו מסד נתונים היסטורי של לקוחות קודמים, שבו רשום עבור כל לקוח את רמת ההכנסה שלו ואת איכות האשראי שהתבררה בפועל. נתונים אלו מסוכמים בטבלה 3.

כפי שניתן לראות בטבלה 3, במסד הנתונים שלנו יש 200 דוגמאות (סכום כל התאים). עבור לקוחות עם הכנסה נמוכה, יש 42 מקרים של אשראי רע ורק 15 מקרים של אשראי טוב. התמונה הפוכה עבור לקוחות עם הכנסה גבוהה: 50 מקרים של אשראי טוב לעומת 18 מקרים של אשראי רע. התבנית מגלה אינטואיציה ברורה: ככל שההכנסה גבוהה יותר,

טבלה 3: נתוני דירוג אשראי לפי רמת הכנסה — ספירת דוגמאות עבור כל צירוף של רמת הכנסה ואיכות אשראי

רמת הכנסה	**אשראי רע**	**אשראי טוב**
נמוכה	42	15
בינונית	35	40
גבוהה	18	50

טבלה 4: pbth

כך פחות סביר שהאשראי יהיה רע.

כעת נניח שמגיע לקוח חדש עם הכנסה נמוכה. מה תהיה ההחלטה שלנו? האם נסווג אותו כבעל אשראי טוב או רע? כדי להשתמש במסווג בייס, עלינו לחשב עבור כל מחלקה את המכפלה $P(X = \text{נמוכה}|Y = c) \cdot P(Y = c)$, ולבחור את המחלקה עם הערך הגבוה יותר. אך לפני שנוכל לעשות זאת, עלינו להמיר את הספירות בטבלה 3 להסתברויות — וזהו הנושא של הסעיף הבא.

דוגמה זו אינה רק תרגיל אקדמי. בעיות דומות מתעוררות בכל מקום שבו יש צורך לקבל החלטות על סמך מידע היסטורי: אישור כרטיס אשראי, אבחון רפואי, איתור הונאות, סינון ספאם ועוד. המתודולוגיה זהה: אנו לומדים מהעבר (הנתונים ההיסטוריים) כדי לקבל החלטות בהווה (סיווג של תצפית חדשה). מסווג בייס מספק מסגרת מתמטית קפדנית לביצוע משימה זו.

3.3 מספירה להסתברות

הנתונים בטבלה 3 מוצגים בצורה של ספירות: כמה פעמים ראינו כל צירוף של רמת הכנסה ואיכות אשראי. כדי להשתמש במסווג בייס, עלינו להמיר ספירות אלו להסתברויות. תהליך זה הוא הלב של הלמידה ממסד נתונים, והוא מבוסס על עיקרון פשוט: **ההסתברות היא היחס בין מספר הפעמים שמאורע מסוים התרחש לסך כל המקרים**.

נתחיל בחישוב ההסתברויות הקדמיות $P(Y = c)$. אלו הן ההסתברויות של המחלקות לפני שראינו כל מידע על התכונה X. ההסתברות הקדמית של מחלקה c מחושבת לפי הנוסחה:

$$P(Y = c) = \frac{N_c}{N} \tag{13}$$

כאן N_c הוא מספר הדוגמאות במחלקה c, ו־N הוא סך כל הדוגמאות במסד הנתונים. במקרה שלנו, סך כל הדוגמאות הוא $N = 42 + 15 + 35 + 40 + 18 + 50 = 200$. מספר הדוגמאות עם אשראי רע הוא $N_{\text{רע}} = 42 + 35 + 18 = 95$, ומספר הדוגמאות עם אשראי טוב הוא $N_{\text{טוב}} = 15 + 40 + 50 = 105$. לפיכך:

$$P(Y = \text{רע}) = \frac{95}{200} = 0.475$$

$$P(Y = \text{טוב}) = \frac{105}{200} = 0.525$$

שימו לב שסכום ההסתברויות הוא 1, כפי שצריך להיות. אנו רואים שבמסד הנתונים שלנו יש מעט יותר לקוחות עם אשראי טוב מאשר עם אשראי רע, אך ההבדל קטן יחסית.

השלב הבא הוא לחשב את ההסתברויות המותנות $P(X = x|Y = c)$ — הנראות. אלו הן ההסתברויות לראות ערך מסוים של התכונה X, בהינתן שאנו יודעים את המחלקה Y. ההסתברות המותנית מחושבת לפי:

$$P(X = x|Y = c) = \frac{\text{count}(X = x, Y = c)}{\text{count}(Y = c)} \tag{14}$$

במילים: אנו סופרים כמה פעמים ראינו את $X = x$ ביחד עם $Y = c$, ומחלקים במספר הכולל של דוגמאות עם $Y = c$. לדוגמה, נחשב את $P(X = \text{נמוכה}|Y = \text{רע})$:

$$P(X = \text{נמוכה}|Y = \text{רע}) = \frac{42}{95} \approx 0.442$$

באופן דומה, נחשב את $P(X = \text{נמוכה}|Y = \text{טוב})$:

$$P(X = \text{נמוכה}|Y = \text{טוב}) = \frac{15}{105} \approx 0.143$$

ההבדל בין שתי ההסתברויות הללו הוא משמעותי. אם אשראי הוא רע, יש סיכוי של כ־44% שההכנסה תהיה נמוכה. לעומת זאת, אם האשראי טוב, יש רק כ־14% סיכוי להכנסה נמוכה. זה מעיד שיש קשר ברור בין רמת ההכנסה לאיכות האשראי.

נוכל לחשב את כל ההסתברויות המותנות עבור כל הצירופים של רמת הכנסה ומחלקה, ולסכם אותן בטבלה 5.

כפי שניתן לראות בטבלה 5, ההסתברויות המותנות עבור כל מחלקה מסתכמות ל־1. למשל, עבור מחלקת האשראי הרע: $0.442 + 0.368 + 0.189 = 0.999 \approx 1$ (ההבדל הקטן נובע מעיגולים). זה הגיוני: אם אנו יודעים שהאשראי רע, ההכנסה חייבת להיות אחת משלוש האפשרויות.

עתה יש בידינו את כל המרכיבים הנדרשים: ההסתברויות הקדמיות $P(Y)$ וההסתברויות המותנות $P(X|Y)$. השלב הבא הוא להשתמש בהן כדי לסווג תצפית חדשה — וזהו תהליך ההחלטה עצמו.

טבלה 5: הסתברויות קדמיות ומותנות מחושבות ממסד הנתונים

הסתברות	**ערך**
$P(Y = \text{ער})$	0.475
$P(Y = \text{בוט})$	0.525
$P(X = \text{הכומנ} \mid Y = \text{ער})$	0.442
$P(X = \text{הכומנ} \mid Y = \text{בוט})$	0.143
$P(X = \text{תינוניב} \mid Y = \text{ער})$	0.368
$P(X = \text{תינוניב} \mid Y = \text{בוט})$	0.381
$P(X = \text{ההובג} \mid Y = \text{ער})$	0.189
$P(X = \text{ההובג} \mid Y = \text{בוט})$	0.476

טבלה 6: pbth

3.4 תהליך הלמידה

תהליך הלמידה של מסווג בייס מתחלק לשני שלבים ברורים. בשלב הראשון, **נלמד מהנתונים** את ההסתברויות הדרושות: תחילה את ההסתברויות הקדמיות $P(Y=c)$, ולאחר מכן את ההסתברויות המותנות $P(X=x|Y=c)$. בשלב השני, **נשתמש בהסתברויות אלו לסיווג** של תצפיות חדשות באמצעות כלל הargmax. זוהי מסגרת קלאסית בלמידת מכונה: תחילה למידה (training), ולאחר מכן חיזוי (prediction).

נראה כיצד תהליך זה מתבצע בפועל באמצעות קוד Python. הקוד שלהלן ממחיש את שני השלבים:

הקוד לעיל ממחיש את שלבי הלמידה והסיווג בצורה ברורה. בשורות 6–12 אנו מגדירים את הנתונים ממסד הנתונים שלנו בצורה של מטריצה. כל שורה מתאימה לרמת הכנסה, וכל עמודה מתאימה לאיכות אשראי. לאחר מכן, בשורות 14–20, אנו מבצעים את שלב הלמידה: מחשבים את ההסתברויות הקדמיות ואת ההסתברויות המותנות. זהו השלב שבו אנו "לומדים" מהנתונים.

בשורות 22–35 מוגדרת פונקציית הסיווג. זוהי מימוש ישיר של כלל הargmax מנוסחה 21. עבור תצפית חדשה (רמת הכנסה), אנו מחשבים את המכפלה $P(X|Y)\cdot P(Y)$ עבור כל מחלקה, ובוחרים את המחלקה עם הערך הגבוה יותר. שימו לב שאיננו מחשבים את המכנה $P(X)$, כי הוא זהה לשתי המחלקות ואיננו משפיע על הargmax.

נבחן את התוצאות עבור שלוש התצפיות האפשריות:

1. **הכנסה נמוכה:**

$$P(X=\text{נמוכה}|Y=\text{רע})\cdot P(Y=\text{רע})=0.442\times 0.475\approx 0.210$$

$$P(X=\text{נמוכה}|Y=\text{טוב})\cdot P(Y=\text{טוב})=0.143\times 0.525\approx 0.075$$

מכיוון ש־0.210 גדול מ־0.075, המסווג יבחר באשראי רע. זה הגיוני: רוב הלקוחות עם הכנסה נמוכה במסד הנתונים שלנו היו בעלי אשראי רע.

2. **הכנסה בינונית:**

$$P(X=\text{בינונית}|Y=\text{רע})\cdot P(Y=\text{רע})=0.368\times 0.475\approx 0.175$$

$$P(X=\text{בינונית}|Y=\text{טוב})\cdot P(Y=\text{טוב})=0.381\times 0.525\approx 0.200$$

כאן הערך הגבוה יותר הוא 0.200, ולכן המסווג יבחר באשראי טוב.

3. **הכנסה גבוהה:**

$$P(X=\text{גבוהה}|Y=\text{רע})\cdot P(Y=\text{רע})=0.189\times 0.475\approx 0.090$$

מסווג בייס לדירוג אשראי

```python
import numpy as np

# Credit data: counts[income_level][credit_quality]
# income: 0=low, 1=medium, 2=high
# credit: 0=bad, 1=good
counts = np.array([
    [42, 15],   # low income
    [35, 40],   # medium income
    [18, 50]    # high income
])

# Stage 1: Learning - Calculate P(Y)
total = counts.sum()
P_bad = counts[:, 0].sum() / total
P_good = counts[:, 1].sum() / total

# Stage 1: Learning - Calculate P(X|Y)
P_X_given_bad = counts[:, 0] / counts[:, 0].sum()
P_X_given_good = counts[:, 1] / counts[:, 1].sum()

# Stage 2: Classification - Apply argmax rule
def classify(income_level):
    """
␣␣␣␣Classify␣credit␣quality␣using␣Bayes␣rule.

␣␣␣␣Args:
␣␣␣␣␣␣␣␣income_level:␣0=low,␣1=medium,␣2=high

␣␣␣␣Returns:
␣␣␣␣␣␣␣␣Predicted␣credit␣quality:␣"bad"␣or␣"good"
␣␣␣␣"""
    # Calculate posterior probabilities (unnormalized)
    p_bad = P_X_given_bad[income_level] * P_bad
    p_good = P_X_given_good[income_level] * P_good

    # Return argmax
    return "bad" if p_bad > p_good else "good"

# Example usage
print(f"Low␣income␣->␣{classify(0)}")
print(f"Medium␣income␣->␣{classify(1)}")
print(f"High␣income␣->␣{classify(2)}")
```


$$P(X = \text{ההובג} | Y = \text{בוט}) \cdot P(Y = \text{בוט}) = 0.476 \times 0.525 \approx 0.250$$

ההבדל כאן משמעותי: 0.250 גדול בהרבה מ־0.090. המסווג בוודאי יבחר באשראי טוב.

התוצאות האלו מייצגות את ההיגיון שאנו מצפים לו: ככל שההכנסה גבוהה יותר, כך סביר יותר שהאשראי יהיה טוב. מסווג בייס הצליח ללכוד את הקשר הזה באופן אוטומטי, ישירות מהנתונים, ללא צורך בהנחות נוספות או כללי החלטה ידניים.

יש לציין נקודה חשובה נוספת: תהליך הלמידה כאן הוא **דו־שלבי**. תחילה למדנו את ההסתברויות הקדמיות $P(Y)$, ולאחר מכן את ההסתברויות המותנות $P(X|Y)$. זוהי מסגרת אופיינית למודלים גנרטיביים (generative models)[1], שבהם אנו לומדים את ההתפלגות המשותפת של X ו־Y, ולא רק את ההסתברות המותנית $P(Y|X)$ ישירות. היתרון של גישה זו הוא שאנו מבינים את המבנה ההסתברותי הבסיסי של הנתונים, ויכולים להשתמש בו לא רק לסיווג אלא גם למטרות אחרות, כמו ייצור דוגמאות חדשות או איתור אנומליות.

לסיכום, מסווג בייס מספק מסגרת פשוטה ועוצמתית לפתרון בעיות סיווג. הוא מבוסס על עקרונות הסתברותיים ברורים, קל ליישום, ומניב החלטות שאפשר להסביר ולהצדיק. למרות הפשטות, הוא עובד היטב במצבים רבים, במיוחד כאשר מספר התכונות אינו גדול מדי ומסד הנתונים מספיק גדול כדי לאמוד באופן אמין את ההסתברויות. בפרקים הבאים נראה כיצד ניתן להרחיב את העקרונות הללו למצבים מורכבים יותר, ונבחן שיטות נוספות לסיווג המבוססות על רעיונות דומים.

4 מודלים להתפלגות

בפרקים הקודמים ראינו כיצד מסווג Bayes משתמש בהסתברויות כדי לקבל החלטות. אך כאשר ניגשנו לדוגמאות מעשיות, הסתכלנו רק על מקרים פשוטים בהם המשתנים היו דיסקרטיים ואפשר היה לספור את התדירויות בטבלה. מה קורה כאשר המידע שלנו הוא רציף? איך נעריך את $P(X|Y)$ כאשר X יכול לקבל אינסוף ערכים אפשריים, כמו גובה, משקל, או טמפרטורה?

התשובה טמונה בבחירה נכונה של **מודל התפלגות** — דרך מתמטית לתאר כיצד הערכים מתפזרים במרחב האפשרויות. בחירה זו איננה רק עניין טכני, אלא החלטה קריטית המשפיעה ישירות על ביצועי המסווג ועל היכולת שלנו להכליל מהדוגמאות שראינו לדוגמאות חדשות. בפרק זה נלמד כיצד לבחור מודלים התואמים לסוג הנתונים, כיצד לאמוד את הפרמטרים שלהם מהדאטה, וכיצד להשתמש בהם בתוך מסווג Bayes.

4.1 בחירת המודל: מדיסקרטי לרציף

כאשר אנו בוחרים מודל התפלגות, השאלה הראשונה שעלינו לשאול היא: מהו סוג הנתונים שאנו מנסים לתאר? האם המשתנה שלנו דיסקרטי — קטגוריה מתוך רשימה קבועה של אפשרויות — או רציף, כמו מדידה פיזיקלית שיכולה לקבל כל ערך ממשי?

ההבחנה הזו איננה רק פורמלית. המודלים המתמטיים שאנו משתמשים בהם, והדרך בהם אנו מעריכים פרמטרים, שונים באופן מהותי בין שני המקרים הללו. כפי שמציינים [1], "השימוש במודל לא מתאים לסוג הנתונים יכול להוביל לאמידות שגויות ולביצועי סיווג גרועים." בואו נתחיל במקרה הפשוט יותר — המשתנים הדיסקרטיים.

4.2 משתנים דיסקרטיים: ספירה ותדירות

כאשר המשתנה שלנו דיסקרטי, המצב הוא יחסית פשוט. נניח שאנו רוצים לסווג אימיילים לספאם או לא-ספאם על סמך נוכחות מילים מסוימות. המשתנה "האם המילה 'זכייה' מופיעה באימייל?" יכול לקבל רק שני ערכים: כן או לא. זהו משתנה **בינארי**, והוא נמצא בבסיס מודל התפלגות פשוט הנקרא Bernoulli distribution.

התפלגות Bernoulli מתארת ניסוי בודד עם שתי תוצאות אפשריות. אם נסמן אחת מהתוצאות כ"הצלחה" (למשל, מילת הספאם מופיעה), אזי ההסתברות להצלחה היא θ, וההסתברות לכישלון היא $1-\theta$. באופן פורמלי:

$$P(X=x|\theta)=\theta^x(1-\theta)^{1-x}, \quad x\in\{0,1\} \tag{15}$$

נוסחה 51 מתארת את ההסתברות לקבל את הערך x (כאשר $x=1$ מייצג הצלחה ו-$x=0$ כישלון) בהינתן הפרמטר θ. כאשר $x=1$, הנוסחה מצטמצמת ל-θ, וכאשר $x=0$ היא נותנת $1-\theta$. זהו ביטוי מתמטי קומפקטי המאפשר לנו לכתוב פונקציה אחת עבור שתי התוצאות.

איך נעריך את הפרמטר θ? הדרך הפשוטה ביותר היא **ספירה**. אם ראינו n אימיילי ספאם בעבר, וב-k מהם הופיעה המילה "זכייה", אזי האומדן הפשוט שלנו יהיה:

$$\hat{\theta} = \frac{k}{n} \tag{16}$$

זהו **אומדן נראות מקסימלית** (Maximum Likelihood Estimate, MLE) — הערך של θ שמסביר את התצפיות שלנו בצורה הטובה ביותר. למרות הפשטות, זהו אומדן יעיל וחזק כאשר יש לנו מספיק דוגמאות אימון [8].

כאשר יש לנו יותר משתי אפשרויות, אנו עוברים להתפלגות Multinomial. דמיינו שאנו מסווגים מסמכים לפי נושא: פוליטיקה, ספורט, כלכלה, או תרבות. כאן יש ארבע קטגוריות אפשריות. עבור כל קטגוריה i יש הסתברות θ_i, כאשר הסכום של כל ההסתברויות הוא 1:

$$\sum_{i=1}^{K} \theta_i = 1 \tag{17}$$

גם כאן, אומדן הפרמטרים הוא ישיר: עבור כל קטגוריה i, נספור כמה פעמים ראינו אותה בדאטה ונחלק במספר הכולל של הדוגמאות:

$$\hat{\theta}_i = \frac{n_i}{N} \tag{18}$$

כאשר n_i הוא מספר הפעמים שראינו קטגוריה i, ו-N הוא המספר הכולל של הדוגמאות. המודל הזה עובד מצוין כאשר הנתונים באמת דיסקרטיים, והמרחב של האפשרויות לא גדול מדי.

4.3 משתנים רציפים: מדיסקרטיזציה להערכת צפיפות

אך מה קורה כאשר המשתנה שלנו רציף? נניח שאנו רוצים לסווג פרחים לפי אורך העלה-הכותרת (petal length). זהו מספר ממשי, והוא יכול לקבל אינסוף ערכים אפשריים — 3.2 ס"מ, 3.21 ס"מ, 3.215 ס"מ, וכן הלאה. הרעיון של "ספירה" כבר לא עובד, כי הסיכוי לראות בדיוק את הערך 3.2 פעמיים הוא אפסי.

במקום לדבר על הסתברות לערך ספציפי, אנו מדברים על **צפיפות הסתברות** (probability density function, PDF). זוהי פונקציה רציפה $p(x)$ שמתארת כיצד "מרוכזת" ההסתברות בכל אזור של ציר הערכים. ההסתברות שהערך יפול בתוך טווח מסוים $[a, b]$ היא האינטגרל של הצפיפות על הטווח הזה:

$$P(a \leq X \leq b) = \int_a^b p(x)\, dx \tag{19}$$

איך נעריך צפיפות הסתברות מהדאטה? הדרך הפשוטה ביותר היא **היסטוגרמה**. אנו מחלקים את ציר הערכים לתאים קטנים (למשל, כל תא ברוחב של 0.5 ס"מ), וסופרים כמה דוגמאות נפלו בכל תא. לאחר מכן, אנו מנרמלים את הספירות כך שהשטח הכולל תחת ההיסטוגרמה יהיה 1. התוצאה היא קירוב גס לפונקציית הצפיפות האמיתית.

אמנם ההיסטוגרמה פשוטה להבנה ולחישוב, אך יש לה מגבלות. הבחירה של רוחב

התאים היא שרירותית וישפיע משמעותית על הצורה של ההתפלגות המוערכת. תאים רחבים מדי יגרמו לאיבוד של פרטים עדינים, ותאים צרים מדי יגרמו ל"רעש" עודף. מה שחסר לנו הוא מודל גמיש יותר, שיכול להתאים את עצמו לצורת הדאטה בצורה חלקה ויעילה.

4.4 מודל גאוסיאני חד-ממדי: עקומת הפעמון

המודל הנפוץ והשימושי ביותר למשתנים רציפים הוא **ההתפלגות הנורמלית** או **הגאוסיאנית** (Gaussian או Normal distribution). זוהי עקומת הפעמון המפורסמת, המופיעה בכל מקום — מגבהים של בני אדם ועד לשגיאות מדידה במעבדה. למרות השכיחות המדהימה שלה בטבע, התפלגות זו איננה "חוק יסוד" של המציאות. היא מופיעה לעיתים קרובות בגלל **משפט הגבול המרכזי** (Central Limit Theorem): כאשר משתנה מושפע מסכום של גורמים רנדומליים רבים ובלתי תלויים, ההתפלגות שלו תהיה קרובה לגאוסיאנית, ללא תלות בהתפלגות של הגורמים הבודדים [1].

התפלגות גאוסיאנית חד-ממדית מוגדרת על ידי שני פרמטרים: הממוצע μ והשונות σ^2. פונקציית הצפיפות היא:

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{20}$$

נוסחה 02 מתארת את צפיפות ההסתברות לקבל ערך x בהינתן ממוצע μ ושונות σ^2. הממוצע קובע את מיקום המרכז של הפעמון, והשונות קובעת את "רוחב" הפיזור. ככל שהשונות גדולה יותר, העקומה רחבה יותר ונמוכה יותר. הקבוע $\frac{1}{\sqrt{2\pi\sigma^2}}$ מבטיח שהשטח תחת העקומה הוא בדיוק 1, כנדרש מפונקציית צפיפות.

איך נעריך את μ ו-σ^2 מהדאטה? גם כאן, אומדן הנראות המקסימלית נותן תוצאות אינטואיטיביות ופשוטות. נניח שיש לנו N דוגמאות $x_1, x_2, \ldots, x_N$. האומדן למוצע הוא פשוט הממוצע האריתמטי:

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{21}$$

והאומדן לשונות הוא הממוצע של הריבועים של הסטיות מהממוצע:

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N} (x_i - \hat{\mu})^2 \tag{22}$$

נוסחאות 12 ו-22 נותנות לנו את אומדני MLE לפרמטרים של ההתפלגות הגאוסיאנית. הן מבטאות את הרעיון הפשוט שהממוצע של הדאטה הוא האומדן הטוב ביותר לממוצע ההתפלגות, והשונות מודדת כמה בממוצע הדוגמאות מתרחקות מהמרכז.

בואו נראה דוגמה קונקרטית. נניח שאספנו מדידות של אורך עלי-כותרת מ-100 פרחים מסוג Iris setosa. הממוצע שקיבלנו הוא 1.46 ס"מ, וסטיית התקן (σ, השורש של השונות) היא 0.17 ס"מ. כעת, עבור כל ערך חדש x, אנו יכולים לחשב את הצפיפות $p(x|\text{setosa})$ באמצעות נוסחה 02. זהו בדיוק מה שאנו צריכים כדי לחשב את $P(X|Y)$ במסווג Bayes.

4.5 סיווג עם מודל גאוסיאני: דוגמה חישובית

כדי להבין כיצד משתמשים במודל הגאוסיאני בתוך מסווג Bayes, בואו נבחן דוגמה מספרית פשוטה. נניח שאנו רוצים לסווג פרחים לשתי קטגוריות: Iris setosa ו-Iris versicolor, על סמך מאפיין בודד — אורך עלה-הכותרת (ב-ס"מ).

לאחר שאספנו דאטה אימון, העריכנו עבור כל קטגוריה את הפרמטרים של ההתפלגות הגאוסיאנית:

$$\text{רובע setosa:} \quad \mu_0 = 1.46, \quad \sigma_0 = 0.17 \tag{23}$$

$$\text{רובע versicolor:} \quad \mu_1 = 4.26, \quad \sigma_1 = 0.47 \tag{24}$$

כעת, כאשר מגיע פרח חדש עם אורך עלה-כותרת של $x = 3.0$ ס"מ, אנו רוצים לחשב לאיזו קטגוריה הוא שייך. על פי כלל Bayes, אנו צריכים לחשב את ההסתברות האחורית לכל קטגוריה:

$$P(Y = k|X = 3.0) \propto P(X = 3.0|Y = k) \cdot P(Y = k) \tag{25}$$

נניח שההסתברויות הקודמות (priors) שוות: $P(Y = \text{setosa}) = P(Y = \text{versicolor}) = 0.5$.
כעת נחשב את הצפיפויות המותנות באמצעות נוסחה 02:

$$\begin{aligned} p(x = 3.0|\text{setosa}) &= \frac{1}{\sqrt{2\pi \cdot 0.17^2}} \exp\left(-\frac{(3.0 - 1.46)^2}{2 \cdot 0.17^2}\right) \\ &\approx 2.35 \times 10^{-9} \end{aligned} \tag{26}$$

$$\begin{aligned} p(x = 3.0|\text{versicolor}) &= \frac{1}{\sqrt{2\pi \cdot 0.47^2}} \exp\left(-\frac{(3.0 - 4.26)^2}{2 \cdot 0.47^2}\right) \\ &\approx 0.18 \end{aligned} \tag{27}$$

אנו רואים שהצפיפות עבור versicolor גבוהה בהרבה מזו של setosa. לכן, המסווג יבחר ב-versicolor כקטגוריה החזויה. זה הגיוני אינטואיטיבית: אורך של 3.0 ס"מ קרוב הרבה יותר לממוצע של versicolor (4.26 ס"מ) מאשר לממוצע של setosa (1.46 ס"מ).

הקוד הבא ב-Python מממש את התהליך הזה:

הקוד הזה מדגים את התהליך המלא: אמידת פרמטרים מדאטה אימון, ושימוש בהם כדי לסווג תצפית חדשה. הפונקציה `norm.pdf` מחשבת את צפיפות ההסתברות הגאוסיאנית, והסיווג נעשה על ידי השוואה של ההסתברויות האחוריות.

4.6 מודל גאוסיאני רב-ממדי: מעבר לממד אחד

עד כה עסקנו במקרה חד-ממדי, בו יש לנו רק מאפיין בודד למדידה. אך ברוב היישומים המעשיים, יש לנו **וקטור מאפיינים** — מספר מדידות שונות על אותו אובייקט. לדוגמה, בבעיית סיווג הפרחים של Fisher, יש לנו ארבעה מאפיינים: אורך ורוחב של עלה-הכותרת, ואורך ורוחב של הגביע. איך נתאר התפלגות על מרחב רב-ממדי?

סיווג בייסיאני עם מודל גאוסיאני חד-ממדי

```python
import numpy as np
from scipy.stats import norm

# Estimate Gaussian parameters from training data
class_0_data = np.random.normal(loc=1.46, scale=0.17, size=100)  #
setosa
class_1_data = np.random.normal(loc=4.26, scale=0.47, size=100)  #
versicolor

mu_0, sigma_0 = np.mean(class_0_data), np.std(class_0_data)
mu_1, sigma_1 = np.mean(class_1_data), np.std(class_1_data)

# Classification function
def classify_gaussian(x, prior_0=0.5, prior_1=0.5):
    """Classify␣new␣observation␣using␣Gaussian␣models."""
    p_x_given_0 = norm.pdf(x, mu_0, sigma_0)
    p_x_given_1 = norm.pdf(x, mu_1, sigma_1)

    posterior_0 = p_x_given_0 * prior_0
    posterior_1 = p_x_given_1 * prior_1

    return 0 if posterior_0 > posterior_1 else 1

# Classify new observation
x_new = 3.0
predicted_class = classify_gaussian(x_new)
print(f"For␣x␣=␣{x_new},␣predicted␣class:␣{predicted_class}")
```

ההתפלגות הגאוסיאנית הרב-ממדית (Multivariate Gaussian Distribution) היא הכללה ישירה של המקרה החד-ממדי לממד d כלשהו. במקום ממוצע בודד μ ושונות σ^2, יש לנו:

- **וקטור ממוצע** $\boldsymbol{\mu}$ בגודל $d \times 1$, שמכיל את הממוצע של כל מאפיין.
- **מטריצת קווריאנס** Σ בגודל $d \times d$, שמכילה את השונות של כל מאפיין על האלכסון, ואת הקווריאנס (מידת הקשר הליניארי) בין כל זוג מאפיינים מחוץ לאלכסון.

פונקציית הצפיפות של התפלגות גאוסיאנית רב-ממדית היא:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right) \tag{28}$$

נוסחה 82 נראית מסובכת במבט ראשון, אך היא מבטאת רעיון דומה לזה של המקרה החד-ממדי. הביטוי $(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})$ נקרא **מרחק מהלנוביס** (Mahalanobis distance) — זהו מידה של המרחק בין הנקודה $\mathbf{x}$ לבין המרכז $\boldsymbol{\mu}$, כאשר המרחק מתוקנן על פי מבנה הקווריאנס. הקבוע $|\Sigma|$ הוא הדטרמיננטה של מטריצת הקווריאנס, והוא מבטיח נרמול נכון.

מטריצת הקווריאנס Σ מקודדת מידע חשוב על הקשרים בין המאפיינים. אם המאפיינים בלתי תלויים זה בזה, המטריצה תהיה אלכסונית (כל הערכים מחוץ לאלכסון יהיו אפס). אם יש קורלציה חיובית בין שני מאפיינים, הקווריאנס ביניהם יהיה חיובי, ואם הקורלציה שלילית — הקווריאנס יהיה שלילי.

אומדן הפרמטרים של התפלגות גאוסיאנית רב-ממדית הוא הכללה ישירה של המקרה החד-ממדי. נניח שיש לנו N דוגמאות אימון $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, כאשר כל $\mathbf{x}_i$ הוא וקטור בגודל $d \times 1$. אומדן MLE לוקטור הממוצע הוא:

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{29}$$

זהו פשוט הממוצע של כל המאפיינים על פני כל הדוגמאות. אומדן MLE למטריצת הקווריאנס הוא:

$$\hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T \tag{30}$$

שימו לב שזהו מכפלה חיצונית (outer product) של וקטור בעצמו, והתוצאה היא מטריצה. האיבר (i, j) במטריצת הקווריאנס מודד את הקווריאנס בין המאפיין ה-i והמאפיין ה-j.

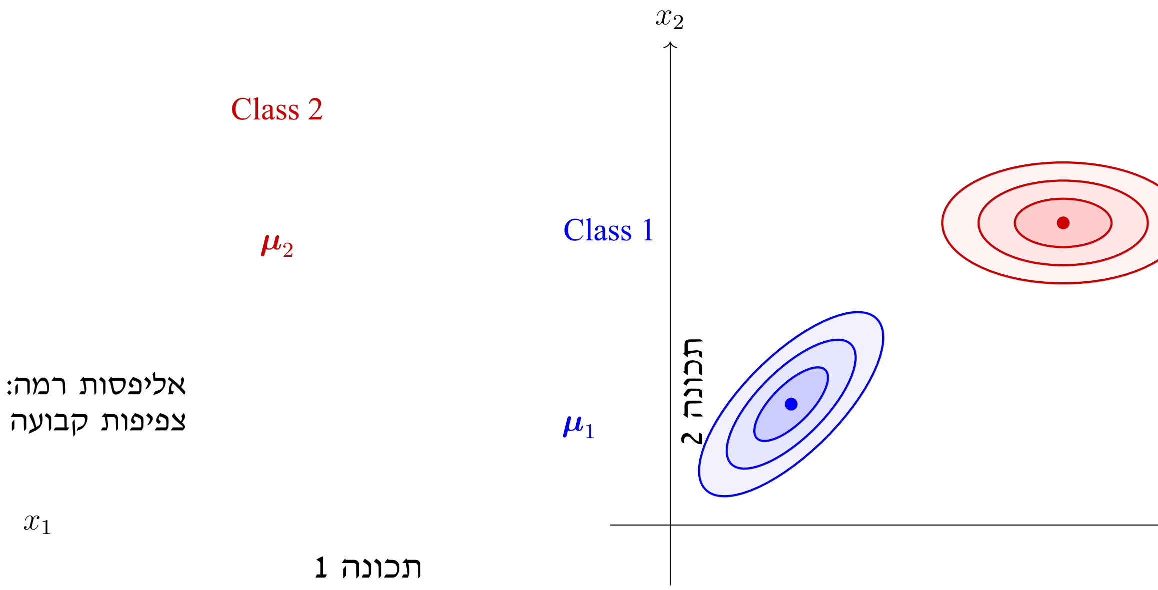
4.7 ייצוג גיאומטרי של התפלגות רב-ממדית

איך נוכל לדמיין התפלגות גאוסיאנית רב-ממדית? במקרה הדו-ממדי ($d = 2$), קווי-רמה של פונקציית הצפיפות יוצרים אליפסות במישור. המרכז של האליפסה הוא וקטור הממוצע $\boldsymbol{\mu}$, והכיוונים והגדלים של צירי האליפסה נקבעים על ידי הערכים העצמיים (eigenvalues) והוקטורים העצמיים (eigenvectors) של מטריצת הקווריאנס Σ.

[כאן דרוש איור: ייצוג גיאומטרי של התפלגות גאוסיאנית דו-ממדית עם אליפסות רמה]

הערה לאיור: נא לצייר מערכת צירים דו-ממדית עם שתי התפלגויות גאוסיאניות שונות, כאשר כל אחת מיוצגת על ידי אליפסות רמה. האליפסות צריכות להיות במיקומים שונים (ממוצעים שונים) ובזוויות שונות (מטריצות קוווריאנס שונות). לדוגמה, אליפסה אחת במרכז (2,2) עם ציר ראשי לאורך הקו x=y, ואליפסה שנייה במרכז (4,6) עם ציר ראשי אופקי. יש לסמן את המרכזים בנקודות, ולהוסיף כיתוב המסביר שהאליפסות מייצגות רמות שונות של צפיפות הסתברות.

איור 6 מדגים כיצד שתי קבוצות שונות (למשל, שני מינים של פרחים) יכולות להיות מיוצגות על ידי התפלגויות גאוסיאניות במרחב דו-ממדי. כל אליפסה מקיפה אזור בו הצפיפות גבוהה, ולכן יש סבירות גבוהה למצוא נקודות מהקבוצה הזו. המרחק והחפיפה בין האליפסות קובעים את קושי בעיית הסיווג: אם האליפסות רחוקות ולא חופפות, הסיווג יהיה קל ומדויק. אם יש חפיפה משמעותית, יהיו תצפיות שקשה לסווג בוודאות גבוהה.



איור 6: ייצוג גיאומטרי של שתי התפלגויות גאוסיאניות דו-ממדיות. האליפסות מייצגות קווי-רמה של צפיפות ההסתברות. המרכז של כל אליפסה הוא וקטור הממוצע, והכיוון והצורה נקבעים על ידי מטריצת הקווריאנס.

4.8 אמידת פרמטרים: מתיאוריה למעשה

ראינו את הנוסחאות לאומדן MLE של פרמטרי ההתפלגות הגאוסיאנית. אך בפועל, האמידה מעלה מספר שאלות מעשיות חשובות. כמה דוגמאות אנו צריכים כדי לקבל אומדן אמין? מה קורה כאשר מספר המאפיינים גדול יחסית למספר הדוגמאות? איך נתמודד עם דאטה חסר או רועש?

גודל המדגם ואיכות האמידה. באופן כללי, ככל שיש לנו יותר דוגמאות אימון, האומדן שלנו יהיה קרוב יותר לערכים האמיתיים של הפרמטרים. אבל כמה זה "מספיק"? עבור התפלגות גאוסיאנית ב-d ממדים, אנו צריכים לאמוד d ממוצעים ו-$\frac{d(d+1)}{2}$ פרמטרים במטריצת הקווריאנס (השונויות והקווריאנסים הייחודיים). בסך הכל, זה $\frac{d(d+3)}{2}$ פרמטרים.

ככלל אצבע, רצוי שיהיה לנו לפחות פי 10 דוגמאות ממספר הפרמטרים כדי לקבל אמידה יציבה [8].

בעיית הממד הגבוה. כאשר מספר המאפיינים גדול (d גדול), מספר הפרמטרים שצריך לאמוד גדל בצורה ריבועית. אם יש לנו מעט דוגמאות יחסית, מטריצת הקווריאנס עלולה להיות "דלילה" (רנק חסר) או לא יציבה מספרית. בפועל, זה אומר שההופכית Σ^{-1} שמופיעה בנוסחה 82 לא תהיה מוגדרת היטב.

פתרון אחד הוא **הנחות פשטות**. למשל, אם נניח שכל המאפיינים בלתי תלויים זה בזה (הנחה ה"naive" של מסווג Naive Bayes), מטריצת הקווריאנס תהיה אלכסונית, ונצטרך לאמוד רק d שונויות במקום $\frac{d(d+1)}{2}$ פרמטרים. אמנם זו הנחה חזקה ולא תמיד נכונה, אך היא מצמצמת דרמטית את מספר הפרמטרים ומאפשרת אמידה אמינה גם עם פחות דאטה.

פתרון נוסף הוא **רגולריזציה** — הוספת איבר קטן לאלכסון של מטריצת הקווריאנס כדי להבטיח שהיא תהיה הפיכה:

$$\hat{\mathbf{\Sigma}}_{\text{reg}} = \hat{\mathbf{\Sigma}} + \lambda \mathbf{I} \tag{31}$$

כאשר λ הוא פרמטר קטן (למשל, 0.01), ו-$\mathbf{I}$ היא מטריצת היחידה. זה מקטין מעט את האומדן של הקווריאנסים, אך מבטיח יציבות מספרית. [9] מציע שיטות מתוחכמות יותר לרגולריזציה, המאפשרות למצוא איזון אופטימלי בין דיוק לגמישות.

הקוד הבא מדגים אמידת פרמטרים עבור מודל גאוסיאני רב-ממדי:

אמידת פרמטרים למודל גאוסיאני רב-ממדי

```
import numpy as np

def estimate_gaussian_params(data):
    """Estimate␣mean␣vector␣and␣covariance␣matrix␣for␣multivariate␣
Gaussian."""
    N, d = data.shape

    # Estimate mean (equation \ref{eq:multivariate_mean_mle})
    mu = np.mean(data, axis=0)

    # Estimate covariance (equation \ref{eq:multivariate_covariance_mle
})
    centered = data - mu
    Sigma = (centered.T @ centered) / N

    return mu, Sigma

def regularize_covariance(Sigma, lambda_reg=0.01):
    """Apply␣regularization␣to␣covariance␣matrix."""
    d = Sigma.shape[0]
    Sigma_reg = Sigma + lambda_reg * np.eye(d)
    return Sigma_reg

# Generate synthetic 2D data for two classes
np.random.seed(42)

# Class 0: mean [2, 3], covariance with correlation
mu_0_true = np.array([2, 3])
Sigma_0_true = np.array([[1.0, 0.5], [0.5, 1.5]])
data_0 = np.random.multivariate_normal(mu_0_true, Sigma_0_true, size
=100)

# Class 1: mean [6, 5], covariance with different correlation
mu_1_true = np.array([6, 5])
Sigma_1_true = np.array([[2.0, -0.8], [-0.8, 1.0]])
data_1 = np.random.multivariate_normal(mu_1_true, Sigma_1_true, size
=100)

# Estimate parameters
mu_0_est, Sigma_0_est = estimate_gaussian_params(data_0)
mu_1_est, Sigma_1_est = estimate_gaussian_params(data_1)

print("Class␣0␣-␣Estimated␣mean:", mu_0_est)
print("Class␣0␣-␣Estimated␣covariance:\n", Sigma_0_est)
```

הקוד הזה מממש את נוסחאות האמידה שראינו, ומדגים כיצד ניתן לאמוד פרמטרים מדאטה דו-ממדי. בפועל, אנו נשתמש באומדנים הללו בתוך מסווג Bayes כדי לחשב את $P(X|Y)$ עבור תצפיות חדשות.

4.9 סיכום: מהבחירה במודל לסיווג

בפרק זה עברנו מסע מעמיק דרך מודלים להתפלגות, מהמקרים הפשוטים של משתנים דיסקרטיים ועד למודלים גאוסיאניים רב-ממדיים מתוחכמים. ראינו כיצד בחירת המודל הנכון תלויה בסוג הנתונים שלנו — אם הם דיסקרטיים או רציפים, אם יש תלות בין מאפיינים, וכמה דוגמאות אימון יש לנו.

המסר המרכזי הוא זה: אין מודל אחד שמתאים לכל בעיה. המודל הגאוסיאני נפוץ ונוח, אך הוא לא תמיד הנכון. כאשר הדאטה באמת מגיע מהתפלגות גאוסיאנית (או קרוב לה), השימוש במודל הזה יהיה יעיל ומדויק. אך כאשר ההתפלגות האמיתית רחוקה מגאוסיאנית — למשל, אם יש כמה "פסגות" במקום פעמון אחד, או אם יש ערכי קיצון (outliers) רבים — המודל הגאוסיאני עלול להוביל לתוצאות גרועות.

הכלים שלמדנו כאן — ספירה לדיסקרטי, אמידת צפיפות לרציף, מודל גאוסיאני חד-ורב-ממדי, ואומדן MLE לפרמטרים — הם אבני הבניין של מסווגים גנרטיביים רבים. בפרק הבא נראה כיצד לשלב אותם עם כללי החלטה אופטימליים, וכיצד להעריך את ביצועי המסווג שבנינו.

הדרך שעברנו מלמדת אותנו על הקשר ההדוק בין תיאוריה מתמטית ליישום מעשי. הנוסחאות שראינו אינן רק ביטויים אבסטרקטיים, אלא כלים שניתן לממש בקוד ולהשתמש בהם כדי לפתור בעיות אמיתיות. ההבנה של המודלים הללו, היתרונות והמגבלות שלהם, היא המפתח לבניית מסווגים חזקים ואמינים.

5 Naive Bayes

5.1 קללת הממדיות

כאשר בוחנים את האלגוריתם של Bayes הקלאסי, מתגלה אתגר מתמטי מהותי שנובע ישירות ממבנה המרחב הסתברותי. כדי לחשב את ההסתברות המותנית $P(X_1, X_2, \ldots, X_n|C)$ עבור n תכונות, נדרש לאמוד את כל צירופי הערכים האפשריים של התכונות בהינתן כל קטגוריה. אם כל תכונה יכולה לקבל $|X|$ ערכים שונים, ויש $|C|$ קטגוריות אפשריות, מספר הפרמטרים שיש לאמוד מגיע ל־$O(|X|^n \cdot |C|)$. זוהי אכן קללה — גידול אקספוננציאלי שהופך את האלגוריתם לבלתי ישים עבור מרחבי תכונות גדולים.

בעיה זו, המכונה **קללת הממדיות** (Curse of Dimensionality), מעמידה חסם פרקטי חמור. עבור 10 תכונות בלבד, כאשר לכל אחת 10 ערכים אפשריים ו־2 קטגוריות, נדרש לאמוד $20 = 2 \cdot 10^{10}$ מיליארד פרמטרים. כמות הנתונים הנדרשת לאמידה סטטיסטית אמינה של פרמטרים רבים כל כך חורגת בהרבה ממה שעומד לרשות רוב היישומים המעשיים. המציאות הזו מובילה למסקנה ברורה: יש למצוא דרך לצמצם את מרחב הפרמטרים מבלי לוותר על כוח החיזוי של המודל.

5.2 הנחת אי־תלות

הפתרון המבריק שמציע אלגוריתם Naive Bayes נעוץ בהנחה מפשטת אך עוצמתית: כל התכונות בלתי תלויות זו בזו בהינתן הקטגוריה. הנחה זו, הידועה כ**הנחת אי־התלות המותנית** (Conditional Independence Assumption), מאפשרת לפרק את ההסתברות המשותפת למכפלה של הסתברויות שוליות:

$$P(X_1, X_2, \ldots, X_n|C) = \prod_{i=1}^{n} P(X_i|C) \tag{32}$$

במקום לאמוד הסתברות משותפת מורכבת אחת, האלגוריתם מתמקד באמידת n הסתברויות פשוטות יותר. כל אחת מהן בוחנת את ההתפלגות של תכונה בודדת בהינתן הקטגוריה, וכך מספר הפרמטרים יורד באופן דרמטי מ־$O(|X|^n \cdot |C|)$ ל־$O(n \cdot |X| \cdot |C|)$. זהו מעבר מגידול אקספוננציאלי לגידול לינארי — שינוי שהופך את האלגוריתם לישים גם במרחבים בעלי מימדים רבים.

חשוב להבין שהנחה זו נקראת "נאיבית" (Naive) מסיבה טובה: במציאות, תכונות רבות אכן תלויות זו בזו. במשימת סיווג דואר זבל, למשל, הופעת המילה "זכית" תלויה באופן ברור בהופעת המילה "פרס". עם זאת, הפשטות המתמטית שמביאה הנחה זו מתגמלת את המודל ביתרונות חישוביים ניכרים, והתוצאות המעשיות מראות שהאלגוריתם מצליח להניב ביצועים טובים גם כאשר ההנחה אינה מתקיימת במלואה.

5.3 נוסחת Naive Bayes

בהינתן הנחת אי־התלות, ניתן לגזור את כלל הסיווג של Naive Bayes ישירות מכלל Bayes המלא. עבור דוגמה חדשה עם תכונות $X = (X_1, X_2, \ldots, X_n)$, הקטגוריה המיוחסת היא:

$$C_{\text{NB}} = \arg\max_{c \in C} P(c) \cdot \prod_{i=1}^{n} P(X_i|c) \tag{33}$$

כאן $P(c)$ היא ההסתברות הקודמת (Prior Probability) של הקטגוריה c, הנאמדת בדרך כלל כשיעור הדוגמאות בקטגוריה זו בקבוצת האימון. המכפלה $\prod_{i=1}^{n} P(X_i|c)$ מייצגת את **הנראות** (Likelihood) של התכונות הנצפות בהינתן הקטגוריה.

בפרקטיקה, חישוב ישיר של מכפלת הסתברויות עלול להוביל ל**תת־זרימה מספרית** (Numerical Underflow), שכן מכפלת מספרים קטנים רבים מתכנסת במהירות לאפס. לכן, נהוג לעבוד ב**מרחב לוגריתמי**:

$$C_{\text{NB}} = \arg\max_{c \in C} \left[\log P(c) + \sum_{i=1}^{n} \log P(X_i|c) \right] \tag{34}$$

המעבר ללוגריתמים משמר את יחסי הגודל תוך המרת המכפלה לסכום, מה שמבטיח יציבות חישובית ומונע בעיות של אובדן דיוק במספרים צפים.

5.4 יתרונות האלגוריתם

האלגוריתם של Naive Bayes מציע שילוב נדיר של פשטות, מהירות ויעילות. היתרון המרכזי שלו טמון בפשטות המתמטית: אין צורך בתהליכי אופטימיזציה איטרטיביים, ואין פרמטרים חיצוניים (Hyperparameters) שיש לכוונן. האימון מסתכם בסריקה אחת של נתוני האימון לצורך אמידת ההסתברויות, והסיווג עצמו דורש רק חיבור וכפל של מספרים. מורכבות זמן הריצה היא $O(n \cdot |C|)$ לכל דוגמה, מה שהופך אותו לאחד האלגוריתמים המהירים ביותר בתחום.

יתרון נוסף הוא יכולתו לעבוד עם כמויות נתונים קטנות יחסית. בעוד שאלגוריתמים מורכבים יותר דורשים אלפי דוגמאות כדי להגיע לביצועים סבירים, Naive Bayes יכול להניב תוצאות טובות גם עם מאות דוגמאות בלבד. זאת בזכות מספר הפרמטרים המצומצם שהוא מאמד.

פרדוקס מרתק הוא שלמרות שהנחת אי־התלות כמעט ולא מתקיימת בפועל, האלגוריתם מצליח להניב ביצועי סיווג טובים במשימות רבות. inazzaP dna sognimoD [4] הסבירו תופעה זו בכך שסיווג נכון אינו דורש אמידה מדויקת של ההסתברויות, אלא רק שמירה על סדר נכון ביניהן. כל עוד $P(c_1|X) > P(c_2|X)$ במקרים שבהם אכן c_1 היא הקטגוריה הנכונה, הסיווג יצליח גם אם הערכים המספריים עצמם רחוקים מהמדויקים. gnahZ [10] הוסיפו והראו שבתנאים מסוימים, Naive Bayes אף מגיע לביצועים אופטימליים למרות שהנחותיו מופרות.

5.5 מגבלות וחסרונות

למרות יתרונותיו הרבים, לאלגוריתם Naive Bayes מגבלות ברורות הנובעות מהנחותיו המפשטות. הבעיה המרכזית מתעוררת כאשר תכונות תלויות מאוד זו בזו. במצבים כאלה, ההנחה שכל תכונה תורמת באופן עצמאי למידע על הקטגוריה אינה נכונה, וכתוצאה מכך האלגוריתם עלול לתת משקל כפול לאותו מידע. לדוגמה, בסיווג טקסטים, אם שתי מילים מופיעות כמעט תמיד ביחד (כמו "ניו יורק"), המודל יתייחס אליהן כאל שני פיסות מידע נפרדות, בעוד שבפועל הן נושאות מידע אחד.

מגבלה נוספת קשורה לטיפול בתכונות רציפות. בעוד שעבור תכונות קטגוריאליות האמידה פשוטה (ספירת תצפיות), עבור תכונות רציפות יש להניח התפלגות מסוימת — בדרך כלל נורמלית. הנחה זו אינה תמיד מתאימה, ועלולה להוביל לאומדנים לא מדויקים. פתרון אפשרי הוא ביצוע **דיסקרטיזציה** (Discretization) של התכונות הרציפות, אך גם זה מצריך החלטות עיצוביות שאינן תמיד ברורות.

בעיית **האפס** (Zero Probability Problem) מהווה אתגר נוסף: אם צירוף מסוים של תכונה וקטגוריה לא הופיע בנתוני האימון, ההסתברות המוערכת תהיה אפס, מה שיגרום לכך שכל המכפלה תתאפס. הפתרון המקובל הוא **החלקה** (Smoothing), כגון **החלקת לפלס** (Laplace Smoothing), שמוסיפה מניין קטן (1 בדרך כלל) לכל הספירות, ובכך מבטיחה שאף הסתברות לא תהיה אפס מוחלט.

לבסוף, האלגוריתם מניח שכל התכונות תורמות באופן שווה לסיווג. במקרים שבהם חלק מהתכונות רועשות או לא רלוונטיות, הן עדיין ישפיעו על התוצאה. אלגוריתמים מתקדמים יותר כוללים מנגנוני **בחירת תכונות** אוטומטית, בעוד ש־Naive Bayes דורש בחירה ידנית מקדימה.

5.6 מימוש בפייתון

המימוש הבא מדגים את עקרונות האלגוריתם באופן מובנה וברור. המחלקה `reifissalCseyaBeviaN` מטפלת בתכונות קטגוריאליות, כוללת החלקת לפלס למניעת בעיית האפס, ועובדת במרחב לוגריתמי ליציבות מספרית:

דוגמת שימוש במסווג:

המימוש כולל את כל המרכיבים החיוניים: אמידת הסתברויות קודמות מתוך שכיחויות בנתוני האימון, אמידת הסתברויות מותנות לכל צירוף של תכונה וערך, החלקת לפלס למניעת הסתברויות אפס, ועבודה במרחב לוגריתמי ליציבות מספרית. הקוד ממחיש כיצד פשטות מתמטית מתורגמת למימוש תכנותי ישיר ויעיל.

מחלקת Naive Bayes - אתחול ואימון

```
import numpy as np
from collections import defaultdict
from typing import Dict, List, Tuple

class NaiveBayesClassifier:
    """Naive␣Bayes␣classifier␣with␣Laplace␣smoothing."""

    def __init__(self, alpha: float = 1.0):
        """Initialize␣classifier␣with␣smoothing␣parameter."""
        self.alpha = alpha
        self.classes: List = []
        self.class_priors: Dict = {}
        self.feature_probs: Dict = defaultdict(dict)
        self.feature_values: Dict = {}

    def fit(self, X: np.ndarray, y: np.ndarray) -> None:
        """Train␣model␣on␣training␣data."""
        n_samples, n_features = X.shape
        self.classes = np.unique(y)

        # Estimate prior probabilities
        for c in self.classes:
            self.class_priors[c] = np.sum(y == c) / n_samples

        # Identify possible values for each feature
        for j in range(n_features):
            self.feature_values[j] = np.unique(X[:, j])

        # Estimate P(X_i | C) for each feature, value, class
        for c in self.classes:
            X_c = X[y == c]
            n_c = X_c.shape[0]
            for j in range(n_features):
                n_values = len(self.feature_values[j])
                for value in self.feature_values[j]:
                    count = np.sum(X_c[:, j] == value)
                    # Laplace smoothing
                    prob = (count + self.alpha) / (n_c + self.alpha *
n_values)
                    self.feature_probs[c][(j, value)] = np.log(prob)
```

מחלקת Naive Bayes - חיזוי וסיווג

```
    def predict_proba(self, X: np.ndarray) -> np.ndarray:
        """Calculate␣log-probabilities␣for␣each␣class."""
        n_samples, n_features = X.shape
        log_probs = np.zeros((n_samples, len(self.classes)))

        for i, c in enumerate(self.classes):
            log_probs[:, i] = np.log(self.class_priors[c])
            for sample_idx in range(n_samples):
                for j in range(n_features):
                    value = X[sample_idx, j]
                    key = (j, value)
                    if key in self.feature_probs[c]:
                        log_probs[sample_idx, i] += self.feature_probs[c
][key]
        return log_probs

    def predict(self, X: np.ndarray) -> np.ndarray:
        """Predict␣class␣labels␣for␣new␣samples."""
        log_probs = self.predict_proba(X)
        return self.classes[np.argmax(log_probs, axis=1)]
```

דוגמת שימוש במסווג Naive Bayes

```python
# יצירתנתונידוגמה  : סיווגמזגאוויר
# תכונות: outlook, temperature, humidity, windy
# קטגוריה: play (yes/no)

X_train = np.array([
    [0, 0, 1, 0],  # sunny, hot, high, false
    [0, 0, 1, 1],  # sunny, hot, high, true
    [1, 0, 1, 0],  # overcast, hot, high, false
    [2, 1, 1, 0],  # rainy, mild, high, false
    [2, 2, 0, 0],  # rainy, cool, normal, false
    [2, 2, 0, 1],  # rainy, cool, normal, true
    [1, 2, 0, 1],  # overcast, cool, normal, true
])

y_train = np.array([0, 0, 1, 1, 1, 0, 1])  # no, no, yes, yes, yes, no,
yes

# אימוןהמודל
nb_classifier = NaiveBayesClassifier(alpha=1.0)
nb_classifier.fit(X_train, y_train)

# חיזוילדוגמהחדשה
X_test = np.array([[0, 1, 1, 0]])  # sunny, mild, high, false
prediction = nb_classifier.predict(X_test)
print(f"חיזוי:␣{'כן'␣if␣prediction[0]␣==␣1␣else␣'לא'}")

# הצגת log-probabilities
log_probs = nb_classifier.predict_proba(X_test)
print(f"Log-probabilities:␣{log_probs}")
```

6 דוגמה מפורטת: Play Tennis

6.1 הצגת הנתונים

נתחיל בבחינת מערך נתונים קלסי בתחום למידת המכונה - מערך Play Tennis. מערך זה מכיל 14 דוגמאות אימון, כאשר כל דוגמה מתארת תנאי מזג אוויר והחלטה האם לשחק טניס או לא.

המשתנים המסבירים (features) הם:

- Outlook - מצב השמיים: Sunny, Overcast, או Rain
- Temperature - טמפרטורה: Hot, Mild, או Cool
- Humidity - לחות: High או Normal
- Wind - רוח: Weak או Strong

המשתנה התלוי (target variable) הוא Play Tennis, עם ערכים Yes או No.

להלן מערך הנתונים המלא:

יום	Outlook	Temp	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

טבלה 7: מערך נתונים Play Tennis

6.2 חישוב הסתברויות המחלקות

הצעד הראשון בסיווג Naive Bayes הוא חישוב ההסתברויות a priori של המחלקות. נספור את מספר ההופעות של כל מחלקה:

- מספר דוגמאות עם Play = Yes: 9 מתוך 14
- מספר דוגמאות עם Play = No: 5 מתוך 14

לכן ההסתברויות הן:

$$P(\text{Yes}) = \frac{9}{14} \approx 0.643 \tag{35}$$

$$P(\text{No}) = \frac{5}{14} \approx 0.357 \tag{36}$$

6.3 חישוב הסתברויות מותנות

כעת נחשב את ההסתברויות המותנות של כל תכונה בהינתן המחלקה. נתחיל בספירת ההופעות של כל ערך תכונה עבור כל מחלקה.

6.3.6.3.1 הסתברויות עבור Outlook

Outlook	מחלקה	ספירה	הסתברות
Sunny	Yes	2	$P(\text{Sunny}\|\text{Yes}) = 2/9 \approx 0.222$
Sunny	No	3	$P(\text{Sunny}\|\text{No}) = 3/5 = 0.600$
Overcast	Yes	4	$P(\text{Overcast}\|\text{Yes}) = 4/9 \approx 0.444$
Overcast	No	0	$P(\text{Overcast}\|\text{No}) = 0/5 = 0.000$
Rain	Yes	3	$P(\text{Rain}\|\text{Yes}) = 3/9 \approx 0.333$
Rain	No	2	$P(\text{Rain}\|\text{No}) = 2/5 = 0.400$

טבלה 8: התפלגות Outlook לפי מחלקה

6.3.6.3.2 הסתברויות עבור Temperature

6.3.6.3.3 הסתברויות עבור Humidity

6.3.6.3.4 הסתברויות עבור Wind

6.4 חישוב עבור דוגמה חדשה

כעת נסווג דוגמה חדשה עם המאפיינים הבאים:

Temperature	מחלקה	ספירה	הסתברות
Hot	Yes	2	$P(\text{Hot}\mid\text{Yes}) = 2/9 \approx 0.222$
Hot	No	2	$P(\text{Hot}\mid\text{No}) = 2/5 = 0.400$
Mild	Yes	4	$P(\text{Mild}\mid\text{Yes}) = 4/9 \approx 0.444$
Mild	No	2	$P(\text{Mild}\mid\text{No}) = 2/5 = 0.400$
Cool	Yes	3	$P(\text{Cool}\mid\text{Yes}) = 3/9 \approx 0.333$
Cool	No	1	$P(\text{Cool}\mid\text{No}) = 1/5 = 0.200$

טבלה 9: התפלגות Temperature לפי מחלקה

Humidity	מחלקה	ספירה	הסתברות
High	Yes	3	$P(\text{High}\mid\text{Yes}) = 3/9 \approx 0.333$
High	No	4	$P(\text{High}\mid\text{No}) = 4/5 = 0.800$
Normal	Yes	6	$P(\text{Normal}\mid\text{Yes}) = 6/9 \approx 0.667$
Normal	No	1	$P(\text{Normal}\mid\text{No}) = 1/5 = 0.200$

טבלה 10: התפלגות Humidity לפי מחלקה

Wind	מחלקה	ספירה	הסתברות
Weak	Yes	6	$P(\text{Weak}\mid\text{Yes}) = 6/9 \approx 0.667$
Weak	No	2	$P(\text{Weak}\mid\text{No}) = 2/5 = 0.400$
Strong	Yes	3	$P(\text{Strong}\mid\text{Yes}) = 3/9 \approx 0.333$
Strong	No	3	$P(\text{Strong}\mid\text{No}) = 3/5 = 0.600$

טבלה 11: התפלגות Wind לפי מחלקה

- Outlook = Sunny
- Temperature = Cool
- Humidity = High
- Wind = Strong

נחשב את ההסתברות המותנית עבור כל מחלקה באמצעות נוסחת Naive Bayes:

6.4.6.4.1 חישוב עבור Play = Yes

$$
\begin{aligned}
P(\text{Yes}|X) &\propto P(\text{Yes}) \cdot P(\text{Sunny}|\text{Yes}) \cdot P(\text{Cool}|\text{Yes}) \\
&\quad \cdot P(\text{High}|\text{Yes}) \cdot P(\text{Strong}|\text{Yes}) && (37) \\
&= \frac{9}{14} \cdot \frac{2}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} && (38) \\
&= 0.643 \cdot 0.222 \cdot 0.333 \cdot 0.333 \cdot 0.333 && (39) \\
&\approx 0.00529 && (40)
\end{aligned}
$$

6.4.6.4.2 חישוב עבור Play = No

$$
\begin{aligned}
P(\text{No}|X) &\propto P(\text{No}) \cdot P(\text{Sunny}|\text{No}) \cdot P(\text{Cool}|\text{No}) \\
&\quad \cdot P(\text{High}|\text{No}) \cdot P(\text{Strong}|\text{No}) && (41) \\
&= \frac{5}{14} \cdot \frac{3}{5} \cdot \frac{1}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} && (42) \\
&= 0.357 \cdot 0.600 \cdot 0.200 \cdot 0.800 \cdot 0.600 && (43) \\
&\approx 0.02057 && (44)
\end{aligned}
$$

6.4.6.4.3 החלטת הסיווג

לאחר נרמול ההסתברויות:

$$P(\text{Yes}|X) = \frac{0.00529}{0.00529 + 0.02057} \approx 0.204 \tag{45}$$

$$P(\text{No}|X) = \frac{0.02057}{0.00529 + 0.02057} \approx 0.796 \tag{46}$$

מסקנה: המסווג חוזה Play = No עם הסתברות של כ-79.6%.

להלן קוד Python שמבצע את החישוב:

חישוב Naive Bayes למערך Play Tennis

```python
"""␣Play␣Tennis␣-␣Naive␣Bayes␣Classification␣"""

# Prior probabilities
P_Yes = 9/14
P_No = 5/14

# Conditional probabilities for the test case
# Outlook = Sunny
P_Sunny_Yes = 2/9
P_Sunny_No = 3/5

# Temperature = Cool
P_Cool_Yes = 3/9
P_Cool_No = 1/5

# Humidity = High
P_High_Yes = 3/9
P_High_No = 4/5

# Wind = Strong
P_Strong_Yes = 3/9
P_Strong_No = 3/5

# Calculate unnormalized posterior probabilities
posterior_Yes = (P_Yes * P_Sunny_Yes * P_Cool_Yes *
                 P_High_Yes * P_Strong_Yes)
posterior_No = (P_No * P_Sunny_No * P_Cool_No *
                P_High_No * P_Strong_No)

# Normalize
total = posterior_Yes + posterior_No
prob_Yes = posterior_Yes / total
prob_No = posterior_No / total

print(f"P(Yes|X)␣=␣{prob_Yes:.4f}")
print(f"P(No|X)␣=␣{prob_No:.4f}")
print(f"Prediction:␣{'Yes'␣if␣prob_Yes␣>␣prob_No␣else␣'No'}")
```

6.5 ניתוח התוצאה

התוצאה שקיבלנו - חיזוי של No - הגיונית כאשר בוחנים את התכונות של הדוגמה החדשה:

1. **Humidity = High**: זוהי התכונה המשפיעה ביותר. מתוך 4 מקרים של High Humidity ו-No, רק 3 מקרים של High Humidity ו-Yes. יחס זה (0.800 לעומת 0.333) מעיד על קשר חזק בין לחות גבוהה להחלטה שלא לשחק.

2. **Outlook = Sunny**: גם כאן רואים העדפה ברורה - 3 מתוך 5 ימים שטושים עם No (יחס של 0.600), לעומת רק 2 מתוך 9 ימים שטושים עם Yes (יחס של 0.222).

3. **Wind = Strong**: רוח חזקה מעט נוטה לטובת No - 0.600 לעומת 0.333.

4. **Temperature = Cool**: זוהי התכונה היחידה שמעט תומכת ב-Yes - 0.333 לעומת 0.200. אולם השפעתה חלשה יחסית לתכונות האחרות.

הדוגמה ממחישה כיצד מסווג Naive Bayes משלב מידע מכל התכונות תוך הנחת עצמאות ביניהן. למרות שהנחה זו אינה מתקיימת תמיד במציאות (למשל, יתכן קשר בין טמפרטורה ללחות), המסווג עדיין מצליח להגיע למסקנות הגיוניות במקרים רבים.

7 נושאים מתקדמים

בפרק זה נחקור נושאים מתקדמים המרחיבים את הבנתנו במסווג Naive Bayes. נתמקד בטכניקות מעשיות לשיפור היציבות המספרית, נבחן שיטות להתמודדות עם בעיות נתונים דלילים, ונחשוף קשרים עמוקים יותר בין Naive Bayes לשיטות סטטיסטיות קלאסיות. הנושאים שנדון בהם הם יסודיים להטמעה מוצלחת של המסווג בסביבות ייצור אמיתיות, שבהן נתונים אינם מושלמים והביצועים חייבים להיות אמינים.

7.1 מניעת גלישה תחתית באמצעות לוגריתמים

אחת האתגרים המעשיים המרכזיים ביישום Naive Bayes היא הבעיה של גלישה תחתית מספרית. כאשר אנו מכפילים הסתברויות רבות זו בזו, במיוחד כאשר כל אחת מהן קטנה מאוד, התוצאה עלולה להיות קטנה כל כך שהמחשב אינו יכול לייצג אותה במדויק. בפועל, זה אומר שכל ההסתברויות עשויות להפוך לאפס, ואנו מאבדים את היכולת להבחין בין מחלקות שונות.

הפתרון הסטנדרטי הוא לעבוד במרחב הלוגריתמי. במקום לחשב את מכפלת ההסתברויות ישירות, אנו מחשבים את הסכום של הלוגריתמים שלהן. זה נובע מהזהות המתמטית הבסיסית:

$$\log\left(\prod_{i=1}^{n} P_i\right) = \sum_{i=1}^{n} \log P_i \tag{47}$$

בהקשר של Naive Bayes, נזכור שאנו מחפשים את המחלקה C^* שממקסמת את $P(C|X)$. לפי כלל Bayes:

$$P(C|X) = \frac{P(C) \cdot P(X|C)}{P(X)} = \frac{P(C) \cdot \prod_{i=1}^{n} P(X_i|C)}{P(X)} \tag{48}$$

כאשר אנו עוברים למרחב הלוגריתמי, המכפלה הופכת לסכום:

$$\log P(C|X) = \log P(C) + \sum_{i=1}^{n} \log P(X_i|C) - \log P(X) \tag{49}$$

מכיוון ש-$P(X)$ זהה לכל המחלקות, אנו יכולים להתעלם ממנו בבחירת המחלקה האופטימלית. לכן, כלל ההחלטה הופך להיות:

$$C^* = \arg\max_{C \in \mathcal{C}} \left[\log P(C) + \sum_{i=1}^{n} \log P(X_i|C)\right] \tag{50}$$

הגישה הזו מבטיחה יציבות מספרית מכיוון שהלוגריתמים של מספרים קטנים הם מספרים שליליים סבירים, ולא מספרים קטנים במיוחד. יתרה מזאת, פעולת החיבור יציבה הרבה יותר מפעולת הכפל כאשר מדובר במספרים קרובים לאפס.

7.2 החלקת לפלס

בעיה נפוצה ביישום Naive Bayes היא התמודדות עם מאפיינים שלא הופיעו כלל בנתוני האימון עבור מחלקה מסוימת. נניח שאנו בונים מסווג דואר זבל, ומילה מסוימת מופיעה בכל הודעות הספאם שראינו אך מעולם לא בהודעות לגיטימיות. כאשר נפגוש הודעה חדשה המכילה את המילה הזו, נקבל $P(\text{לגיטימי}|\text{מילה}) = 0$, ובגלל הכפל הזה כל ההסתברות $P(X|\text{לגיטימי})$ תהפוך לאפס, ללא קשר לכל המאפיינים האחרים.

החלקת Laplace (Laplace smoothing), הידועה גם בשם **תוספת אחד** (add-one smoothing), מציעה פתרון אלגנטי. הרעיון הבסיסי הוא להוסיף ספירה קטנה של "תצפיות דמה" לכל ערך אפשרי של כל מאפיין, כאילו ראינו כל צירוף אפשרי לפחות פעם אחת. זה מונע הסתברויות אפס מבלי לשנות באופן דרמטי את התפלגויות ההסתברות שלמדנו.

הנוסחה הכללית להחלקת Laplace היא [11], [12]:

$$P(x|c) = \frac{\text{count}(x, c) + \alpha}{\text{count}(c) + \alpha \cdot |V|} \tag{51}$$

כאשר:

- $\text{count}(x, c)$ הוא מספר הפעמים שהערך x הופיע במחלקה c
- $\text{count}(c)$ הוא מספר כל המופעים במחלקה c
- α הוא פרמטר ההחלקה (בדרך כלל $\alpha = 1$)
- $|V|$ הוא גודל המילון (מספר הערכים השונים האפשריים)

כאשר $\alpha = 1$, אנו למעשה מוסיפים תצפית אחת לכל ערך אפשרי. המכנה גדל ב-$|V|$ כדי לשמור על כך שסכום כל ההסתברויות יהיה 1. בחירת ערך α קטן יותר (למשל $\alpha = 0.1$) מפחיתה את ההשפעה של ההחלקה, בעוד ערך גדול יותר מוסיף יותר משקל לאומד האחיד.

7.3 אומדן פרמטרים: MLE מול MAP

כאשר אנו לומדים מודל Naive Bayes מנתוני אימון, אנו למעשה מאמדים את הפרמטרים של המודל — ההסתברויות המותנות $P(X_i|C)$ והסתברויות הקדם $P(C)$. קיימות שתי גישות עיקריות לאומדן פרמטרים: **אומד נראות מקסימלית** (Maximum Likelihood Estimation, MLE) **ואומד בדיעבד מקסימלי** (Maximum A Posteriori, MAP).

אומד MLE מחפש את הפרמטרים θ שממקסמים את הנראות של הנתונים:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} P(D|\theta) \tag{52}$$

כאשר D מייצג את נתוני האימון. זוהי הגישה הסטנדרטית הפשוטה: אנו מחפשים את הפרמטרים שהופכים את הנתונים שראינו לסבירים ככל האפשר.

אומד MAP, לעומת זאת, לוקח בחשבון גם התפלגות קדם על הפרמטרים עצמם [1]:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} P(\theta|D) = \arg\max_{\theta} P(D|\theta) \cdot P(\theta) \tag{53}$$

כאן אנו מיישמים את כלל Bayes ברמת הפרמטרים. התפלגות הקדם $P(\theta)$ מאפשרת לנו להטמיע ידע מוקדם או העדפות לגבי הפרמטרים. למשל, אם אנו מאמינים שההסתברויות צריכות להיות חלקות ולא קיצוניות, נוכל לבחור התפלגות קדם שמעניקה משקל נמוך לערכים קיצוניים.

מעניין לציין ש**החלקת Laplace** היא למעשה צורה של אומדן MAP עם התפלגות קדם מסוג Dirichlet. כאשר אנו מוסיפים α לכל ספירה, אנו למעשה אומרים שיש לנו אמונה קדם-מוקדמת שכל ערך אפשרי צריך להופיע לפחות α פעמים. כאשר $\alpha = 1$, זו התפלגות קדם אחידה; כאשר $\alpha > 1$, אנו מעדיפים התפלגות חלקה יותר.

7.4 הקשר ל-Linear Discriminant Analysis

קיים קשר מרתק בין Naive Bayes לבין שיטה קלאסית אחרת בלמידת מכונה הנקראת **ניתוח בחנים ליניארי** (Linear Discriminant Analysis, LDA). LDA היא טכניקה שפותחה על ידי Ronald Fisher בשנות ה-30 [13], והיא משמשת עד היום הן לסיווג והן לצמצום מימדים.

LDA מניחה שכל מחלקה מתפלגת נורמלית (גאוסיאנית) ושכל המחלקות חולקות את אותה מטריצת קוואריאנס. תחת הנחות אלו, ניתן להראות שגבולות ההחלטה האופטימליים בין מחלקות הם **ליניאריים** — קווים ישרים במרחב המאפיינים. זה בניגוד לשיטות כמו **ניתוח בחנים ריבועי** (Quadratic Discriminant Analysis, QDA), שבו כל מחלקה יכולה להיות בעלת מטריצת קוואריאנס שונה, והגבולות הם ריבועיים.

מסתבר ש-LDA הוא למעשה מקרה פרטי של **Gaussian Naive Bayes** — גרסת Naive Bayes למאפיינים רציפים שבה כל מאפיין בהינתן המחלקה מתפלג נורמלית. כאשר אנו מוסיפים להנחה זו את ההנחה ש**מטריצת הקוואריאנס אלכסונית** (כלומר, המאפיינים מותנית בלתי-תלויים — בדיוק הנחת Naive Bayes!), ושכל המחלקות חולקות את אותה מטריצה, אנו מקבלים בדיוק את LDA.

בפירוט מתמטי: ב-Gaussian Naive Bayes, הנראות של דוגמה x בהינתן מחלקה c היא:

$$P(x|c) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi\sigma_{i,c}^2}} \exp\left(-\frac{(x_i - \mu_{i,c})^2}{2\sigma_{i,c}^2}\right) \tag{54}$$

כאשר אנו דורשים ש-$\sigma_{i,c}^2 = \sigma_i^2$ לכל המחלקות (שונות משותפת), ועוברים ללוגריתמים, נקבל ביטוי שהוא ליניארי ב-x, מה שמוביל לגבול החלטה ליניארי — בדיוק כמו LDA [9].

הקשר הזה חושף תובנה עמוקה: Naive Bayes, למרות פשטותו, קשור באופן הדוק לשיטות סטטיסטיות קלאסיות ומוכחות. ההבדל העיקרי הוא שבעוד LDA מאמדת את מטריצת הקוואריאנס המלאה (אפילו אלכסונית), Naive Bayes מניח עצמאות מוחלטת ומתעלם מהקוואריאנסים. זו הפשטה שמאפשרת לימוד יעיל יותר עם פחות נתונים, אך בעלות של אובדן מידע על מתאמים בין מאפיינים.

7.5 מימוש טכניקות מתקדמות

לסיום, נציג מימוש של הטכניקות המתקדמות שדנו בהן. הקוד הבא ממחיש שימוש בלוגריתמים למניעת גלישה תחתית והחלקת Laplace לטיפול בהסתברויות אפס.

Advanced Naive Bayes - אתחול ואימון

```
import numpy as np
from typing import Dict, List
from collections import defaultdict

class AdvancedNaiveBayes:
    """Advanced␣Naive␣Bayes␣with␣log␣probabilities."""

    def __init__(self, alpha: float = 1.0):
        self.alpha = alpha
        self.class_log_prior = {}
        self.feature_log_prob = defaultdict(dict)
        self.classes = []
        self.vocab_size = 0

    def fit(self, X: np.ndarray, y: np.ndarray):
        """Train␣with␣log␣probabilities␣and␣smoothing."""
        n_samples = X.shape[0]
        self.classes = np.unique(y)
        self.vocab_size = X.shape[1]

        # Calculate log prior for each class
        for c in self.classes:
            n_c = np.sum(y == c)
            self.class_log_prior[c] = np.log(n_c / n_samples)

        # Calculate log likelihood with Laplace smoothing
        for c in self.classes:
            X_c = X[y == c]
            for feature_idx in range(self.vocab_size):
                count = np.sum(X_c[:, feature_idx])
                total = np.sum(X_c) + self.alpha * self.vocab_size
                prob = (count + self.alpha) / total
                self.feature_log_prob[c][feature_idx] = np.log(prob)
```

המימוש הזה ממחיש שני עקרונות מרכזיים: **ראשית**, כל החישובים מתבצעים במרחב הלוגריתמי, מה שמונע גלישה תחתית גם כאשר אנו מכפילים הסתברויות קטנות מאוד. **שנית**, החלקת Laplace מובנית בשיטת `tif`, ומבטיחה שאף מאפיין לא יקבל הסתברות אפס. הפרמטר `ahpla` ניתן לכוונון בהתאם למאפייני הנתונים והדומיין.

Advanced Naive Bayes - חיזוי ושימוש

```python
    def predict(self, X: np.ndarray) -> int:
        """Predict␣class␣with␣maximum␣log␣probability."""
        log_probs = {}
        for c in self.classes:
            log_prob = self.class_log_prior[c]
            for idx, value in enumerate(X):
                if value > 0:
                    log_prob += self.feature_log_prob[c][idx]
            log_probs[c] = log_prob
        return max(log_probs, key=log_probs.get)

# Example usage
X_train = np.array([[1, 1, 0, 0], [1, 1, 1, 0],
                    [0, 0, 1, 1], [0, 1, 1, 1]])
y_train = np.array([0, 0, 1, 1])

model = AdvancedNaiveBayes(alpha=1.0)
model.fit(X_train, y_train)
print(f"Prediction:␣{model.predict(np.array([1,␣0,␣1,␣0]))}")
```

הטכניקות המתקדמות שהוצגו בפרק זה הן חיוניות ליישום מוצלח של Naive Bayes בעולם האמיתי. עבודה במרחב הלוגריתמי מבטיחה יציבות מספרית, החלקת Laplace מתמודדת עם דלילות נתונים, הבנת ההבדל בין MLE ל-MAP מעמיקה את האינטואיציה הסטטיסטית, והקשר ל-LDA מחבר אותנו למסורת ארוכה של שיטות סטטיסטיות קלאסיות.

8 הערכה וסיכום

8.1 מדדי הערכה לסיווג

לאחר שבנינו מסווג Naive Bayes, עלינו להעריך את ביצועיו. בפרק זה נציג את המדדים העיקריים להערכת מודלים של למידת מכונה, ונסביר כיצד להשתמש בהם כדי לבחון את איכות הסיווג שלנו.

8.1.8.1.1 דיוק (Accuracy)

המדד הבסיסי והאינטואיטיבי ביותר הוא **דיוק** — אחוז הדוגמאות שסווגו נכון מתוך כל הדוגמאות:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{55}$$

כאשר:

- TP (True Positive): דוגמאות חיוביות שסווגו נכון כחיוביות
- TN (True Negative): דוגמאות שליליות שסווגו נכון כשליליות
- FP (False Positive): דוגמאות שליליות שסווגו בטעות כחיוביות
- FN (False Negative): דוגמאות חיוביות שסווגו בטעות כשליליות

למרות שהדיוק הוא מדד פופולרי, יש לו מגבלות. בבעיות עם **חוסר איזון** בין המחלקות (imbalanced classes), דיוק גבוה עשוי להטעות. לדוגמה, אם 99% מהמיילים הם לגיטימיים, מסווג שתמיד מנבא "לא ספאם" ישיג דיוק של 99%, אך הוא חסר תועלת [14].

8.1.8.1.2 דיוק חיובי (Precision)

דיוק חיובי מודד: מתוך כל הדוגמאות שסווגו כחיוביות, כמה באמת היו חיוביות?

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{56}$$

מדד זה חשוב כאשר עלות False Positive גבוהה. לדוגמה, במערכת אבטחה שמזהה פורצים: אזעקות שווא (FP) גורמות להתערבות מיותרת ולביזבוז משאבים.

8.1.8.1.3 שיעור זיהוי (Recall/Sensitivity)

שיעור הזיהוי מודד: מתוך כל הדוגמאות החיוביות האמיתיות, כמה זיהינו?

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{57}$$

מדד זה קריטי כאשר עלות False Negative גבוהה. בבדיקה רפואית למחלה מסוכנת, חשוב לזהות את כל החולים, גם במחיר של אזעקות שווא.

8.1.8.1.4 ציון F1

לעיתים קרובות קיים **trade-off** בין Precision ל-Recall: שיפור אחד פוגע בשני. ציון F1 הוא ממוצע הרמוני של שני המדדים:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{58}$$

הממוצע ההרמוני מעניש חוסר איזון: אם אחד המדדים נמוך מאוד, גם F1 יהיה נמוך. זהו מדד מאוזן המתאים לרוב היישומים [15].

8.1.8.1.5 עקומת ROC ושטח מתחת לעקומה (AUC)

עקומת ROC (Receiver Operating Characteristic) מציגה את הקשר בין True Positive Rate (Recall) ל-False Positive Rate עבור סף החלטה משתנה:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{59}$$

השטח מתחת לעקומה (AUC - Area Under Curve) נע בין 0 ל-1, כאשר AUC = 0.5 מייצג ניחוש אקראי ו-AUC = 1.0 מייצג מסווג מושלם. מדד זה שימושי להשוואה בין מודלים שונים [14].

8.2 מטריצת הבלבול

מטריצת הבלבול (Confusion Matrix) היא כלי ויזואלי המסכם את ביצועי המסווג. היא מציגה טבלה 2×2 (לבעיית סיווג בינארי) המשווה בין התוויות האמיתיות לתוויות המנובאות.

8.2.8.2.1 מבנה המטריצה

טבלה 12: מטריצת בלבול לסיווג בינארי

		ערך מנובא	
		חיובי	שלילי
ערך אמיתי*2	חיובי	TP	FN
	שלילי	FP	TN

המטריצה מאפשרת לראות במבט אחד היכן המודל מצליח והיכן הוא טועה.

8.2.8.2.2 דוגמה: סינון ספאם

נניח שבדקנו 100 מיילים, מתוכם 40 ספאם ו-60 לגיטימיים. המסווג שלנו זיהה:
מהמטריצה נחשב:

טבלה 13: מטריצת בלבול לסינון ספאם

		ניבוי		
		ספאם	לגיטימי	סה"כ
אמת*2	ספאם	35	5	40
	לגיטימי	3	57	60
	סה"כ	38	62	100

- **דיוק**: $(35+57)/100 = 0.92$ — 92% מהמיילים סווגו נכון
- **Precision**: $35/(35+3) = 0.921$ — 92.1% מהמיילים שסומנו כספאם היו באמת ספאם
- **Recall**: $35/(35+5) = 0.875$ — 87.5% מהספאם זוהה
- **F1**: $2 \cdot 0.921 \cdot 0.875/(0.921 + 0.875) = 0.897$

8.2.8.2.3 ניתוח הטעויות

- **False Positive (FP = 3)**: 3 מיילים לגיטימיים סומנו בטעות כספאם. זה עלול להוביל לאובדן מיילים חשובים.
- **False Negative (FN = 5)**: 5 מיילי ספאם לא זוהו והגיעו לתיבת הדואר הנכנס.

בהתאם ליישום, נבחר את סף ההחלטה: אם FP יקרים יותר (לא רוצים לאבד מיילים חשובים), נעלה את הסף; אם FN יקרים יותר (מעדיפים לחסום יותר ספאם), נוריד את הסף.

8.3 יישומים מעשיים

מסווג Naive Bayes משמש במגוון רחב של תחומים בשל פשטותו, יעילותו, וביצועיו הטובים בבעיות טקסט ומידע קטגורי.

8.3.8.3.1 סינון ספאם

אחד היישומים הקלאסיים והמוצלחים ביותר של Naive Bayes הוא **סינון מיילי ספאם**. המודל לומד מאוסף מיילים מתוייגים (ספאם / לגיטימי) ומזהה דפוסים במילים ובביטויים.

[3] הראו כי Naive Bayes יעיל במיוחד בסינון ספאם, משום שהוא:

- **מהיר**: סיווג מייל נעשה בזמן אמת
- **מסתגל**: ניתן לעדכן את המודל עם מיילים חדשים ללא אימון מחדש מלא

- **עמיד**: מתמודד טוב עם מילים נדירות ועם וריאציות שפתיות

דוגמה: מילים כמו "viagra", "winner", "free money" מקבלות הסתברות גבוהה בהינתן ספאם, בעוד מילים כמו "meeting", "report", "invoice" מאפיינות מיילים לגיטימיים.

8.3.8.3.2 אבחון רפואי

במערכות תמיכה בהחלטה רפואית, Naive Bayes משמש לסיווג חולים על בסיס תסמינים ובדיקות. [5] תיארו שימוש במודל בייסיאני פשוט לאבחון מחלות לב וסרטן.
לדוגמה, באבחון מחלת לב:

- **תכונות**: גיל, לחץ דם, כולסטרול, דופק במנוחה, כאבים בחזה
- **מחלקות**: "מחלת לב" / "בריא"

המודל מחשב:

$$P(\text{מחלה} \mid \text{תסמינים}) \propto P(\text{מחלה}) \cdot \prod_i P(\text{תסמין}_i \mid \text{מחלה})$$

יתרון נוסף: המודל מספק **פלט הסתברותי**, המאפשר לרופא להעריך את רמת הוודאות ולקבל החלטה מושכלת.

8.3.8.3.3 סיווג מסמכים וטקסטים

Naive Bayes נמצא בשימוש נרחב ב-**Natural Language Processing (NLP)**:

- **סיווג נושאים**: זיהוי אם מאמר עוסק בספורט, פוליטיקה, כלכלה וכו'
- **ניתוח סנטימנט**: האם ביקורת על מוצר חיובית או שלילית?
- **זיהוי שפה**: קביעת שפת המסמך (עברית, אנגלית, ערבית...)
- **סיווג כוונות (Intent Classification)**: בצ'אטבוטים, זיהוי מה המשתמש מבקש

בניתוח סנטימנט, לדוגמה:

- ביקורת: "The product is amazing, fast delivery and great quality!"
- מילות מפתח חיוביות: amazing, fast, great
- המודל מחשב $P(\text{positive} \mid \text{review}) > P(\text{negative} \mid \text{review})$
- תוצאה: סנטימנט חיובי

8.3.8.3.4 מערכות המלצה ומיון תוכן

Naive Bayes משמש גם ב:

- **סינון תוכן אוטומטי**: במערכות ניהול תוכן, מיון אוטומטי של מאמרים לקטגוריות

- **המלצות מותאמות אישית**: חיזוי האם משתמש יתעניין בתוכן מסוים על בסיס העדפותיו הקודמות

- **זיהוי הונאות**: במערכות פיננסיות, זיהוי עסקאות חשודות

8.4 מימוש מדדי הערכה

כעת נממש את מדדי ההערכה בפייתון, כך שנוכל להעריך את ביצועי המסווג שלנו בצורה מעשית.

8.4.8.4.1 חישוב מטריצת הבלבול

noitatnemelpmI xirtaM noisufnoC

```python
import numpy as np

def confusion_matrix(y_true, y_pred):
    """
␣␣␣␣Calculate␣confusion␣matrix␣for␣binary␣classification.

␣␣␣␣Parameters:
␣␣␣␣␣␣␣␣y_true:␣Array␣of␣true␣labels␣(0/1)
␣␣␣␣␣␣␣␣y_pred:␣Array␣of␣predicted␣labels␣(0/1)

␣␣␣␣Returns:
␣␣␣␣␣␣␣␣Dictionary␣with␣TP,␣TN,␣FP,␣FN
␣␣␣␣"""
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)

    # True Positives: predicted 1 and truth is 1
    TP = np.sum((y_pred == 1) & (y_true == 1))

    # True Negatives: predicted 0 and truth is 0
    TN = np.sum((y_pred == 0) & (y_true == 0))

    # False Positives: predicted 1 but truth is 0
    FP = np.sum((y_pred == 1) & (y_true == 0))

    # False Negatives: predicted 0 but truth is 1
    FN = np.sum((y_pred == 0) & (y_true == 1))

    return {'TP': TP, 'TN': TN, 'FP': FP, 'FN': FN}
```

8.4.8.4.2 חישוב מדדי הערכה

noitatnemelpmI scirteM noitaulavE

```python
def evaluation_metrics(y_true, y_pred):
    """
␣␣␣␣Calculate␣complete␣evaluation␣metrics.

␣␣␣␣Returns:
␣␣␣␣␣␣␣␣Dictionary␣with␣Accuracy,␣Precision,␣Recall,␣F1
␣␣␣␣"""
    cm = confusion_matrix(y_true, y_pred)
    TP, TN, FP, FN = cm['TP'], cm['TN'], cm['FP'], cm['FN']

    # Accuracy: percentage of correct classifications
    accuracy = (TP + TN) / (TP + TN + FP + FN) if (TP + TN + FP + FN) >
0 else 0

    # Precision: of those classified positive, how many truly positive
    precision = TP / (TP + FP) if (TP + FP) > 0 else 0

    # Recall: of all positives, how many did we identify
    recall = TP / (TP + FN) if (TP + FN) > 0 else 0

    # F1: harmonic mean of Precision and Recall
    f1 = 2 * precision * recall / (precision + recall) if (precision +
recall) > 0 else 0

    return {
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall,
        'f1': f1,
        'confusion_matrix': cm
    }
```

8.4.8.4.3 דוגמה: הערכת מסווג ספאם

elpmaxE noitaulavE reifissalC mapS

```
"""Classification␣results␣(1=spam,␣0=legitimate)"""
y_true = [1, 0, 1, 1, 0, 0, 1, 0, 1, 0]
y_pred = [1, 0, 1, 0, 0, 1, 1, 0, 1, 0]

# Calculate evaluation metrics
metrics = evaluation_metrics(y_true, y_pred)

print("Evaluation␣Results:")
print(f"Accuracy:␣{metrics['accuracy']:.3f}")
print(f"Precision:␣{metrics['precision']:.3f}")
print(f"Recall:␣{metrics['recall']:.3f}")
print(f"F1␣Score:␣{metrics['f1']:.3f}")
print(f"\nConfusion␣Matrix:")
print(f"TP={metrics['confusion_matrix']['TP']},␣"
      f"TN={metrics['confusion_matrix']['TN']}")
print(f"FP={metrics['confusion_matrix']['FP']},␣"
      f"FN={metrics['confusion_matrix']['FN']}")
```

פלט:

```
:stluseR noitaulavE
008.0 :ycaruccA
057.0 :noisicerP
057.0 :llaceR
057.0 :erocS 1F

:xirtaM noisufnoC
5=NT ,3=PT
1=NF ,1=PF
```

8.5 סיכום הספר

עברנו מסע ארוך מהבנת עקרונות הסתברות בסיסיים ועד לבניית מסווג Naive Bayes מלא. הגיע הזמן לסכם את הדרך ולהפנים את הלקחים המרכזיים.

8.5.8.5.1 סקירת הפרקים

פרק 1: מבוא ומוטיבציה

- למידת מכונה עוסקת בלמידה מדאטה ללא תכנות מפורש

- סיווג הוא בעיית ליבה: השמת תווית למופע חדש
- Naive Bayes הוא מודל פשוט, יעיל ואפקטיבי

פרק 2: יסודות הסתברות

- הסתברות היא שפה למדידת אי-ודאות
- כללי בייס מאפשר לעדכן אמונות לאור ראיות חדשות
- הסתברות מותנית היא המפתח להסקה סטטיסטית

פרק 3: משפט בייס

- $P(A|B) = P(B|A) \cdot P(A)/P(B)$
- הפיכת "האצות → הביס" ל-"הביס → האצות"
- יישום מעשי: בדיקות רפואיות, עדכון אמונות

פרק 4: מודל Naive Bayes

- הנחת העצמאות המותנית: $P(X|Y) = \prod_i P(x_i|Y)$
- למרות ה"נאיביות", המודל עובד היטב בפועל
- שלושה וריאנטים: Gaussian, Multinomial, Bernoulli

פרק 5: מימוש אלגוריתם

- אימון: חישוב Prior ו-Likelihood מהדאטה
- ניבוי: מכפלת הסתברויות וחישוב log למניעת underflow
- עיבוד טקסט: Tokenization, Vectorization

פרק 6: שיפורים ואופטימיזציה

- Laplace Smoothing למניעת הסתברות אפס
- Log probabilities ליציבות מספרית
- בחירת תכונות, טיפול במילים נדירות

פרק 7: דוגמאות יישום

- סינון ספאם, ניתוח סנטימנט
- השוואה ל-scikit-learn
- דוגמאות קוד מלאות ומעשיות

פרק 8: הערכה וסיכום (זה)

- מדדי הערכה: Accuracy, Precision, Recall, F1
- מטריצת בלבול ושימושים מעשיים
- סיכום ולקראת המשך הדרך

8.5.8.5.2 מסקנות מרכזיות

1. פשטות אינה חולשה

Naive Bayes מוכיח שאלגוריתם פשוט יכול להניב תוצאות מצוינות. הנחת העצמאות, למרות שאינה מדויקת, מספקת קירוב טוב למציאות מורכבת.

2. יסודות תיאורטיים חשובים

הבנת משפט בייס, הסתברות מותנית, והנחות המודל מאפשרת לנו להשתמש בו נכון, לזהות מתי הוא מתאים, ולשפר אותו בהתאם.

3. איזון בין תיאוריה למעשה

מצד אחד, יש לנו תיאוריה מתמטית מוצקה; מצד שני, צריך להתמודד עם בעיות מעשיות כמו underflow, תכונות חסרות, ונתונים רועשים.

8.5.8.5.3 מתי להשתמש ב-Naive Bayes?

מתאים במיוחד עבור:

- **סיווג טקסטים**: ספאם, סנטימנט, נושאים
- **נתונים קטגוריים**: תכונות בדידות רבות
- **דאטה מוגבל**: עובד טוב גם עם מעט דוגמאות
- **צורך במהירות**: אימון וניבוי מהירים מאוד
- **פלט הסתברותי**: כאשר רוצים רמת ביטחון, לא רק תווית

פחות מתאים עבור:

- **תכונות תלויות מאוד**: למשל, פיקסלים בתמונה
- **יחסים מורכבים**: כאשר התלות בין תכונות קריטית
- **נתונים רציפים מורכבים**: רשתות נוירונים עשויות להתאים יותר

8.6 לקראת המשך הדרך

Naive Bayes הוא צעד ראשון מצוין בעולם למידת המכונה, אך העולם הזה רחב ומרתק בהרבה. בסעיף זה נציין כיוונים מתקדמים להמשך לימוד ומחקר.

8.6.8.6.1 נושאים מתקדמים בלמידה בייסיאנית

רשתות בייסיאניות (Bayesian Networks)

Naive Bayes מניח עצמאות מותנית מלאה. **רשתות בייסיאניות** מאפשרות לדגמן תלויות חלקיות בין תכונות באמצעות גרף מכוון אציקלי (DAG). כל צומת מייצג משתנה, וקשתות מייצגות תלויות.

דוגמה: במודל רפואי, עישון משפיע על סרטן ריאות ועל ברונכיטיס, אך ברונכיטיס לא משפיע ישירות על סרטן. רשת בייסיאנית יכולה לתאר יחסים אלו במדויק.

מודלים היררכיים בייסיאניים

כאשר הדאטה מאורגן בהיררכיה (למשל, תלמידים בכיתות בבתי ספר), ניתן להשתמש במודלים היררכיים שלומדים גם מדוגמאות פרטניות וגם מקבוצות.

תהליכים גאוסיים (Gaussian Processes)

הרחבה של גישה בייסיאנית לפונקציות רציפות, שימושית במיוחד לבעיות רגרסיה עם אי-ודאות.

8.6.8.6.2 אלגוריתמי למידה קשורים

רגרסיה לוגיסטית (Logistic Regression)

דומה ל-Naive Bayes, אך לומד משקולות לתכונות ללא הנחת עצמאות. מתאים כאשר יש קורלציה בין תכונות.

מכונות וקטור תומך (Support Vector Machines)

מחפש היפרמישור המפריד בין מחלקות עם מרווח מקסימלי. עובד טוב בחללים ממדיים גבוהים.

עצי החלטה ויערות אקראיים (Random Forests)

אלגוריתמים מבוססי חוקים, קלים לפרשנות, ויכולים ללכוד אי-לינאריות מורכבות.

רשתות נוירונים עמוקות (Deep Learning)

לומדות ייצוגים היררכיים מורכבים, מתאימות לתמונות, טקסט, דיבור. דורשות הרבה דאטה וכוח חישוב.

8.6.8.6.3 נושאים קריטיים בלמידה חישובית

Overfitting ו-Underfitting

איך למצוא איזון בין מודל פשוט מדי (לא לומד מספיק) למורכב מדי (לומד רעש)?

Cross-Validation

חלוקה של הדאטה ל-training/validation/test להערכה מהימנה של ביצועים.

Feature Engineering

יצירת תכונות חדשות מהנתונים הגולמיים — אחד הצעדים החשובים ביותר בפרויקט ML.

פרשנות ושקיפות (Interpretability)

למידת מכונה משמשת יותר ויותר בהחלטות קריטיות (רפואה, משפט, כספים). יש צורך להבין למה המודל מחליט מה שהוא מחליט.

8.6.8.6.4 משאבים להמשך לימוד

ספרים מומלצים:

- gninraeL enihcaM dna noitingoceR nrettaP מאת Christopher Bishop — מבוא מקיף ללמידה סטטיסטית
- evitcepsreP citsilibaborP A :gninraeL enihcaM מאת Kevin Murphy — עומק תיאורטי ומעשי
- gninraeL lacitsitatS fo stnemelE ehT מאת Hastie, Tibshirani, Friedman — קלאסיקה בתחום

קורסים מקוונים:

- Andrew Ng's Machine Learning ב-Coursera
- fast.ai — למידה עמוקה מעשית
- קורסי MIT OCW בלמידת מכונה

ספריות ותוכנה:

- scikit-learn — הספרייה המובילה למידת מכונה בפייתון
- PyTorch / TensorFlow — למידה עמוקה
- Jupyter Notebooks — סביבה אינטראקטיבית לניסויים

8.6.8.6.5 רפלקציה פילוסופית: חשיבה הסתברותית

מעבר לטכניקות ולקוד, Naive Bayes מלמד אותנו גישה מחשבתית: **חשיבה הסתברותית**. בעולם מלא באי-ודאות, אנחנו לא יכולים תמיד לדעת בוודאות. אבל אנחנו יכולים:

- **לכמת אי-ודאות:** להעניק ציון הסתברות לאמונות
- **לעדכן אמונות:** לשנות דעה לאור ראיות חדשות (כלל בייס)
- **לקבל החלטות רציונליות:** לבחור פעולה שמקסימת תוחלת תועלת

Naive Bayes מדגים שאפילו מודל פשוט, עם הנחות "נאיביות", יכול לספק תובנות עמוקות כאשר הוא מבוסס על עקרונות הסתברותיים מוצקים.

8.6.8.6.6 מילים לסיום

למידת מכונה היא תחום דינמי ומרגש, שממשיך להתפתח במהירות. Naive Bayes, למרות גילו ופשטותו, נשאר כלי רלוונטי וחשוב. הוא משמש נקודת התחלה מצוינת להבנת עקרונות יסוד, ומספק בסיס למעבר למודלים מתקדמים יותר.
אנו מקווים שספר זה העניק לך:

- **הבנה עמוקה** של עקרונות הסתברות, משפט בייס, ו-Naive Bayes
- **יכולת מעשית** לממש, לשפר, ולהעריך מסווגים
- **השראה** להמשיך לחקור את עולם למידת המכונה והבינה המלאכותית

הדרך רק מתחילה. בהצלחה במסע שלך בעולם הדאטה והלמידה החישובית!