השוואות סטטיסטיות בין מסווגים

<u>הסבר כללי על בחירת מבחן סטטיסטי</u>:

איך בוחרים מבחן סטטיסטי?

1. <u>השוואה פרמטרית / לא פרמטרית</u>: בהשוואה פרמטרית מניחים שלנתונים יש התפלגות ידועה מראש (לדוגמה: התפלגות נורמלית, t, F, z וכו'). בהשוואה לא פרמטרית לא מניחים כלום לגבי ההתפלגות, אלא מחשבים נטו בהתאם לנתונים שהתקבלו.
אם ההנחות לגבי התפלגות עבור מבחן פרמטרי (לדוגמה t-test) מתקיימות, אז עדיף מבחן פרמטרי (יותר אמין). אם הן לא מתקיימות, אז לא בהכרח נקבל תוצאות אמינות ונעדיף מבחן לא פרמטרי.
לכל מבחן יכול להיות הנחות נוספות שצריך להניח לפני שמבצעים את המבחן (גם בפרמטרי וגם בלא פרמטרי).

<u>למה הכוונה בתוצאות לא אמינות</u>:

א. קבלת p-value נמוך ממה שהוא באמת => סיכוי גבוה יותר לקבלת type-1 error (לשלול את הנחת האפס למרות שלא היה צריך לשלול אותה).

ב. power נמוך => סיכוי גבוה יותר לקבלת type-2 error (לא לשלול את הנחת האפס למרות שהיה צריך לשלול אותה).

2. <u>השוואה בזוגות (paired) / לא בזוגות (unpaired)</u>: אם ההשוואה היא על אותה אוכלוסייה (אותם נתונים), נבצע השוואה בזוגות. אם בין 2 אוכלוסיות שונות, נבצע השוואה לא בזוגות.

<u>דוגמה עבור השוואת תוצאה ממוצעת בין 2 מודלים</u>: עבור paired, נחסיר בכל איטרציה בין התוצאות של 2 המודלים, ולבסוף ניקח את הממוצע של כל החיסורים. עבור unpaired, ניקח את ממוצעי התוצאות של כל מודל, ולבסוף נחסיר בין הממוצעים.

3. <u>השוואה בין 2 מודלים או יותר</u>: אם צריך לבצע רק השוואה אחת (לדוגמה השוואה בין תוצאות של 2 מודלים), נעשה את ההשוואה. אם צריך לעשות יותר מהשוואה אחת (3+ מודלים), צריך קודם כל לבדוק האם קיים הבדל כלשהו בין הקבוצות (למשל במבחן ANOVA). אם לא נמצא הבדל, נוכל להגיד שאין שום הבדל בין הקבוצות ושם סיימנו. אם מצאנו שקיים הבדל, ממשיכים למבחני post-hoc בין כל זוג בשביל לבדוק בין אילו קבוצות (למשל בין אילו מודלים) קיים הבדל.

<u>הערה</u>: רפרנס בסוף.

השוואה סטטיסטית בין 2 מודלים על אותו מאגר נתונים:

מבחנים מומלצים עבור איטרציה אחת:

Non-parametric:

1. McNemar's test.
2. For AUC only: Delong's test (קוד בסוף).

מבחנים מומלצים עבור מספר איטרציות:

Parametric:

1. 5 x 2 cross-validation t-test (2-fold CV for 5 iterations).
2. 100 iterations of random resampling or 10×10-fold cross-validation with the Nadeau and Bengio correction to the paired Student t-test ("corrected resampled t-test").

Non-parametric:

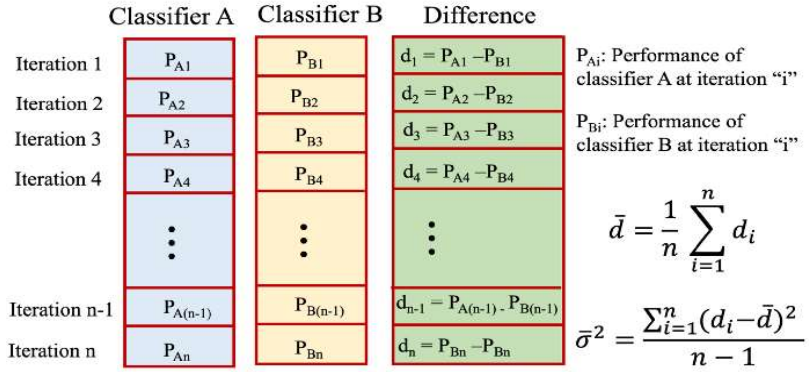
1. Wilcoxon signed-rank test.

הבעיה עם מספר איטרציות (כולל ערבוב הנתונים בכל איטרציה): צריך להניח שאין תלות בין הדגימות (תוצאות המודל), אבל בגלל הערבוב יש תלות (אותם נבדקים יהיו פעם אחת באימון ופעם אחרת במבחן).

הכי מומלץ (לדעתי): corrected resampled paired t-test. אני אישית יישמתי על 100 iterations of random resampling.

Corrected resampled paired t-test: (Proposed by Nadeau and Bengio)

נתחיל עם מבחן t רגיל:



$$t = \frac{\bar{d}}{\sqrt{\sigma^2 \cdot \left(\frac{1}{No.\,iterations}\right)}}$$

<u>בעיה עם t-test רגיל</u> : צריכה להתקיים ההשערה שהדגימות בלתי תלויות. כלומר, חוסר תלות בין הנבדקים שהיו באיטרציה i לאלו שהיו באיטרציה j. בכל איטרציה אני מערבב את כל הדטא כך שנבדקים שהיו פעם אחת בtrain/val, בפעם אחרת יהיו בtest => יש תלות.

<u>השפעה</u> : קבלת p-value נמוך יותר ממה שהוא באמת. סיכוי יותר גבוה לעשות טעות מסוג 1.

בפועל, בגלל התלות, השונות לא תגדל ככל שנוסיף איטרציות (נגדיל את k). ככל ש-k גדל (והשונות לא גדלה) כך t גדל וייתן לנו p-value יותר קטן נטו בגלל שביצענו יותר איטרציות => ניתן להקטין את p-value על ידי הגדלת כמות האיטרציות.

<u>פתרון</u> : להתאים את המשוואה באופן הבא :

$$t\ modified = \frac{\bar{d}}{\sqrt{\sigma^2 \cdot \left(\frac{1}{No.\,iterations} + \frac{No.\,subjects\ (test)}{No.\,subjects\ (train)}\right)}}$$

<u>הערה</u> : עבור two-tailed t-test, צריך להכפיל את הp-value שמתקבל ב-2. למה? כי מחפשים הבדל מ-2 הכיוונים של ההתפלגות אז צריך להתחשב גם באפשרות שמודל A קטן מB וגם שA גדול מB.

<u>פונקציה במטלב</u> :

```
function pValue = twoTailedModifiedTtest(PerformanceClassifierA, PerformanceClassifierB,
numSubjTrainVal, numsubjTest)

% corrected resampled two tailed t-test based on Nadeau and Bengio.
% ref: https://link.springer.com/content/pdf/10.1023/A:1024068626366.pdf

n = length(PerformanceClassifierA); % No. iterations

% difference between two models results
difference = PerformanceClassifierA - PerformanceClassifierB;
differenceMean = sum(difference, 'omitnan') / n; % mean
differenceVar = sum((difference - differenceMean).^2, 'omitnan') / (n - 1); % variance
% differenceVar = (difference - differenceMean)' * (difference - differenceMean) / (n - 1); %
variance

% variance after correction
differenceVarModified = differenceVar * (1/n + numsubjTest/numSubjTrainVal);

% t value
t = differenceMean / sqrt(differenceVarModified);

% Probability of larger t-statistic (two tailed)
pValue = (1 - tcdf(abs(t), n-1)) * 2;

end
```

<u>השוואה סטטיסטית בין 3+ מודלים על אותו מאגר נתונים לאורך מספר איטרציות</u>:

מבחנים מומלצים:

<u>Parametric</u>:

Analysis of Variance (ANOVA).

<u>אופן פעולה</u>: מבצעים מבחן ANOVA בשביל לבדוק האם קיים הבדל באופן כללי בין הקבוצות. אם אין, סיימנו. אם יש, ממשיכים למבחני post-hoc בשביל למצוא בין אילו זוגות קיים אותו הבדל.

The Three Assumptions for ANOVA:

1. <u>Normality</u>: The distribution of the response variable is normally distributed.
2. <u>Homogeneity of variances</u>: The variances of the differences between all combinations of related groups must be equal.
3. <u>Independence</u>: Each of the observations should be independent.

<u>תנאי 1</u>: אם אין לנו התפלגות נורמלית => לעשות מבחן לא פרמטרי.

<u>תנאי 2</u>: יש מבחן בשם Mauchly's Test שבודק משהו דומה (בדיקת sphericity במבחן RANOVA).
להבנתי יש דרך להתמודד במקרה והתנאי לא מתקיים (הוספת פרמטר לנוסחה), אבל לא חקרתי מעבר לזה.

<u>תנאי 3</u>: יש תלות בתוצאות באיטרציות שונות כי עובדים על אותו דטא מעורבב => תנאי לא מתקיים.

<u>פתרון במקרה שתנאי 3 לא מתקיים</u>:

במבחן ANOVA צריך לבצע בפועל מבחן F. קיים קשר ריבועי בין מבחן F למבחן t ($F = t^2$).

לכן, ניקח את ההתאמה שעשינו עבור מבחן t, נעלה אותה בריבוע ונוסיף אותה למבחן F ב-ANOVA.

במטלב, נבצע ANOVA, נקבל תוצאה ל-F ונכפיל אותה במקדם הבא:

$$F\ modified = \frac{F}{\left(1 + No.iterations \cdot \frac{No.subjects\ (test)}{No.subjects\ (train)}\right)}$$

רשימת מקורות:

השוואה בין 2 מודלים – איטרציה אחת – חישוב AUC:

For AUC only: Delong's test.

בלוג עם הסבר לשיטה + רפרנס למאמר המקורי:

https://glassboxmedicine.com/2020/02/04/comparing-aucs-of-machine-learning-models-with-delongs-test/

השוואה בין 2 מודלים - מספר איטרציות – בשיטת corrected paired t-test:

מאמר של נאדיו ובנג'יו (מי שהמציאו את השיטה) 2003:

https://cirano.qc.ca/files/publications/99s-25.pdf

ספר של ML בו הציגו את הנוסחה של נאדיו ובנג'יו 2011:

https://www.wi.hs-wismar.de/~cleve/vorl/projects/dm/ss13/HierarClustern/Literatur/WittenFrank-DM-3rd.pdf

עוד ספר של ML בו הציגו את הנוסחה של נאדיו ובנג'יו (יש להם טעות בנוסחה) 2011:

https://github.com/linux08/machine-learning-books/blob/master/Evaluating%20Learning%20Algorithms%20-%20A%20Classification%20Perspective%202011.pdf

השוואה בין יותר מ-2 מודלים (מספר איטרציות) גם עבור מאגרי נתונים שונים:

מאמר בו מדברים על מבחן אנובה (פרמטרי) ופרידמן (לא פרמטרי):

https://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf

הערה: בפועל לא נעזרתי בזה.. מתעסקים שם בעיקר במצבים שבהם בוחנים על מאגרי נתונים שונים.

2 מאמרים בהם עושים התאמה לנוסחה במבחן אנובה בשביל להתמודד עם התלות בין הדגימות:

https://www.jstor.org/stable/20152345#metadata_info_tab_contents

https://link.springer.com/content/pdf/10.3758/BF03206822.pdf?pdf=button

ב-2 המאמרים יש נוסחאות לt-test וANOVA מתוקנים בשביל להתמודד עם בעיית התלות בין הדגימות באיטרציות שונות. בשניהם הנוסחאות מכילות פרמטר של קורלציה בין הדגימות ללא ערך מדויק שאפשר להציב. אבל, הם מציינים קשר ריבועי בין t-test וANOVA => אפשר לקחת נוסחה מוכנה לt-test ולהמיר אותה לANOVA.

מבחן דלונג להשוואת AUC בין 2 מודלים - איטרציה אחת:

```
function DeLongTest2CompareAUCsResults = DeLongTest2CompareAUCs(pred_probA,
pred_probB, true_labls)

% Comparing AUCs of Machine Learning Models with DeLong's Test

% pred_probA: probability scores of model A (values range: [0, 1])
% pred_probB: probability scores of model B (values range: [0, 1])
% true_labls: true labels (values are the same for both models and can be: 0 | 1)

% reference:
% https://glassboxmedicine.com/2020/02/04/comparing-aucs-of-machine-learning-models-
with-delongs-test/

% divide into positive class and negative class
posA = pred_probA(true_labls == 1);
negA = pred_probA(true_labls == 0);
posB = pred_probB(true_labls == 1);
negB = pred_probB(true_labls == 0);

% Calculating the Empirical AUC (theta) for Model A and Model B
thetaA = CalculateEmpiricalAUC(posA, negA);
thetaB = CalculateEmpiricalAUC(posB, negB);

% Structural Components V10 and V01
[v10A, v01A] = CalculateV10andV01(posA,negA);
[v10B, v01B] = CalculateV10andV01(posB,negB);

% Matrices S10 and S01
[s10, s01] = CalculateS10andS01(v10A, v01A, v10B, v01B, thetaA, thetaB);

% Calculating the Variance and Covariance
s = s10 / length(v10A) + s01 / length(v01A);
varA = s(1, 1);
varB = s(2, 2);
varAB = s(1, 2);

% Calculation of the Z Score
z = (thetaA - thetaB) / sqrt(varA + varB - 2*varAB);

% Using the Z Score to Obtain a P-Value
p_value = (1-normcdf(z)) * 2;

DeLongTest2CompareAUCsResults = struct('aucA', {thetaA}, 'aucB', {thetaB}, 'varA',
{varA}, 'varB', {varB}, 'z_score', {z}, 'p_value', {p_value});

end

function auc = CalculateEmpiricalAUC(pos, neg)

heaviside = CalculateHeaviside(pos, neg);
auc = sum(heaviside, 'all') / (length(pos) * length(neg));

end

function heaviside = CalculateHeaviside(pos,neg)

heaviside = (pos > neg') + 0.5 * (pos == neg');

end
```

```
function [v10, v01] = CalculateV10andV01(pos,neg)

% Structural Components V10 and V01

% pre-allocation
v10 = zeros(size(pos));
v01 = zeros(size(neg));

% v10
% loop over all positive subjects (each for v10 indx)
for i = 1 : length(pos)

    % loop over all negative subjects and sum heavisides
    for j = 1 : length(neg)
        v10(i) = v10(i) + CalculateHeaviside(pos(i),neg(j));
    end
end

% normalize by No. negatives
v10 = v10 / length(neg);

% v01
% loop over all negative subjects (each for v01 indx)
for i = 1 : length(neg)

    % loop over all positive subjects and sum heavisides
    for j = 1 : length(pos)
        v01(i) = v01(i) + CalculateHeaviside(pos(j),neg(i));
    end
end

% normalize by No. positives
v01 = v01 / length(pos);

end

function [s10, s01] = CalculateS10andS01(v10A, v01A, v10B, v01B, thetaA, thetaB)

% Matrices S10 and S01
s10AA = sum((v10A - thetaA) .* (v10A - thetaA)) / (length(v10A) - 1);
s10BB = sum((v10B - thetaB) .* (v10B - thetaB)) / (length(v10A) - 1);
s10AB = sum((v10A - thetaA) .* (v10B - thetaB)) / (length(v10A) - 1);
s10 = [s10AA s10AB ; s10AB s10BB];

s01AA = sum((v01A - thetaA) .* (v01A - thetaA)) / (length(v01A) - 1);
s01BB = sum((v01B - thetaB) .* (v01B - thetaB)) / (length(v01A) - 1);
s01AB = sum((v01A - thetaA) .* (v01B - thetaB)) / (length(v01A) - 1);
s01 = [s01AA s01AB ; s01AB s01BB];

end
```