

מבוא לבינה מלאכותית – תרגול 11

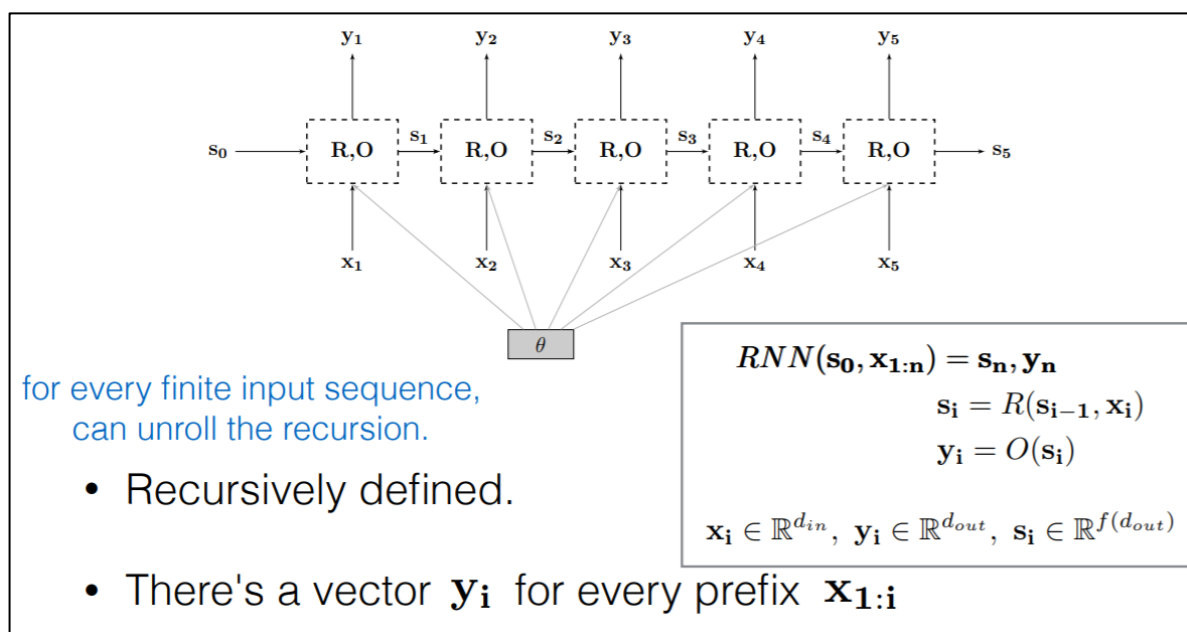
נושאים:

- Recurrent Neural Networks (RNNs)
- מדדי איכות ללמידה לא מפוקחת

:Recurrent Neural Networks (RNNs)

RNN הוא שם למשפחה של רשתות נוירונים שהייחוד שלהן הוא שהן באות להתמודד עם בעיות למידה (מפוקחת – כאן יש תיוגים) על סדרות או רצפים, באורך שאינו בהכרח קבוע (למשל, הקלטת קול היא רצף בזמן, טקסט הוא סדרה של מילים, חלבון הוא רצף של חומצות אמינו). השם RNN מגיע מכך שיש במודלים אלה מעין רקורסיה, או רכיב שחוזר על עצמו (אנחנו לא יכולים לאפשר יותר מדי שינויים ביחס שלנו לכל רכיב ברצף, מכיוון שאנחנו לא יודעים בכמה כאלה ניתקל, ולכן אנחנו מפעילים שוב ושוב אותו הדבר).

באופן כללי זה נראה (ומצויר) כך – נסמן את רצף הקלט שלנו בתור $x_{1:n}$, כאשר כל יחידה בקלט מתוארת באמצעות וקטור באותו ממד. למודל יש מצב פנימי התחלתי s_0 שמתעדכן בעקבות כל אלמנט ברצף, ולכל "נקודת זמן" המודל מוציא פלט y_i ומעדכן את המצב שלו:



על s_i אפשר לחשוב בתור ה"זיכרון" של המודל. אינטואיטיבית, כאשר מכניסים אלמנט קלט למודל, s_i מתעדכן כך שחלק מהאינפורמציה המצויה בו נשמרת, וחלק נדרס ומשוכתב כדי לקבל את אינפורמציה חדשה מהקלט שנכנס. כך, s_i שומר גם מידע לוקלי מה-"זמנים" האחרונים שעליהם המודל עבר (x_{i-1}, x_{i-2} וכו'), וגם מידע גלובלי מתחילת המשפט. איזה מידע וכמה לוקלי? זה תלוי בסוג המודל ונלמד על ידו.

כתוצאה מכך, הפלט ה- i של המודל y_i יהיה תלוי לא רק ב- x_i אלא גם ב- $x_{1:i}$.

מודל RNN ראשון:

המודל הפשוט ביותר של RNN (מכונה גם Elman RNN) הוא המודל המוגדר באופן הבא:

$$s_i = \sigma_{s,i}(W_s x_i + U_s s_{i-1} + b_s)$$
$$y_i = \sigma_{y,i}(W_y s_i + b_y)$$

כאשר $\sigma_{s,i}, \sigma_{y,i}$ פונקציות אקטיבציה (לא חייבות להיות זהות ומכאן האינדקסים), W_s, U_s, W_y, U_y מטריצות משקלים ו- b_s, b_y הם וקטורי bias נלמדים (לא תלויים ב- i ! זו בדיוק הרקורסיביות).

הערות:

1. מבחינת האינטואיציה לגבי המצב s_i , כאן כל המצב הפנימי משוכתב לכל קלט שנכנס.
2. תאורטית, המודל אמור להיות חזק מאוד – עבור אלמנט ברצף הוא מסוגל לזכור את כל האלמנטים הקודמים לו. בפועל, קשה לאמן אותו, והסיבה היא Vanishing/Exploding Gradients – מכיוון ששערוך הנגזרות בתהליך האימון (backpropagation) כולל העלאת מטריצות המשקולות בחזקה גבוהה, ההכפלה הזו מספר רב של פעמים גורמת לערך הגראדיינט לעלות או לרדת אקספוננציאלית, ואז צעד העדכון של הפרמטרים ענקי או זניח בהתאמה, ואף אחד מהם לא עוזר לאימון להיות יעיל.

לכן, נלמד כאן מודל RNN אחר שנקרא LSTM (Long Short-Term Memory):

LSTM מכיל מספר רכיבים שנקראים שערים (gates), והם נלמדים כדי להיות אחראיים על כמה מתוך המצב הקודם "לזכור" וכמה "לדרוס". פורמלית, זה נראה כך:

$$i_t = \sigma(W_{ii}x_t + U_{hi}h_{t-1} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + U_{hf}h_{t-1} + b_{hf})$$
$$o_t = \sigma(W_{io}x_t + U_{ho}h_{t-1} + b_{ho})$$
$$g_t = \tanh(W_{ig}x_t + U_{hg}h_{t-1} + b_{hg})$$
$$c_t = c_{t-1} \odot f_t + g_t \odot i_t$$
$$h_t = o_t \odot \tanh(c_t)$$

הרבה להסביר:

- הם השערים i, f, o פונקציית האקטיבציה שמופעלת עליהם היא תמיד סיגמואיד, כי תמיד נרצה שהערכים שיקבלו יהיו בין 0 ל-1.
- g נקרא cell gate או gate gate, הוא מבצע בעצם אותה פעולה כמו ב-RNN הקלאסי.
- c הוא המצב הפנימי, אפשר לשים לב שהשערים קובעים כמה הוא מתעדכן וכמה נשמר (\odot) הוא כפל איבר-איבר) – אם רכיב של הווקטור f (forget gate) קרוב ל-0, אז אנחנו נמחק את התוכן שהיה קודם בתא לטובת יותר ממה שנכנס כאן, ואם קרוב ל-1 אנחנו נשמור יותר. בנוסף, אם רכיב של הווקטור i (input gate) קרוב ל-0 אז נכתוב פחות אינפורמציה על הזיכרון הפנימי שלנו על סמך הקלט הנוכחי, ולהפך אם קרוב ל-1.
- h הוא הפלט שלנו. שימו לב שגם נעשה בו שימוש לחלק הבא, וגם הוא מהווה וקטור פלט לתהליך בקלט הנוכחי. הוא מתקבל מתוך המצב הפנימי הכולל שלנו c שעליו מפעילים את השער o (output gate), ששולט בכמה מתוך כל אלמנט בווקטור הסופי נרצה להוציא.

הערה: פתרנו את שתי הבעיות שהיו כאן קודם: ראשית, המודל לומד כמה לזכור וכמה לשכוח מתוך הקלט בכל "זמן", על סמך הקלט ועל סמך המצב הפנימי יחד. בנוסף, השימוש בשערים גורם לחישוב הגראדיינט שלא להיות תלוי ישירות במכפלת מטריצות, אלא שהמטריצות מוכפלות בשערים.

הערות לגבי נושאים מתקדמים יותר:

- למה ללמוד רצף אך ורק מההיסטוריה שלו? קיימת משפחה של מודלים של bidirectional RNNs, שבהם מוציאים ייצוג לכל אלמנט ברצף משרשור של וקטורים מתוך RNN אחד שמתקדם "קדימה" לאורך הרצף ו-RNN שני שהתקדמות בו היא "אחורה" לאורך הרצף (או מקבלת את הרצף ההפוך).
- Deep RNNs – ניתן גם "לערום" RNN אחד על גבי השני, למשל (עומק 2) פלט של RNN אחד שהתאים וקטור לכל אלמנט ברצף יהיה קלט של RNN שני שיקבל את הפלטים של ה-RNN הראשון ויחזה מהרצף שלהם את הפלט הכולל שנרצה לחזות.
- אפשר גם לתהות מה אם נרצה ללמוד מרצף באורך אחד, רצף באורך שני (למשל, במשימות של תרגום). כאן נכנסים מודלים של encoder-decoder (או attention encoder-decoder), שמטרתם היא בשלב ראשון לקודד את רצף הקלט לווקטור, ובשלב שני להוציא מווקטור הייצוג הזה את הפלט באורך שלא דווקא שווה לרצף הקודם.

מימוש של מודלים של RNN קיים כבר בחבילות של למידת מכונה בפיתון, ראו למשל:

עבור Pytorch – https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html (משימה של חיזוי חלקי דיבר במשפטים באנגלית – תיוג many-to-many, כלומר כל קלט מתויג ומבצעים על כולם חיזוי).

עבור Keras – <https://towardsdatascience.com/understanding-lstm-and-its-quick-implementation-in-keras-for-sentiment-analysis-af410fd85b47> (משימה של חיזוי סנטימנט – sentiment analysis – של משפטים, כלומר האם משפטים הם חיוביים או שליליים. זו משימת many-to-one, כלומר נכנס רצף וקטורי קלט ל-RNN ולומדים רק מתוך וקטור פלט אחד – האחרון).

יש עוד המון דוגמות, לקראת סוף הסמסטר ומועד פרסום העבודה (מתי שלא יהיה), אני ממליץ לאמץ את גוגל כחברכם הטוב ביותר וכל מה שלא מובן או שמחפשים עבורו מימוש, כדאי לחפש כי יש מי שכבר עשו את זה קודם. כמובן שגם אני כאן לעזור.

מדדים להערכת איכות בלמידה לא מפוקחת:

בלמידה לא מפוקחת, אין לנו תיוגים. שלא כמו בלמידה מפוקחת, שם אפשר בקלות לבדוק כמה טוב אנחנו מסוגלים לחזות תיוג של דוגמה, כאן נצטרך כלים קצת יותר מתוחכמים כדי לבדוק כמה טוב תהליך למידה שעשינו. נתמקד בעיקר במשימה של הערכת איכות ב-clustering, שם נראה בין היתר איך נרצה לבחור את מספר האשכולות המתאים לנו ביותר.

מה לא לוקחים כמדד איכות:

נראות לבדה לא מספיקה להערכת איכות המודל שלנו. כלומר, אם נבדוק מודל מסוים עם פרמטרים שונים בו, לא בהכרח נעדיף לקחת את הפרמטרים שנותנים למודל ערך נראות גבוה יותר. למשל, אם נבצע אשכול של n דגימות ל- k אשכולות, אז k שייתן נראות מיטבית יהיה $k = n$, הרי כל אשכול יכול בדיוק דגימה אחת וזה יהיה מושלם מבחינת צפיפות ומרחק משאר האשכולות. למרות זאת, זה כמובן לא מה שנרצה (השתמשנו ביותר מדי פרמטרים וקיבלנו overfitting).

אז מה כן יעזור לנו? (יש כל מיני אפשרויות. ראו גם תרגול 9 משנה שעברה)

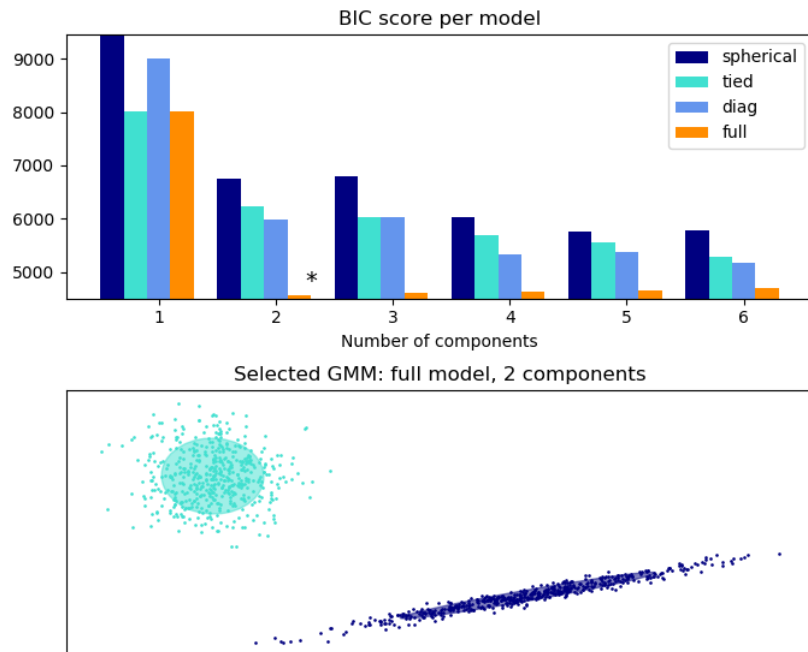
הערה כללית – עדיף לבחור על סמך יותר ממדד אחד, כי לעתים קרובות הם מראים תוצאות שאינן זהות. מכיוון שאין ground truth, אין לנו תשובה נכונה באופן מוחלט (למשל, לא ידוע מספר האשכולות הנכון) וצריך לנסות לחפש את התשובה המתאימה ביותר.

אפשרות אחת – AIC ו-BIC:

אלה שני מדדים שבהם מחשבים את לוג הנראות המקסימלית ל- L הנקודות שיש לנו, ומתקנים אותה לפי מספר הפרמטרים החופשיים (כמו מספר האשכולות שבחרנו), k . ככל שנבחר יותר פרמטרים חופשיים, ככה נשלם יותר במדד שלנו.

מדד BIC (Bayesian Information Criterion) - $k \ln(n) - 2L$. ככל שיהיה נמוך יותר, כך טוב יותר.

הערה – המדד הזה טוב יותר כאשר $n \gg k$.



מדד AIC (Akaike Information Criterion) - $2k - 2L$. גם כאן נמוך יותר פירושו טוב יותר. אפשר לשים לב שהמדד לא תלוי ב- n , וזה טוב למצב שבו יש לנו תצפיות שתלויות זו בזו, שם אנחנו לא רוצים שכמות הדגימות תשפיע.

אפשרות שנייה – p-value:

כזכור, p-value הוא גודל שמוודד, בהנחת נכונות של השערה מסוימת (למשל, שמספר האשכולות האמיתי הוא איזשהו k), את הסיכוי שקורה מצב קיצוני לפחות כמו זה שאנחנו רואים. למשל, אם מנסים למצוא נקודות חריגות מתוך אוסף נקודות קיים ולפי אשכול שביצענו, נניח שהאשכול הזה מגדיר את התפלגויות הנקודות בכל cluster (ב-GMM זה יהיה גאוסיאן רב ממדי סביב כל מרכז אשכול) ונחפש מה ה-p-value של כל נקודה להשתייכותה ל-cluster שהתאמנו לה. אם זה יהיה ערך נמוך, כנראה שהנקודה הזו חריגה כי היא לא מתאימה לאף אשכול.

אפשרות שלישית – מדדים מבוססי מרחק עבור אשכול:

אלה מדדים ספציפיים לאשכול. הרעיון המרכזי של כולם הוא שבאשכולות, אלגוריתם איכותי ייתן מרחקים פנימיים (בין נקודות מאותו אשכול) קטנים לעומת מרחקים חיצוניים (בין נקודות מאשכולות שונים) גדולים. שימו לב שלא לכל מקרי האשכול זה נכון, אלא בעיקר לאשכולות קמורים.

Dunn index – נסמן עבור שני אשכולות i, j את המרחק ביניהם (תלוי איך נגדיר אותן, אבל לא משנה לענייננו) ב- $d(i, j)$ ואת המרחק בתוך אשכול (שוב, כמה הגדרות ופחות מעניין כרגע) ב- $d(i)$. הפעם, הערך שנמדוד הוא הערך המופיע מטה, וככל שנקבל מספר גבוה יותר כך טוב יותר:

$$DI = \frac{\min_{i,j} d(i,j)}{\max_i d(i)}$$

Silhouette score – זהו מדד יותר אינפורמטיבי מהקודם, כי אפשר לחשב אותו לכל נקודה בדאטא בנפרד. הוא גם שימושי יותר כי הוא נוטה שלא לתת פתרונות טריוויאליים.

עבור קדקוד i ששייך לאשכול j , נגדיר: $a(i)$ – מרחק ממוצע משאר הנקודות ב- j , $b(i)$ – מרחק מינימלי של i מנקודה מחוץ ל- j . אז:

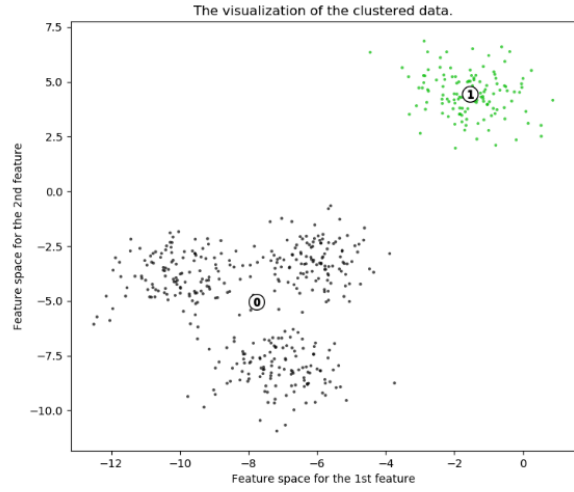
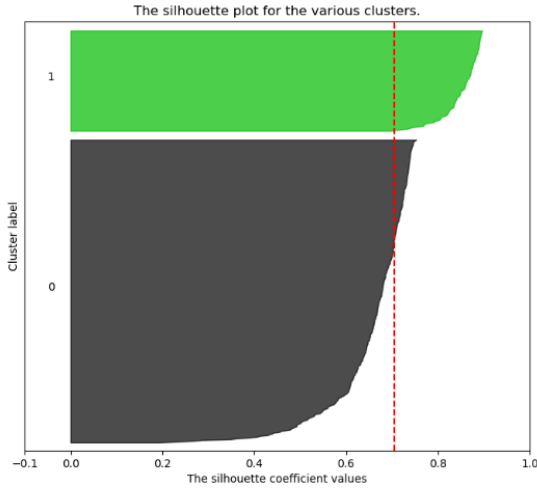
$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

המדד עבור אשכול יהיה ממוצע של המדד לכל אחת מהנקודות באשכול, והמדד עבור כלל המערכת יהיה ממוצע על פני כל המדדים של האשכולות.

$S(i)$ נע בין 1- (הנקודה ממש לא שייכת לאשכול) ל-1 (הנקודה מתאימה בצורה מושלמת לאשכול).

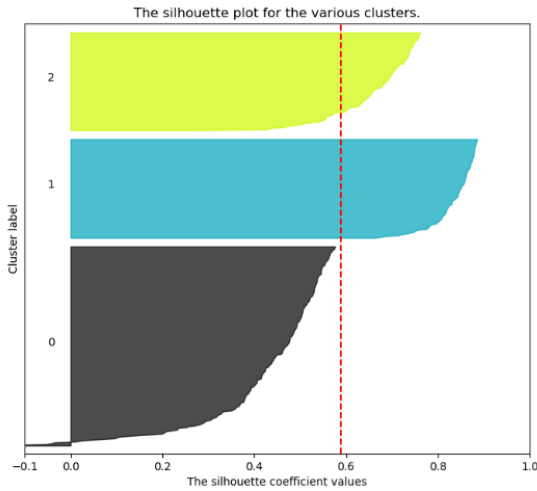
דוגמה לבחירת מספר אשכולות בעזרת silhouette score:

Silhouette analysis for KMeans clustering on sample data with n_clusters = 2



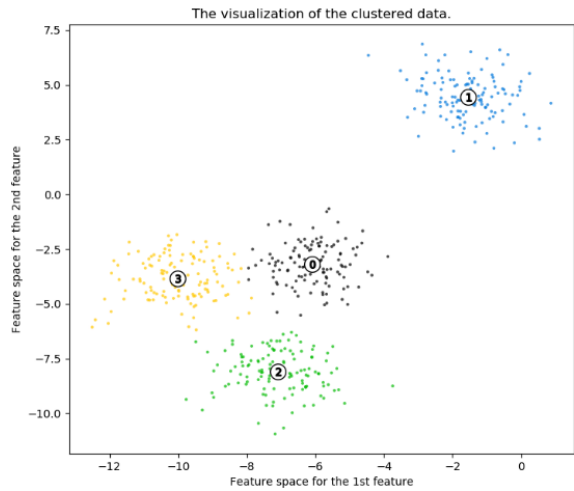
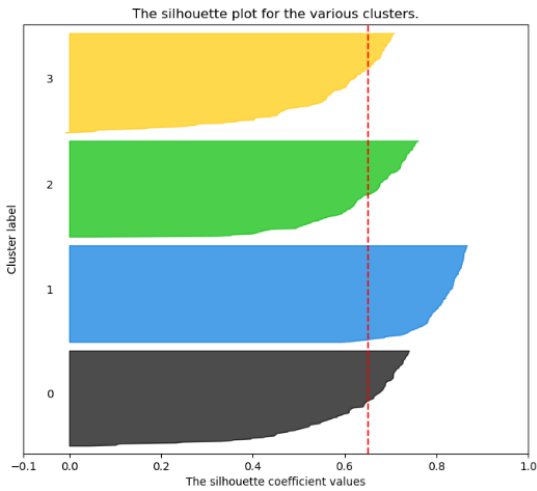
עבור $n = 2$, ה-silhouette score היה 0.7049787496083262

Silhouette analysis for KMeans clustering on sample data with n_clusters = 3



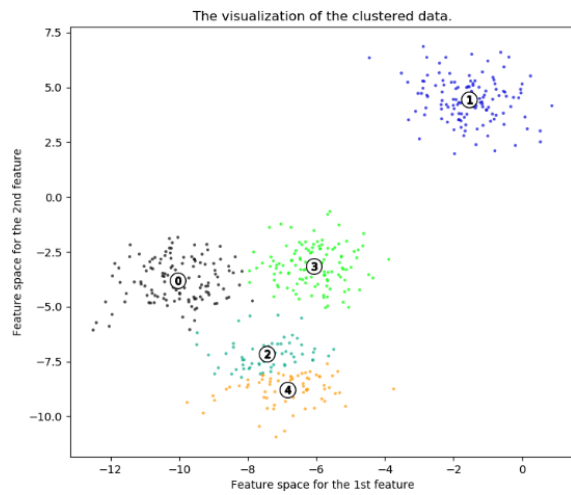
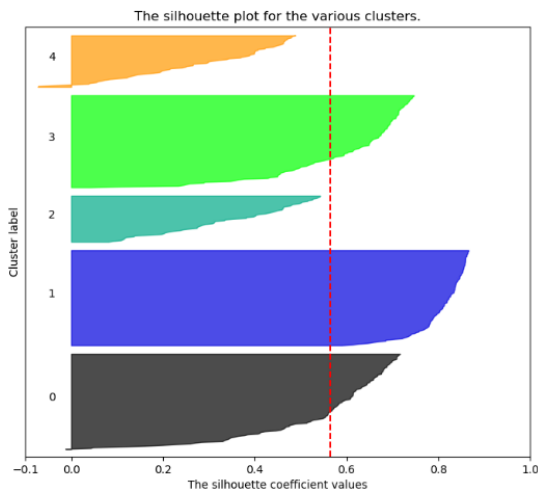
עבור $n = 3$, ה-silhouette score היה 0.5882004012129721

Silhouette analysis for KMeans clustering on sample data with n_clusters = 4



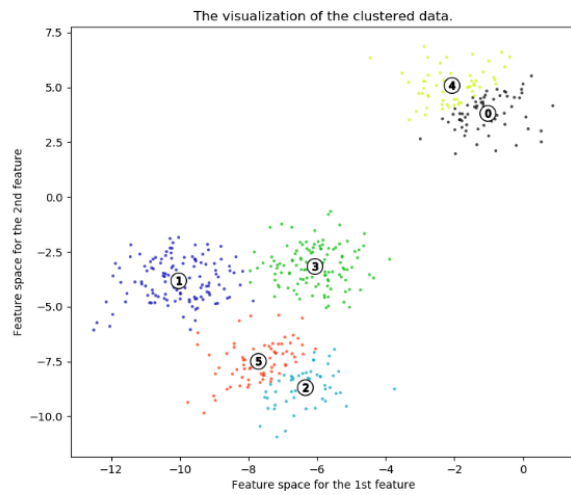
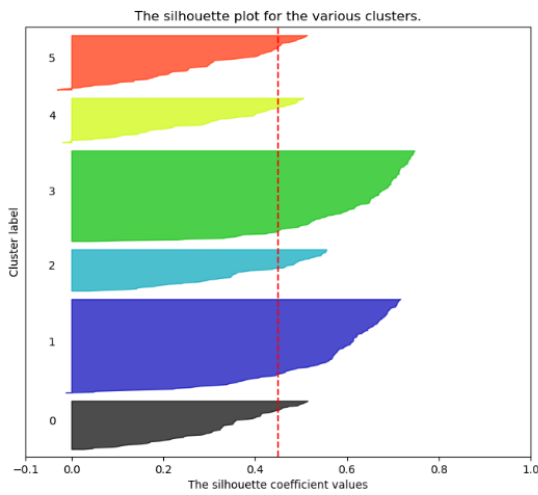
עבור $n = 4$, ה-silhouette score היה 0.6505186632729437

Silhouette analysis for KMeans clustering on sample data with $n_clusters = 5$



עבור $n = 5$, ה-silhouette score היא 0.56376469026194

Silhouette analysis for KMeans clustering on sample data with $n_clusters = 6$



עבור $n = 6$, ה-silhouette score היא 0.4504666294372765

בחירה מבוססת על גודל המדד לבדה משאירה אותנו לקחת 2 אשכולות. אולם, הסתכלות על כל אשכול לגופו מראה שאולי עדיף יותר לקחת 4 אשכולות.