סטטיסטיקה

שונות- הפיזור של ההתפלגות, כמה התצפיות קרובות לממוצע

התפלגות נורמלית- מורכבת ממוצע ושונות

רגרסיה לינארית- משתמשים במשוואה של קו ישר, על מנת למדל את הקשר בין x לy.

מסמנים נקודה בyi, נסמן את המרחק של כל הנקודות yi לקו הממוצע של הנקודות yi .

ניקח את סכום המרחקים(השאריות-residual) , ככל שסכום ריבועי השאריות קטן יותר כך הפונקציה מתארת את הנתונים בצורה טובה יותר.

שלבים של למידת מכונה

1. איסוף נתונים
2. בדיקת המידע וויזואליזציה שלו
3. עיבוד מוקדם של המידע
4. בניית מודל
5. הערכה של המודל

**סוגי למידת מכונה**

**Supervised learning**- משהו שמנסים לחזות (מה המחיר של בית, האם זה כלב או חתול), יש משתנה-label שמנסים לחזות אותו, המשתנים שאנו משתמשים בהם כדי לחזות את המשתנה הם features, משמש לחיזוי.

**לדוגמה:**מקבלים נתונים של תפוחים וקאפקייקס, מכניסים את הנתונים למודל של למידת מכונה, המודל לומד מה זה תפוח ומזה קאפקייקס, לאחר מכן משתמשים בנתונים שלא היו באימון ומודל מוצלח יצליח להבין האם הנתונים מהפרט שהתקבל הוא תפוח או קאפקייק.

**רגרסיה**- מנסים לחזות משתנה מספרי

**סיווג**- מנסים לחזות חלוקה לקטגוריות

**Unsupervised learning**- מנסים להבין דברים על הנתונים, לדוגמה clustering- חלוקה של הנתונים לקבוצה, density estimation- לחזות איך הנתונים מתפלגים, צמצום ממדים.

**לדוגמה:** מקבלים נתונים של תפוחים וקאפייקס, מכניסים לאימון על מודל של למידת מכונה ומגדירים שיחלק את הנתונים ל2 קבוצות. אין labels, הוא מחלק את הנתונים בצורה שנראית הגיונית, לא מנסים לחזות משהו, משמש לאנליזה.

על מנת להעריך כמה מוצלח המודל, מחלקים את המודל לtraining data ו test data -בדרך כלל חלוקה של 80% אימון 20% מבחן

שיטה נוספת היא חלוקה נוספת לvalidation data.

**היפר-פרמטרים:** פרמטרים שמשתמשים בהם כדי להגדיר את האלגוריתם, לדוגמה להחליט את עומק עץ ההחלטה (כמה רמות), על מנת לשפר את הביצועים של המודל.

**קרוס ולידציה k fold**- בונים 5 עצי החלטה שונים באמצעות שימוש בחלקים שונים מהנתונים על מנת להעריך כמה טובים הביצועים של כל סוג עץ.

במטריצה כל שורה זה עץ, המלבנים הכחולים בכל שורה זו הולדיציה- ה20% מהנתונים שהם המבחן של העץ.

**קרוס ולידציה של leave one out**- כל פעם להשאיר חלק אחר מהנתונים מחוץ לאימון ולבסוף לבחון עליו.

-לנקות את הנתונים מדברים לא רלוונטיים, ערכים לא תקינים, ערכים חסרים

**Imputation**

התהליך של להחליף נתונים חסרים

אופציה 1: ניתן להוריד תצפיות או משתנים

צריך לחשוב טוב לפני שעושים זאת כדי לא לאבד מידע

אופציה 2: ניתן לתת לכל הערכים החסרים ערך חדש

אופציה 3: אמדים סטטיסטים(ממוצע, חציון, שכיח(mode) )

להחליף את הערכים החסרים באמדים סטטיסטיים, תהליך זה מתבצע עבור כל feature בנפרד

אופציה 4: KNN -אלגוריתם שמחפש את K התצפיות הכי קרובות לתצפית הנוכחית, כך האלגוריתם מנבא את הערך של הערכים החסרים

**אופציה 5: MICE**:

1.עבור כל אחד מהfeatures תחילה מנסים להשלים את כל הערכים החסרים עם הממוצע

2. מחסירים מfeature אחד את הערך החסר

3. משתמשים ברגרסיה לינארית כדי להשלים את הערך החסר שהורדנו מהfeature.

4. מבצעים זאת עבור כל ערך חסר בכל feature.  
5.מחסירים מהתוצאה האחרונה שקיבלנו מתהליך זה, את הנתונים שהשלמנו עם הממוצע בשלב 1

6.חוזרים על שלבים 2-5 עד שההפרשים הם מאוד קטנים או אחרי מספר איטרציות שקבענו מראש

**קורלציות**

התהליך של לזהות קשרים בין features. כאשר יש קורלציה יותר מדי גבוהה בין features אז נשקול להוריד אחד מהfeatures. קורלציה גבוהה מדי בין משתנים אומרת שהfeatures אומרים את אותו הדבר.

**קורלציית פירסון**- מעריך את היחס הלינארי בין 2 משתנים המשכיים. ככל שאנו קרובים יותר ל1 אז יש יותר קורלציה חיובית בין המשתנים (יחס ישר), ככל שמתקרבים ל-1 יש קורלציה שלילת יותר גדולה (יחס הפוך). ככל שמתקרבים ל0 אין קורלציה.

**קורלציית ספירמן-** בודק את היחס המונוטוני בין 2 משתנים

קשר מונוטוני עולה- ככל שxעולה אז y מעולם לא יורד

קשר מונוטני יורד- ככל שx עולה אז y מעולם לא עולה

קשר לא מונוטוני- ככל שx עולה אז y לפעמים עולה ולפעמים יורד

\*קורלציה אומרת שיש קשר בין המשתנים אבל לא אומר שמשתנה אחד גורם לשני (לא בהכרח קשר של סיבה ותוצאה בין המשתנים).

**מטריצת קורלציות-** לוקחים את המשתנים, ובודקים את הקורלציות בין כל המשתנים, כל תא במטריצה מראה את רמת הקורלציה בין המשתנים. ככל שהצבע יותר קרוב לכחול אז יש יותר קורלציה בין המשתנים. באלכסון המרכזי תמיד יהיה 1 כי זה יהיה הקורלציה בין משתנה לעצמו.

באופציית ויזואליזציה השנייה של המטריצה האלכסון המרכזי מראה בהיסטוגרמה את ההתפלגות של המשתנה.

**טרנספורמציה**

**סטנדרטיזציה-** שינו התפלגות הנתונים להתפלגות נורמלית

Text

Description automatically generated

**נורמליזציה-** שינוי והגבלת ערכי הfeatures לטווח מסוים

Text

Description automatically generated

**One hot encoding-**כאשר יש משתנה קטגורי, כמו צבעים, אז המודלים לא עובדים איתם טוב בדרך כלל, לכן נחליף את המשתנים הקטגורים לעמודות כאשר בכל שורה יש 1 בשורה אחת אשר מסמן את הערך הקטגורי הנוכחי ובכל שאר השורה יהיו אפסים.

**Model Evaluation**

משתמשים ב2 מתודות שונות על הtraining set:

רגרסיה לינארית- לא משנה איך נזיז את הישר, הוא אף פעם לא יתפוס את המגמה, אם היא לא לינארית, לכן למודל זה יש bias גדול בדרך כלל, אך זה מודל שלא משתנה יותר מדי בין datasets שונים, לדוגמה בין הtraining לtest-שונות נמוכה.

Curved line- מודל אשר יש לו פחות bias, יכול לתפוס את כל המגמה בין הנתונים, אך מודל זה רגיש לdataset שבו משתמשים. עבור datasets שונים נקבל curved line שונה- bias נמוך, שונות גבוהה.

**Underfit**-מודל שלא תופס את המגמה בצורה טובה-high bias, low variance

**Overfit**- כאשר המודל מותאם יותר מדי לdataset מסוים ולא יתאים לdatasets אחרים- Low Bias, High Variance

המטרה שלנו היא למזער את הטעות הכללית

טעות כללית=Err

-טעות כללית שתמיד קיימת

קיים יחס tradeoff בין מורכבות המודל לטעות הכללית

נדרש למצוא את הנקודה הספציפית בין גודל השונות והbias, בה בטעות הכללית היא הנמוכה ביותר. **לדוגמה** ניתן להגביל את העומק של העץ, ניקח כמה עומקים שונים ונמדוד איזה עץ הוא האופטימלי.

**Low bias and low variance**- המודל יחזה את הנתונים באופן מדויק

**Low bias and high variance**-המודל נותן תוצאות יחסית מדויקות אבל משתנה מאוד כאשר נותנים לו להתאמן על datasets שונים.

**High bias and low variance**- המודל לא מדויק, אבל נותן תוצאות דומות עבור datasets שונים.

**High bias and high variance**- המודל לא מדויק ומשתנה מאוד כאשר נותנים לו להתאמן על datasets שונים.

**2 class confusion matrix**

Text

Description automatically generatedA picture containing table

Description automatically generated

בדוגמה המספרים בטבלה מייצגים כמה נתונים יש לנו בכל הגדרה.

Graphical user interface, diagram

Description automatically generated with medium confidence**Accuracy**- מודד כמה חיזויים נכונים עשינו מתוך כלל החיזויים

Text

Description automatically generated with medium confidence

Specifity- כמה ניחשת נכון שהם שליליים מתוך כל השליליים

**Sensitivity**- כמה צדקנו בחיזוי חיובי

כמה חשבת שהם חיוביים וצדקת חלקי כל מה שהוא חיובי

כאשר יותר חשוב לזהות חיוביים נכון

כאשר יותר חשוב לזהות שליליים נכון

Graphical user interface, text, application

Description automatically generated with medium confidence

Text

Description automatically generatedPrecision- כמה תצפיות ניחשת נכון שהן חיוביות חלקי כל מה שחשבת שהוא חיובי

Chart

Description automatically generated with medium confidence

יש הרבה מדדים לבדוק את דיוק המודל כיוון שיכולות להיות בעיות שונות, לדוגמה:

אם יש טעות בזיהוי classאחד יותר מאשר השני זה קשור לspecifity.

תלוי גם במה המטרה של המודל על פי מטרת המודל נבחר מדדים שונים להעריך את המודל.

F1 Score- ממוצע ממושקל בין recall ו precision

Multi class confusion matrix

מסתכלים על הערכים של TP,TN,FP,FN על פי כל קלאס בנפרד. הTrue positive יהיה התא של אותו קלאס גם בשורה וגם בטור והשאר יהיו בהתאם.

Table

Description automatically generated

**מדדים של רגרסיה**

**MSE-** סכום הריבועים של הטעות חלקי מספר התצפיות, משווים את הערך בין המודלים ובוחרים את המודל עם הMSE הנמוך ביותר

**MAE-** סכום הטעויות בערך מוחלט חלקי מספר התצפיות

**R Square**- מודד כמה שונות מתוך המשתנה התלוי אפשר להסביר באמצעות המודל, ככל שהוא יותר גבוה אז המודל יותר טוב. מחשבים את סכום ההפרש בין ערך התצפית האמיתי למה שניחשת שיהיה ערך התצפית **חלקי** סכום ההפרש בין ערך התצפית האמיתי לממוצע ערכי התצפיות האמיתיות. מפחיתים את התוצאה מ1

A picture containing chart

Description automatically generatedText, letter

Description automatically generated

**עץ קלסיקפציה**- לחזות לאיזו מחלקה שייכת הרשומה, בדוגמה: האם בן אדם אוהב סרטים או לא.

בודקים על פי כל תכונה בנפרד איך מתחלקים הרשומות

מחשבים את הgini עבור כל מחלקה שקיבלנו בכל חלוקה

טווח ערכי ג'יני: 0-1

Chart

Description automatically generatedText, letter

Description automatically generated

עבור כל חלוקה מחשבים בכל node כמה התקבלו מכל קבוצה של הlabel.

Total Gini- סך כל התצפיות בצד אחד חלקי סך כל התצפיות \* gini של אותו צד (סוכמים עבור כל Node את התוצאות).

**התמודדות בעצי החלטה עם משתנה נומרי:** מבצעים מיון בסדר עולה של הרשומות על פי התכונה, מחשבים את הממוצע עבור כל 2 תצפיות עוקבות, לאחר מכן מחשבים את רמת הטהורות באמצעות חישוב gini עבור כל ממוצע( נחלק ל2 קבוצות- תצפיות שקטנות שוות לממוצע ותצפיות שגדולות מהממוצע). בוחרים את הממוצע עם הערך gini הכי נמוך ומשתמשים בו לחלוקה על פי התכונה

עץ רגרסיה

Graphical user interface, text, application

Description automatically generatedText, letter

Description automatically generated

Y גג הוא הממוצע של התצפיות לתכונה זו

Pruning

הורדה של עלים על מנת להימנע מoverfitting.

עבור עצי רגרסיה:

1. נחשב את סכום ה SRR עבור עץ באמצעות סכימה של ערכי הSSR של כל העלים
2. מוסיפים קנס לסכום הSRR על פי רמת המורכבות של העץ
3. בוחרים את העץ עם הסכום SRR הכי נמוך

Graphical user interface, text, application

Description automatically generatedChart, scatter chart

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Bootstrap** a statistical procedure that resamples a single dataset to create many simulated samples.

לכל תצפית יש הסתברות שווה להיכלל בתוך הdataset החדש. הגודל של הdataset החדש יהיה בגודל של המקורי. תצפית יכולה להיבחר כמה פעמים לdataset החדש.

**Bagging-** יוצרים n עצים מתוך datasets שונים של אותו data. לאחר מכן משתמשים בעצים שהתקבלו ביחד על מנת לחזות את הסיווג של הנתונים, על פי הבחירה של רוב העצים או הממוצע (אם זה משתנה מספרי).

**Random Forest**

יוצר n עצים במקום עץ אחד, אך לא לוקחים datasets מתוך הdata, אלא גם לוקחים רק חלק מהfeatures בכל שלב חלוקה- על מנת לבחור באיזה feature להשתמש בכל חלוקה בעץ, הוא בוחר כל פעם feature אחד מתוך רק חלק מהfeatures.

ניתן להשתמש בתצפיות שלא נכללו בdataset לבניית עץ ספציפי זה, על מנת לבדוק את רמת הדיוק של העץ-out of bag accuracy- We can measure how accurate our random forest is by the proportion of out-of-bag samples that were correctly classified.

Text

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedOut-of-bag Error: the proportion of out-of-bag samples that were incorrectly classified.

**AdaBoost**

תחילה כל התצפיות מקבלות את אותו משקל -1 חלקי מספר התצפיות

מחשבים את הGini עבור כל סטאמפ כמו בעץ החלטה (מחלקים את הנתונים על פי כל תכונה לפי הממוצע שמחלק הכי טוב).

Diagram

Description automatically generatedDiagram

Description automatically generated

עבור כל סטאמפ מחשבים כמה הוא טעה

מספר הפעמים שהוא טעה חלקי מספר התצפיות=טעות כוללת

מחשבים מה המשקל של הסטאמפ

כאשר המשקל הכי גבוה שיכול להיות לסטאמפ הוא 1

על מנת לבחור מה הסטאמפ הבא שנבדוק נבצע את התהליך הבא:

Diagram

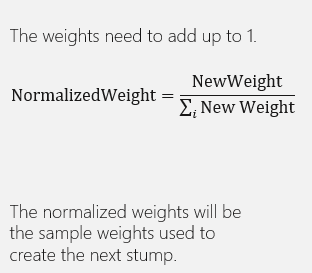
Description automatically generatedרוצים לתת חשיבות יותר גדולה לתצפיות שסווגו לא נכון, כדי שהסטאמפ הבא שיבדוק את התצפית יסווג אותה נכון

Table

Description automatically generated

כיוון שאנחנו רוצים שסך כל המשקלים של התצפיות יהיה שווה ל1 אז מנרמלים את משקלי התצפיות באמצעות חלוקת כל משקל של תצפית בסך כל משקלי התצפיות

Graphical user interface, text, application

Description automatically generated

שיטה 2

עבור כל משקל של תצפית מגדירים טווח, כאשר הטווח של כל תצפית הוא המשקל של התצפית הקודמת עד סכום שמקלים של התצפיות עד התצפית הנוכחית.

בוחרים מספר רנדומלי בין 0 ל1 ובוחרים את התצפית שלטווח שלה המספר נכנס

מוסיפים את התצפית שנבחרה לdataset חדש

חוזרים על שלבים 2 ו3 n פעמים, כאשר n הוא מספר התצפיות בdataset

כיוון שלתצפיות עם משקל גדול יותר יהיה טווח גדול יותר, יש יותר סיכוי שהן יבחרו, כי יש יותר סיכוי שהמספר הרנדומלי יהיה בטווח שלהן,כך שבdataset החדש שנבנה השורות שסווגו לא נכון, יופיעו יותר פעמים ולכן החשיבות שלהן תהיה יותר גדולה, כיוון שהיא תשפיע יותר על הgini.

מבצעים את תהליך שוב עם הdataset החדש, בוחרים את התנאי עם הפיצול הכי טוב, ומבצעים את הפיצול על הdataset החדש.אחרי שבנינו את כל הסטאמפס אז מחשבים עבור כל החלטה את סכום המשקלים של הסטאמפס שבחרו בהחלטה זו. ההחלטה עם המשקל הכי גבוה היא ההחלטה שנבחרת**.**

אם זה Regression AdaBoost אז מחשבים את הממוצע הממושקל שמתקבל מהתוצאות של כל הסטאמפים.

**היפר-פרמטרים:**

אפשר לקבוע כמה סטאמפים רוצים- הדיפולט הוא 50(n\_estimators)

בדיפולט adaboost בונה stumps- עצים בעומק 1 אפשר גם לקבוע עומק אחר (base\_estimator)

**Gradient Boosting Regression**

1. מחשבים את הממוצע של כל ערכי הlabel (מה שמנסים לחזות)
2. מחשבים את השאריות – ההפרשים בין הערך האמיתי של הלייבל בכל תצפית לממוצע שחושב
3. בונים עץ רגרסיה ומנסים לחזות את השאריות עבור כל תצפית

לוקחים תצפית מהtrain ומנסים לעשות חיזוי: לוקחים את החיזוי הראשוני( הממוצע של הלייבלס) ומחברים עםlearning rate כפול החיזוי של השארית מהעץ  
זה יהיה החיזוי החדש. אם יש כמה תצפיות בכל עלה אז מחשבים את הממוצע שלהם.

הlearning rate הוא היפר-פרמטר, בdefault הוא 0.1

1. מחשבים שאריות חדשות: מחסירים את החיזוי החדש מהערך המקורי
2. Graphical user interface, text, application, email

   Description automatically generatedחוזרים על שלבים 2-4 עד שמספר העצים הנדרש מתקבל או שהשאריות לא גורמות להבדל משמעותי

**KNN**

Chart, scatter chart

Description automatically generated

Text, letter

Description automatically generated

ערכים מאוד קטנים של K מובילים לoverfitting, ערכים מאוד גדולים של K מובילים לbias מאוד גדול.הערך הכי טוב לK תלוי בנתונים, מנסים ובודקים מה נותן את החלוקה הטובה ביותר

על מנת לחלק features קטגורליים נמיר אותם ל0 ו1.אם משנים את המדגם מהנתונים ועבור כל מדגם נקבל תוצאה שונה- שונות גדולה –overfitting עבור מדגמים שונים regressor מאוד שונה.

זה מצב לא טוב כי אנחנו רוצים להשתמש באותו regressor למדגמים שונים. כאשר נשתמש ביותר מדי קלאסטרים אז נקבל bias גדול אבל שונות נמוכה, הregressor לא משתנה ממדגמים שונים אך עושה טעויות בסיווג לקלאסטרים, רוצים להגיע למצב מאוזן בין variance לbias

**קלסיפיקציה-** מסתכלים מהו הלייבל הכי נפוץ בין כל K השכנים ובוחרים בו

**רגרסיה-** מחשבים את הממוצע של הלייבל של כל K השכנים ובוחרים בו

Graphical user interface, text, application, email

Description automatically generated

Weight- עושה ממוצע משוקלל לפי המרחק של תצפית מהתצפית החדשה

Metric- באיזה סוג מרחק משתמשים, דיפולט זה אוקלידי

**KNN לעיבוד תמונה:**

ניתן להשתמש בKNN לעיבוד תמונה, באמצעות שימוש בפונקציה reshape ניתן להפוך כל פיקסל בתמונה לfeature ואז לבדוק את ערכו.

במקרה של תמונות שחור לבן ניתן לבדוק אם הפיקסל שחור ואז מקבל את הערך 1 או לבן ואז מקבל את הערך 0.

כך באמצעות KNN נקבל קלאסטרים לפי מה שחור ומה לבן

אם במקרים הבאים אותם פיקסלים יהיו לבנים פחות או יותר אז נוכל לזהות איזו ספרה מופיעה בתמונה.

**KD Trees**

בונים L עצים כאשר בוחרים feature באופן רנדומלי ומחלקים לפי התצפית עם הערך של החציון של תכונה זו. מבצעים את התהליך מספר פעמים כך שכל צומת בעץ היא החציון של החלוקה לפי תכונה זו.

כאשר נותנים תצפית חדשה, מריצים אותה בכל העץ. כאשר מגיעים לעלה הרוולנטי בעץ- בוחרים את השכן הכי קרוב בכל עלה. בודק גם בעלה השני של החלוקה האחרונה שנבדקה האם יש תצפית קרובה יותר בו.

כאשר יש לנו את המרחקים הכי קרובים של העלים מכל העצים, בוחרים בעלה עם המרחק הכי קטן.

Chart, diagram

Description automatically generated

**LSH-Locality Sensitive Hashing**

יוצרים L hash tables , חותכים את הdata לקבוצות שונות באופן רנדומלי

כל התצפיות בחתך מסוים נכנסים ביחד לאותו hash code

בהינתן תצפית חדשה מחפשים באמצעות טבל הhash את השכן הקרוב ביותר

קיימת גם אופציה לבצע חתכים על פי הצפיפות של הנתונים- נגריל את המיקומים של החתכים באמצעות הגרלה של אינדקס של נקודה מתוך הנקודות הקיימות( במקום להגריל ערך בין הערך של הנקודה מינמלית לערך של הנקודה המקסימלית-uniform,) ונבצע את החתך בנקודה שהתקבלה. כך ההסתברות לקבל חתך באזורים עם יותר נקודות יהיה גדול יותר, לכן יהיו יותר חתכים באזורים עם צפיפות גבוהה. כך נקבל בכל תא הסתברותית בערך אותה כמות נקודות (data driven)

Graphical user interface, text, application, email

Description automatically generatedChart, scatter chart

Description automatically generated

**Clustering**

פונקציות לחישוב מרחק בין קלאסטרים: מרחק אוקלידי, מרחק מנהטן, קורלציית פירסון\ ספירמן

אם עלול להיות הבדל גדול מבחינת מספר תצפיות בכל קלאסטר אבל להשתמש בחישוב זווית על מנת לבדוק את המרחק, או לנרמל את הנתונים ואז להשתמש במרחק אוקלידי.

**K-Means**

K-Means יכול ליצור קלאסטרים רק בצורת עיגול

**Graphical user interface, text, application, email

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence**

מדד WCSSהוא אחד הדרכים המקובלות ביותר להעריך את תוצאות החלוקה לאשכולות בה .K-meansהיתרון שלמדד זה הוא האפשרות לראות באופן כמותי את מידת ההצלחה של המודל, כלומר לקבל מספר ממשי שמכמת את הצלחת המודל. מנגד, WCSSהוא מספר ללא תחום מסוים והוא דורש פרשנות, כיוון שהערך והמשמעות שלו משתנים ממודל למודל. ערך מסוים יכול להיחשב תוצאה טובה במקרה מסוים, ובמקרה אחר זאת עשויה להיחשב תוצאה רעה מאוד. ניתן להשוות WCSSבין מודלים אך ורק כאשר יש להם את אותו מספר אשכולות ואותו מספר תצפיות. באופן פורמלי, ערך זה מחושב באופן הבא:

**A picture containing schematic

Description automatically generated**

Chart, line chart

Description automatically generatedאשר Kהוא מספר האשכולות, ו-𝑛 הוא מספר הנקודות במדגם.  
ישנו trade-offבין השאיפה למזער את מדד ה- WCSSובין מספר האשכולות הרצוי: ככל שמספר האשכולות גדול יותר, כך ה- WCSSיקטן. הדבר מתיישב עם ההיגיון – פיזור סנטרואידים רבים )כלומר, חלוקה ליותר אשכולות( על פני הנתונים יוביל לכך שבהכרח סכום המרחקים של התצפיות מהסנטרואידים יקטן או לא ישתנה. כיוון שתצפית משויכת לסנטרואיד הקרוב אליה ביותר, אם התווסף סנטרואיד שקרוב לנקודה מסוימת –ה- WCSSקטן. ואם הסנטרואיד רחוק מכל שאר הנקודות במדגם יותר מהסנטרואידים הקיימים – חלוקת התצפיות לאשכולות לא תשתנה, וערך ה- WCSSלא ישתנה. לכן מצד אחד, נרצה לבחור Kגדול שימזער את ה- ;WCSSמצד שני, הסיבה שהשתמשנו ב- K-meansמלכתחילה היא בכדי לפשט את הנתונים למספר סביר של אשכולות, כזה שיאפשר לנו לערוך אנליזה נוחה. שיטת המרפק ) במדד ה- WCSSהוא מתון במידה סבירה. אין דרך חד משמעית לקבוע שה- Kהנבחר הוא האופטימלי. בדרך זו ניתן לשכנע מדוע ה- Kשנבחר הוא הנכון, אך ההחלטה הסופית נתונה לשיקול דעתו של המשתמש.

**Graphical user interface, text, application, email

Description automatically generated**

**Elbow Method**

באמצעות שיטה זו בוחרים מה הK קלאסטרס שהכי כדאי להשתמש בהם.

באמצעות חישוב של sum of squared errors מוצאים את הנקודה שבה אין הבדל משמעותי בדיוק אם מוסיפים קלאסטר נוסף.

Sum of squared errors- מחשבים את הממוצע של הנתונים, לאחר מכן את המרחק של כל תצפית מהממוצע, מעלים בריבוע את המרחק של כל תצפית מהממוצע ואז סכומים את התוצאות

GMM-Gaussian Mixture Models

יוצר קלאסטרים שהם לא רק בצורת עיגול

תחילה בוחרים רנדומלית נקודות שהם יהיו מרכזי הקלאסטרים.

הנקודות האלו הופכות להיות ערכי הממוצע של התפלגות נורמלית

מחשבים עבור כל נקודה את ההסתברות שלה להיות חלק מהקלאסטר באמצעות log likelihood

מעדכנים את ערכי הפרמטרים של ההתפלגות על מנת להגדיל את הסיכוי שהנקודות יהיו חלק מהקלאסטר. לאחר מכן מחשבים שוב מה הסיכוי של כל נקודה להיות חלק מהקלאסטר וחוזרים על התהליך עד שאין כמעט שינוי במעבר בין הלקאסטרים.

**Text

Description automatically generated**פאי k הוא אחוז התצפיות ששייך לקאסטר מספר k מסך כל התצפיות

**Text

Description automatically generated**

\

**Chart

Description automatically generated**

**Diagram, schematic

Description automatically generated**

**בK means**:

השונות היא זהה עבור כל הקלסטרים

Expectation- מחשבים את ההסתברות שכל נקודה שייכת לכל קלאסטר

בוחרים את הקלאסטר עם ההסתברות הגבוהה ביותר שהנקודה שייכת אליו

כיוון שהשונות זהה לכל הקלאסטרים אז מחשבים את המרחק של הנקודה מכל קלאסטר . הקלאסטר שאליו המרחק הכי קצר- אליו משייכים את הנקודה

Maximization- לחשב את המרכז מחדש של כל קלאסטר, מרכז הקלאסטר-ממוצע הערכים בקלאסטר.

מבצעים את תהליך זה שוב ושוב על מנת לשפר את הסיווג לקלאסטרים של הנקודות.

**Soft K means**

מניחים עדיין שהשונות היא אותו דבר עבור כל קלאסטר

נותנים לכל נקודה את ההסתברות שהיא שייכת לכל קלאסטר

**תהליך:**מחשבים את ההסתברות של כל נקודה להיות שייכת לכל קלאסטר חלקי סכום כל ההסתברויות וכך נקבל וקטור באורך K של משקלים עבור כל נקודה- נקבל ערכים בין 0 ל1 שסכום הערכים בקלאסטר הוא 1 . לאחר מכן נחשב את המרכז של כל קלאסטר- סכום המשקלים של כל הנקודות בקלאסטר זה חלקי מספר הנקודות בקלאסטר.

**עבור Full GMM**

הממוצע שהוא מרכז הקלאסטר הוא הממוצע המשוקלל של הנקודות: סכום הנקודות כפול המשקל של כל נקודה, לחלק לסכום המשקלים של הנקודות.

**Hierarchical Clustering**

Agglomerative Clusterting

1. מגדירים כל תצפית להיות קלאסטר בפני עצמה
2. מבצעים השוואה של כל 2 תצפיות ביחד ובונים מטריצה אשר מבטאת כמה כל 2 תצפיות דומות, בדרך כלל מתבצע באמצעות מרחק אוקלידי, שומרים את מטריצת הדמיון
3. יוצרים קלאסטר ביחד של 2 התצפיות הכי דומות
4. חוזרים על שלבים 2-3 עד שכל התצפיות נמצאות בקלאסטר אחד או שמגיעים לכמות הרצויה של קלאסטרים

שיטות קישור- מגידרות איך מחשבים את המרחק בין קלאסטרים:

Complete linkage- מחשבים את המרחק המקסימלי בין 2 הנקודות הכי רחוקות בכל קלאסטר

Single linkage- לוקחים את המרחק הכי קטן (בין 2 הנקודות הכי קרובות מכל קלאסטר)

Average linkage- מחשבים את המרחק בין הממוצע של כל קלאסטר

Graphical user interface, application

Description automatically generated

המטריצה היא מטריצת מרחקים

במקרה זה המרחק הכי קטן הוא בין תצפית 1 ל2

הופכים את תצפיות 1,2 לקלאסטר

מחפשים את המרחק הכי גדול בין הקלאסטר שנוצר לבין כל אחת מהתצפיות ויוצרים מזה מטריצה.

מוצאים את המרחק הכי קטן כאשר הקלאסטר עכשיו נחשב כתצפית אחת, במקרה זה תצפיות 3,4

לכן 3,4 יהפכו לקלאסטר.

מחשבים שוב את המרחק המקסימלי, בין הקלאסטרים, במקרה זה 0.8 ויוצרים מטריצה חדשה

לבסוף יוצרים קלאסטר של כל התצפיות יחד.

Diagram

Description automatically generated with medium confidenceמצד שמאל ניתן לראות את הדנדוגרמה שהתקבלה מתהליך זה, באמצעותה ניתן להבין כמה קלאסטרים כדאי לעשות: בודקים מה ההפרש הכי גדול בין המרחקים המקסימליים בדנדוגרמה בין מרחקים עוקבי’ , נבחר בקו האנכי הכי ארוך שמסמל את ההפרש הכי גדול ונחתוך אותו עם קו אופקי לפני החלוקה לקאסטרים הבאה. מספר החיתוכים של הקו האופקי עם קווים אנכיים יהיה מספר הקלאסטרים שכדאי לעשות (שאלגוריתם יחזיר).

**היפר-פרמטרים:**

**Text, letter

Description automatically generated**

n\_clusters-מספר הקלאסטרים שהוא יגיע אליו

Affinity- איך לחשב את המרחק

Linkage-הדיפולט הוא word-מיזעור של השונות בתוך הקלאסטר

Distance\_threshold- ניתן להגדיר מרחק שמעליו לא מבצעים חיבור לקלאסטר

**\***לכל קבוצת נתונים מתאים Linkage אחר שיחלק את הנתונים בצורה הטובה ביותר, נדרש לנסות את הסוגים השונים.

**Divisive Clustering**

1. תחילה מתייחסים לכל התצפיות כקלאסטר אחד
2. יוצרים מטריצת קירבה(affinty) באמצעות המשוואה המופיעה אשר מחשבת את המרחק בין כל נקודה

A picture containing diagram

Description automatically generated

1. מחשבים את הסכום של המרחקים עבור כל קלאסטר אפשרי, כאשר מוסיפים תצפית לקלאסטרים הקיימים: במונה סוכמים את המרחק בין כל נקודה לשאר הנקודות ובמכנה מחשבים את מספר התצפיות בקלאסטר ומעלים בחזקת אלפא- ערך בין 0 ל2 שהוא היפר-פרמטר.
2. בוחרים את הקלאסטר עם המרחק הכי גדול ומחלקים את המטריצה מחדש לפי הקלאסטר שיצרנו.
3. חוזרים על שלבים 2-4 עד עד שמגיעים למספר הקלאסטרים המבוקש או שכל קלאסטר מורכב מתצפית אחת

Graphical user interface

Description automatically generated with medium confidence

נניח כי יש 3 תצפיות

1. יוצרים מטריצת אפיניטי, כל תצפית עם עצמה שווה 1

2. מחשבים את הסכום לפי הנוסחה לכל פיצול אפשרי

אם תצפית 1 תהיה קלאסטר וכל השאר קלאסטר שני אז נקבל על פי המשוואה, כאשר אלפא שווה לחצי נקבל . כך גם עבור כל שאר התצפיות כשהן לבד.

עבור הקלאסטר 1,2 וכל השאר קלאסטר נקבל 1 עם 1 (1), 2 עם 2 (1) ו1 עם 2, שווה 0.4- זה הסכום במונה חלקי גודל הקלאסטר (2) בחזקת חצי=1.69

נבצע את זה לכל קלאסטר של 2 תצפיות.

3. נחלק לפי הסכום המקסימלי-1,3

לכן 1 ו3 הם קלאסטר ביחד ו2 בנפרד

4. חוזרים על תהליך זה עד שכולם בתוך קלאסטר 1

מסתכלים על הדנדוגרמה מלמעלה- קודם מפצלים את 2 ואז את 1 ו3

ניתן באמת לראות מההתחלה במטריצת אפיניטי ש1 ו3 הכי דומים (הערך הכי גבוה)

זה קורה בזמן אקספפונציאלי- מאוד לא יעיל, לכן יש כל מיני דרכים לשפר את הזמן ריצה.

אלתגוריתם שפחות בשימוש כי יש כאלה שעובדים יותר טוב.

**Mean Shift**

1. מגדירים שכל תצפית היא מרכז קלאסטר ובוחרים את הערך של h (רדיוס או bandwidth)
2. עושים קלאסטרינג ביחד של כל התצפיות שהן ברדיוס אחת של השנייה
3. מחשבים מחדש את המרכז של כל קלאסטר- שהוא הממוצע של כל התצפיות בקלאסטר

חוזרים על שלב 2-3 עד שאין שינוי בקלאסטרים.

ערכים קטנים יותר של h יוצרים יותר קלאסטרים קטנים יותר

ערכים גדולים יותר של h יוצרים פחות קלאסטרים יותר גדולים

צריך למצוא את הh שמחלק את הנתונים בצורה הכי טובה.

Graphical user interface, text, application, email

Description automatically generated

bandwidth- הערך לפיו מחליטים מה גודל הרדיוס

Max-iter- מספר האיטרציות המקסמילי שהאלגוריתם יעשה, כדי למנוע ממנו לרוץ עד אינסוף במקרים מסוימים

**shift אדפטיבי**-ניתן להגדיר לכל נקודה את האזור שמסביבה הנקודה האמיתית נמצאת, לא חייב להיות עיגול (לדוגמה אליפסה)- לכל קלאסטר יכול להיות h אחר.

לכל נקודה יש אליפסה שמאתרת את הערך האמיתי שלה

h- האליפסה

על מנת לחשב צפיפות בנקודה מסוימת מחשבים את הפונקציה שמסומנת בf גג.



Text

Description automatically generated

A picture containing diagram

Description automatically generated

**KDE-Kernel Density Estimators**

אמדים לצפיפות של נתונים. לא משתמשים בפרמטר מסוים כדי להגדיר את צפיפות הנתונים (לא כמו התפלגות נורמלית לדוגמה שמשתמשים בתוחלת ושונות)

אומדים את הצפיפות של ההתפלגות, הרבה פעמים, משתמשים כדי להסתכל ויזואלית על המידע, יכול לשמש לקלאסטרינג משמש כשיטה מקבילה אבל באופן רציף להיסטוגרמה.

Text, letter

Description automatically generated

איקס הוא משתנה מקרי שיש לו צפיפות f , הh גם פה הוא bandwith, סוג של רדיוס. בודקים האם הנקודה X היא בין x-h וx+h זאת אומרת מה ההסתברות שX הוא בין x-h לx+h כאשר h שואף ל0

נגדיר אינדיקטור שהוא שווה ל1 אם התצפית x שייכת לטווח ו0 אם לא

נעשה סכום של כל האחדים והאפסים ונחלק ב2 כפול h כפול מספר התצפיות – נקבל את ההסתברות שx שייך לתוך הטווח הזה.

הפונקציה W היא פונקציית משקל שאם היא מקבלת x שהערך מוחלט שלו קטן מ1 אז התוצאה היא 0.5 אחרת 0

ניתן לכתוב גם את המשוואה האחרונה מהשקופית הקודמת כמו המשוואה השנייה בשקופית זו

נגדיר את W שמקבלת , זאת אומרת שאם נמצא בטווח שבין x-h לx אז אם נחשב

נקבל מספר קטן מh. לכן נקבל בתור ערך לW מרחק שהוא קטן h חלקי h לכן נקבל מW 0.5

זאת אומרת שנקבל את אילו תצפיות נכנסות לטווח x-h,x+h

Text, letter

Description automatically generated

אם יהיה לנו שהוא מספר קטן מx-h אז הערך שW יהיה מרחק גדול מh חלקי h ולכן נקבל מW 0

לכן אם נמצא בטווח x-h,x+h אז W יהיה שווה ל0.5, אחרת הוא יהיה שווה 0

כפי שהסברנו המשוואה השנייה היא לא רציפה אז ניתן להגדיר kernel estimator –המשוואה השלישית שבה הפונקציה K היא פונקציית kernel שיכולה להשתנות וh הוא הbandwidth

Chart, line chart

Description automatically generatedA picture containing text

Description automatically generated

הkernel estimator אומד את הצפיפות לפי sum of bumps. כל פס אדום על ציר x הוא תצפית. עבור כל נקודה מגדירים kernel גאוסיני, כאשר כל נקודה היא התוחלת ויש שונות קבועה לכולן.

Chart, histogram

Description automatically generated

לכל הטווח, במקרה זה 0-4 עושה סכום של הbumps –ערכי הf(x) כאשר x הוא הקו שהיא תצפית והf(x) הוא הערך של ההתפלגות הגאוסיינית באותו ערך x. זה יוצר את ההתפלגות שרואים בקו האדום (הסכום של ערכי ההתפלגויות בנקודה)ניתן להגדיר kernel functions בצורות שונות אבל יעבוד על אותו העיקרון. ככל שמגדילים את הbandwidth אז פונקציית הקרנל הופכת לחלקה יותר, ככול שמקטינים אז הפונקציה עם יותר bumps

לא חייבים להגדיר bandwidth ידינית, אפשר לכתוב bw=“silverman” ואז זה מחשב את הbandwidth המתאים

**SVM Classifier**

**Graphical user interface, text, application

Description automatically generated**

מחפשים את קו האמצע בין 2 התצפיות הכי קרובות מכל מחלקה((hyperplane ויוצרים support vectors, אלו קוים אשר בתוכם יכולים להיכנס outliers על מנת למנוע overfitting (soft margins). משתמשים בcross validation על מנת להחליט אילו soft margins הם הכי טובים- עם כמה שפחות variance וbias (כמה שפחות מיסקלסיפקציות אבל בלי overfitting).  
עוברים לממד גובה יותר על מנת לאפשר חלוקה של הנתונים ל2 מחלקות. בחורים את הממד שהכי טוב להעלות אליו את הנתונים באמצעות cross validation.

The Kernel Trick: kernel functions only calculate the relationship between every pair of points as if they are in a higher dimension – they don’t actually do the transformation.

The kernel trick reduces the amount of computation required for SVM.

הינו מודל למידה מונחית המשמש לניתוח נתונים לצורך סיווג, חיזוי ורגרסיה.  
המודל מקבל אוסף של דוגמאות מתויגות במרחב 𝑛-ממדי, ומנסה למצוא מישור המפריד בצורה טובה כמה שניתן. בין דוגמאות האימון השייכות לקטגוריות השונות. המסווג הנוצר באמצעות מודל SVMהינו לינארי, כאשר חלוקת הדוגמאות במרחב הווקטורי נעשית באופן כזה שייווצר מרווח גדול ככל האפשר בין המישור המפריד לבין הנקודות הממוקמות הכי קרוב אליו. מרווח זה מכונה שוליים  
) ,(marginכאשר בצד האחד של השוליים נמצאות דוגמאות עם labelאחד, ובצד השני נמצאות הדוגמאות עם ה- labelהשני. את המישור המפריד ניתן לייצג באמצעות אוסף הנקודות המקיימות כאשר הוא וקטור נורמלי של המישור.*Text, letter

Description automatically generated*

במצב הפשוט ביותר, המשוואה עבור כל אחד מצדדיו של המפריד הינה פונקציה לינארית של המאפיינים וכל  
הדוגמאות אשר סווגו נכונה. מצב זה מכונה "**הפרדה קשיחה"** בו האלגוריתם מוצא את המישור עם השול הרחב ביותר האפשרי, ולא מאפשר לדוגמאות להיות בין הווקטורים התומכים. זוהי למעשה הפרדה מושלמת, והווקטורים התומכים הם למעשה הנקודות בקצוות השוליים, כפי שניתן לראות באיור:

*Chart, scatter chart

Description automatically generatedText

Description automatically generated with low confidence*

*Text, letter

Description automatically generated*

*Text, letter

Description automatically generatedText, letter

Description automatically generated*

Chart

Description automatically generated

Chart

Description automatically generated

Chart

Description automatically generated with medium confidence

צריך שהמרחק בין הקלאסים יהיה כמה שיותר גדול על מנת שהסיווג יהיה כמה שיותר גדול

*מסמנים את נוסחת המרחק לכן כאשר אנחנו מנסים למזער את הנוסחה שבה דאבליו או בטא במכנה, אז אנחנו מחפשים דאבליו או בטא כמה שיותר גדול.*

Text

Description automatically generatedחלוקה ליותר מ2 קלאסים:

Text

Description automatically generated

**PCA**

אלגוריתם להורדת ממדים- יותר מדי פיצ'רים מגדילים את הזמן ריצה וגם קשה לייצג את המידע מבחינה ויזואלית על מנת להסיק ממנו דברים. משמש למציאת קלאסטרים, משמש למצוא תצפיות חריגות רוצים למצוא את z אשר מחזיק כמה שיותר מידע מהדטא המקורי x עם כמה שפחות פיצ'רים

עדיין יש n תצפיות אבל j פיצ'רים אבל לא k כאשר j<k

1. בשלב הראשון מחשבים את מטריצת הco variance של הנתונים X

האלכסון הראשי יהיה השונות של כל פיצ'ר והשאר יהיו השונות המשותפת בין כל שני פיצ'רים לפי העמודה והשורה. השונות המשותפת של x1 עם x2 היא כמו השונות המשותפת של x2 עם x1

גודל המטריצה הוא KxK מספר הפיצ'רים\* מספר הפיצ'רים.

2. לאחר מכן מבצעים פירוק עצמי של המטריצה- מקרה פרטי של SVD(Singular Value Decompostipm) שמוציא לנו את A- המטריצה של הווקטורים העצמיים של מטריצת הco variance.

3. לוקחים את המטריצה המקורית של הנתונים X ומכפילים אותה במטריצה של הוקטורים העצמיים A ונקבל את מטריצה Z שהיא המטריצה עם הפיצ'ירים המצומצמים.

העמודה הראשונה בZ תהיה הפיצ'ר שתופס הכי הרבה שונות מהנתונים, הטור לאחר מכן שתופס הכי הרבה שונות מתחתיו וכו'..הכפלה של מטריצת הנתונים בווקטור העצמי הראשון תיתן את הטור הראשון = כך הלאה לPC האחרים. ימשיכו להיווצר PCs עד שנקבל 90% מהנתונים מבחינת השונות (זה הדיפולט ניתן גם לשנות בהרצת האלגוריתם). הווקטורים מראים לנו אילו ערכים השפיעו על יצירת הקלאסטרים. לדוגמה ככל שsepal width משפיע יותר על התצפית, כך היא תלך יותר לכיוון הוקטור של sepal width, לכן התצפיות בקלאסטר הכחול הן בעלות sepal width גדול- ערך יותר גדול של התכונה= יותר קרוב לווקטור של התכונה והולך לכיוון שלו.

כאשר מציגים ויזואלית ניתן להשתמש רק ב2-3 PCs

Diagram

Description automatically generatedאם מנסים לעשות חיזוי ניתן להשתמש ביותר.

בדוגמה ציר x הוא PC1 וציר y הוא PC2. האחוזים הם כמה שונות כל PC תופס.

על מנת ליצור וקטור עצמי אשר תופס כמה ממדים של הנתונים מחשבים את סכום המרחקים של הטלי התצפיות(projections) על קו וקטור. עושים זאת עבור מספר וקטורים בעלי זווית שונה על מנת למצוא את הווקטור עם סכום המרחקים המרובעים הגדול ביותר(Sum of Squared Distances).

זה יהיה PC1.

Loading scores- ערך החשיבות שלוקחת כל תכונה על הווקטור העצמי (ערכים מ0 עד 1)

סכום המרחקים המרובעים גדול ביותר אשר יצר את PC1 נקרא הערך העצמי של PC1.

השורש של הערך העמי הוא הsingular value של PC1.

PC2 יהיה הקו האנכי לPC1 בראשית הצירים.

משתמשים בהטלי המרחקים של כל תצפית מהPCs על מנת למקם אותם בגרף כאשר PC1 הוא ציר X וPC2 הוא ציר Y.

ניתן להבין כמה אחוזים משונות הנתונים הPC מסביר באמצעות חלוקה של סכום המרחקים המרובעים של הPC במספר התצפיות פחות 1

Text

Description automatically generated

נסכום את התוצאות של שני הPCs ונחלק כל סכום מרחק המרובעים של PC בסכום הכולל וכך נקבל את אחוז השונות שהPC מסביר.

Text, letter

Description automatically generated

**SVD-Singular Value Decomposition**

אלגוריתם להפחתת ממדים, הבסיס של PCA.

**t-SNE**

אלגוריתם להורדת ממדים, משמש למציאת קלאסטרים, משמש לויזואליזציה של הנתונים, מאפשר להבין את המבנה של הנתונים בצורה טובה. מחליטים מראש את מספר הממדים שיצא. זמן הריצה של t-sne הוא ארוך מאוד

1. מחשבים את המרחקים בין כל זוג נקודות בממד הגבוה (לפי נורמה אוקלידית) ואז נקבל מטריצה שבה יש את המרחק של כל נקודה מהשאר.
2. עושים סטנדרטיזציה למרחקים על מנת לקבל ערכים בין 0 ל1, משתמשים בהתפלגות גאוסיאנית על מנת ליצור פיזור טוב של המרחקים- עבור כל נקודה מייצרים התפלגות נורמלית כאשר מרכז ההתפלגות היא הנקודה וממקמים את כל מרחקי הנקודות ממנה על ההתפלגות הנורמלית. כאשר הסטיית תקן של ההתפלגות שווה 1.מחלקים את המרחק של כל נקודה מהנקודה שבמרכז ההתפלגות בסכום המרחקים של כל הנקודות ממנה על מנת לקבל סטדריזציה מ0 ל1.
3. שמים את הנקודות על גרף דו ממדי, מחשבים את המרחקים בין כל זוג נקודות בגרף זה ואז נקבל מטריצה שבה יש את המרחק של כל נקודה מהשאר (מרחקים בדו ממד)
4. האלכסון הראשי במטריצות יהיה 0 כי המרחק של נקודה מעצמה הוא 0.
5. מזיזים את המרחקים בין הנקודות בדו ממד על מנת שהמרחק יהיה יותר קרוב למרחק במטריצה ברב מימד (המימד הגבוה) ואז המטריצה בדו מימד תייצג את המרחק בין הנקודות ברב מימד.
6. מחשבים את המרחק בין ההתפלגויות בין 2 ערכי המרחקים –של הממד הגובה והממד הנמוך ומנסים לצמצם את המרחק ביניהן- מעדכנים את המרחק בין הנקודות בצורה איטרטיבית כך שהמרחקים במימד הנמוך יהיו דומים למרחקים בממד הגבוה.

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated

Preplexity-כמה שכנים קרובים אנחנו רוצים

n\_iter-מספר האיטרציות עד שהאלגוריתם עוצר

\*t-SNE מאוד משופע מהיפר פרמטרים, לכן הוא משמש בעיקר לויוזאליזציה של הנתונים, אך לא לחיזוי.

**SOM-Self Organizing Maps**

משמש גם לקלאסטרינג וגם להורדת ממדים. עובד פחות טוב על פיצ'רים קטגוריים.

**Text

Description automatically generated**

**Text, letter

Description automatically generated**

מאתחלים נוירונים במקומות רנדומליים, בודקים את המרחק של כל נוירון לתצפיות, התצפית שהכי קרובה לנוירון הופכת להיות מקושרת לנוריון זה והנוירון מתקרב אליה. כיוון שהנוירונים מקושרים אז זה גם מושך ברמה מסוימת את שאר הנוירונים בכיוון זה.

חוזרים על תהליך זה עד שהנוירונים מתמקמים במקום מסוים בלי לזוז.

כך התצפיות שקרובות לנוירון מסוים הופכות להיות קלאסטר, לכל נוירון יהיה קלאסטר משלו.

Som חד ממדי-מגדיר את הנתונים על מקום ספציפי בציר אחד. Som דו ממדי- מחלק באמצעות 2 תכונות. Som עושה מספר איטרציות ולכן מטפל באותה תצפית מספר פעמים, יכולה לעבור לקלאס אחר.עבור קלט במרחב N מימדי ופלטי הנוירונים m:

1. בחר באופן רנדומלי וקטור משקולות wi עבור I , כאשר I רץ מ1 עד m (מספר הנוירונים ברשת)
2. בחר קלט רנדומלי X
3. תזהה נוירון "מנצח" K:
   1. כאשר המרחק האוקלידי הוא המינימאלי וכך ניתן להבין שהם הדומים ביותר
4. עדכן את כל וקטורי המשקלות עבור כל הנוירונים I שהם ב"שכונה" של הנוירון K

תחילה המשקלים מאותחלים רנדומלית. לאחר מכן בוחרים קלט רנדומלי x . מזיזים את כל הנוירונים על פי הנוירון x הזה שבחרנו. נוירונים שהכי קרובים זזים הכי הרבה, הכי רחוקים זזים הכי פחות

ממשיכים עד שהנוירונים מפסיקים לזוז. כך שומרים על יחס המרחקים בין הנוירונים בממד הנמוך יותר. Loss function of SOM:



Graphical user interface, text, application

Description automatically generated

**היפר-פרמטרים:**

קצב הלמידה, סוג הפונקציה אשר משמשת למדידת מרחק, מספר איטראציות מקסימלי

**Edison**

משתמש ב mean shift על מנת לחלק את התמונה, בעלי שני ערכי h : אשר משמש להתייחסות לתוכנות מרחביות, המיקום של הפיקסל בתמונה, ו אשר משמש להתייחסות לתכונות של צבע. באמצעות מודל זה ניתן לבצע קלאסטרינג לפי אובייקטים בתמונה וכך לזהות אובייקטים שונים בתמונות.

**Viola Jones- אלגוריתם לזיהוי פרצופים בתמונות**

הקלט הוא תמונה בשחור לבן.

Harr Features- חלוקה של אזור מרובע מסוים בתמונה לפיקסלים בצבע שחור ופיקסלים בצבע לבן, על מנת לבדוק את ההפרש בין סכום הפיקסלים השחורים ללבנים. כל פיקסל מכיל מספר אשר מסמל כמה הוא כהה

Integral image- סכימה של הפיקסלים בכל שורה ובכל טור- כל פיקסל מכיל את ערך האפרוריות של כל הפיקסלים שמעליו ומשמאלו. המטרה היא לייצג את התמונה במספר תאים קטנים יותר במטריצה על מנת לחסוך בזמן חישוב.

האלגוריתם מזהה כך את האזורים בתמונה בעלי ה haar features החשובים על מנת לזהות אם יש פרצוף בתמונה או לא. האלגוריתם כולל adaboost כאשר המסווג הראשון מסווג את התכונה הכי חשובה, לאחר מכן המסווג השני מסווג את התכונה החשובה אחריה וכו..

Diagram

Description automatically generated

Table

Description automatically generated

Attentional Cascade- שיטה לשילוב בין מסווגים במבנה מדורג אשר מגדיל את מהירות הזיהוי ע"י התמקדות באזורים רלוונטיים בתמונה. היתרון בשיטות כאלו אשר מנסות להתמקד באזורים מסויימים בעיקר בתמונה היא שלעיתים ניתן לזהות מהר מאוד היכן בתמונה נמצא האובייקט עיבוד מורכב יותר מתבצע על האזורים האלו שזוהו.

**Neural Networks**

לדוגמה יש לנו תרופה שמרפאה מחלה, במינון שונים. במינון נמוך התרופה לא הייתה אפקטיבית (0) במינון בינוני היא כן (1).רשת נוירונים יכולה להתאים את עצמה לפרבולות ולא רק קוים ישרים.

רשתות נוירונים בנויות ממספר שכבות כאשר השכבה הראשונה מקבלת את הקלט והשכבה האחרונה מוציאה את הפלט. כאשר מבצעים סיווג אז מקבלים כפלט את ההסתברות של תצפית להיות שייכת לכל קלאס, נשייך את התצפית לקלאס עם ההסתברות הכי גבוהה. אם זה רגרסיה נקבל מספר. על הקשרים בין הצמתים יש מספרים אשר מהווים את השיפוע ax+b בין הצמתים.

כאשר מכפילים את הערך בקלט שקיבלנו במספר, המספר שמכפילים בו נקרא "משקל"(a), המספר שמוסיפים נקרא bias(b). הצמתים השונים יכולים לבצע פונקציות אקטיבציה שונות על הנתונים:

Softplus,relius,sigmoid- ניתן להגדיר בהיפר פרמטרים. לכל צומת יכול להיות פונקציית אקטיבציה שונה. מנסים פונקציות אקטיבציות שונות עד שמקבלים את רמת הדיוק הכי גבוהה.

במקרה שלנו, המינון של התרופה הוא בין 0 ל1. נתחיל מ0 בצומת הראשון, מחשבים את ערך שמתקבל מהמעבר לצומת הבא ומוסיפים את ערך הbias, מפעילים את הפונקציה של הצומת הבא, במקרה זה softplus. התוצאה שקיבלנו היא 2.25 . כרגע זה החיזוי שקיבלנו עבור מינון שהוא 0.

כאשר נשים תצפית אחרת(מינון=0.1) נבצע את אותו תהליך ונקבל ערך אחר. נעבור כך על כל התצפיות ונקבל עקומה לכמה התרופה אפקטיבית במינונים שונים. לא בהכרח משתמשים כל הערכים של פונקציית האקטיבציה, אלא רק בחלק, לכן ניתן ליצור מספר עקומות שונות ולפי החלק שמשתמשים בפונקציית אקטיבציה. מכפילים במשקל של הקשת הבאה לצומת הבא ומקבלים עקומה חדשה. בנוסף שולחים את כל התצפיות לnode אחר ומשתמשים במשקלים אחרים מה שנותן עקומה אחרת. סוכמים את העקומה של 2 הצמתים ומקבלים פרבולה אשר חוזה את יעילות התרופה למינונים שונים.

Backpropagation-שיטה למציאת הערכים של המשקלים והbias לאורך הקשרים בין הצמתים ברשת. מתחילים מהקשר לצומת האחרון, ובאמצעות מזעור של הערך שמתקבל מסכום ההפרשים המרובעים בין הערך שנובא לערך האמיתי, מוצאים מה הערך שצריך להיות למשקלים ולbiases. מוצאים את הערך האופטימלי, באמצעות שימוש בgradient descend בו יוצרים פונקציה של ערכי ההפרשים המרובעים, ומחפשים את הנקודה בה הנגזרת של הפונקציה שווה ל0- נקודות מינימום.

באמצעות back-propagation בדרך "הלוך" כל המשקלים מושמים בצמתים וההשפעה שלהם מתפשטת על פני הרשת .מתקבלת תוצאה בשכבה האחרונה ואז במעבר בכיוון ההפוך , המשקלים מתוקנים שכבה אחר שכבה בעזרת חוק תיקון הטעות.   
**הError Signal** מאפשר לחשב מה יהיה השינוי בw-המשקלים. f זה הloss function-רוצים שהתוצאה שלה תהיה 0 כדי שלא תהיה טעות

𝑓(𝑥,𝑤+𝛿𝑤)=𝑓(𝑥,𝑤)+(𝑑𝑓/𝑤)∗𝛿𝑤

רוצים שהערך של (𝑑𝑓/𝑤)∗𝛿𝑤 יהיה שלילי כדי שהתוצאה שלנו 𝑓(𝑥,𝑤+𝛿𝑤) יהיה יותר קטן- פחות טעות.***הפרספטרון****הוא אלגוריתם  ממשפחת האלגוריתמים הלומדים. זהו אלגוריתם למידה "און ליין", משמע הוא לומד תוך כדי ריצה מדגימות שאבחן בזמן פעולתו. מטרתו של האלגוריתם היא לאבחן בין סוגים שונים של דגימות  אותן הוא מקבל.*  
***היפר פרמטרים:***

*מספר הנוירונים ומספר השכבות, הסיווג הסופי הוא לפי השכבה האחרונה.*

Learning rate: *גדול מדי- נפספס את נקודת המינימום המקומי, קטן מדי- לאלגוריתם ייקח הרבה זמן.*

**הכנה למבחן**

תהליך בניית מודל- מקבלים דטא, להסביר על pre processing, להסביר מה צריך לעשות מבחינת הכנה של נתונים לכל אלגוריתם, איך כל אלגוריתם עובד מבחינה רעיונית.

**בעץ חייבים לעשות השלמה של ערכים חסרים**

**KNN עובד יותר טוב עם scaling**

קיבלנו אחוז דיוק גבוה בtrain ואחוז דיוק נמוך בtest- יש overfitting, להסביר מה אפשר לעשות.

**שאלות של רשת נוירונים:**

סיווג:

שכבת הקלט- כמות הנוירונים היא כמות הפיצ'רים שמתקבלים

שכבת הפלט -כמות הנוירונים היא מספר המחלקות

רגרסיה:

שכבת הקלט- כמות הנוירונים היא כמות הפיצ'רים שמתקבלים

שכבת הפלט- נוירון אחד שנותן את התוצאה.

**Loss functions**

בtSNE- ההסתברות בין זוג נקודות בממד הגבוה לבין ההסתברות בממד הנמוך צריכה להיות כמה שיותר דומה- מינימום גלובלי

עצים- ממזערים entropy או gini, לוקאלי

SVD/PCA- הייצוג בעזרת האלגוריתם יהיה כמה שיותר קרוב לדטא המקורי- יתפוס כמה שיותר שונות

**Loss functions regression:**

**Loss functions classification**

**Loss functions- גלובלי או לוקאלי**

SVM- נרצה למזער את המרחק בין הsupport vectors על מנת למנוע כמה שיותר טעויות סיווג , לכן נרצה למזער את -גלובלי

K-Means: למזער את סכום המרחקים מהמרכז של כל קלאסטר, לוקאלי- כי הוא מתחיל מנקודות רנדומליות ולכן ימצא מינימום לוקאלי

עץ החלטה,adaboost,random forest-מחלק את הנתונים לקבוצות שבהן הנתונים דומים- לוקלי

SVD-שומר את הנתונים במימד נמוך- גלובלי

**שאלה שחוזרת על עצמה**

Adaboost,random forest , bagging- מה משותף לשלושת האלגוריתמים: הם super classifiers- משתמשים בהרבה weak learns ומבצעים אגריגציה ביניהם כדי לקבל תוצאות יותר מדויקות. למה כדי להשתמש באחד מאלו ולא עץ החלטה?

הם עושים אגריגרציה של כמה weak learns ולכן מקבלים תוצאה יותר מדויקת בדרך כלל מאשר עץ אחד.

Adaboost משתמש בweak learns הקודמים כדי להגיע לתוצאה עם הlearner הנוכחי, ה2 האחרים לא.

Adaboost וbagging יכולים להשתמש בweak learns שהם לא עצים. Random forest יכול להשתמש רק בעצים.

**איך ממשים bagging עם svm?**

לוקחים dataset ועושים עליו svm

לוקחים dataset אחר ועושים עליו svm..

לפי הרוב\ ממוצע ביניהם בוחרים את התוצאה

**אילו בעיות יכולות להיות כאשר משתמשים בrandom forest עם svm?**

Random forest עובד רק על עצים.

כאשר עושים הגרלה על הפיצ'רים בrandom forest אז מקבלים רק חלק מהפיצ'רים כל פעם, ואז נקבל עצים אשר כל פעם מחלקים באמצעות svm לפי פיצ'רים שונים

ניתן לפתור את זה באמצעות שימוש בadaboost במקום ואז כל עץ יחלק עם svm לפי פיצ'ר אחד.

**אילו בעיות יהיו לשימוש בadaboost עם svm?**

Adaboost נותן משקלים לכל weak learner ובsvm אין משקלים

ניתן לפתור את זה באמצעות לתת משקל לכל נקודה ואז להשתמש בsvm.

**שלבים של :pre processing**  לעשות one hot encoding על משתנים קטגוריים, להוריד משתנה ייחודי לכל שורה כמו תעודת זהות, לבדוק קורלציה ולהוריד תכונות עם קורלציה גבוהה, השלמת נתונים, לבצע נורמליזציה למספרים בטווח מסוים, לדוגמה בין 0 ל1.

SVM לא עובד עם תכונות קטגוריות ולכן לעשות one hot encoding.אלגוריתמים של עצים יכולים לעבוד עם משתנים קטגוריים. עצים לא יודעים לעבוד עם נתונים חסרים, ולכן נדרש לעשות imputation כדי להשלים ערכים חסרים.

KNN עובד יותר טוב עם תכונות שעשו עליהם scaling

אם בתהליך האימון המודל מאוד מדויק ועל הסט מבחן הוא ממש לא מדויק. הסיבה לכך יכולה להיות שהנתונים לא מאוזנים- לעשות preprocessing לlabel, או overfitting- צריך להשתמש בהיפר פרמטרים אחרים. כאשר יש הרבה מאוד פיצ'רים יש זמן ריצה ארוך מאוד, יכול להיות שיש הרבה פיצ'רים זבל, לכן מחפשים קורלציות על מנת להוריד פיצ'רים או להשתמש בPCA על מנת להוריד ממדים.

gridsearchCV בודק את כל השילובים האפשריים בין פרמטרים- לוקח המון זמן, אם חשוב יותר הדיוק אז נשתמש בgridSearch.

RandomizedSeachCV לוקח שילובים אפשריים רנדומליים, לא עובר על כל השילובים, מאפשר זמן ריצה קצר יותר במקרים שבהם הדיוק פחות חשוב והזמן ריצה יותר חשוב.

**Confusion Matrix**

Sensitivity- בודק כמה טוב אנחנו מזהים משהו חיובי (לדוגמה אם מגדירים זיופים כחיובי אז נרצה לבדוק את הsensitivity).

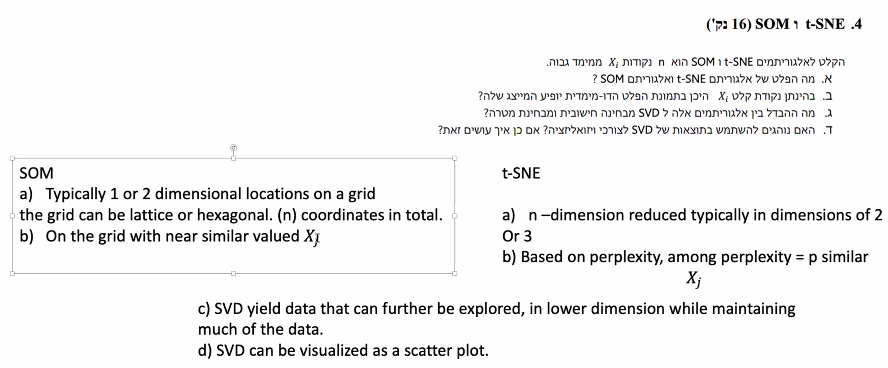
צריך סט ולידציה כדי לעשות cross validation של המודל על מנת למצוא את ההיפר פרמטרים שבונים את המודל הכי טוב. לאחר מכן בודקים את התוצאה של המודל הכי טוב על סט המבחן.

בSVM אם ניתן לclass מסוים משקל גדול יותר כך נגדיל את החשיבות של לסווג קלאס מסוים.

לדוגמה יותר חשוב לזהות קלאס של טפסים שמסווגים כמזויפים.

**עצי החלטה**

כאשר אומרים לנו לבנות עץ החלטה נדרש לחלק את הגרף לקווים על פי הקלאסים , כל קו הוא עוד עומק בעץ ובונים את העץ לפי זה.



מבחינה חישובית som וt-SNE מתחילים בצורה רנדומלית וSVD לא מתרחש בצורה רנדומלית

Table

Description automatically generated

Chart, scatter chart

Description automatically generated

Kernel SVM- פונקציה שמתמשים בה כדי להעלות את הנתונים לממד גבוה יותר על מנת למצוא hyperplane להפריד אותם ל2 מחלקות.



Overfitting לdata מסוים, רמות דויק שונות לkernels שונים.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **אלגוריתם** | **דרישות** | **יתרונות** | **חסרונות** | **Loss Function** |
| **עץ החלטה** | לא יכול nulls | ניתן לראות את תהליך קבלת ההחלטה ולתרגם אותו לשפות אחרות | בעל high variance – רגיש לסט נתונים שנותנים להתאמן עליו.  עלול ליצור עץ מאוד מסובך בשימוש ברגרסיה  לא בעל דיוק גבוה | סיווג: Gini/ entropy  רגרסיה: SSR |
| **Adaboost** | לא יכול nulls, רגיש לoutliers ולכן עדיף scaling | לומד מהטעיות של המסווג הקודם | זמן ריצה ארוך | סיווג: Gini/ entropy  רגרסיה: SSR |
| **Random Forest** |  | דיוק גבוה, משתמש בעצים שונים בתכונות שונות וכך ניתן להבין חשיבות של תכונות, אין צורך בנרמולים | יקר מבחינת זמן ומקום | סיווג: Gini/ entropy  רגרסיה: SSR |
| **Gradient Boosting** |  | מדויק, אין צורך בpre processing של משתנים קטגוריים ומידע חסר | רגיש לoutliers, לוקח הרבה זמן, מושפע מהיפר פרמטרים | SSR |
| **KNN** | עדיף לעשות scaling, נדרש להגדיר מראש את מספר השכנים | לא צריך אימון, יכול להתמודד עם סיווג כאשר הdata מחולק באופן מסובך | תצפיות חריגות עלולות להיכנס לקלאסטר אחר, יקר מאוד עם dataset גדול ועם הרבה ממדים | אין |
| **K-Means** | נדרש להגדיר מראש את מספר השכנים, להוריד שורות כפולות, לא יכול לקבלnulls | - | יוצר רק קלאסטרים בצורת עיגול, יקר מאוד עם dataset עם הרבה ממדים | ,WCSS סכום מרחקי הנקודות ממרכז הקלאסטרים |
| **GMM** |  | מדויק יותר מK-Means, יכול ליצור קלאסטרים בצורות שונות | מקלאסטר לפי התפלגות נורמלית בלבד | Negative logliklihood |
| **Hierarchical Clustering** |  | לא דורש הגדרה מראש של מספר הקלאסטרים, מראה בצורה ברורה את תהליך הקלסטרינג | מאוד רגיש לoutliers | Affinity -רמת דמיון בין קלאסטרים |
| **KDE** |  | מבצע קלאסטרינג לפי התפלגיות שונות | מושפע מאוד מהיפר פרמטרים | משתנה לפי הkernel |
| **PCA** | חייבים סטנדריזציה | משפר ביצועים של אלגוריתמים, מונע overfitting, מאפשר ויזואלזציה, מראה את חשיבות התכונות | אובדן מידע | מזעור של איבוד מידע |
| **t-SNE** |  | מתמודד בצורה יעילה עם נתונים לא לינארים | בעל זמן ריצה ארוך, רגיש מאוד לשינויים בהיפר-פרמטרים | Kullback leibler divergence  צמצום ההבדל בין כל זוג נקודות בממד הגבוה לממד הנמוך |
| **SVM** | דורש scaling של הנתונים כדי למנוע זמן ריצה ארוך מאוד | דיוק גבוה, אפקטיבי בממדים גבוהים, יעיל מבחינת מקום | בעל זמן ריצה ארוך, כמות קטנה של ערכים חריגים עלולה לפגוע רבות בדיוק המודל, לא מתאים לdatasets גדולים, מתקשה להפריד בין קלאסים דומים | נרצה למזער את המרחק בין הsupport vectors על מנת למנוע כמה שיותר טעויות סיווג |
| **SOM** |  | הורדת הממדים מאפשר לצפות בנתונים בקלות, אפשר לתת משקלים לתכונות שונות | הקלאסטרים תלויים אחד בשני ולכן פחות מדויקים, נדרש לבחור את מספר הנוירונים מראש | מופיע למעלה |
| **Bagging** |  | מונע overfitting, מאפשר אופטימיזציה של היפר פרמטרים, מאפשר שימוש בdataset קטן | דורש יותר סיבוכיות מקום וזמן |  |
| **Neural Networks** | לא יכול לקבל nulls | מגיע לרמת דיוק גבוהה | סיבוכיות זמן גדולה | סיווג:cross entropy  רגרסיה:  Least Squared |
| **KD Tree** |  | מהיר | לא מדויק בממדים גבוהים, עלול לפספס את השכן הקרוב אם הוא בעלה אחר ברמה קודמת | אין |
| **LSH** |  | מהיר | עלול לפספס את השכן הכי קרוב אם הוא בתא אחר | אין |