מבוא למערכות לומדות - חורף תשפ״ג

מספר קורס - 236756

HW2 wet

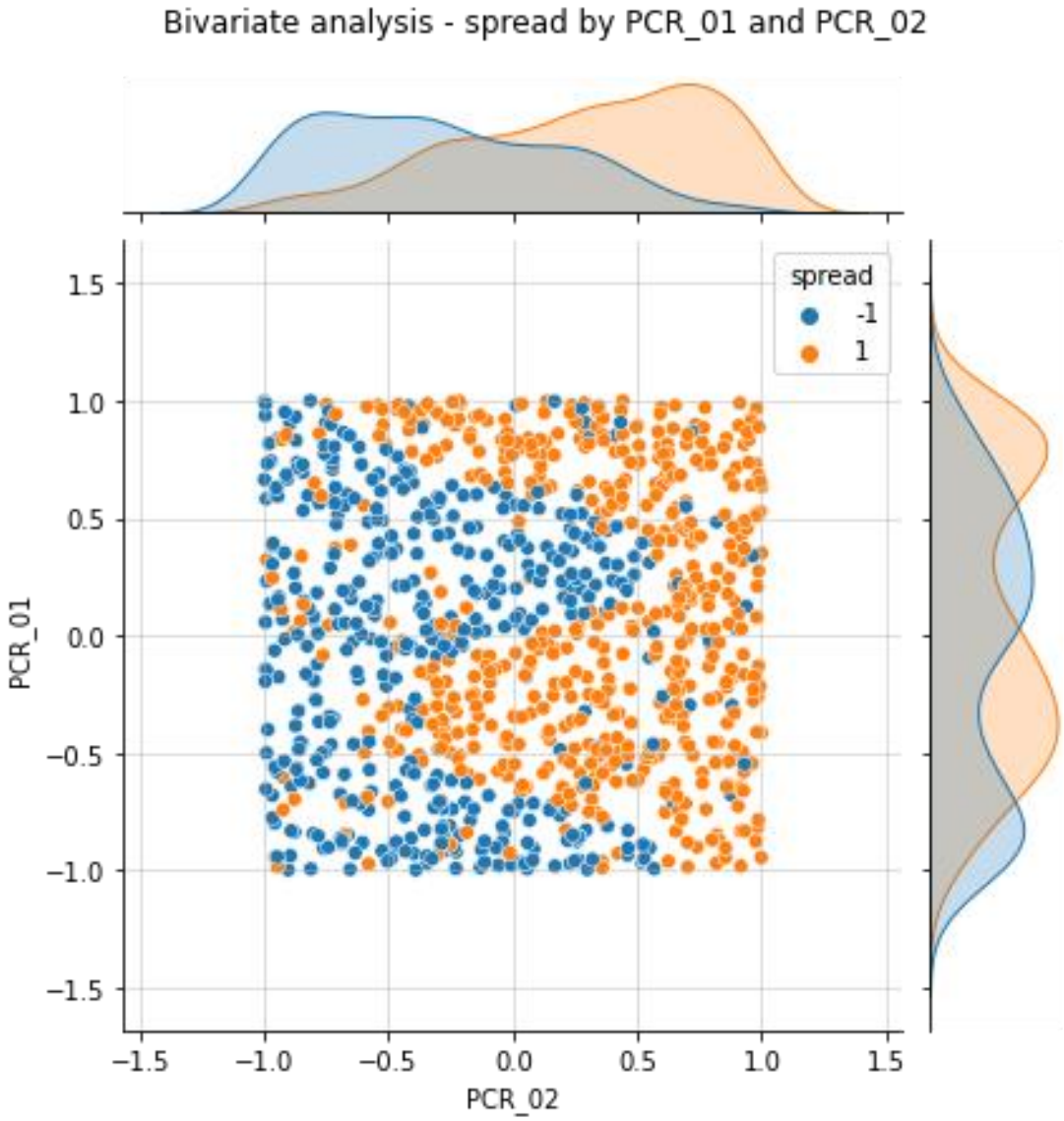
לוי הורביץ

313511602

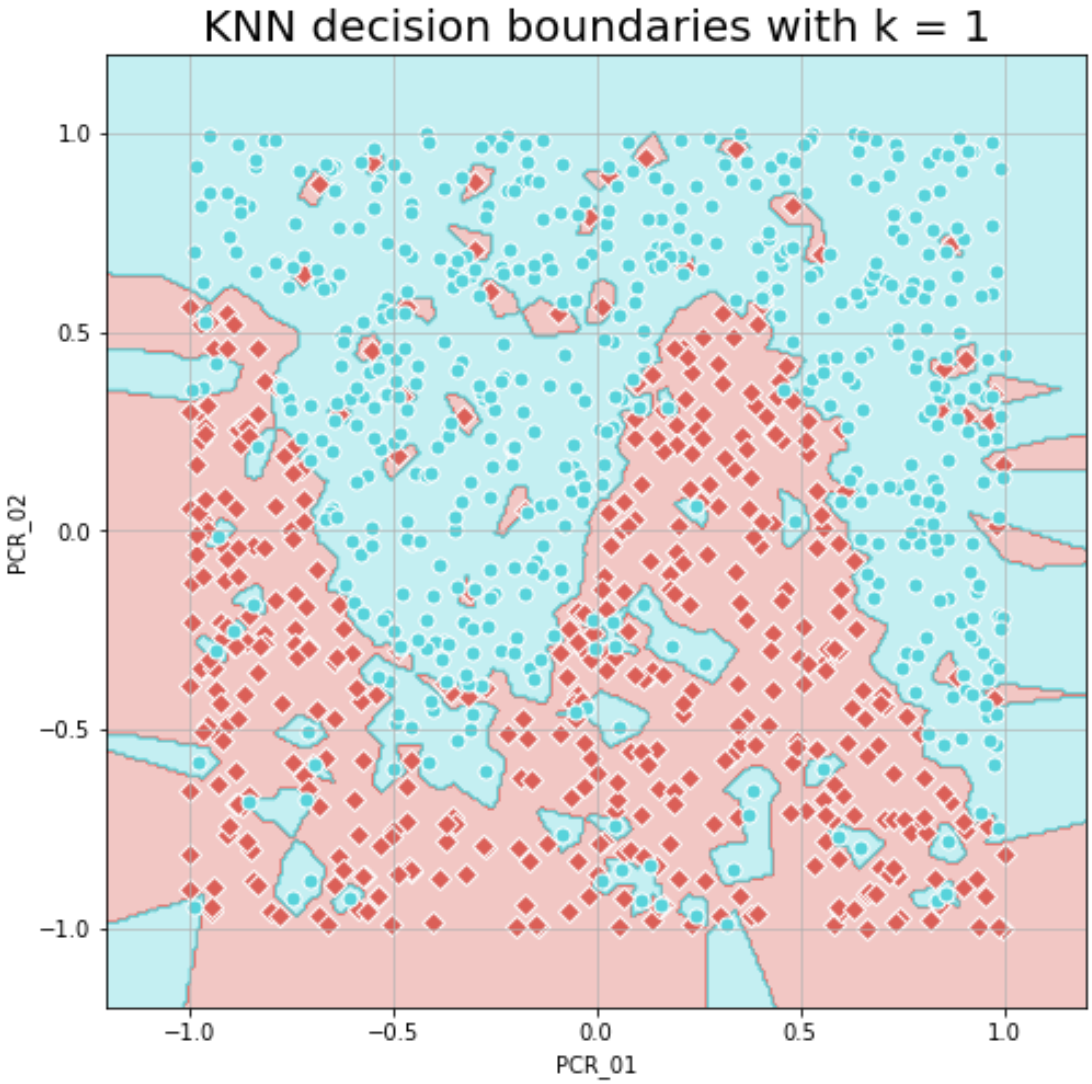
נועה דיקמן

315478867

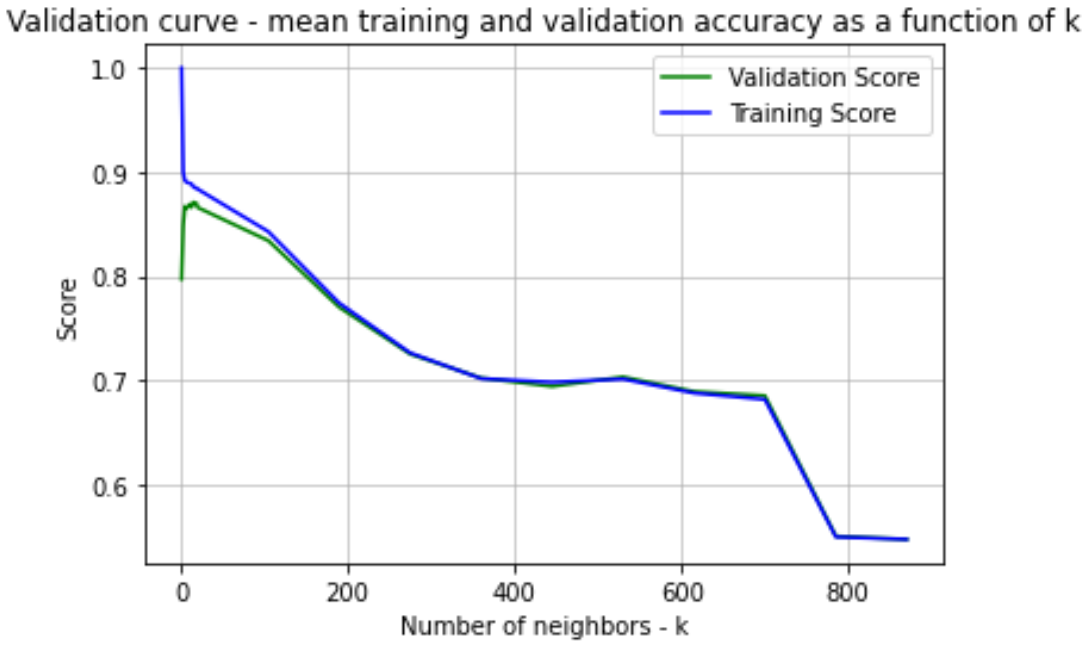
סעיף 1:


Bivariate analysis - spread by PCR_01 and PCR_02
spread
-1
1
PCR_01
PCR_02
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5

סעיף 2:


KNN decision boundaries with k = 1
PCR_02
PCR_01
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

סעיף 3:


Validation curve - mean training and validation accuracy as a function of k
Validation Score
Training Score
Score
Number of neighbors - k
1.0
0.9
0.8
0.7
0.6
0
200
400
600
800

Best k is: 15

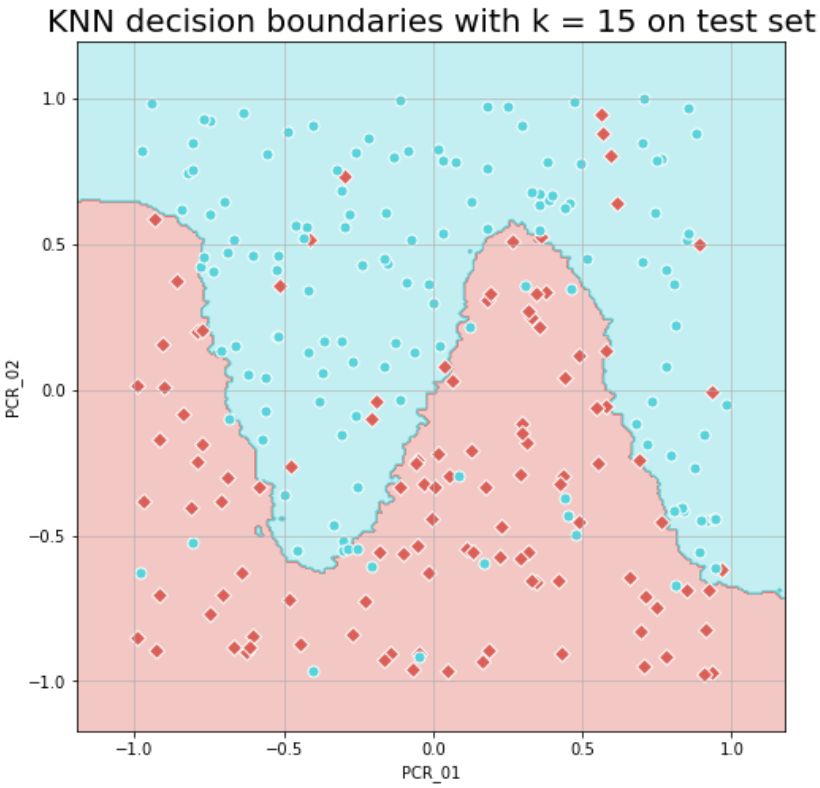
Mean validation accuracy of model with best k: 0.871

Mean train accuracy of model with best k: 0.886

אנו נמצאים במצב של overfitting עבור ערכים של k קטנים או שווים ל-15, מכיוון שאנו רואים בגרף שבהתחלה התוצאות של ה-validation set עבור ערכים של k כנ"ל הן די נמוכות בהשוואה לתוצאות של סט האימון. על סט האימון אנחנו מקבלים תוצאות מאוד גבוהות עבור ערכי k קטנים, מכיוון שכל נקודה היא הקרובה ביותר לעצמה, ולכן כאשר בודקים את התוצאות על סט האימון, מקבלים תוצאות מאוד גבוהות, אבל זה גורם להכללה להיות לא טובה, וזה אכן מה שקיבלנו כאשר בדקנו את התוצאות על ה-validation set.

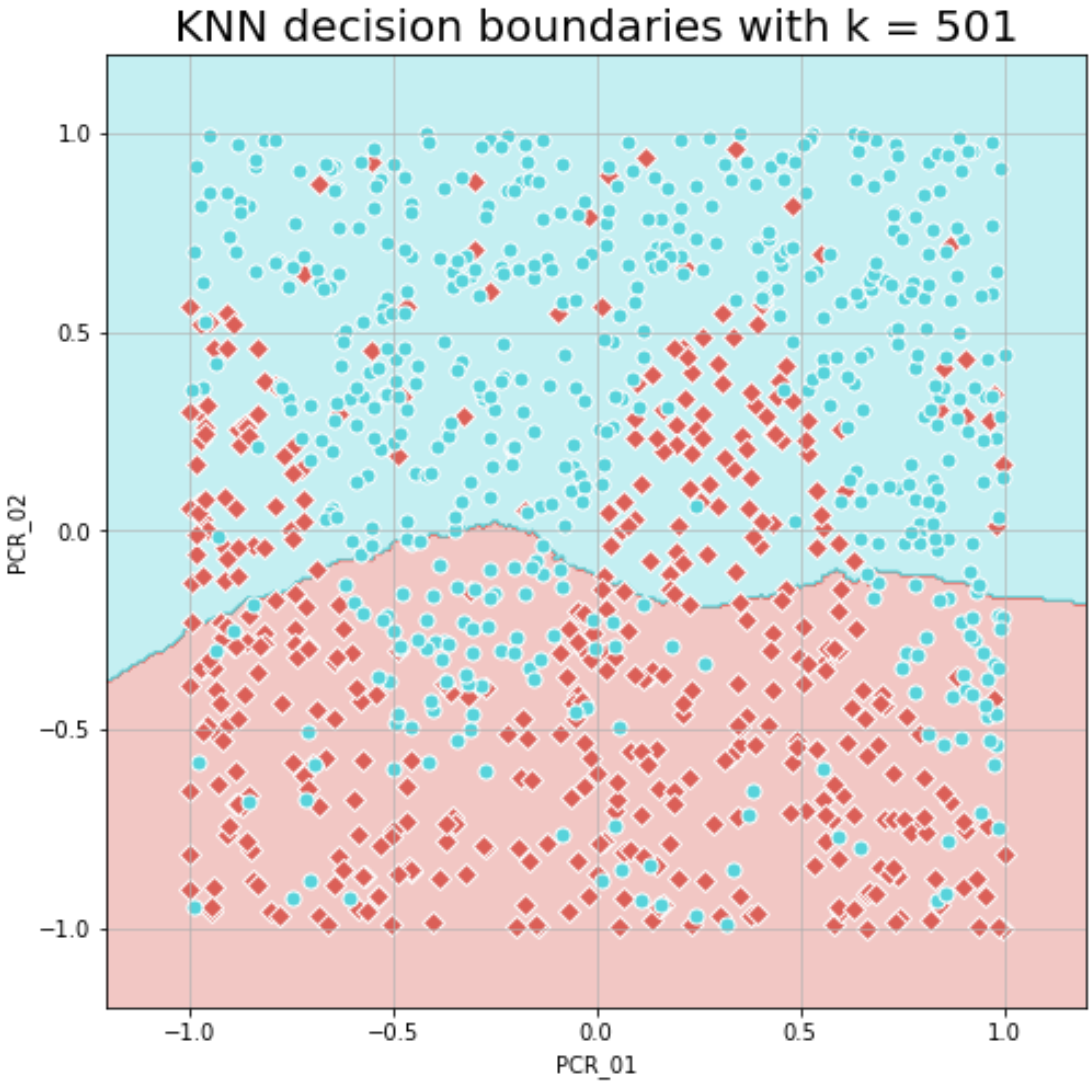
לאחר מכן, יש ירידה באחוז הדיוק גם עבור סט האימון וגם עבור ה-validation set כאשר מסתכלים על ערכי k גדולים מ-15. זה מצב של underfitting בו ניסינו לקחת יותר מדי נקודות קרובות בשביל לאמן את המודל שלנו, אבל כפי שראינו בסעיף 1 את איך שהנתונים מתפלגים – הם מתפלגים בצורה של מצבורי נקודות כמו מלקחיים מאותו סוג באזור מסוים. לכן, אם ניקח במצב הזה יותר מדי שכנים, אז בעצם נגרום למודל שלנו להתחשב גם בנקודות שלא בתוך אותו המצבור נקודות, ובכך בעצם מאוד לפגוע במודל, כי ברור לנו שנקודות שלא בתוך אותו מצבור, לא יכולות ללמד אותנו על הפרדיקציה שצריך להביא לנקודה אותה אנו חוזים. כלומר, במצב של k גדול מדי, אז אנחנו מקבלים מצב של underfitting בגלל הסיבה שתיארנו לעיל.

סעיף 4:



Test set score of KNN model with best k: 0.856

סעיף 5:



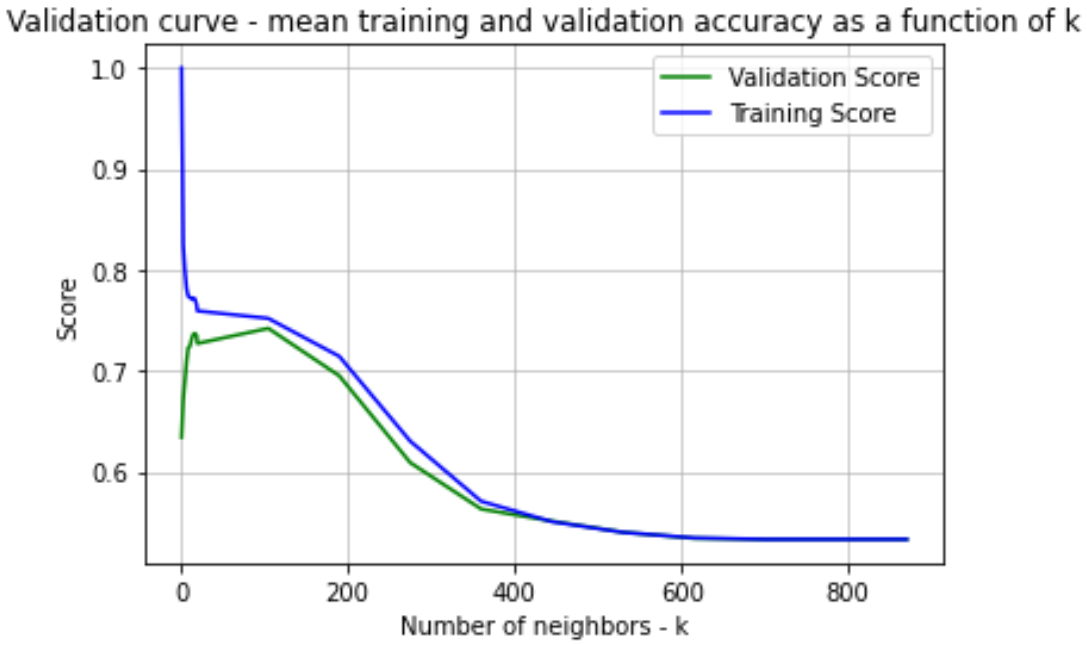
הגרפים של ה-decision boundaries של שני המודלים k=1 ו- best_k נמצאים בסעיפים הקודמים (סעיף 2 וסעיף 4). ה-score של שלושת המודלים הללו הן:

```
Train set score with k = 1: 1.0
Test set score with k = 1: 0.796

Train set score with k = 501: 0.695
Test set score with k = 501: 0.7

Train set score with best k: 0.885
Test set score with best k: 0.856
```

ניתן לראות שעל סט האימון התוצאות של המודל עם k=1 הן מושלמות, אבל זה מכיוון שכל נקודה היא הקרובה ביותר לעצמה, ולכן בוודאות נקבל שהדירוג הוא 1, אבל כאשר מסתכלים על סט המבחן, באמת הניקוד של המודל הזה נמוך מאשר הניקוד של המודל עם ה-k הטוב ביותר. הסיבה לכך היא שכפי שניתן לראות מהגרפים של ה-decision boundaries של שני

המודלים, הגרף של המודל עם k=1, פחות חלק ויותר רגיש לנקודות שלא בדיוק נמצאות במקום. כלומר, כפי שניתן לראות את המיקום של הנקודות ב- joint plot שיצרנו בסעיף 1, הנקודות הן די מופרדות, אך עדיין יש נקודות שנמצאות לא בדיוק באזור שמוקף בנקודות עם אותה תווית, אז באזור הקרוב ביותר לנקודה כזו, אם ניקח k=1, אז לא נוכל להכליל בצורה טובה את המודל שלנו, כי הוא יושפע מכך שהנתונים לא מופרדים בצורה מושלמת ועדיין יש מעט נקודות שחורגות מהאזור שלהן. זאת לעומת מצב שבו לוקחים k_best (במקרה שלנו 15), ששם ניתן להכליל בצורה טובה הרבה יותר מכיוון שהמצב שתיארנו קודם לא יקרה, ואכן קיבלנו דיוק טוב הרבה יותר על סט המבחן. משא"כ ב k= 501 אז ההערכה העשית בצורה גסה מאוד מכיוון שהוא מתחשב כמעט בכל השכנים לצידו, גם של הערכים השגויים (כמו שפירטנו בסעיף 3) וקשה לו לרדת לרזולוציות קטנות.

סעיף 6:



Best k is: 105

Mean validation accuracy of model with best k: 0.742

Mean train accuracy of model with best k: 0.752

Train set score with best k: 0.753

Test set score with best k: 0.76

ישנם מספר הבדלים בין התוצאות שקיבלנו בסעיף זה לבין סעיף 3: הראשון, ה-k הטוב ביותר, והשני הערכים של k עבורם יש overfitting ו-underfitting, ובנוסף התוצאות של המודל שאימנו בסעיף 6 פחות טובות על סט האימון והמבחן.

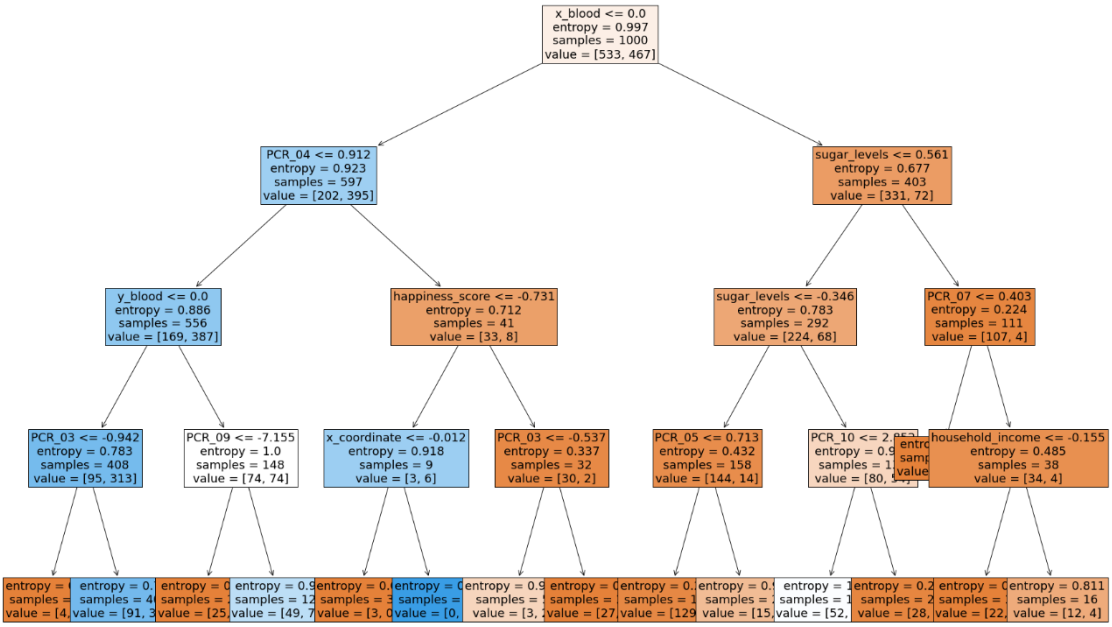
הסיבה לכך היא כנראה בגלל שבשביל להשתמש במודל של KNN אנחנו צריכים שהנתונים יהיו פרידים בצורה כלשהי, כלומר שניתן יהיה להגדיר אזורים מסוימים שכל נקודה שנמצאת בהם תקבל תווית מסוימת. עבור הפיצ'רים PCR_01 ו-PCR_02, זה אכן המצב, אבל עבור שאר הפיצ'רים זה לא מתקיים בצורה טובה (כפי שראינו בקשר לכמה פיצ'רים ב-HW1). לכן, כאשר אנחנו מתחשבים בכל הפיצ'רים בשביל לאמן את המודל KNN, אנחנו בעצם מכניסים פיצ'רים שהם לא פרידים להיות חלק מהמודל שלנו, ובהתאם דבר זה פוגע ביכולת של המודל להכליל ולתת פרדיקציה טובה.

הערכים של k עבורם יש overfitting לא כל כך השתנו בצורה דרסטית וזה הגיוני לאור המנגנון בו KNN עובד, מכיוון שכפי שאמרנו לעיל, KNN מסתמך על המרחק בין כל זוג נקודות (מרחק אוקלידי במקרה שלנו), ולכן כאשר בודקים את הדיוק של המודל על סט האימון, אז כל נקודה היא הקרובה ביותר לעצמה. לעומת זאת, על ה-validation set מצב זה יגרור overfitting, מכיוון שאנו מסתמכים על מספר קטן מדי של נקודות, שעלול לגרום למצב שאם יש נקודה עם תווית שונה בתוך מצבור נקודות עם תווית אחרת, אז הנקודה הזו עלולה להשפיע על הסביבה הקרובה ביותר אליה, למרות שזו לא הפרדיקציה ההגיונית במקרה זה.

הערכים של k עבורם יש underfitting יחסית השתנו, כלומר הפכו ממצב בו יש underfitting כאשר k=15 למצב בו יש כבר underfitting כאשר k גדול מ 105. מכיוון שהשתמשנו במודל הזה בכל הפיצ'רים ולא רק ב-PCR_01 ו-PCR_02, אז כפי שהסברנו לעיל, סביר להניח שהמודל החדש לא פריד כפי שהיה במודל הקודם שבו היו רק את שני הפיצ'רים הללו, אלא כרגע המודל יהיה פחות פריד בהשוואה למודל הקודם. מכיוון שהמנגנון של KNN עובד לפי מדידת המרחקים בין כל זוג וקטורים במקרה זה, אז אם יש פיצ'רים שהם לא פרידים בצורה טובה יחסית, זה ישפיע על התוצאות של המודל, מכיוון שאנו מעשה פרדיקציות במטרה לבדוק את השכנים הקרובים ביותר של נקודה כלשהי, על פי נתונים שהם לא פרידים בצורה טובה.

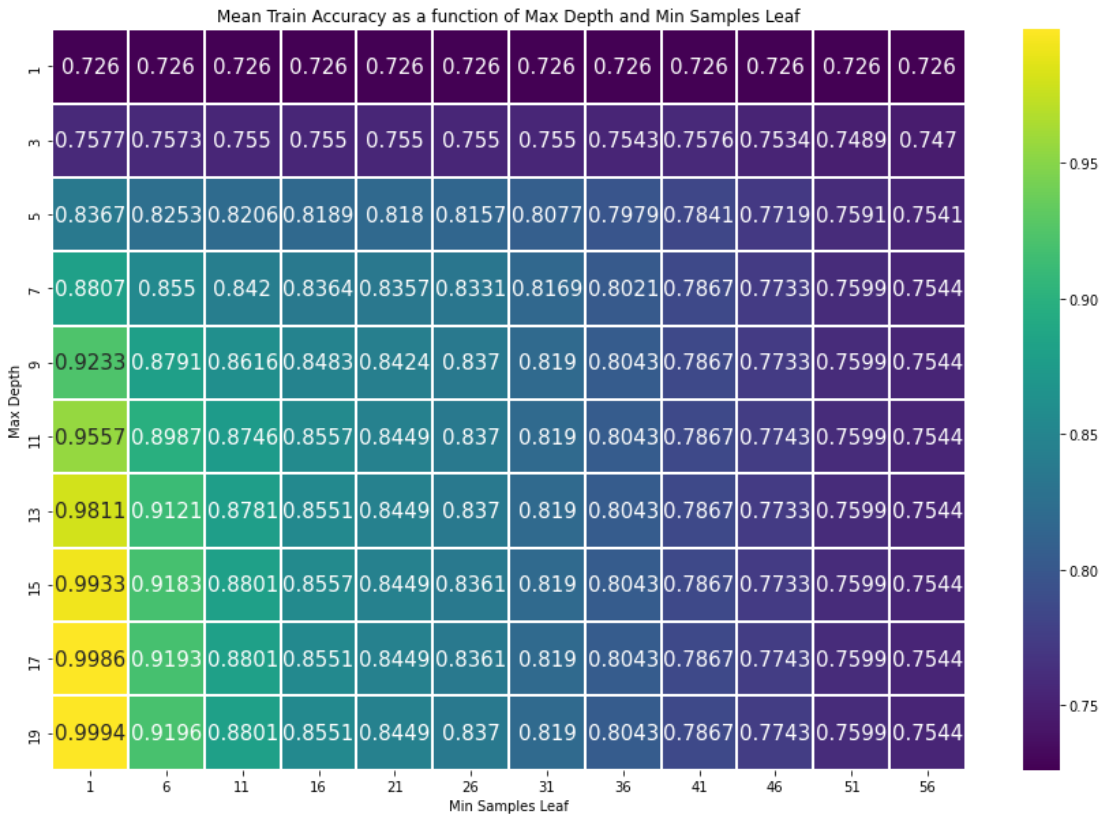
בסופו של דבר הביצועים של המודל אכן נהיו פחות טובים במקרה זה, בגלל הסיבה שעכשיו ציינו, אבל עדיין הביצועים היו יחסית בסדר, וזה כנראה בגלל שעדיין חלק מהפיצ'רים עדיין פרידים, ובנוסף כנראה שהקומבינציה של הפיצ'רים אחד עם השני גרמה לכך שעדיין ניתן היה להשתמש במסווג מסוג KNN בשביל לעשות פרדיקציה.

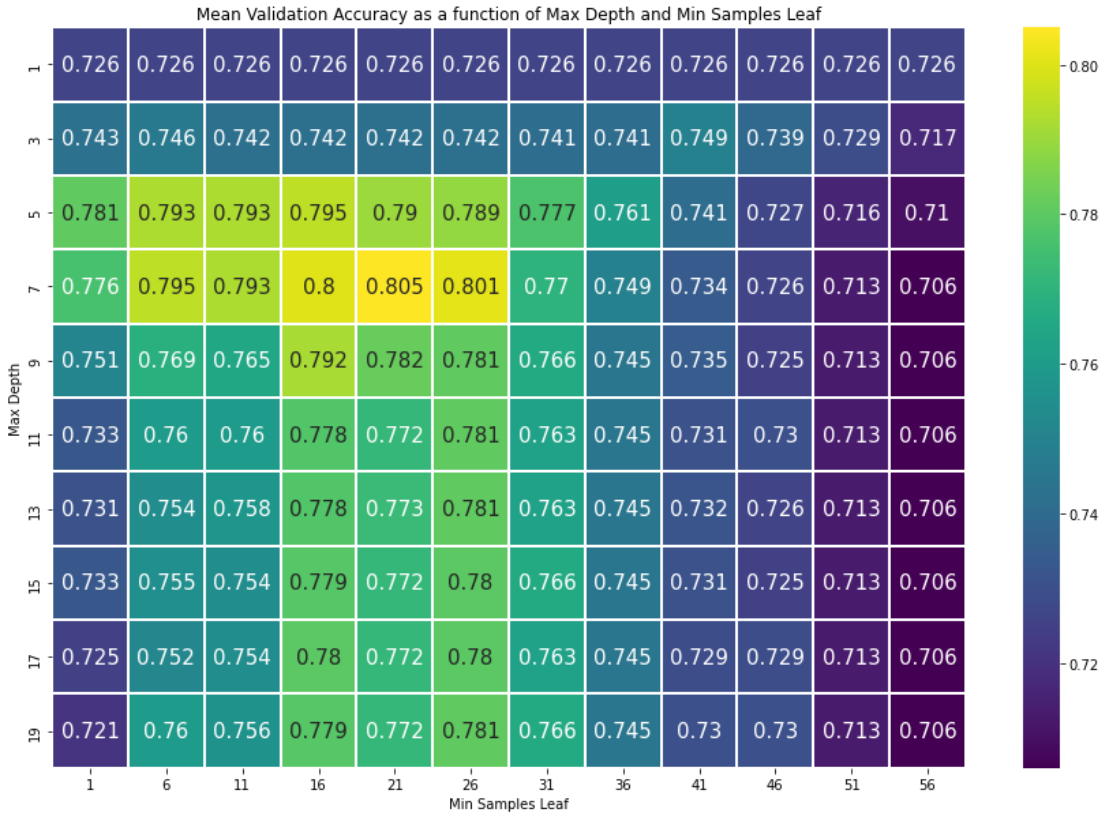
סעיף 7:



Decision Tree - Train Score: 0.787

סעיף 8:





Best parameters: {'max_depth': 7, 'min_samples_leaf': 21}

Best score: 0.805

אם נתבונן ב-heat maps שיצרנו נוכל למצוא קומבינציה שיוצרת underfitting, לדוגמה הקומבינציה הבאה: max depth = 1, min samples leaf = 11. עבור קומבינציה זו אנו רואים שגם על סט האימון וגם על ה-validation set, אנו מקבלים תוצאות יחסית נמוכות בהשוואה לשאר התוצאות שאנחנו מקבלים עבור פרמטרים אחרים. כפי שלמדנו בניתוח התיאורטי בהרצאה, אכן קיבלנו שגם על סט האימון וגם על סט הוולידציה קיבלנו תוצאה נמוכה, בדיוק כפי שקורה במצב של underfitting. תוצאה זו גם הגיונית מאוד עבורנו מכיוון שקבענו את ערך הפרמטר max depth להיות 1, כלומר יש לנו עץ בעומק 1, ולכן הוא יכול לכלול בתוכו את המורכבות והסיבוכיות של המודל, מכיוון שהוא מתחשב בפרמטר יחיד בסך הכל. לכן המודל עם הפרמטרים הללו באמת לא מצליח להביא פרדיקציות טובות על הנתונים שלנו ונמצא במצב של underfitting.

כמו כן נוכל להסיק מה-heat maps קומבינציה בה יש overfitting, לדוגמה עבור הקומבינציה הבאה: max depth = 19, min samples leaf = 1. עבור קומבינציה זו אנו רואים שהדיוק על סט האימון שלנו הוא כמעט מושלם – 0.9972! אבל על סט הוולידציה אנחנו מקבלים דיוק יחסית נמוך בהשוואה לדיוקים אחרים שהצלחנו להגיע עליהם על סט הוולידציה. הסיבה לכך היא שעבור קומבינציה זו של פרמטרים אנחנו נמצאים במצב של overfitting. כלומר, הגדלנו את סיבוכיות המודל שלנו יותר מדי, ובכך גרמנו לדיוק על סט האימון לעלות יותר, אבל על סט הוולידציה הדיוק נפגע. מצב זה תואם את הניתוח התיאורטי שראינו בכיתה שבמצב של overfitting הדיוק על סט האימון רק ממשיך לעלות, אבל על סט הוולידציה הדיוק מתדרדר. בנוסף, זה גם הגיוני לנו שעבור קומבינציה זו אכן נקבל מצב של overfitting כי בעצם הבאנו

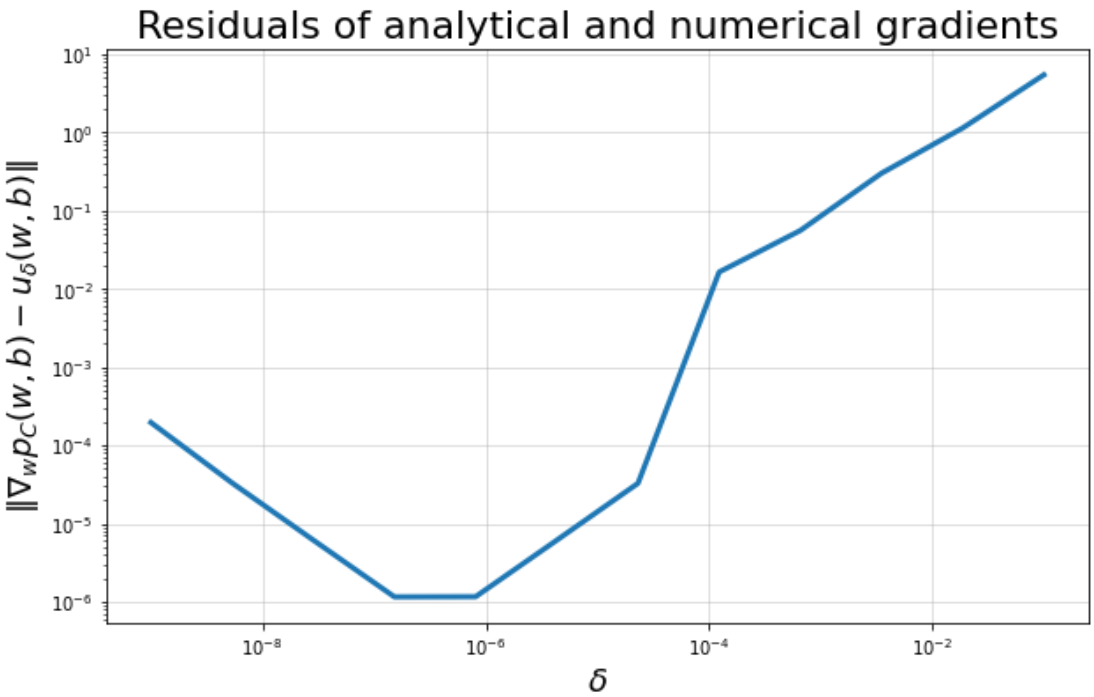
למודל אפשרות לבנות עץ בעל עומק די גדול (עומק 19), ובנוסף, כמות הדגימות מהנתונים שלנו שצריכה להיות בכל עלה של העץ יכולה להיות אפילו דגימה אחת. כלומר, במצב זה עץ ההחלטה ינסה ממש לתפור את כל הנקודות אחת לאחת על מנת להגיע לדיוק מושלם. וזה אכן מה שקרה, אבל הבעיה שזה יעזור רק על סט האימון, אבל על סט הוולידציה זה לא יועיל, ולכן הגענו למצב של overfitting.

סעיף 9:

Decision Tree - Train Score: 0.836

Decision Tree - Test Score: 0.84

סעיף 10:



נשים לב שככל שδ קטנה, הקירוב שלנו נהיה יותר טוב עד לרמה מסוימת (10^{-6}-10^{-7}) ושם ההפרש שוב מתחיל לגדול. נסביר למה עבור δ קטנים הקירוב טוב, ונסביר למה עבור ערכי δ קטנים מדי ההפרש מתחיל שוב לגדול.

לפי הגדרת הגבול, ככל שδ קטנה יותר, ארגומנט הגבול אמור לשאוף לגרדיאנט בנקודה, כי בעצם אנחנו פועלים לפי הגדרת הגבול ושאיפה ל-0, ולכן כצפוי ככל שδ קטנה אנחנו מתקרבים להגדרת הנגזרת ושואפים באמת לגרדיאנט ולכן הקירוב שלנו טוב.

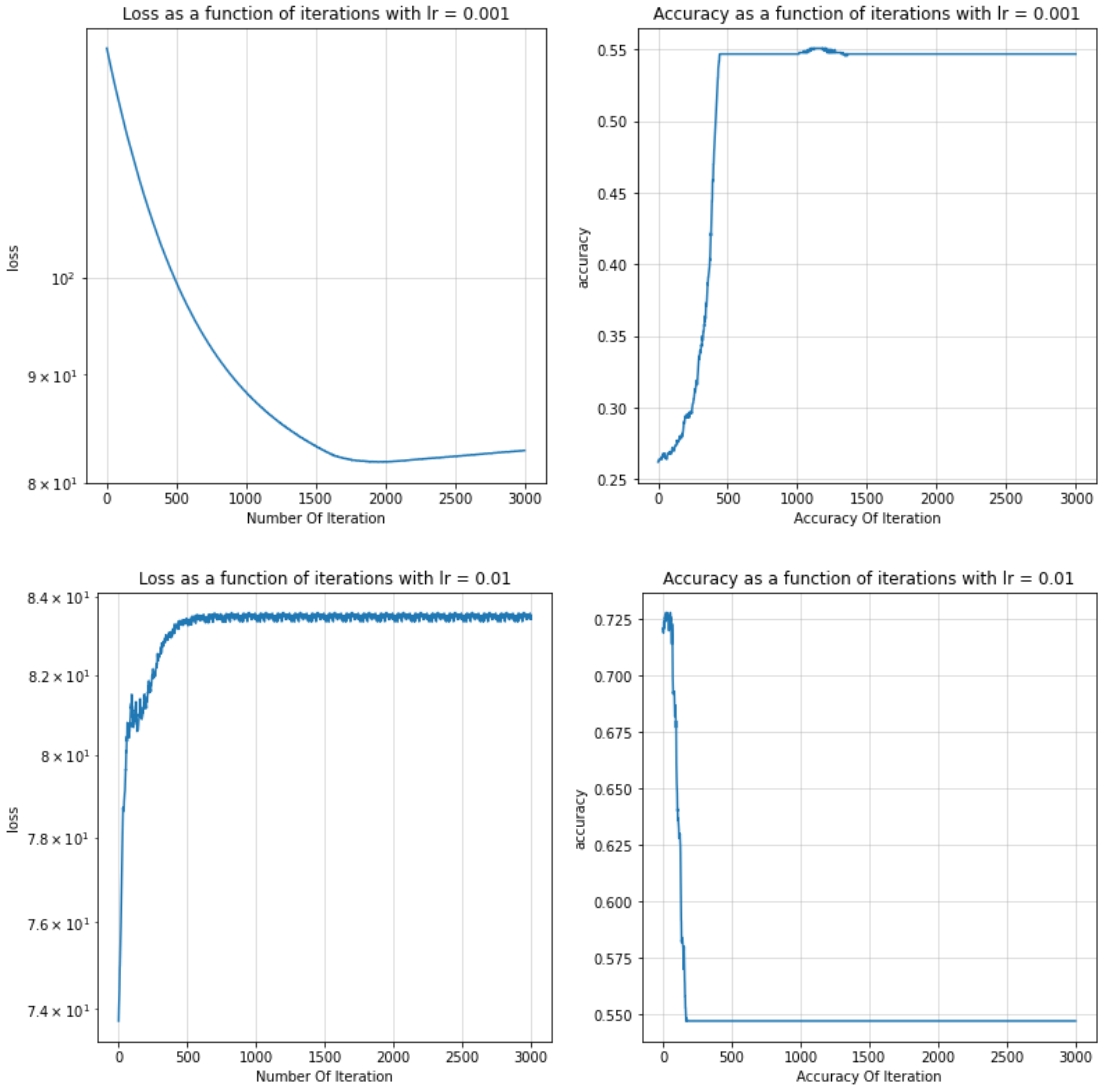
עם זאת, עבור ערכי δ קטנים מדי אנו מבצעים חלוקה במספרים מאוד קטנים, ולכן מקבלים "התפוצצות", מכיוון שחילוק בערכים קטנים כל כך זה כמו הכפלה במספר גדול מאוד ולכן זה

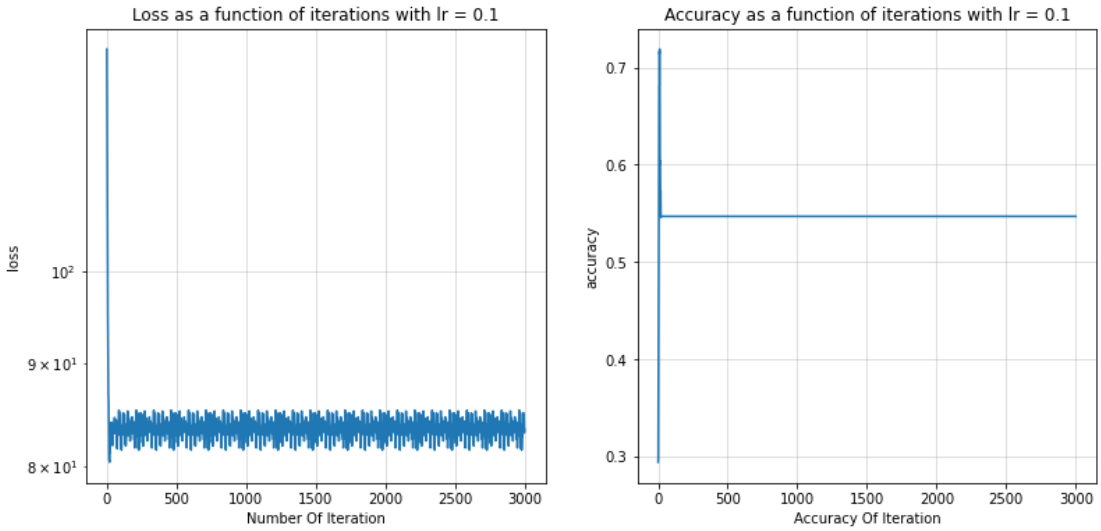
גורם לשגיאות ולאי-דיוקים נומריים, ולכן למרות שבעצם היינו צריכים לקבל קירוב טוב יותר, אנו מקבלים במקום זאת שהקירוב שלנו נפגע וההפרש גדל.

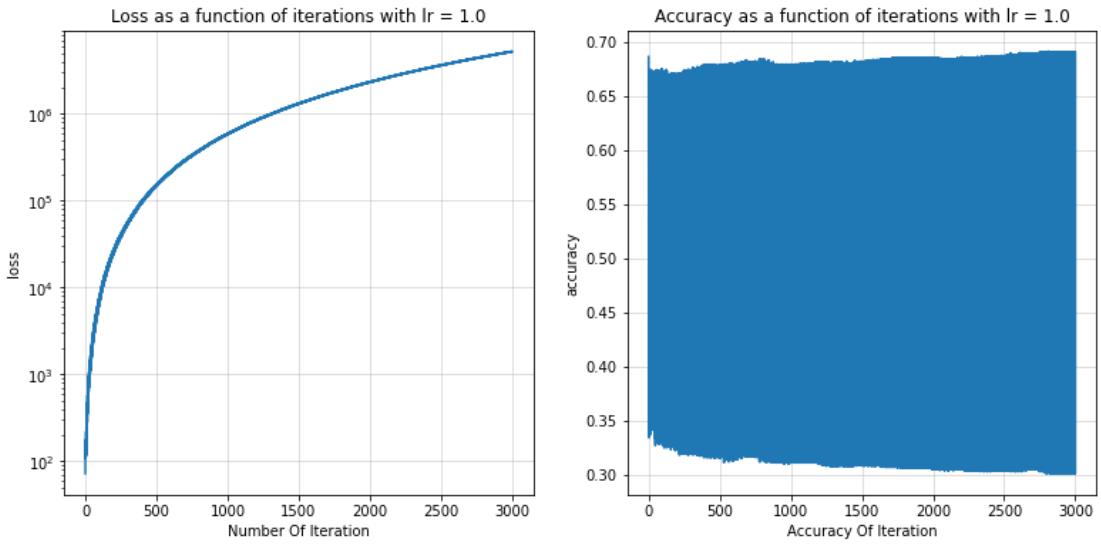
בנוסף, כאשר ערכי δ גדולים יחסית, אז שוב אנחנו מקבלים שהשגיאה גדולה, הסיבה לכך היא שבמצב זה אנחנו לא מתקרבים להגדרת הגבול, אלא לוקחים קירוב גס שלה. כלומר, אנחנו מחלקים במספרים גדולים יחסית, ולכן לא מקיימים את עניין ההשאפה ל-0 של δ. לכן, גם במצב זה אנחנו מקבלים שהשגיאה גדולה.

סעיף 11:

א.

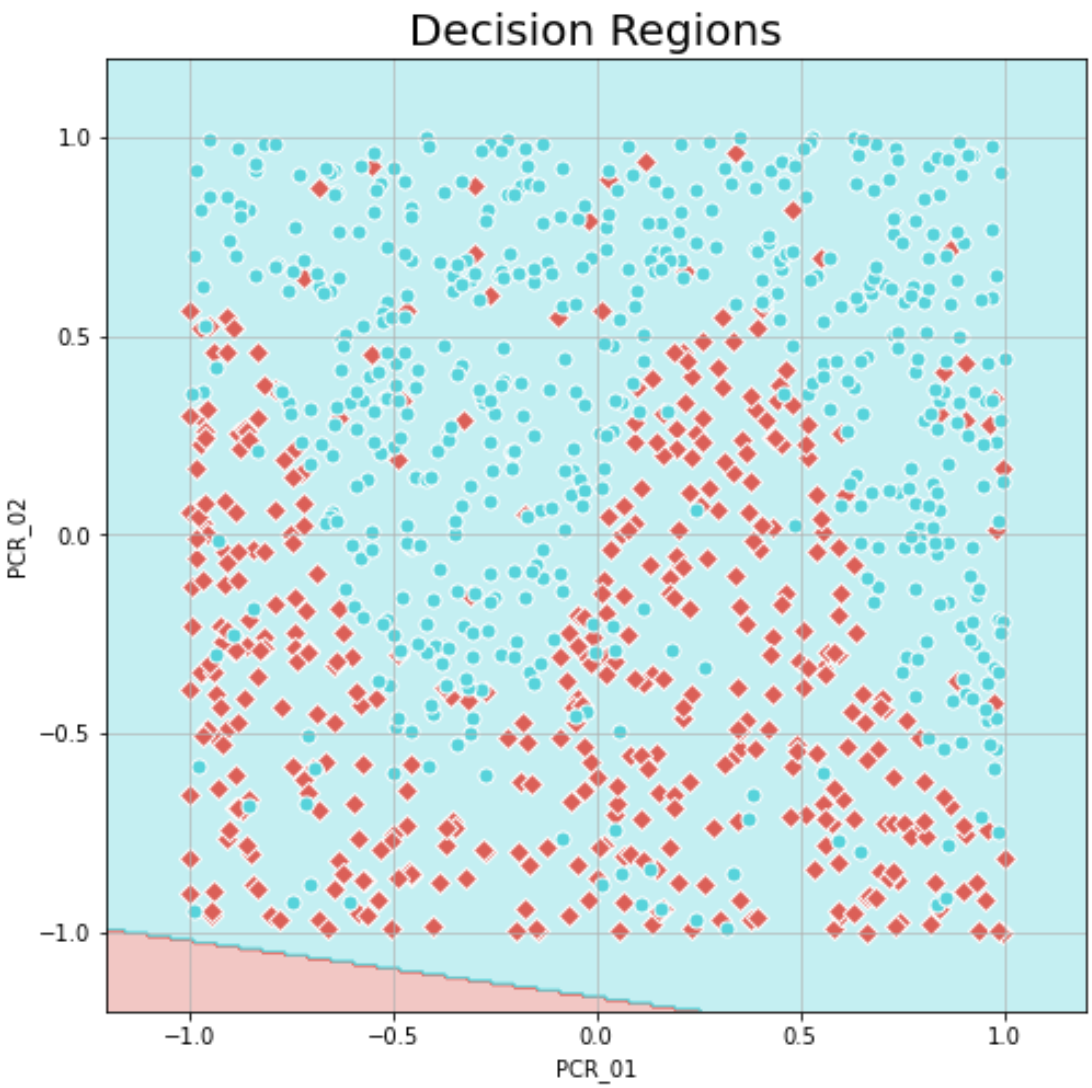






אפשר לראות שב lr = 0.1 ההתכנסות לערך 0.56 מתרחשת כמעט באיטרציות הראשונות לכן נבחר בו. בשונה מכל השאר שמתבדרים או מתכנסים לערך הזה בקצב יותר איטי.

ב.



ג.

max accuracy and min loss for **lr** = 0.1

max accuracy: 0.719

max accuracy iteration: 11

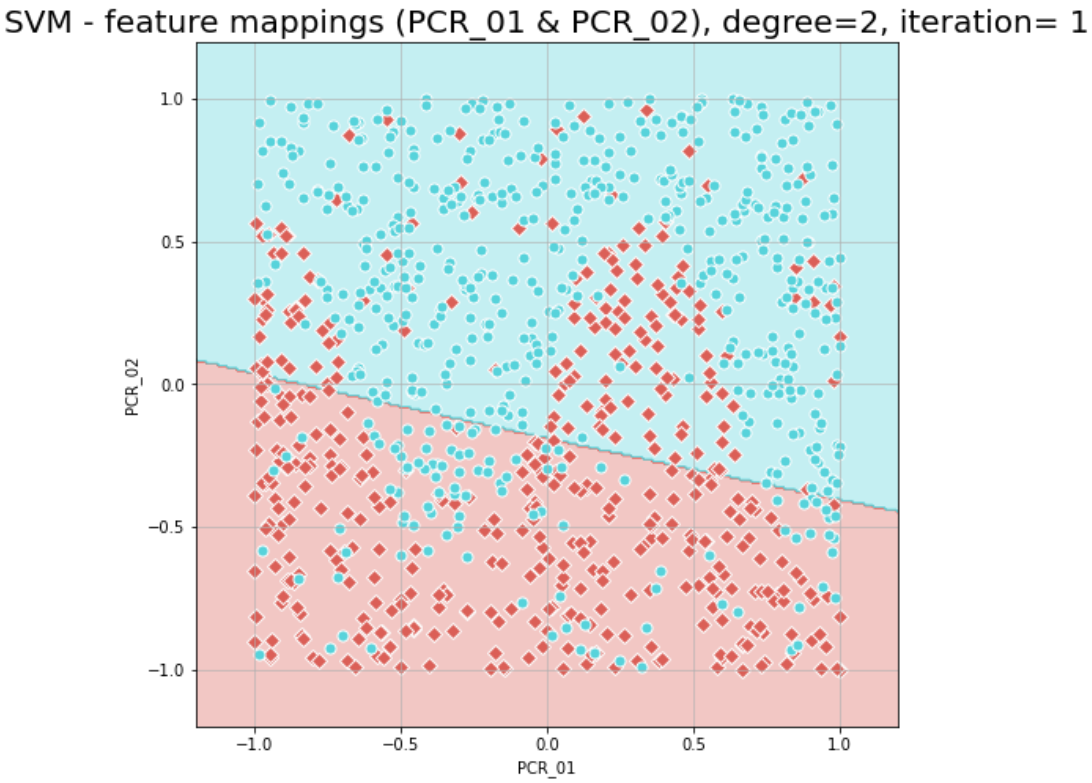
min loss: 80.375

min loss iteration: 19

קיבלנו שיש שוני בין האיטרציות מבחינת הדיוק והאובדן.

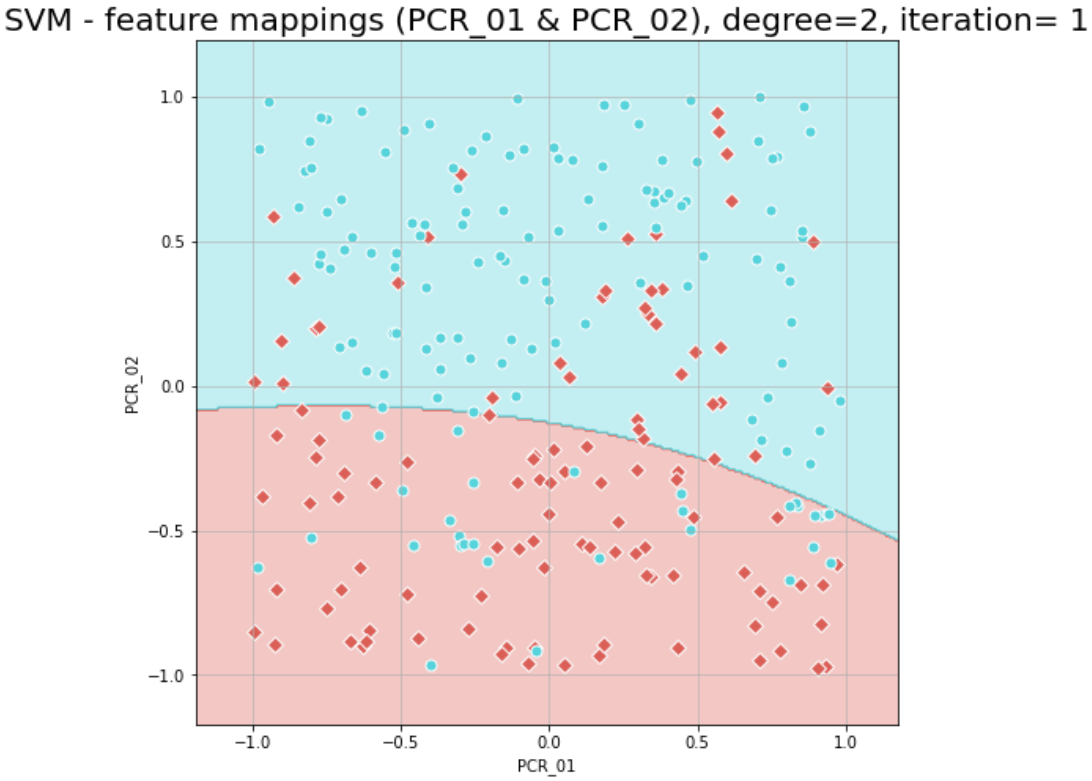
השוני בין האיטרציות נובע מכך שהפעולה שבודקת את רמת הדיוק והפעולה שבודקת את האובדן פועלים בצורה נפרדת אחת מהשנייה ולכן סדר הדגימות שונה אצל כל אחת מהן.

סעיף 12:



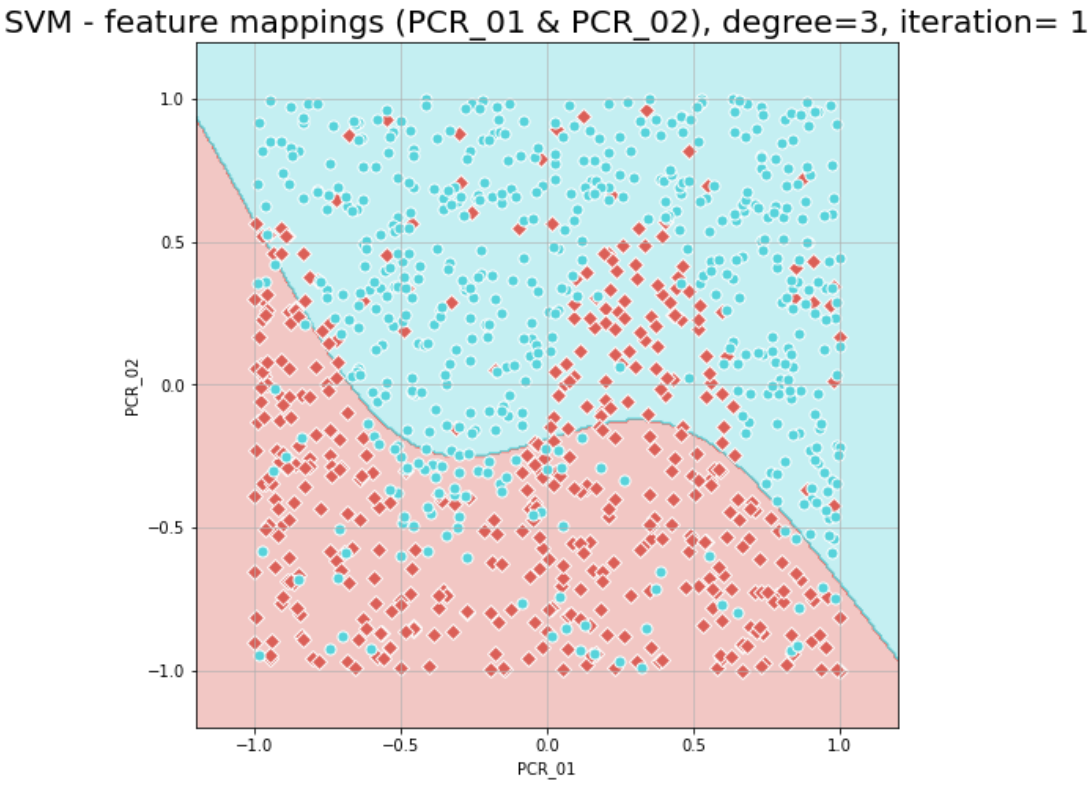
Accuracies = 0.726

Test:



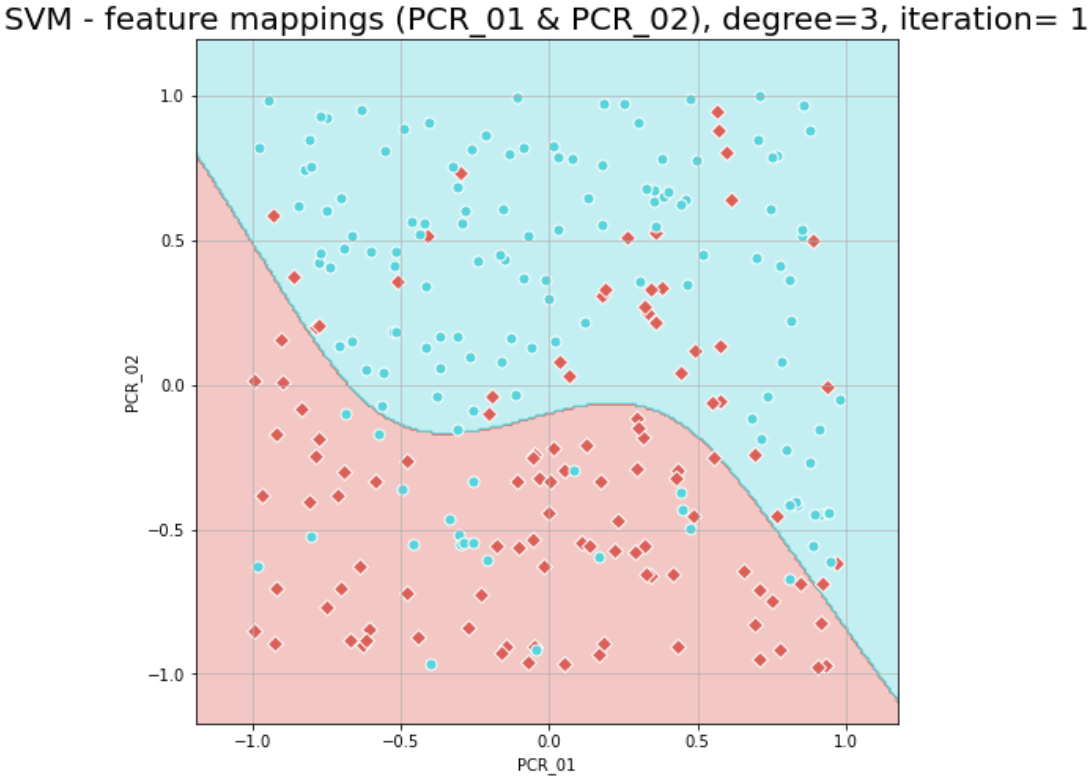
Accuracies = 0.716

train :



Accuracies = 0.779

test :



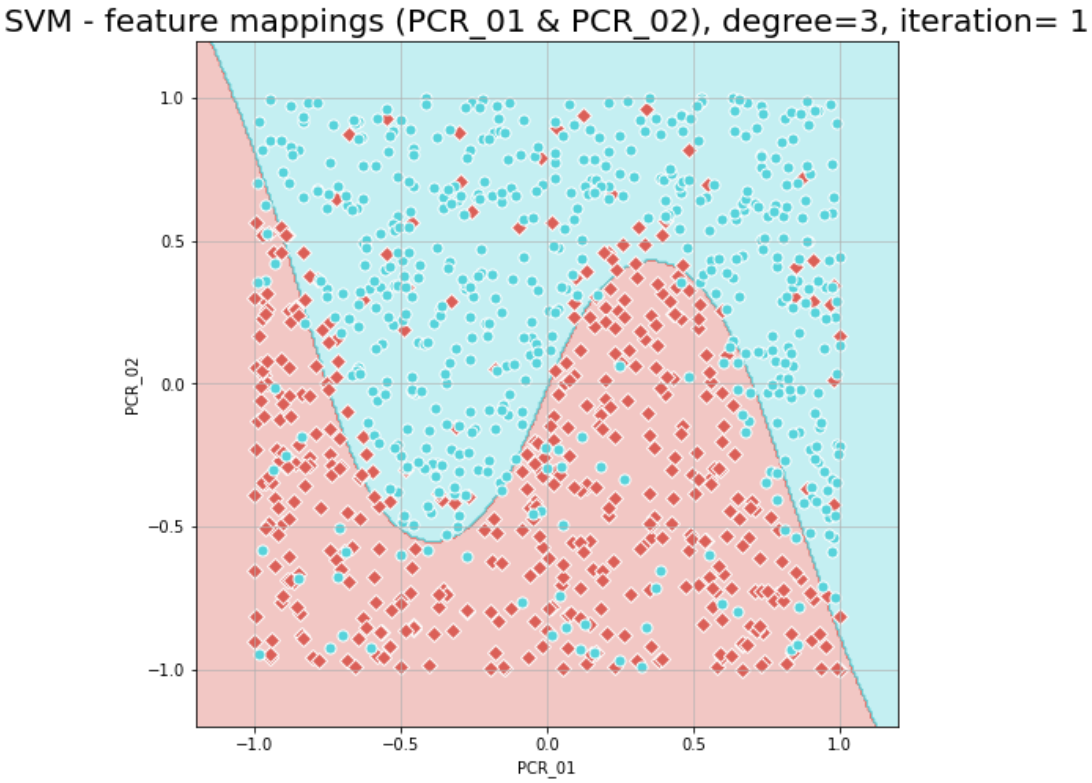
Accuracies = 0.776

אפשר לראות שכאשר הדרגה 2 אנחנו מקבלים סוג של פרבולה וכאשר הדרגה 3 אנחנו נקבל מפריד שדומה לפולינום מדרגה 3.

זאת אומרת, ככל שהדרגת הפולינום גדולה יותר אזי רמת הדיוק משתפרת בהתאם, הן בסט אימון והן בסט של המבחן, מכיוון שיש יותר גמישות ל SVM ויותר יכולת להפריד בין שני התחומים של הערכים השונים.

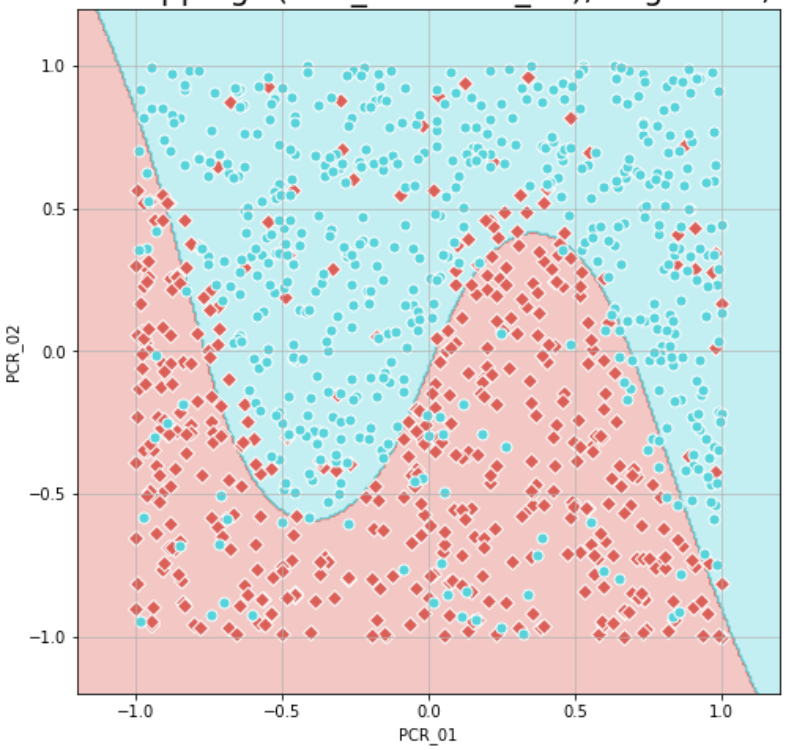
אפשר לראות זאת באמת ברמת דיוק של כל דרגה ונראה שבדרגה ה 3 רמת הדיוק גדולה יותר.

סעיף 13:



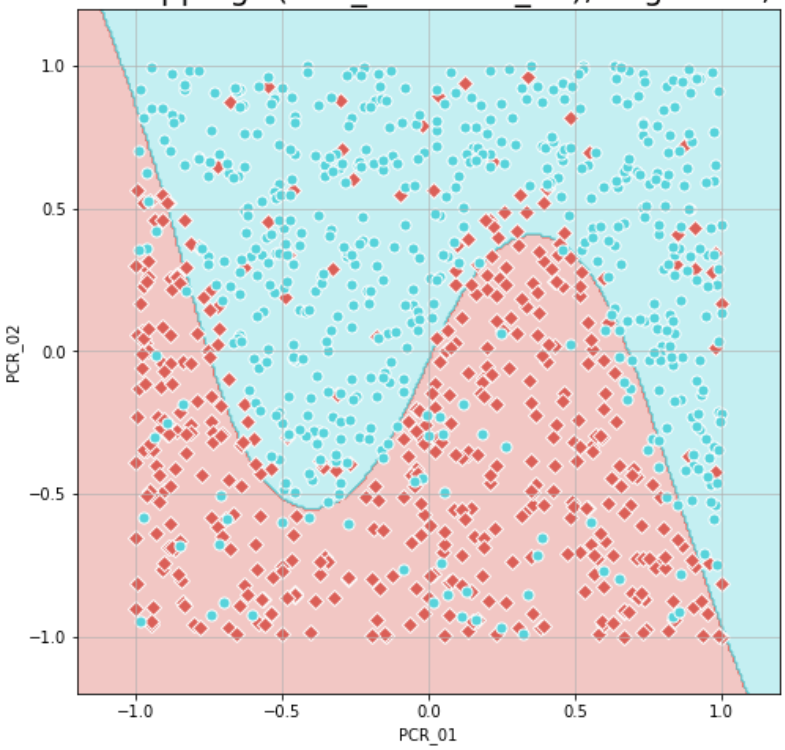

```
Accuracies = 0.852
```

SVM - feature mappings (PCR_01 & PCR_02), degree=3, iteration= 2



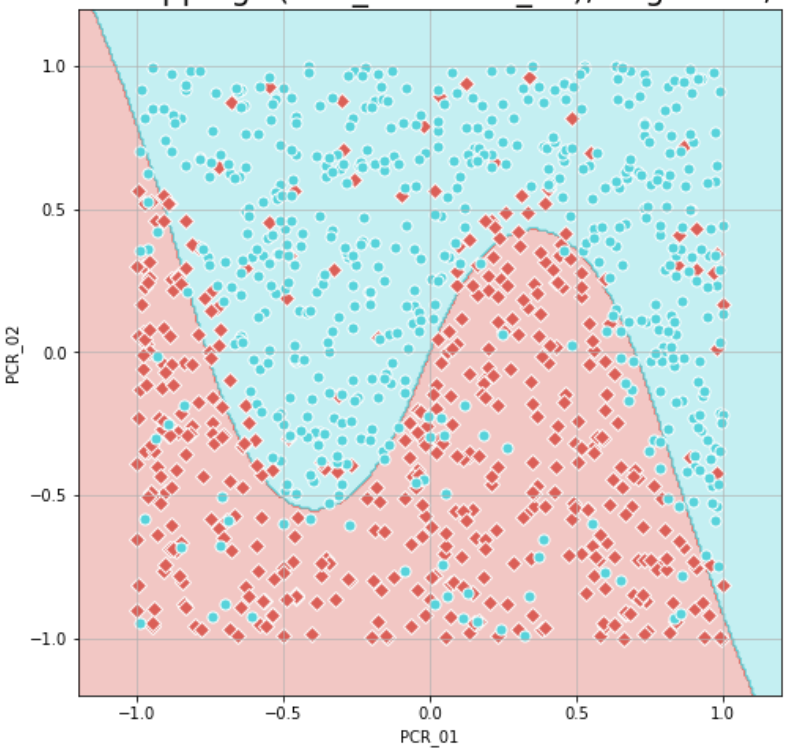
Accuracies = 0.849

SVM - feature mappings (PCR_01 & PCR_02), degree=3, iteration= 3



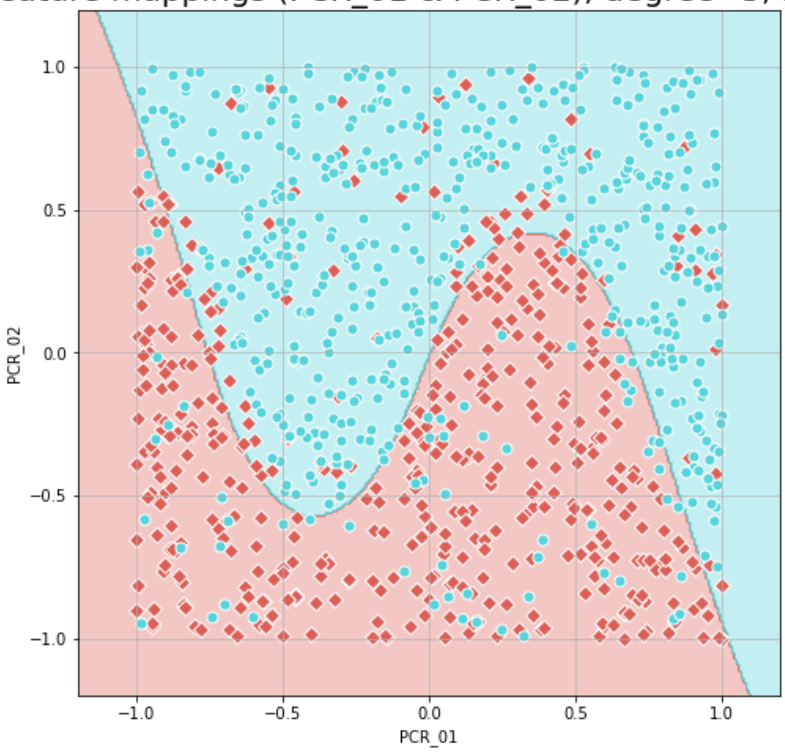
Accuracies = 0.851

SVM - feature mappings (PCR_01 & PCR_02), degree=3, iteration= 4



Accuracies = 0.849

SVM - feature mappings (PCR_01 & PCR_02), degree=3, iteration= 5



Accuracies = 0.851

Mean accuracy = 0.85
Standard deviation = 0.001

שונות הגרפים נובעת מהעובדה שהאלגוריתם שהשתמשנו בו הוא סטוכסטי. מכיוון שאנו משתמשים בSGD, אנו מגרילים מחדש מדגם **אקראי** (בגודל של 32 דגימות – ככה בחרנו) ובאמצעותו אנו מחשבים את הגרדיאנט. לכן, זה הגיוני שנקבל תוצאות מעט שונות בכל פעם שנפעיל את האלגוריתם וגם אחוזי דיוק קצת שונים.

נשים לב שבכל 5 הפעמים שהפעלנו את האלגוריתם קיבלנו תוצאות די זהות מבחינת האזור שנבחר להיות בעל תווית אדומה על ידי האלגוריתם ובעלות אחוז דיוק כמעט זהה. סביר להניח שזה נובע מכך שאין כל מיני נקודות קיצוניות בנתונים שלנו או משהו אחר שיכול להסיט את הגרדיאנט במקרה שאותה נקודה תיבחר, ולכן בחירה של נקודות בשביל למצוא את כיוון הגרדיאנט היא די דומה פחות א יותר עבור כל מדגם של 32 נקודות שאנו בוחרים, ולכן יוצא שבסוף אזורי ההחלטה בכל פעם די דומים וכמו כן אחוזי הדיוק.

סעיף 14:

נרצה להראות ש -
$\lim_{\gamma->\infty} sign(\sum_{i\in[m],a_i>0} a_i y_i e^{-\gamma||x-x_i||_2^2}) = y_{i^*}, i^* = argmin_{i\in[m]}||x-x_i||_2^2$
נפריד למקרים :

1 $y^* = 1$ **:**

במקרה זה נרצה להראות ש :

$$\lim_{\gamma->\infty} sign(\sum_{i\in[m],a_i>0} a_i y_i e^{-\gamma||x-x_i||_2^2}) = 1$$

כלומר :

$$\lim_{\gamma->\infty} \sum_{i\in[m],a_i>0} a_i y_i e^{-\gamma||x-x_i||_2^2} > 0$$

ואכן :

$$\lim_{\gamma->\infty} \sum_{i\in[m],a_i>0} a_i y_i e^{-\gamma||x-x_i||_2^2} \geq^{\forall i: ai \geq a_i^{min}} \lim_{\gamma->\infty} a_i^{min} \sum_{i\in[m],a_i>0} y_i e^{-\gamma||x-x_i||_2^2} =$$

$$\lim_{\gamma->\infty} a_i^{min}(e^{-\gamma||x-x_{i^*}||_2^2} + \sum_{i\in[m],a_i>0,i\neq i^*} y_i e^{-\gamma||x-x_i||_2^2}) >^{\forall i: y_i=-1}$$

$$\lim_{\gamma->\infty} a_i^{min}(e^{-\gamma||x-x_{i^*}||_2^2} - \sum_{i\in[m],a_i>0,i\neq i^*} e^{-\gamma||x-x_i||_2^2}) >^{\{*\}} 0$$

2 $y^* = -1$:

במקרה זה נרצה להראות ש :

$$\lim_{\gamma->\infty} sign(\sum_{i\in[m],a_i>0} a_i y_i e^{-\gamma||x-x_i||_2^2}) = -1$$

כלומר :

$$\lim_{\gamma->\infty} \sum_{i\in[m],a_i>0} a_i y_i e^{-\gamma||x-x_i||_2^2} < 0$$

ואכן :

$$\lim_{\gamma->\infty} \sum_{i\in[m],a_i>0} a_i y_i e^{-\gamma||x-x_i||_2^2} \leq^{\forall i: ai \leq a_i^{max}} \lim_{\gamma->\infty} a_i^{max} \sum_{i\in[m],a_i>0} y_i e^{-\gamma||x-x_i||_2^2} =$$

$$\lim_{\gamma->\infty} a_i^{max}(-e^{-\gamma||x-x_{i^*}||_2^2} + \sum_{i\in[m],a_i>0,i\neq i^*} y_i e^{-\gamma||x-x_i||_2^2}) <^{\forall i: y_i=1}$$

$$\lim_{\gamma->\infty} a_i^{max}(-e^{-\gamma||x-x_{i^*}||_2^2} + \sum_{i\in[m],a_i>0,i\neq i^*} e^{-\gamma||x-x_i||_2^2}) =$$

$$\lim_{\gamma->\infty} -a_i^{max}(e^{-\gamma||x-x_{i^*}||_2^2} - \sum_{i\in[m],a_i>0,i\neq i^*} e^{-\gamma||x-x_i||_2^2}) <^{\{*\}} 0$$

$\{*\}$: נשים לב בשבני המקרים a_i^{max}ו־ a_i^{min} לא משפיעים על הסימן מהיותם חיוביים ולכן כל מה שנותר להראות הוא ש־

$$e^{-\gamma||x-x_{i^*}||_2^2} - \sum_{i\in[m],a_i>0,i\neq i^*} e^{-\gamma||x-x_i||_2^2} > 0$$

לצורך הנוחות נסמן : $d_i = ||x-x_i||_2^2$ וב־d_i $i^{**} = argmin_{i\in[m],i\neq i^*} d_i$.

$$\sum_{i\in[m],a_i>0,i\neq i^*} e^{-\gamma d_i} <^{\forall x<y:e^{-y}<e^{-x}} \sum_{i\in[m],a_i>0,i\neq i^*} e^{-\gamma d_{i^{**}}} = (m-1)e^{-\gamma d_i^{**}} <^{demand} e^{-\gamma}$$

מתקיים ש־ $m \geq 1$ ו־ $e^x > 0$ לכן : $(m-1)e^{-\gamma d_i^{**}} \geq 0$ אז ההפעלת לוג על שני האגפים הסימן נשמר .

$$log(m-1) < -\gamma(d_{i^*} - d_{i^{**}}) = \gamma(d_{i^{**}} - d_{i^*}) \longrightarrow \gamma > \frac{log(m-1)}{d_{i^{**}} - d_{i^*}}$$

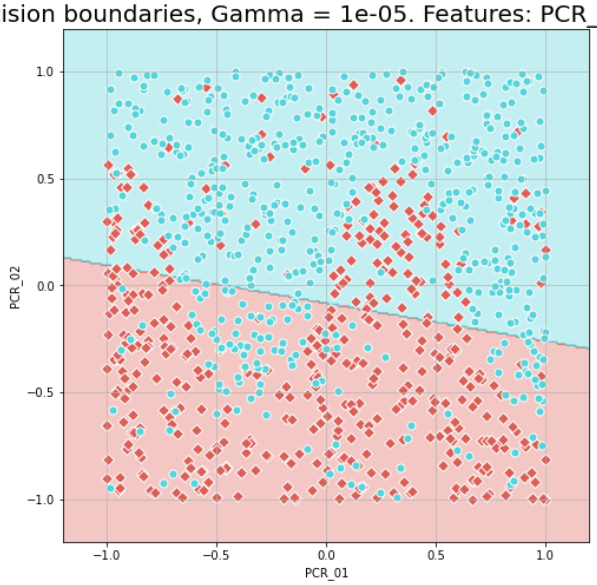
*הסימן נשמר כי המונה והמכנה חיוביים ־ m מספר שלם ו־מהגדרתם $d_{i^{**}} > d_{i^*}$.
לכן בגבול $\gamma \longrightarrow \infty$ הפרמטר γ מקיים את התנאי וכתוצאה מכך

$$\sum_{i\in[m],a_i>0,i\neq i^*} e^{-\gamma d_i} < e^{-\gamma d_{i^*}} \longrightarrow e^{-\gamma d_{i^*}} - \sum_{i\in[m],a_i>0,i\neq i^*} e^{-\gamma d_i} > 0$$

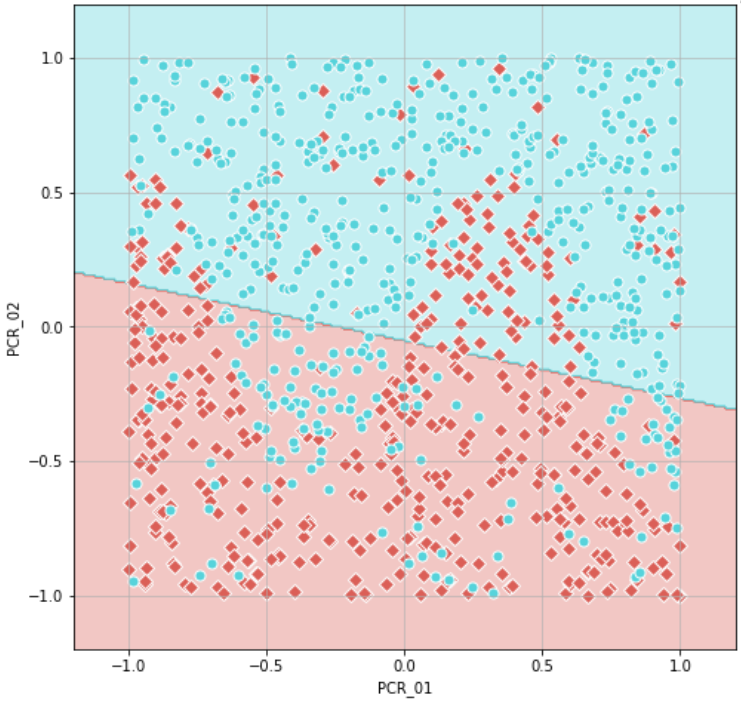
ולכן טענה $\{*\}$ מתקיימת.

סעיף 15:

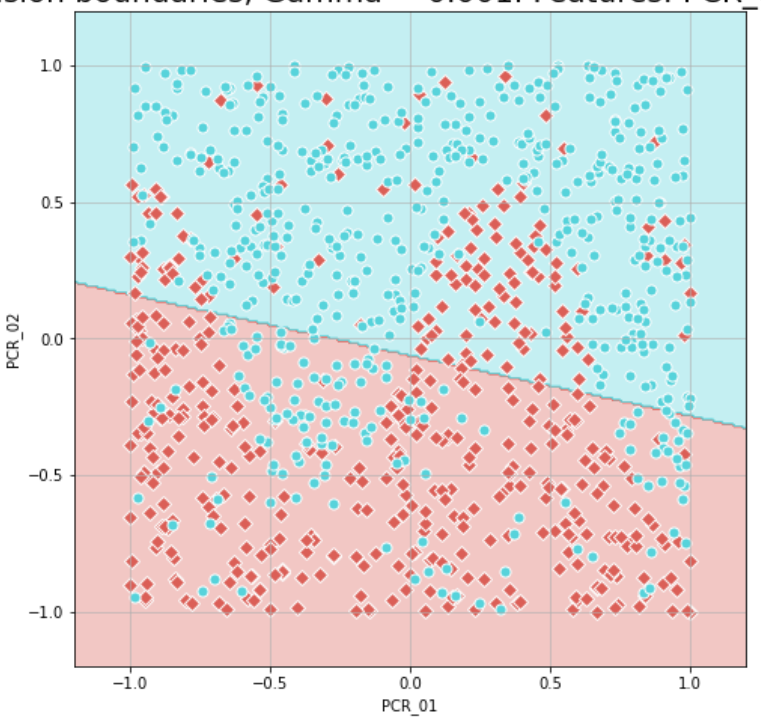
SVC Decision boundaries, Gamma = 1e-05. Features: PCR_01 & PCR_02



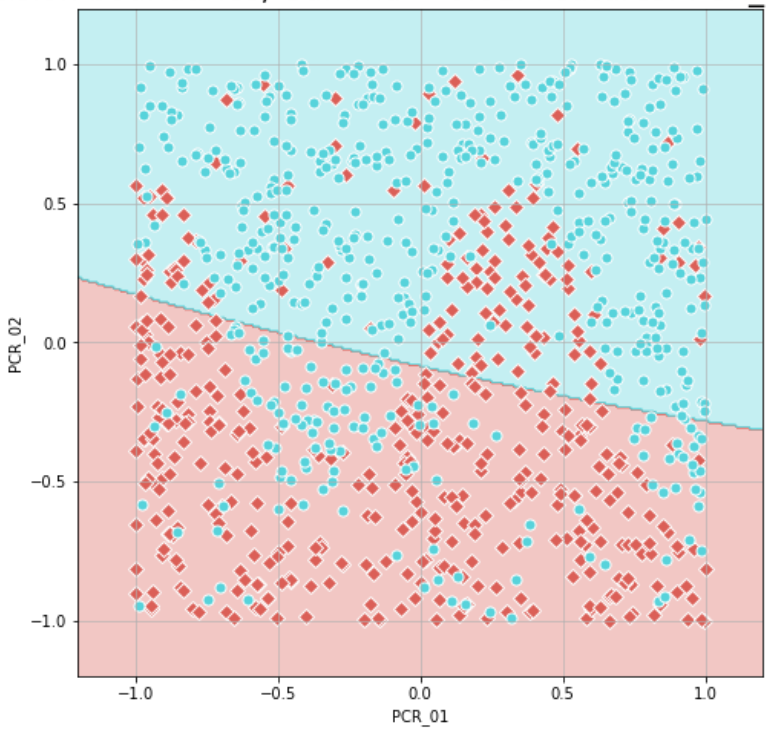
SVC Decision boundaries, Gamma = 0.0001. Features: PCR_01 & PCR_02



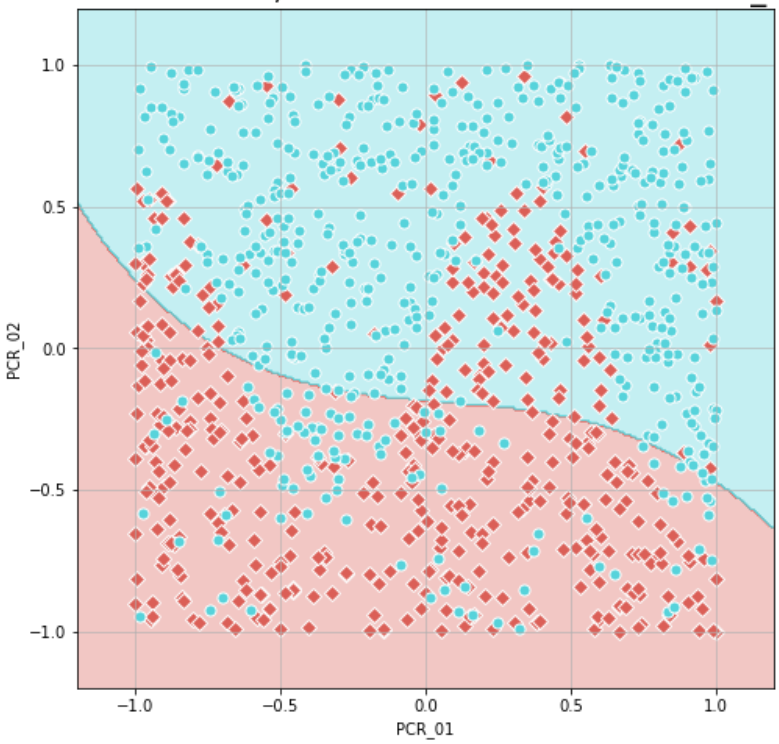
SVC Decision boundaries, Gamma = 0.001. Features: PCR_01 & PCR_02



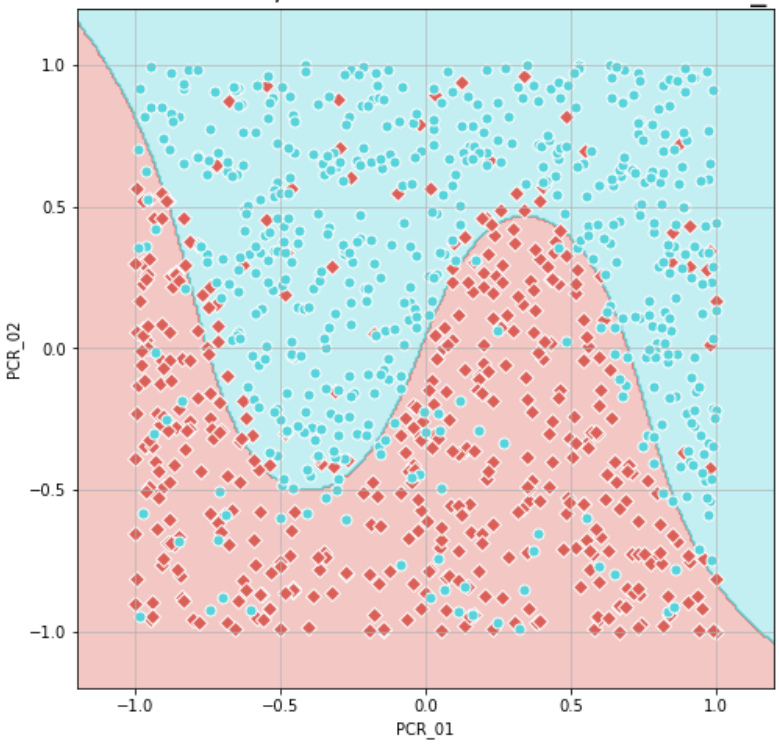
SVC Decision boundaries, Gamma = 0.01. Features: PCR_01 & PCR_02



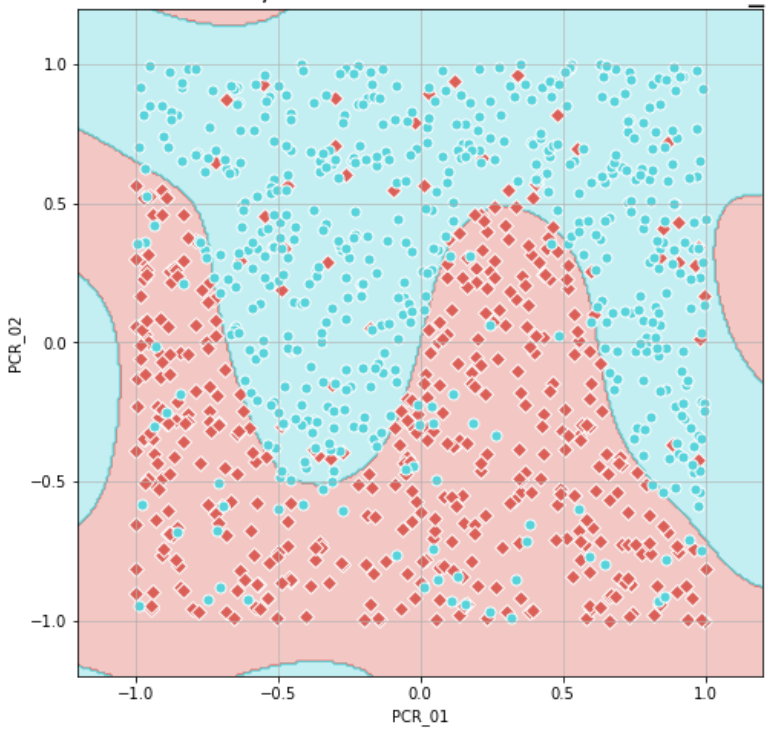
SVC Decision boundaries, Gamma = 0.1. Features: PCR_01 & PCR_02



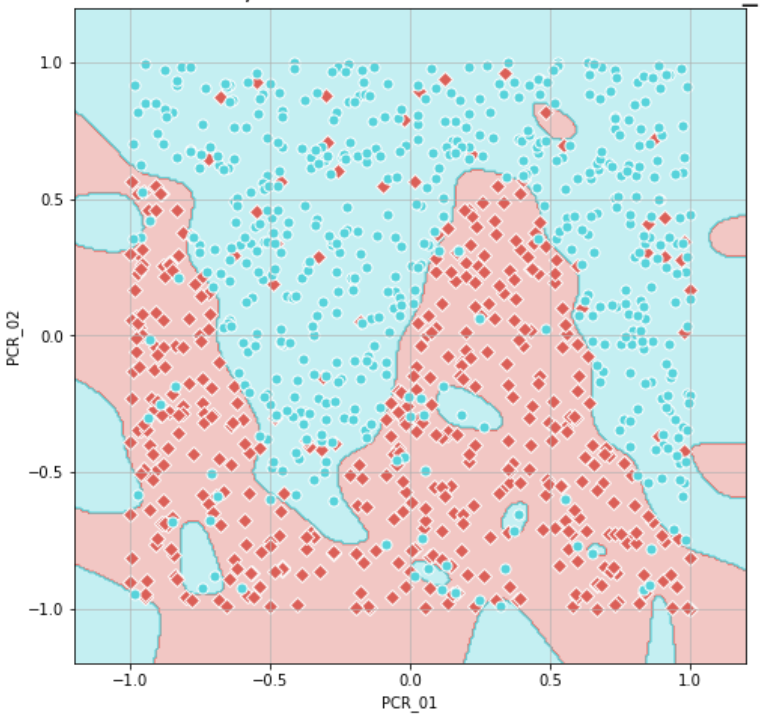
SVC Decision boundaries, Gamma = 1.0. Features: PCR_01 & PCR_02



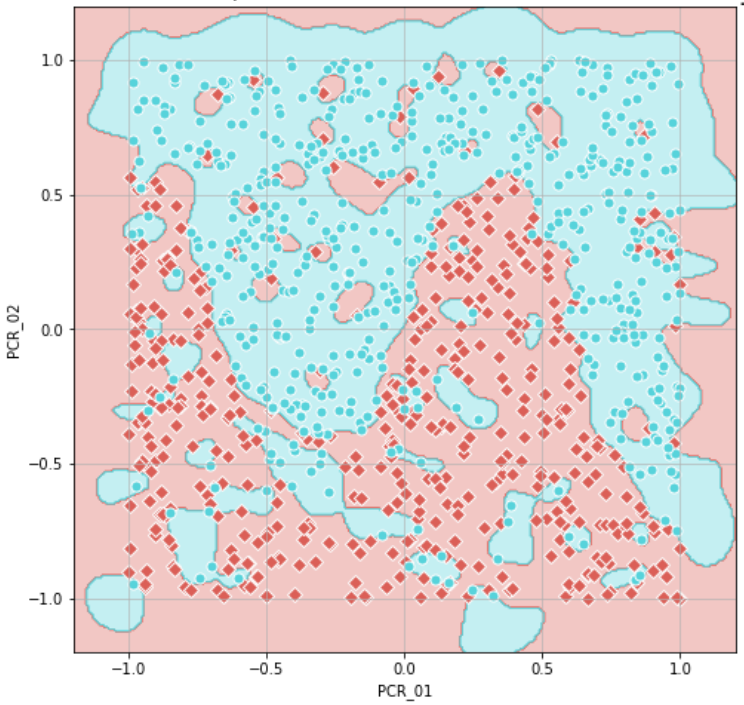
SVC Decision boundaries, Gamma = 10.0. Features: PCR_01 & PCR_02

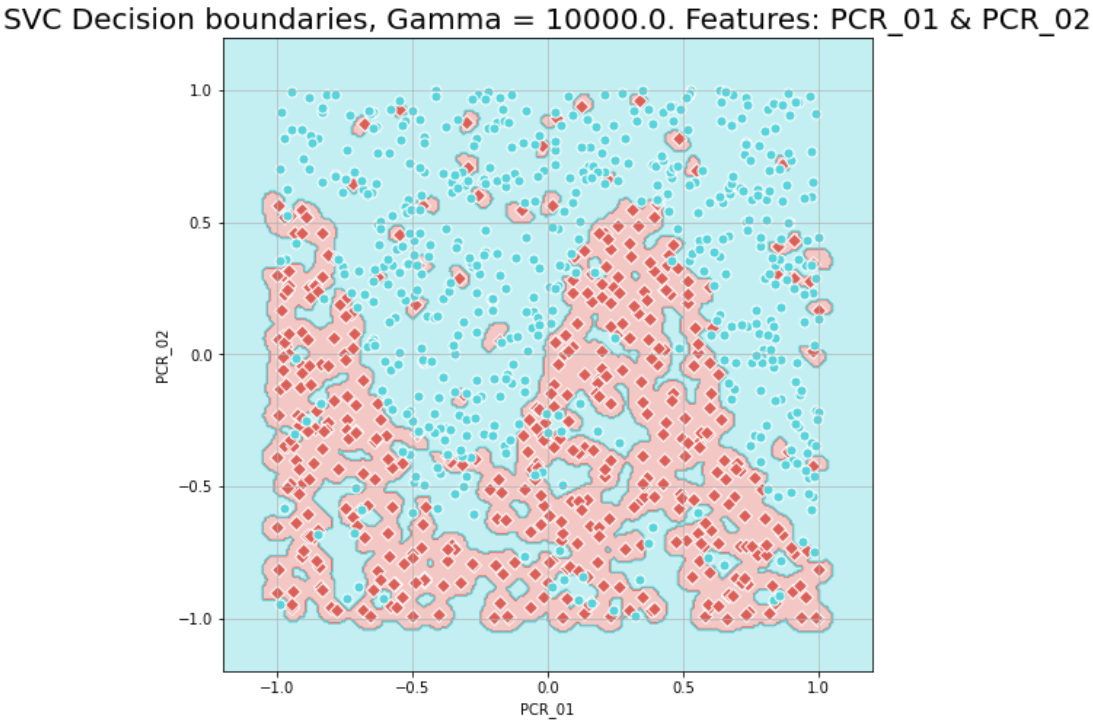


SVC Decision boundaries, Gamma = 100.0. Features: PCR_01 & PCR_02

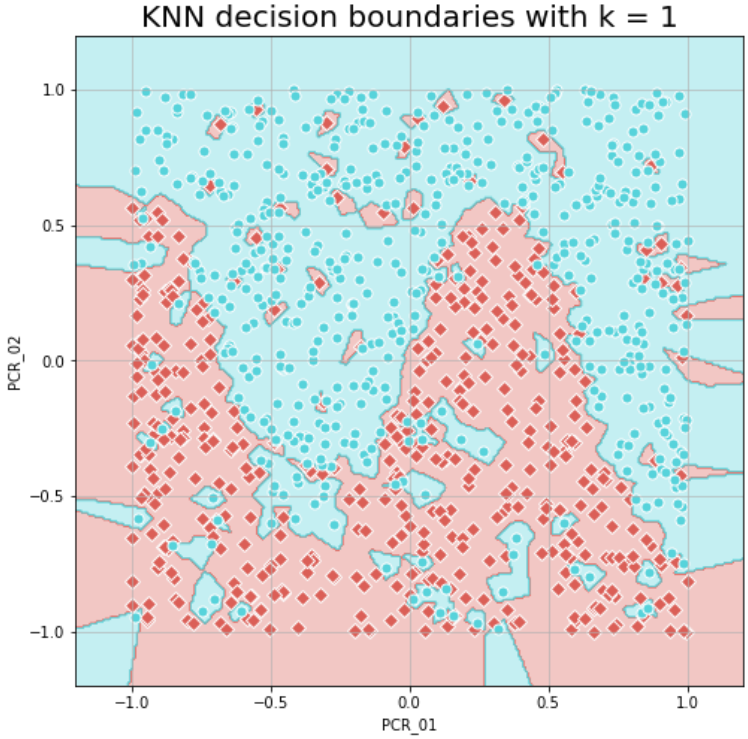


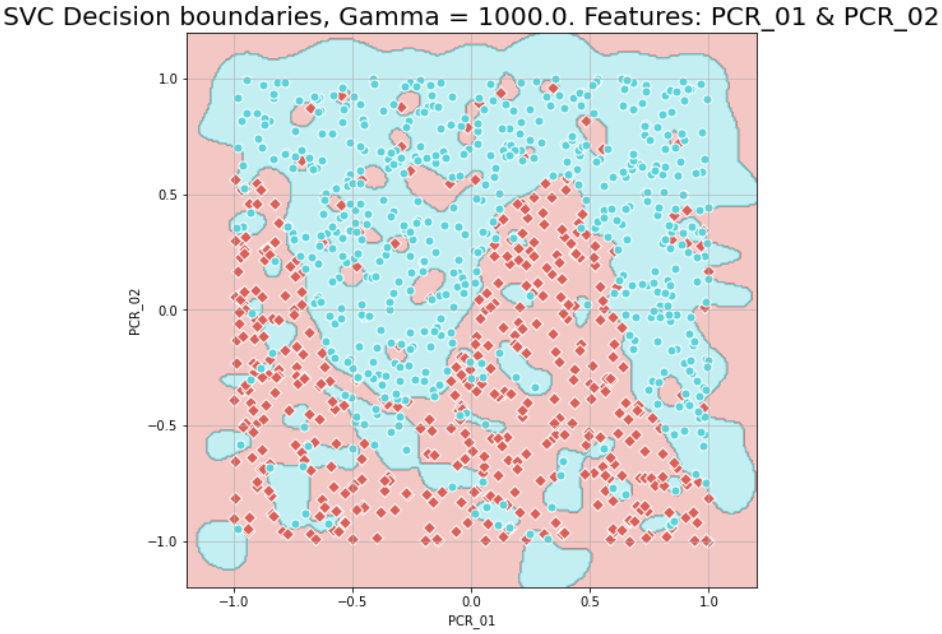
SVC Decision boundaries, Gamma = 1000.0. Features: PCR_01 & PCR_02




SVC Decision boundaries, Gamma = 10000.0. Features: PCR_01 & PCR_02
PCR_02
PCR_01
1.0
0.5
0.0
−0.5
−1.0

סעיף 16:

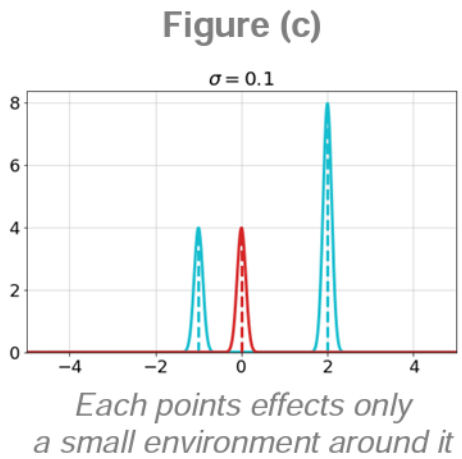

KNN decision boundaries with k = 1
PCR_02
PCR_01
1.0
0.5
0.0
−0.5
−1.0



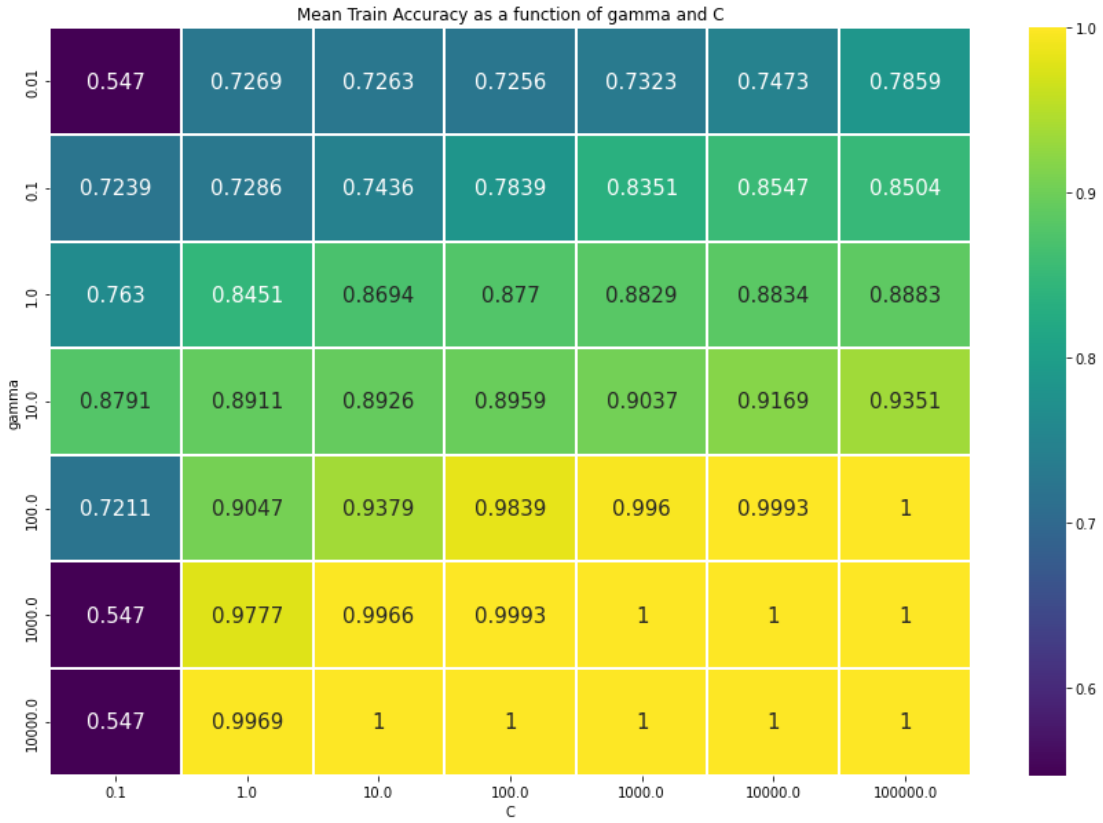
המודלים יצאו אכן יחסית דומים במובן שאזורי ההחלטה שהם מגדירים מתאימים לכל מצבור של נקודות תווית מתאימה (אם מסתכלים היטב רואים שבגרף של ה-RBF kernel גם יש אזורים צבועים באדום מסביב למקבצי הנקודות האדומות, אך האזור הזה צמוד מאוד לנקודות). אבל כפי שניתן לשים לב, עדיין יש הבדלים משמעותיים מאוד בין מודל ה-KNN עם שכן אחד, שבו אנו מסתכלים רק על הנקודה הקרובה ביותר, ואין התייחסות לשאר הנקודות. לעומת זאת, עבור RBF יש התייחסות לכל הנקודות, כאשר לנקודות יותר קרובות יש יותר "משקל". לכן, במצב בו יש מספר נקודות בעלות קרבה יחסית דומה (וזה מצב שיכול להיות נפוץ אצלנו מכיוון שה-data שלנו מורכב בעיקר מכמה גושים מרכזיים בהם יש נקודות צפופות), עבור KNN אנו נסתכל רק על שכן אחד קרוב ביותר ונתעלם מהשכנים האחרים למרות שהם קרובים מאוד במרחק, ולעומת זאת ב-RBF כן נתחשב בנקודות אחרות.

הסבר נוסף שניתן להביא הוא: השתמשנו ב-gamma גדול מאוד, ולכן על פי הנוסחה של ה-RBF kernel, נקבל שההשפעה של כל נקודה על הפרדיקציה של נקודה חדשה היא קטנה מאוד, מכיוון שאנו מכפילים את המרחק במספר שלילי (יש מינוס לפני ה-gamma בנוסחה), ולכן האקספוננט שואף ל-0 במקרה זה. כלומר, ההשפעה שיש לכל נקודה בסט האימון שלנו על פרדיקציה של נקודה חדשה היא קטנה מאוד, אלא אם כן הנקודה החדשה קרובה מאוד לנקודה בסט האימון שלנו שאז אכן נקבל השפעה של הנקודה בסט האימון. כלומר, במצב כזה בעצם נקבל התנהגות שכל נקודה בסט האימון משפיעה רק על הסביבה הקרובה ביותר שלה, וזה בדיוק מה שקיבלנו בגרף.

זה בעצם התיאור הגרפי שמתאר את המצב בו אנו נמצאים כאשר אנו משתמשים בערך gamma גדול מאוד:



סעיף 17:




```
Best parameters: {'C': 10.0, 'gamma': 10.0}
Best score: 0.878
```

קודם כל נבחין כי הקומבינציה שיוצרת את הביצועים הטובים ביותר היא עבור C=10, gamma=10. עבור נתונים אלה הגענו ל-87.8% דיוק על ה-validation set.

כעת, נתבונן ב-heat maps שיצרנו ונוכל למצוא קומבינציה שיוצרת underfitting, לדוגמה הקומבינציה הבאה: C=0.1 ו-gamma=0.01. עבור קומבינציה זו אנו רואים שגם על סט האימון וגם על ה-validation set, אנו מקבלים תוצאות יחסית נמוכות בהשוואה לשאר התוצאות שאנחנו מקבלים עבור פרמטרים אחרים. כפי שלמדנו בניתוח התיאורטי בהרצאה, מכיוון שקיבלנו שם על סט האימון וגם על סט הוולידציה תוצאה נמוכה, אז אנחנו במצב של underfitting בו למודל שיצרנו אין סיבוכיות מספיק גדולה על מנת להכיל את המורכבות של סט הנתונים.

תוצאה זו גם די הגיונית מכיוון שקבענו את ערך הפרמטר gamma להיות 0.01. על פי הניתוח התיאורטי שראינו בהרצאה, וכאשר gamma מאוד קטן, אז המודל לא מצליח להבין את המורכבות של סט הנתונים. זה נובע מכך שבאופן אינטואיטיבי gamma קובע מהו טווח ההשפעה של דגימה אחת, ומכיוון ש-gamma קטן אז טווח ההשפעה של כל דגימה הוא די גדול – כלומר הוא בעצם מתנהג בסוג של יחס הפוך לגודל הרדיוסים שמרכיבים את הכדורים שנוצרים. לכן סך הכל, העובדה ש-gamma הוא קטן גורמת לכך שהמודל יתנהג בצורה די דומה למודל ליניארי (אבל קצת יותר מורכב מכיוון שבסוף אנחנו משתמשים ב-RBF kernel ולכן יש בעצם כמה היפר-מישורים שמגדירים את אזורי ההחלטה. אבל באופן כללי יש למודל צורה ליניארית פחות או יותר). לכן במצב זה, המודל עם הפרמטרים הללו באמת לא מצליח להביא פרדיקציות טובות על הנתונים שלנו כי הוא פשטני מדי ונמצא במצב של underfitting. הפרמטר C גם הוא משפיע במצב הזה על דיוק המודל מכיוון ש-C בעצם קובע את סיבוכיות

מחלקת המודלים שלנו, ולכן הוא בעצם קובע כמה נעניש על כל שגיאה. מכיוון שבחרנו ערך C=10, כלומר אנו מענישים באופן שווה למידת ההתרחקות. אבל, כאשר מסתכלים על אותה השורה בטבלה, רואים שעבור ערכי C גדולים יותר, אז פחות מקבלים underfitting וזה בגלל שבמצב כזה נעניש יותר על כל התרחקות, ולכן אנו מתקרבים במצב זה למודל שמבין את המורכבות של סט הנתונים.

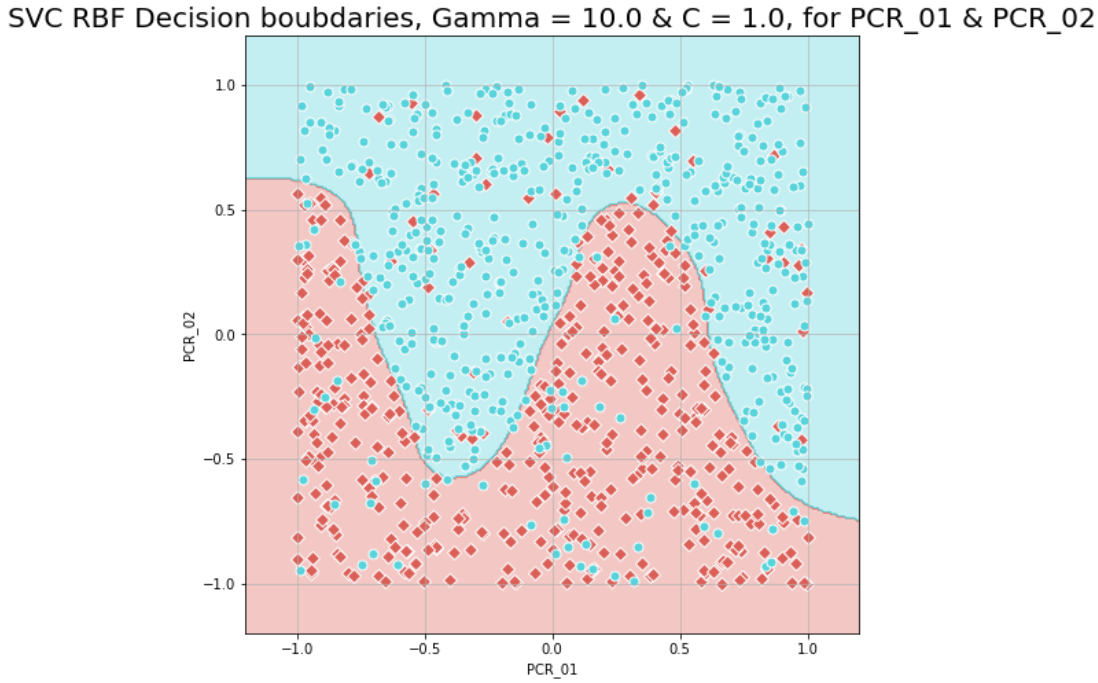
עבור ערכי gamma בתחום של 1 עד 10, אנו רואים שערכי C לא משפיעים מאוד על דיוק המודל (עבור כל ערכי C הדיוק נשאר יחסית זהה). אבל, עבור ערכי gamma גדולים מ-10 הדיוק שוב מתחיל לרדת.

נוכל להסיק שעובר ערכים אלו של gamma יש מצב של overfitting. לדוגמה, הקומבינציה הבאה גורמת ל-overfitting: C = 10, gamma = 10000. עבור קומבינציה זו אנו רואים שהדיוק על סט האימון שלנו הוא מושלם –1! אבל על סט הוולידציה אנחנו מקבלים דיוק יחסית נמוך בהשוואה לדיוקים אחרים שהצלחנו להגיע עליהם על סט הוולידציה. הסיבה לכך היא שעבור קומבינציה זו של פרמטרים אנחנו נמצאים במצב של overfitting. כלומר, הגדלנו את סיבוכיות המודל שלנו יותר מדי, ובכך גרמנו לדיוק על סט האימון לעלות יותר, אבל על סט הוולידציה הדיוק נפגע. מצב זה תואם את הניתוח התיאורטי שראינו בכיתה שבמצב של overfitting הדיוק על סט האימון רק ממשיך לעלות, אבל על סט הוולידציה הדיוק מתדרדר.

בנוסף, זה גם הגיוני לנו שעבור קומבינציה זו אכן נקבל מצב של overfitting כי בעצם כפי שלמדנו בהרצאה, הפרמטר gamma אחראי באופן אינטואיטיבי על רדיוס הכדורים, כאשר הוא מתנהג ביחס הפוך לגודלו. כלומר במצב זה בעצם אנו מתאימים לכל נקודה אדומה, כדור מאוד קטן שמקיף רק אותה ומתאים רק אליה. כלומר, יצרנו בעצם מודל שמתאים את עצמו באופן מושלם לסט האימון, ולכן שגיאת ההכללה עליו היא מאוד גדולה, מכיוון שהוא למד את סט האימון באופן מושלם ויצר לכל נקודה כדור עבורה (כי הוא ממימד אינסופי), ויצר כדורים מאוד קטנים סביב כל נקודה אדומה, ולכן כאן נעשה פרדיקציה על נקוד האדומה נוספת, כנראה שלא נפגע בכדור אדום ונטעה, ולכן שגיאת ההכללה במצב זה היא גדולה וקיבלנו מצב של overfitting. הפרמטר C גם הוא חשוב כאן, אנו מסתכלים כעת על C=10. כלומר, העונש שמביאים על כל חריגה גדול פי 10 מהמרחק של החריגה. זה ערך לא גדול כל כך של C, והוא כי שלמדנו אחראי על הסיבוכיות של מחלקת המודלים. אבל כפי שניתן לראות בטבלה, עבור ערך כל כך גדול של gamma, זה כבר לא כל כך משנה אילו ערכי C נבחר כי בכולם עדיין נישאר במצב של overfitting.

סעיף 18:

נסתכל על המודל עם פרמטרים אופטימליים עבור C וGamma:




```
The test score of the SVC model with the best C and gamma is: 0.864
```

בעזרת מודל זה הצלחנו להגיע לאחוז דיוק של 86.4% על ה-test set. ניזכר כי עבור מודל ה-KNN שאימנו בשאלה 4 הגענו ל85.6% דיוק, כלומר אחוזי הדיוק די שווים פחות או יותר. ואפילו אפשר לראות שמודל RBF טוב יותר מבחינת הרמת דיוק אם כי רק ב 0.8 אחוז (שזה יחסית זניח במקרה זה), אבל חשוב לשים לב לנקודות הבאות: מודל ה-RBF הוא מודל סטוכסטי ולכן היינו יכולים לקבל אחוזי דיוק מעט יותר גבוהים או נמוכים, ולכן אין כנראה הבדל משמעותי מבחינת הביצועים עבור 2 המודלים. עם זאת, מבחינות אחרות אכן יש הבדלים: אם נסתכל על הגרף של אזורי ההחלטה של שני המודלים, ניתן לראות שאזורי ההחלטה של המודל שהשתמש ב-RBF הן יותר חלקים, דבר שיכול לסייע כאשר רוצים להכליל את המודל. נקודה חשובה נוספת היא מבחינת השימוש בזיכרון - נשים לב ש-KNN ידרוש מאיתנו יותר שימוש בזיכרון, מכיוון שאנו צריכים לשמור את כל סט האימון ורק עליו לבצע פרדיקציות במקום לשמור רק את התוצאה של ה-RBF.