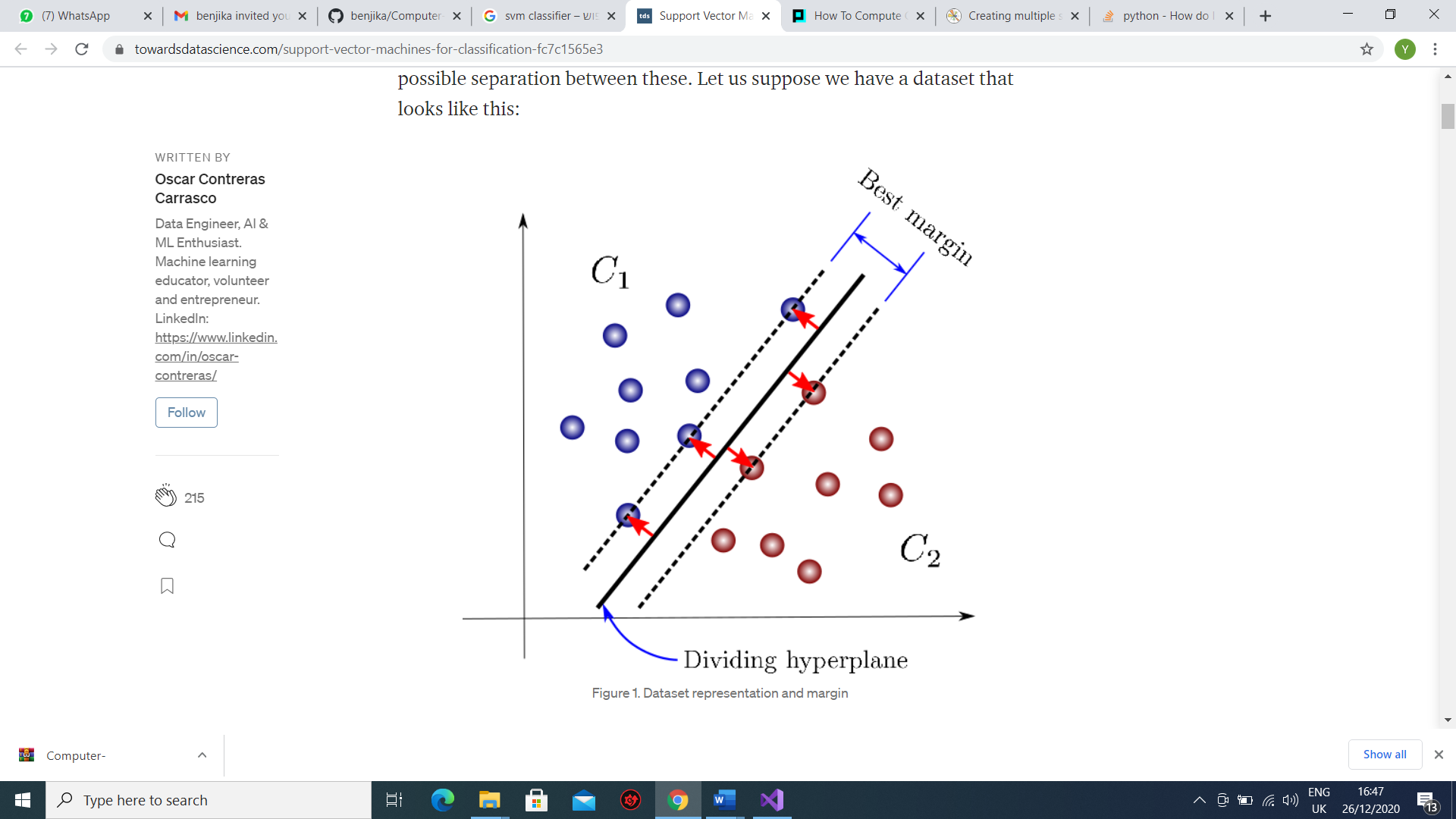
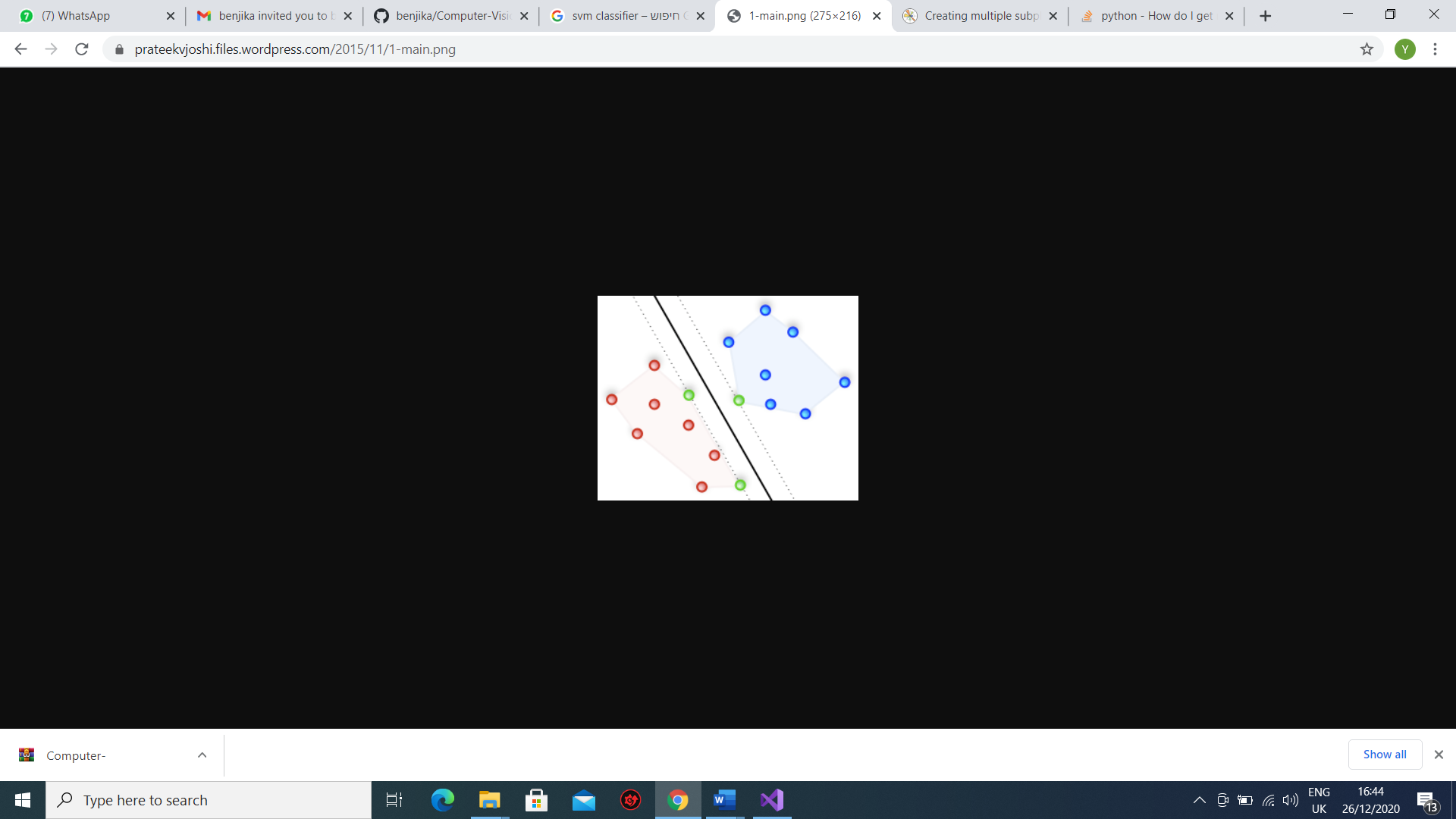
**דו"ח עבודה ראייה ממוחשבת – חלק 1**





הקדמה:

בחרנו לממש את מודל SVM בעזרת HOG. HOG נראתה לנו שיטה מעט מאתגרת ולכן היה לנו מעניין להתמודד איתה.

הערה לבודקת: על מנת להקל את תהליך הפיתוח והבדיקה הוספנו משתנה Phase הנמצא

ב-Params.

1. כאשר Phase= "TuningPhase" הקלאסים יכוונו ל-Fold1 ולפונקציית TrainWithTuning.
2. כאשר "ModelTrainPhase" Phase=הקלאסים יכוונו ל-Fold2 ולפונקציית Train.
3. כאשר "Assignment" Phase=הקלאסים יכוונו ל- class\_indicesולפונקציית Train.

כיוונון פרמטרי המודל

Image & Hog params Tuning

ערכנו מחקר באינטרנט ולמדנו שנהוג להשתמש ב-HOG כאשר התמונה בגודל 64X128(במיוחד עבור זיהוי אנשים). מכיוון שמגוון האובייקטים רחב היה קשה לגבל אינטואיציה איזה פרמטרים טובים עבורנו. היה צוררך למצוא גודל האופציה הטובה ביותר עבור שילוב בין גודל תמונה לגודל מסגרת(CELL) היא כאשר שניהם בכפולה של 8. האופציות הטובות ביותר היו :

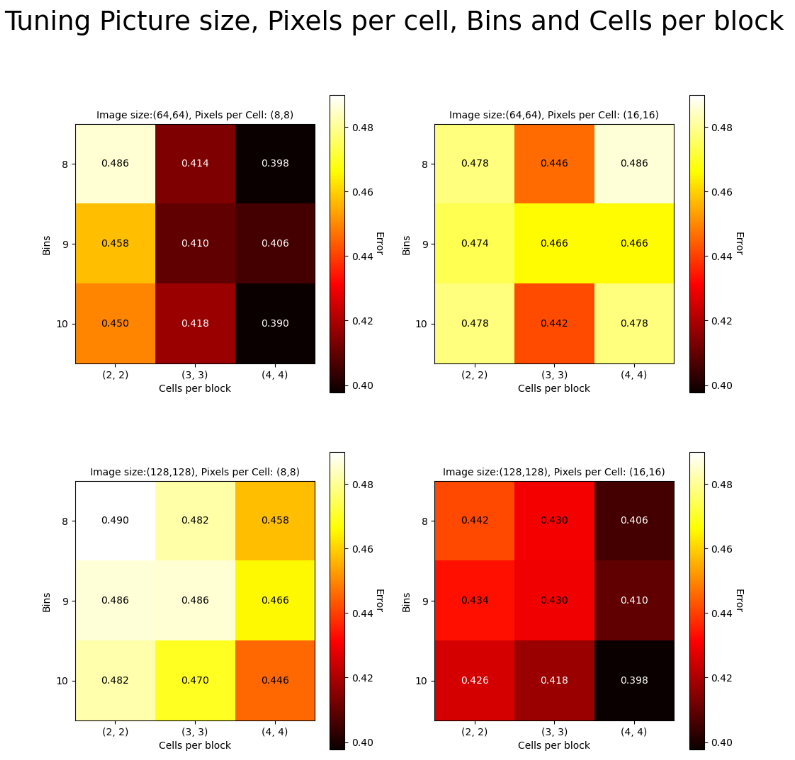
**Image:** image size – [(64,64), (128,128)]

**Hog:** pixels per cell – [(8,8), (16,16)] Bins – [8,9,10] Cells per block – [(2,2), (3,3), (4,4)]

את הבדיקה עשינו בעזרת ריצה בלולאה על כל הפרמטרים של התמונה וה-HOG (הלולאות נמחקו משיקולי אסתטיקה).בכל איטרציה בדקנו את הפרמטרים של התמונה וה-HOG עם כל השילובים של ה-SVM בעזרת GridSearch ולקחנו את השילוב הטוב ביותר. עבור ה-SVM עשינו טיונינג בנפרד לאחר מכן. את התוצאות הכנסנו לאובייקט והדפסנו מפות חום. ניתן לראות את התוצאה בפונקציית plot\_pics\_hog\_tuning\_heatmap() בקובץ. מצאנו כי הפרמטרים הטובים ביותר הם:

**Image:** image size – (64,64)

**HOG:** pixels per cell – (8,8) Bins – 10 Cells per block – (4,4)

נשים לב כי על אף שבחרנו את הפרמטרים שנותנים את הביצועים הטובים ביותר, יש סיכוי שאם היינו מגדילים את כמות ה-Bins וה-Cells per block היינו מצליחים למצוא נקודת מינימום עדיפה אבל זה כבר מעבר לטווח הערכים שנראו לנו סבירים ויכל לגרום ל-Over fitting לקלאסים וסאמפלים שעליהם אנו מבצעים טיוניג. נרצה להשאר בטווחים הסבירים ומתוכם לקחת את המינימום.

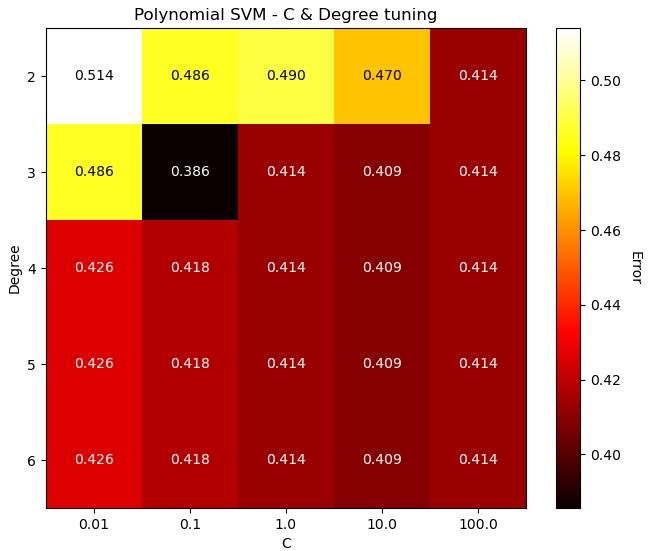
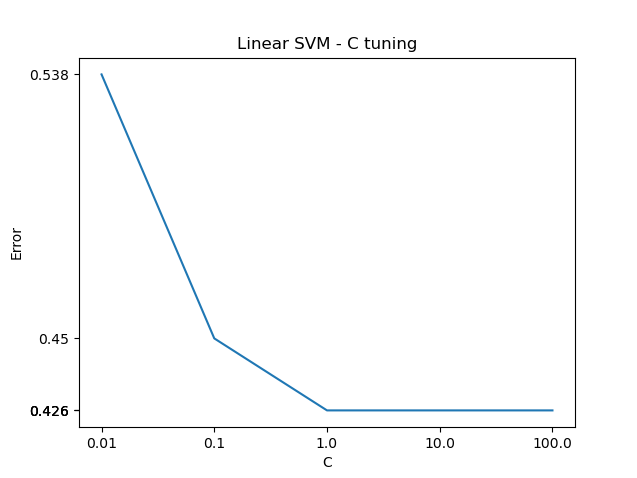
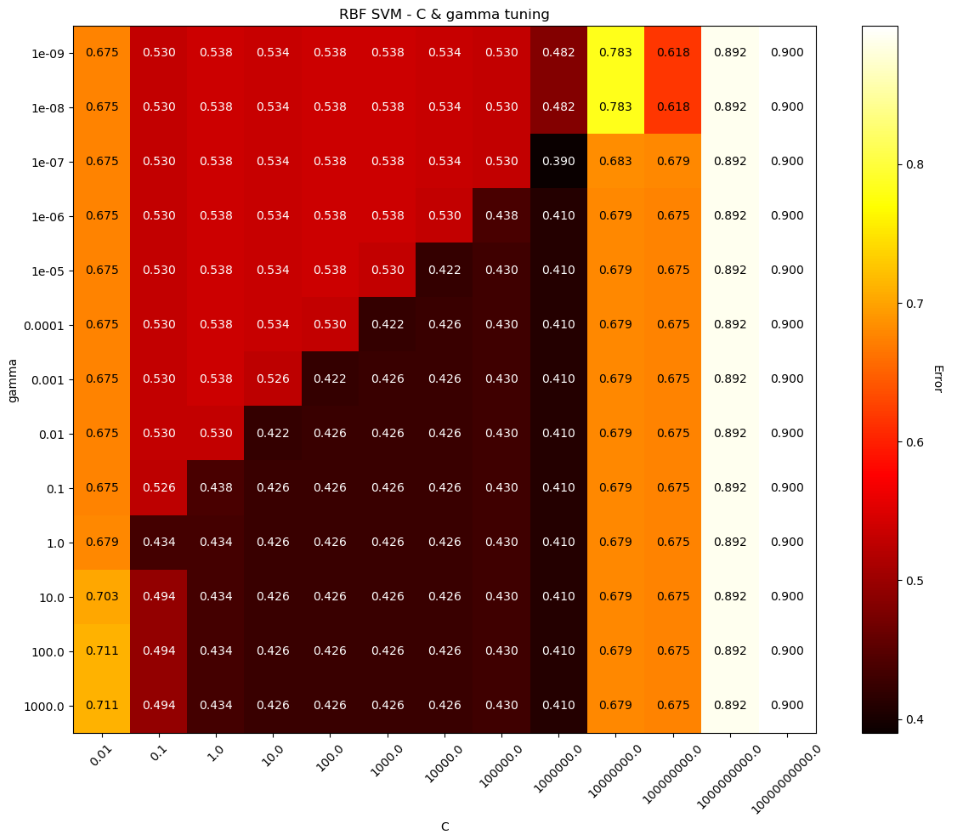
SVM Tuning

בגלל כמות הפיצ'רים והקלאסים הגדולה ציפינו להגיע לביצועים לא טובים במיוחד במסווג הלינארי מכיוון שהוא אינו גמיש מספיק. בחרנו את המסווג הפולינומיאלי מכיוון שנוח לכוון את הדרגה שלו ולהדגים כיצד מימד יותר גבוה עוזר למצוא הפרדה לינארית. את מסווג RBF בחרנו מכיוון שלא יצא לנו ללמוד אותו סכתה ורצינו לבחון את הביצועים שלו. מכיוון שמשימת הסיווג נוגעת ב-10 קלאסים שונים, היינו צריכים להגדיר את המסווג לשיטת One VS Rest, היוצרת מסווגים ככמות הקלאסים ובכל פעם בודקת שייכות לקלאס אחר. בסוף הריצה נותן מידת ודאות לגבי כל קלאס ומסווגת את האובייקט אל הקלאס עם הוודאות הגבוהה ביותר.

בדקנו מהם פרמטרים סבירים עבור בעיות מסוג זה. הכנסנו את הפרמטרים שמצאנו שיהיו העדיפים ביותר. מראש החלטנו שבמידה ויהיו מספר קומבינציות טובות נבחר את בעלת המימד הקטן ביותר על מנת להמנע מ-OverFit. בנוסף עובדה שאנו מבצעים טיונינג על עשירית מהקלאסים הקיימים ועל כל קלאס על 25 סאמפלים בלבד מקשה .

{'kernel': ['linear'], 'C': [0.01, 0.1, 1, 10, 100]},  
 {'kernel': ['poly'], 'C': [0.01, 0.1, 1, 10, 100], ‘degree': [2, 3, 4, 5, 6]}

ובזכות מפת החום נבחר :

{'kernel': ['poly'], 'C': 0.01, ‘degree': 3}

הבדיקה של הקרנל הלינארי איששה את הנחה שהוא מוגבל בביצועים שלו. ברוב המוחלט של הקומבינציות בקרנל הפולינומיאלי הגיעו לגודל שגיאה עדיף. גם אם היינו רוצים לשפר את הקרנל הלינארי לא היה ניתן מכיוון שהוא לא הצליח להשתפר לאחר C=1. גם ב-RBF נראה שבהחלט מצאנו נקודת מינימום אבל הנקודת הביצועים במסווג הפולינומיאלי עדיפים. אנו מרוצים מהתוצאה בקרנל הפולינומיאלי כי ניתן לראות בבירור שהגענו למינימום ומבחינתנו זאת תוצאה מעולה. כמו במקרה התמונה וה-HOG רצינו להשאר בערכים הסבירים של C ולא לנסות לדייק את המסווג הפולינומיאלי יותר .

גישת k-fold cross validation -

הפונקציה מחלקת את קבוצת הנתונים ל k קבוצות נפרדות בעלות גודל זהה.

לאחר מכן עבור כל קומבינציית פרמטרים במערך שבחרנו , הפונקציה מאמנת את המודל על k-1 קבוצות וקבוצה אחת נועדת בשביל לבחון את ביצועי המסווג.

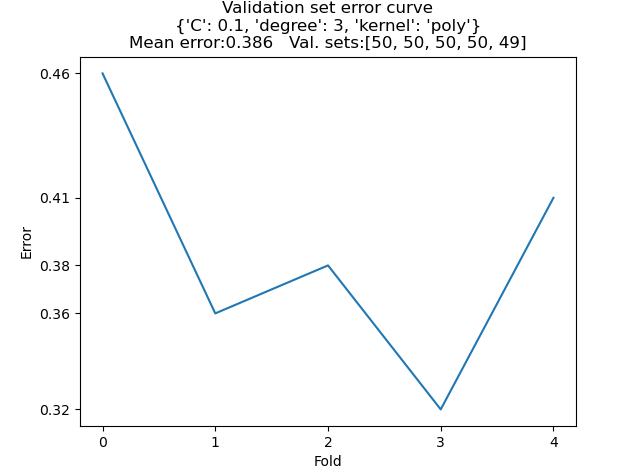
חוזרים על תהליך זה k פעמים , ולאחר מכן מחשבים את הממוצע של ביצועי המסווג .

תהליך זה מצריך עומס חישובי רב יותר אך , מניב תוצאות טובות יותר.

בעזרת הספרייה StratifiedKFold נחלק ל-5 פולדים בגדלים [50,50,50,50,49]

בעזרת StratifiedKFold ניקח את כל ה-Train-set, נערבב, נחלק ל-5 ונעשה טיונינג לפרמטרים.

הTrain-set מסודר לפי הקלאסים וערבוב יתן אימון יותר אפקטיבי.



לקחנו את ביצועי כל validation-set בקלסיפייר הטוב ביותר שמצאנו. מבחינתנו המסווג הטוב ביותר הוא המסווג עם Mean error rate הנמוך ביותר. לקחנו את המסווג בעל השגיאה הממוצעת הנמוכה ביותר- 0.386 הוצאנו לגרף וקיבלנו את הביצועים שלו בכל סבב Fold:

Kernel - פרמטר זה מייצג את צורת פונקציית המסווג. יכול להיות לינארי, פולינומיאלי, RBF, Sigmoid.

כידוע לא כל הנתונים ניתן לסווג על ידי מסווג ליניארי ולכן יש לבדוק בתהליך הכוונון גם צורות שונות לא ליניאריות עבור המסווג ולבחור את הצורה המניבה את התוצאות הטובות ביותר עבור סט הנתונים שלנו.

C – פרמטר של הכללה(Regulazation). ככל ש-C יותר גבוה כך אנו מענישים יותר על טעות והפרמטר פחות מוכלל. C חייב להיות חיובי.

Degree - פרמטר זה מייצג את מעלת הפולינום במקרה שהמסווג שלנו הוא פולינומי. ככל שיותר גבוה המסווג יהיה בעל יותר קימורים. חייב להיות בדיד.

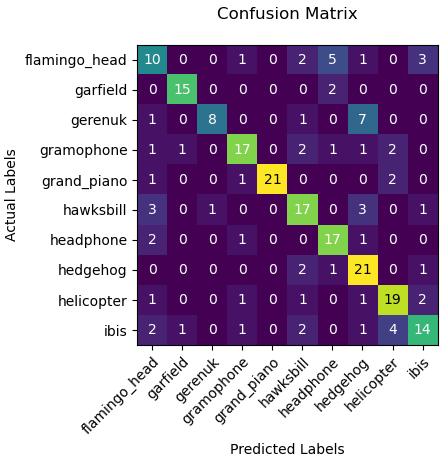
Gamma – משתנה המגדיר עד כמה נהיה מוכנים "לעקם" את ההיפר מישור על מנת לתפוס כל נקודת. Gamma נמוכה משמע נהיה מוכנים לעקם יותר את ההיפר מישור.

תוצאות ביצועי המסווג על Fold 2

לאחר שהזנו את סט הפרמטרים הטובים ביותר שנבחרו בתהליך הכוונון עבור אימון המסווג SVM

בדקנו את ביצועי המסווג על ה-Test-set ב-Fold 2 וקיבלנו Error rate: 0.299

בסוף הריצה נשמור את תוצאות ה-Test לקובץ pickle בתיקיית Results.

**

Confusion Matrix

מטריצה זו נועדה כדי להעריך את איכות הפלט של מסווג בערכת הנתונים. האלמנטים האלכסוניים מייצגים את מספר התמונות שבהן הסיווג היה נכון. בכל שורה נראה איך התמונות של כל קלאס סווגו ובכל עמודה איך במציאות מתחלקות התמונות שהמסווג החליט כי שייכות לקלאס מסוים. ככל שהערכים האלכסוניים של המטריצה גבוהים יותר כך המסווג טוב יותר, דבר המצביע על תחזיות נכונות רבות.

מסקנות:

מהתבוננות ב Confusion Matrix , כפי שכבר צוין בהסבר, האלמנטים האלכסוניים מייצגים את מספר התמונות שבהן הסיווג היה נכון. ניתן לראות כי ביצועי הסיווג של המודל שלנו היו הגבוהים ביותר עבור הקבוצות: grand piano , hedgehog, אשר בהם המודל צדק ב 21 תמונות כל אחת, עם זאת עדיין המודל שלנו לא הצליח לסווג באופן מושלם אף אחת מהקבוצות מתוך 10 הקבוצות שהיה עליו לסווג.

לעומת זאת ניתן לראות כי ביצועי הסיווג של המודל שלנו היו הנמוכים ביותר עבור הקבוצות: gerenuk , flamingo head. הסיבה לכך לדעתנו נעוצה מעצם הצורה המורכבת של דמות הפלאמינגו ושל דמות הצבי. אם נתבונן בתמונות בעלות השגיאה הגדולה ביותר המשתייכות לקבוצות gerenuk , flamingo head , נוכל לראות כי צורת הדמות של ראש הפלמינגו יחסית מורכבת ודינמית בשל הצוואר הדק הארוך והמפוטל שלה , אצל הצבי צורת הקרניים הארוכות והרגליים הארוכות והדינמיות (בחלק מהתמונות הצבי שוכב ובחלקן רץ , עומד או מכופף). עם זאת בהשוואה אליהן אם נתבונן בתמונות בעלות השגיאה הגדולה ביותר השייכות לקבוצות המצליחות של המודל שלנו: grand piano , hedgehog ניתן לראות כי צורת הדמות שלהן ניתנות לתיאור כצורה אחידה ויחסית סטטית אשר לא משתנה הרבה בין תמונה לתמונה. לקיפוד צורה חצי מעגלית בעל רגליים קצרות ביותר , ולפסנתר צורה רחבה ,אחידה וסטטית. מכאן אנו מסיקים כי המודל שלנו מתקשה לסווג קבוצות בהן מצויים איברים ארוכים ודינמיים, וקל לו לסווג קבוצות בהן הצורה נשארת אחידה ,סטטית ורחבה.

2 התמונות בעלות השגיאה הגדולה ביותר בכל קלאס

עבור בדיקת Class-Score של כל תמונה אצל כל קלאסיפייר השתמשנו בפונקציית decision\_function אשר מחזירה וודאות של כל מסווג עבור כל תמונה.

מתוצאות הסיווג חישבנו את ה שגיאה של כל תמונה ולקחנו מכל קלאס את ה-2 בעלות השגיאה הגדולה ביותר. ניתן לראות שהמסווג מתקשה עם תמונות שהedges שלו פחות ברורים לו כמו במקרה ה-Hedgehog, Ibis, הGerenuk ועוד.

לדעתנו ניתן יהיה להקטין את גודל שגיאת המודל על ידי אימון המודל שלנו על מאגר גדול יותר של תמונות עבור כל קלאס. זאת אומרת להגדיל את ה-split ל0.8. אסור לנו לשכוח כי האימון של המודל המסווג שלנו היה על לא יותר מ 25 תמונות עבור כל קלאס , שזה לדעתנו כמות קטנה יחסית של מידע בכדי לבנות את המודל המסווג שלנו. בנוסף, כוונון הפרמטרים על תמונות מכל הקלאסים ולא רק מ-Fold 1 היה יכול לגרום לפרמטרים להתאים למספר רב יותר של קלאסים ולא רק ל-Fold1.

דבר נוסף אשר יכול להקטין לנו את שגיאת המודל הוא איסוף (אם ניתן) ואימון על יותר תמונות בהן הדמות נבלעת ברקע , כך המודל שאנו נאמן יידע להתמודד בצורה טובה ויעילה יותר עם תמונות אשר הדמות בהן נבלעת ברקע.