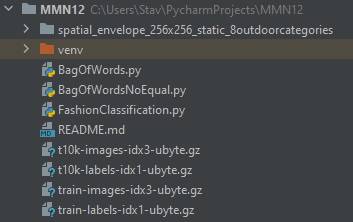
**ממן 12 מבוא לראייה ממוחשבת**

סתיו כהן

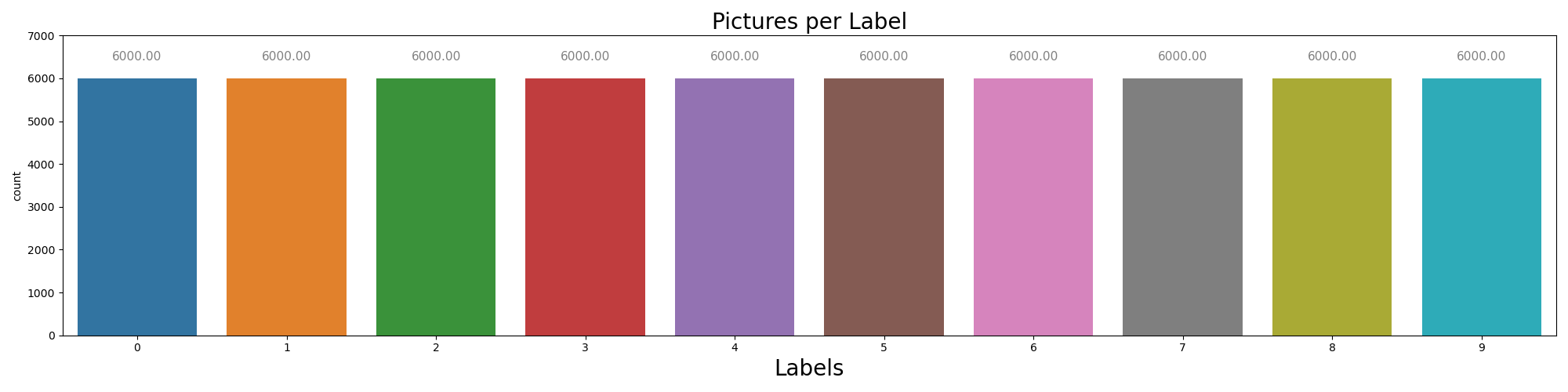
תאריך 16.11.20

****

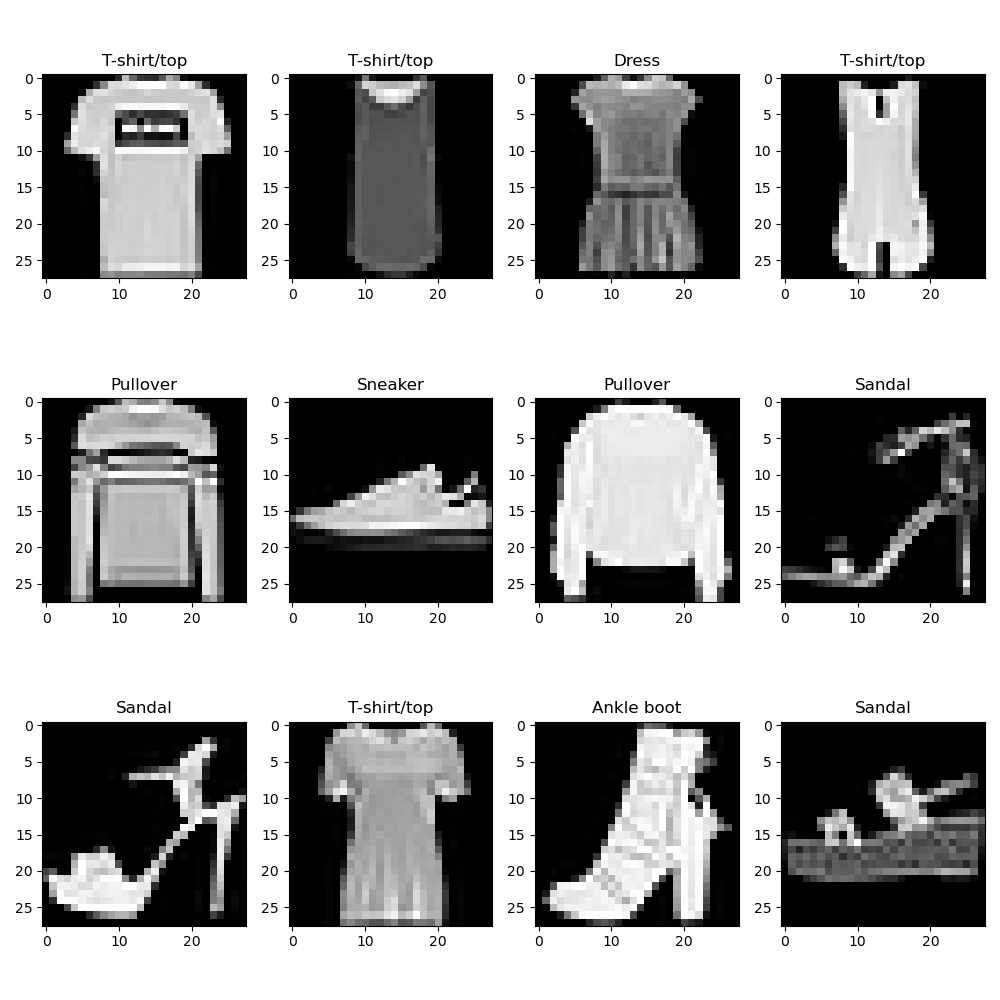
**כך נראה מבנה הממן אצלי, לא יכולתי לצרף את קבצי התמונות אך כן חשוב לי להראות כיצד המבנה נראה כדי לנסות למנוע בזבוז זמן בצד הבודק על חיבור קבצי הפרויקט כראוי במידה ותרצה להריץ את כל הקוד, הסברים על הקבצים יגיעו בהמשך. 😊**

**הקדמה**

ראשית בדקתי כמה תמונות יש מכל מחלקה,ניתן לראות כי מסד הנתונים מאוזן למפות(כמצופה מMnist) ובכל קטגוריה יש 6000 תמונות.

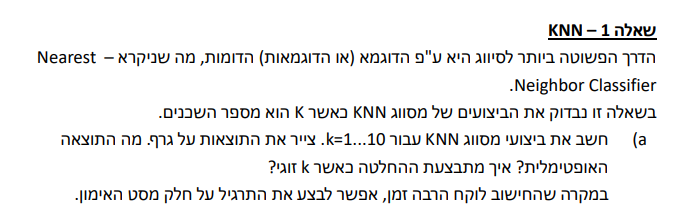
****

אציג את 12 התמונות הראשונות במסד הנתונים עם הכותרת המתאימה להם לפי סיווג הקטגורויות

****

**שלושת השאלות בחלק זה של הממן נמצאות בקובץ בשם FashionClassification**

**ניתן להוריד תיעוד ליד כל אחת מהן בשביל להפעיל אותם לחוד**

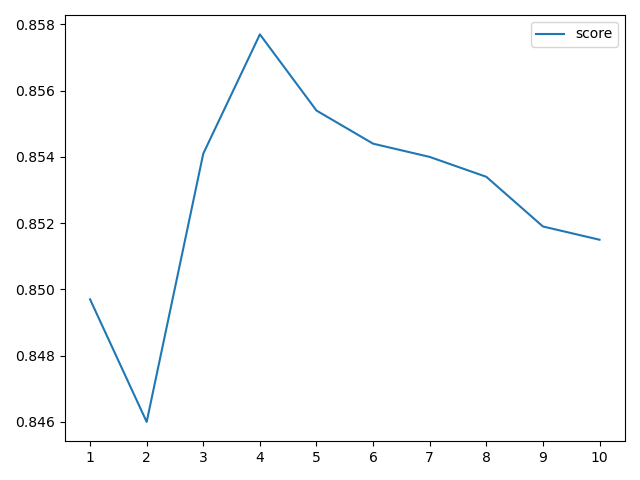


ההמלצה היא לא להשתמש בK זוגי אלא רק בK אי זוגי מכיוון שיכולה לקרות לנו סיטואציה שנרצה לסווג "נקודה" כלשהי על פי 4 השכנים הקרובים שלה ויהיה לנו מצב של תיקו, 2 שכנים מקטגוריה A 2 שכנים מקטגוריה B, לכן האלגוריתם יצטרך לבצע הכרעה כיצד לסווג את הנקודה, לכן אנו נרצה להימנע מסיטואציות כאלו.

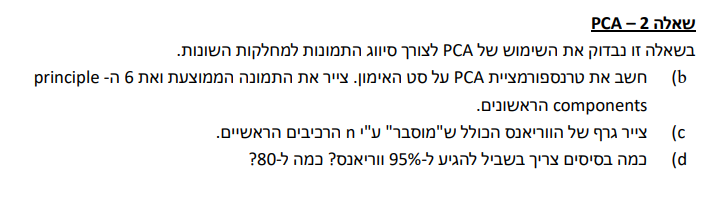
חלק מהרעיונות לפתרון בעיה זו היא לבצע בחירה רנדומלית של אחת הקטגוריות, לקחת את הקטגוריה הראשונה שמצאנו או לעלות\להקטין את מספר הK עד שנקודה זו תסווג ללא מצב של שיוון.

בשאלה זו השתמשתי בספרייה SKLearn.

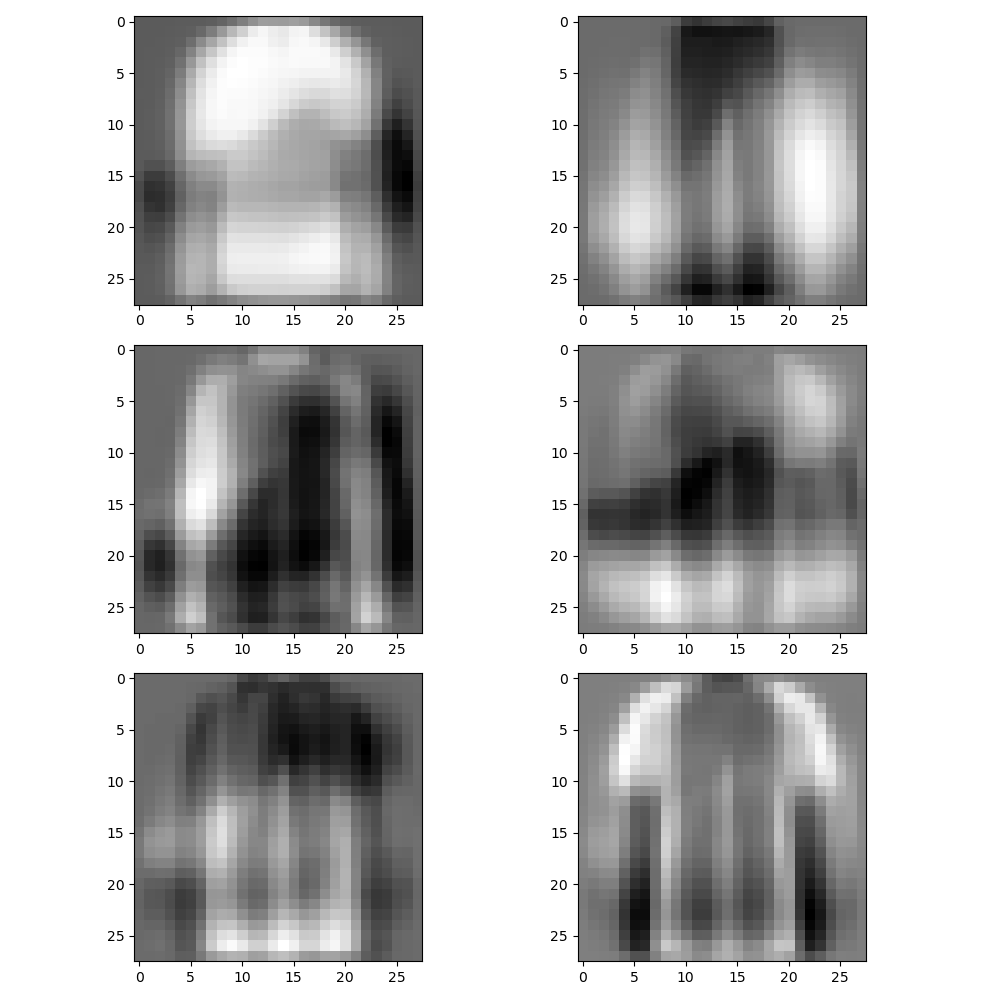
ביצעתי את החישוב על כל הDataSet , חישוב זה לקח לי באזור ה30 דק עם שימוש בn\_jobs=-1 , אך כמובן זה מעמיס על המעבד והמחשב יוצא מכלל שימוש למשך כ30 דק ( לפחות זה מה שקרה אצלי), ניתן למחוק את n\_jobs אך זה יאט את זמן החישוב.



ניתן לראות כי התוצאה האופטימלית היא חישוב לפי 4 שכנים קרובים, אך שאר התוצאות לא כאלה רחוקות מימנה שזה דיי הפתיע אותי (ניתן להסתכל על מדדי הScore בציר Y).

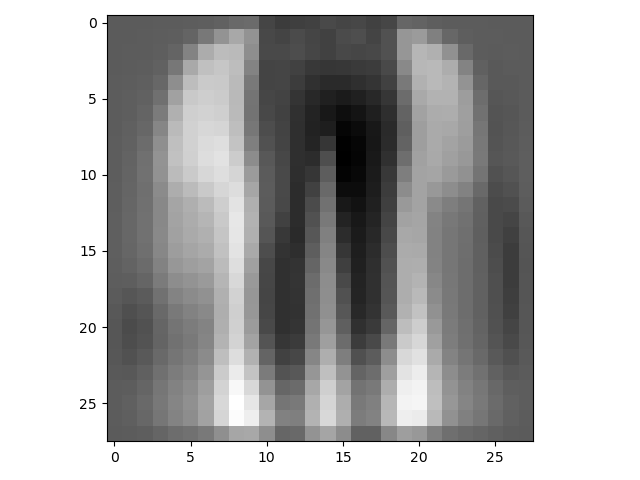
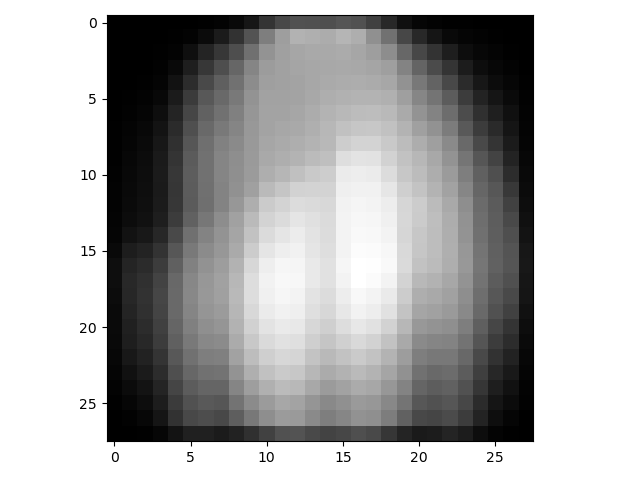


בשאלה זו השתמשתי בPCA מהספרייה SKLearn, ראשית מצאתי את 6 הPrinciple Components הראשונים שהם הכי משפיעים.



לאחר מכן מצאתי את התמונה הממוצעת (התמונה הימנית) שיצאה דיי מבולגנת אך ניתן לראות בבירור(יחסית) מכנסיים ואם מתאמצים אפשר גם לראות צורת חולצה ונעל.

החלטתי גם לנסות לחבר את ששת התמונות לתמונה אחת ולראות את התוצאה(התמונה השמאלית, לא שייך לתרגיל)

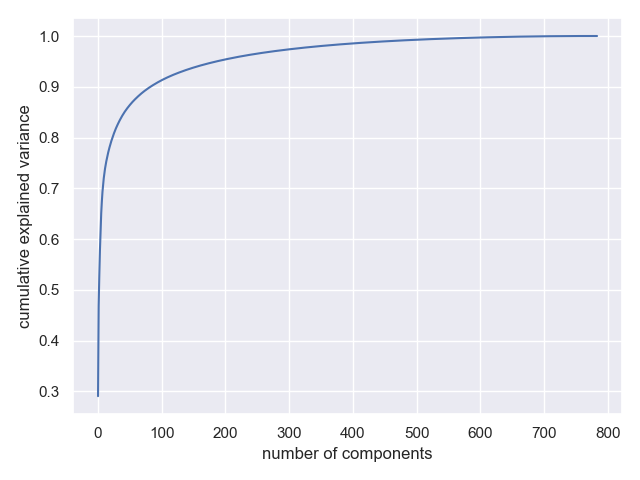


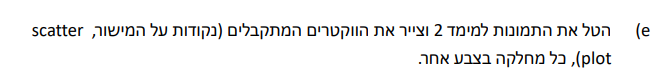
**סעיף C +D:**

ציירתי גרף כולל של הווריאנס "המוסבר" ובדקתי במדויק מתי מגיעים לדיוק של 80% ומתי לדיוק של 95%



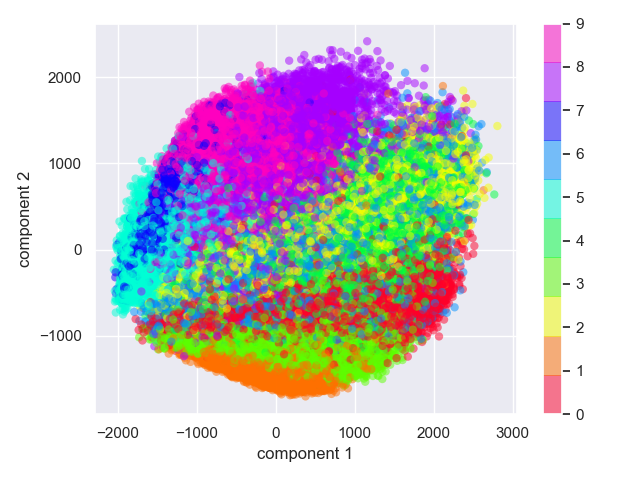
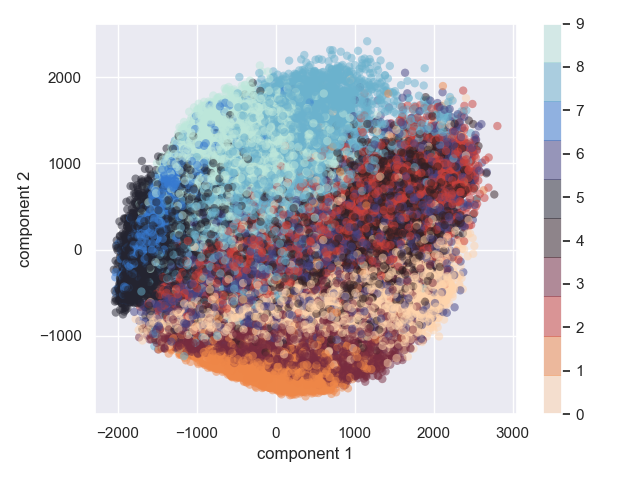
ניתן לראות בגרף שבאזור ה186 והלאה יש ההאטה רצינית במומנטום של הגרף ונראה כי 186 יתנו לנו 95% דיוק וכדי להגיע ל100% דיוק נצטרך 784 בסיסים, הפרש ענק בכמות הבסיסים!



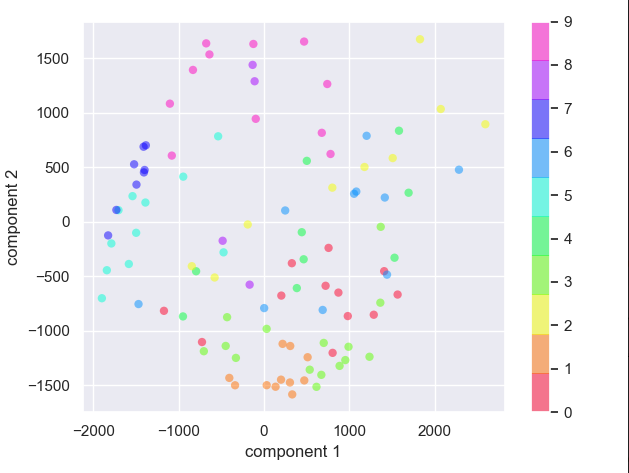


הטלתי את התמונות למימד 2 וציירתי את הוקטורים המתקבלים כנקודות על המישור בשתי צבעים שונים כדי לראות הבדלים שלא תלויים רק בצבעים שבחרתי.

המסקנה המרכזית שאני לוקח משרטוט זה היא שטכניקת הPCA לא פותרת את הבעיה שלנו, אנו כן רואים כיצד חלק מהקטגוריות חופפות לאחרות אך אין בינהן הפרדה כלשהי.



ניסיתי גם לבדוק פיזור זה עם 100 נק בלבד אך גם כאן ניתן לראות בבירור כי PCA לא פותר את בעיית החלוקה.





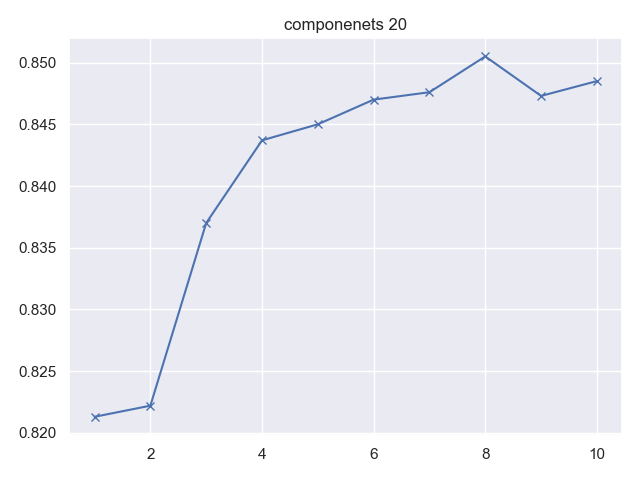
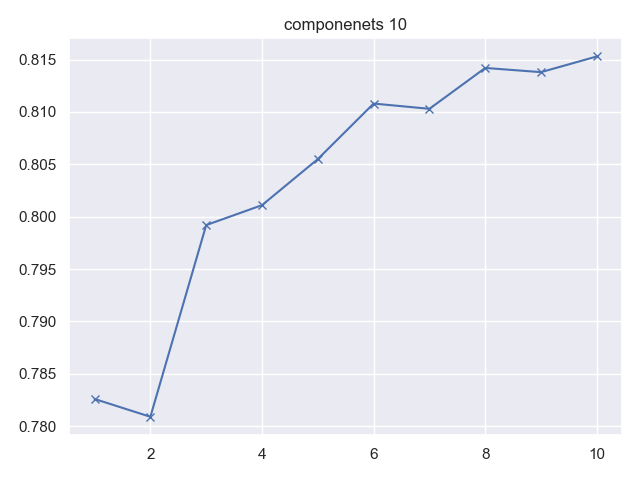
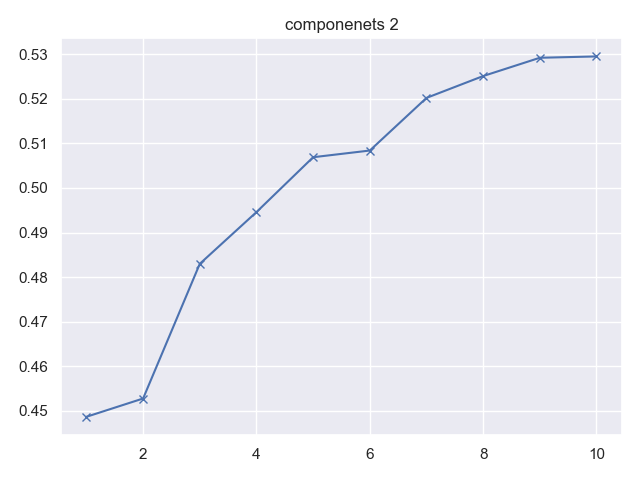
חזרתי על שאלה 1 , בניתי פונקציית עזר בשם KNNHelper אשר תעשה את שערוך הKNN כל פעם לפי K ולפי המימד המתאים גם כאן נעשה שימוש בn\_jobs=-1.

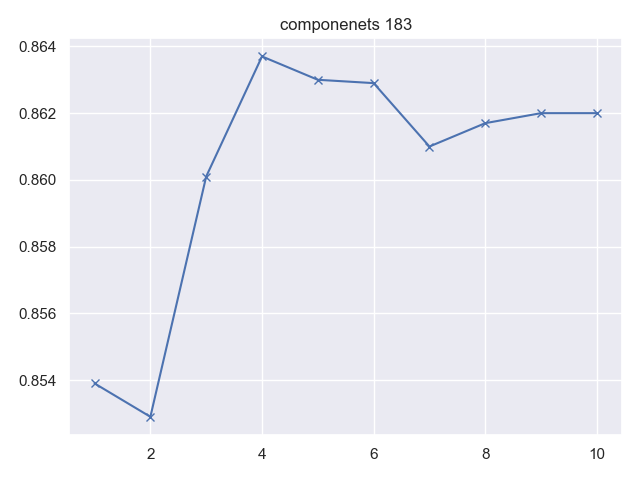
הצגתי את שלושת הגרפים המתאימים לפי כמות הבסיסים המוצגת בכל גרף בכותרת, ציר הX מציג את כמות הK שכנים, ציר Y מציג את מדד הScore של SKLearn .

ניתן לראות שימוש ב20 בסיסים וK=8 הוא האופטימלי ביותר 0.85. מה שמעניין מאוד שאנו קרובים מאוד ל Score מסעיף 1 בו רצנו על כל הDataSet בלי לבצע הטלות! רק כאן יש הבדל משמעותי מאוד והוא זמן הריצה ! לקח לי בערך דקה לסיים לרוץ על שלושת האפשרויות של הבסיסים לעומת כחצי שעה שצרתי על כל הDataset ואחוזי הדיוק דומים מאוד 0.85 כאן מול 0.858 על כל הDataSet .

שימוש בPCA עם כמות בסיסים נכונה בהחלט תורם לזמן הריצה משמעותית ומחסיר טיפה מהדיוק, מעניין מאוד.

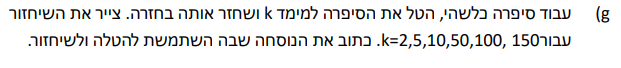
לעומת זאת שימוש ב2 בסיסים נופל משמעותית בניקוד 0.53





בדקתי בדיקה נוספת של 183 בסיסים (בהם אנו מגיעים ל95% שונות מוסברת לפי הסעיפים הקודמים)

גם כאן המהירות השתפרה פלאים (לקח באזור 2 דק ) והדיוק השתפר עוד יותר.



ביצעתי הטלה למימדים הנתונים בעזרת שימוש בפונקציות transform , inverse\_transform

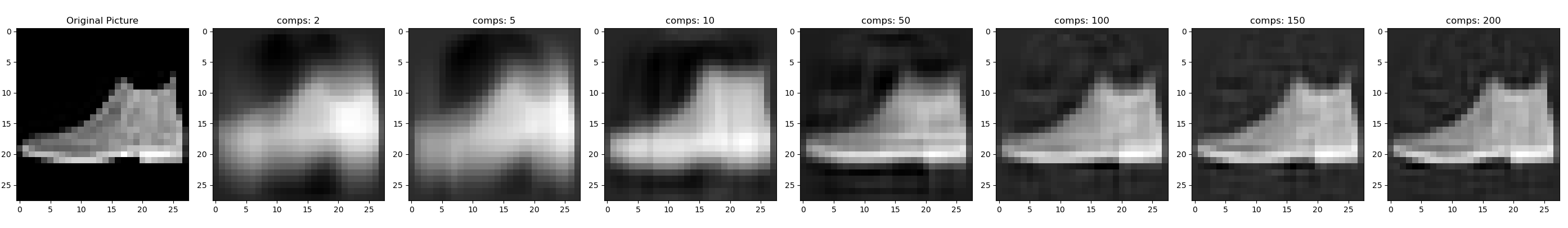
מהספרייה SkLearn

הפונקציה של Transform מוגדרת כך

return np.dot(X, self.components\_.T)

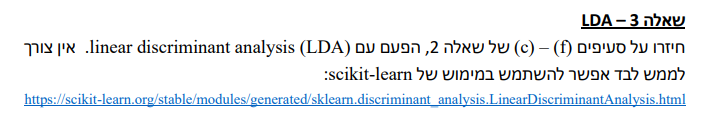
ו Inverse\_transform מוגדרת כך:

return np.dot(X, self.components\_) + self.mean\_



ציירתי את השחזור עבור הבסיסים הנתונים בשאלה, נראה כי לאחר 150 התוצאה כבר דומה נורא לתמונה המקורית(השמאלית ביותר)

החלטתי לנסות גם 200 בסיסים, ישנו הבדל אם מסתכלים טוב אך הוא אינו משמעותי נורא.

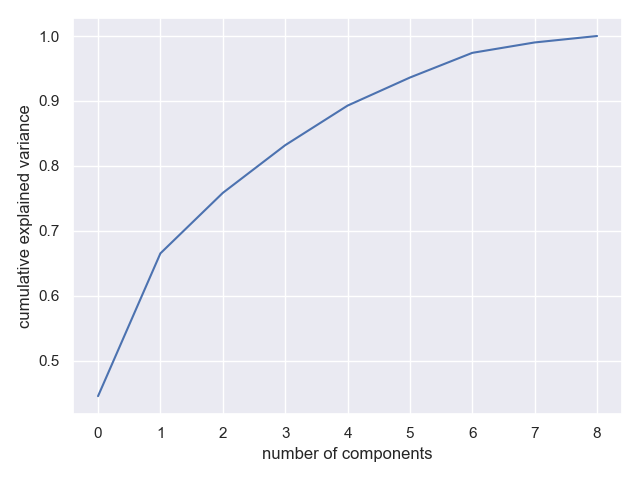


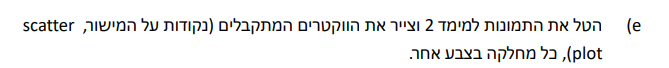


ניתן לראות שהיו צריכים בסה"כ 8 בסיסים בשביל להגיע ל100 אחוז!

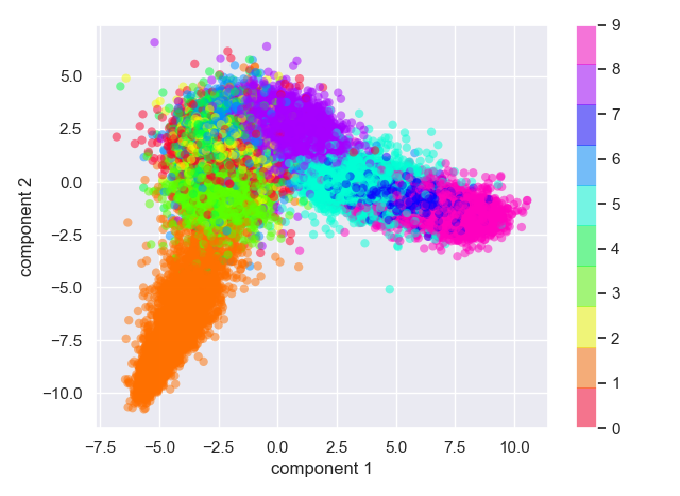


אחרי 3 בסיסים הגענו ל80% ואחרי 6 הגענו ל95%



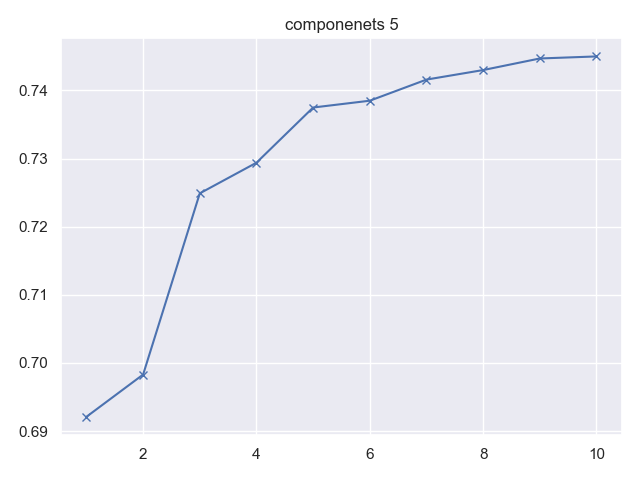
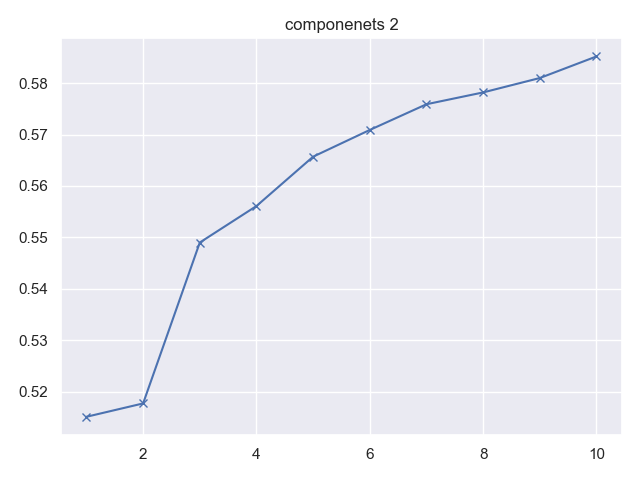


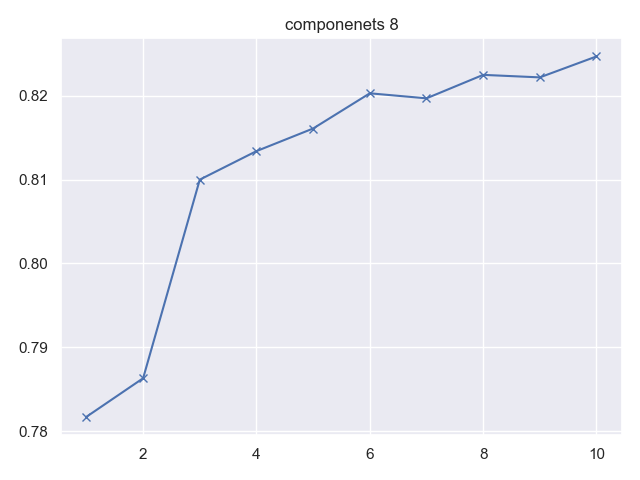
גם כאן נראה שהבעיה לא נפתרה לחלוטין אבל אפשר לראות גבולות ברורים הרבה יותר בין חלק מהמחלקות לדוגמא מחלקה 1 שדיי מבודדת משער המחלקות, אך עדיין מחלקות 0,2,4 דיי מבולגנות.

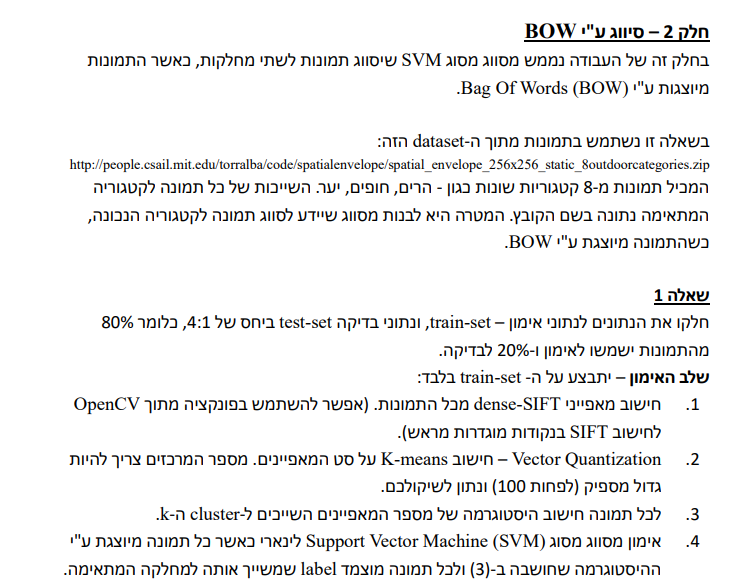


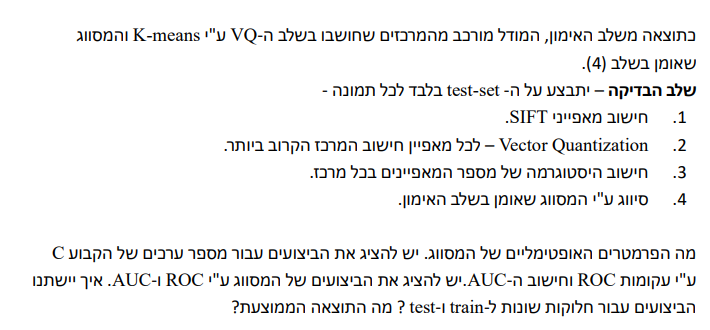


גם כאן ניתן לראות שככל שאנו משתמשים ביותר בסיסים אנו נקבל יותר דיוק בKNN (לא ניתן היה לבצע הטלה למימדים 10,20 כי מקסימום בסיסים שהיה למודל זה הוא 8, ניתן לראות זה בסעיפים הקודמים)









ראשית התחלתי בבדיקה של כמות התמונות מכל 8 הקטגוריות של בסיס הנתונים, גיליתי כי לכל קטגוריה יש מספר שונה של תמונות לכן החלטתי לבדוק מי הקטגוריה עם הכי פחות תמונות ומצאתי כי לקטגוריה Highway יש הכי הרבה פחות תמונות-260 תמונות סה"כ לעומת הקטגוריה opencountry עם 410 תמונות סה"כ

 לכן החלטתי לנקוט 2 גישות שונות בממן זה:

1. החלטתי לקחת מכל קטגוריה רק 260 תמונות סה"כ ולחלק אותם לסט האימון המבחן וכך לשמור על איזון בכמות התמונות של כל קטגוריה בכל סט. קובץ BagOfWords

2. ניסיתי לקחת כמות יחסית (באחוזים) מכל קטגוריה לסט האימון וסט המבחן, דבר זה יגרום לחוסר איזון של תמונות מכל קטגוריה בכל אחד מהסטים. קובץ BagOfWordsNoEqual

החלטתי לבדוק את שני גישות ולראות מה הגישה הנכונה לפתרון בעיה זו.

אתחיל בגישה 1:

**התחלתי בחילוק של סט הנתונים וסט האימון ברטיו של 4:1**



אלו היו הגדלים של הסטים.

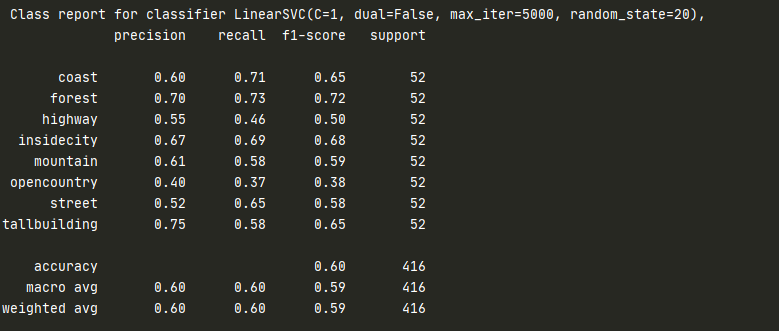
השתמשתי באלגוריתם Sift בעזרת התקנה של הספרייה Contrib Opencv כי בספריה הראשית אין אפשרות להשתמש באלגוריתם זה כיום (בגלל פטנט רשום)

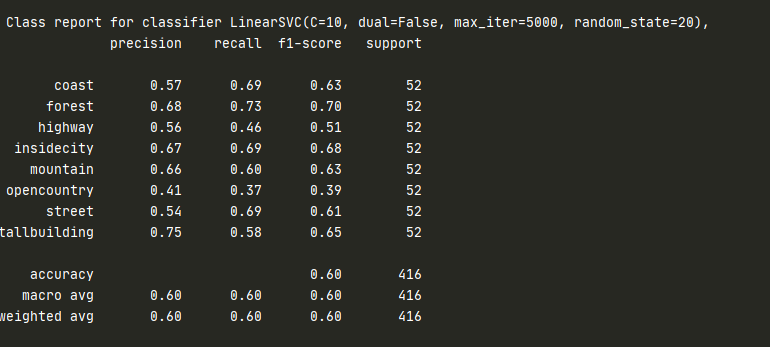
בניתי MiniBatchKMeans ובניתי היסטוגרמה לשם מימוש האלגוריתם BagOfVisualWords (שם מתאים יותר)

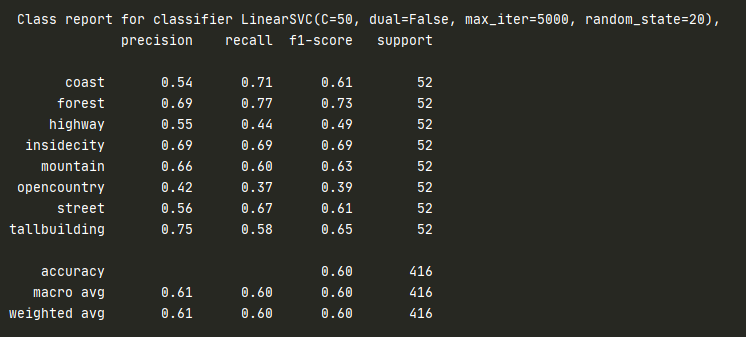
לאחר מכן השתמשתי בפונקציית SVM לחיזוי התיוג של תמונות על סט המבחן.

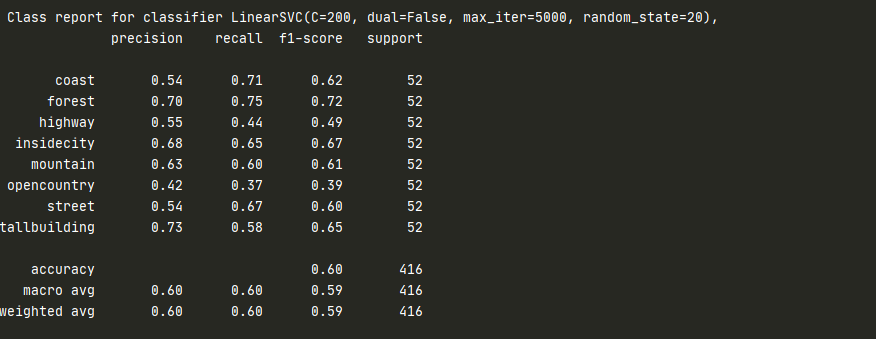
לבסוף הדפסתי את דיוק המודל באמצעות שימוש בMetrics.classification\_report לשם קבלת הדיוק של המודל.

ניסיתי כמה פרמטרים שונים לבניית Kmeans אך K=100 היה הפרמטר שקיבלתי עבורו את האחוז דיוק הגבוה ביותר, ניסיתי גם כאן ערכים שונים לפרמטר הרגולציה C אך קיבלתי תוצאות סופיות דיי דומות למרות שהתרחשו שינויים קלים בדיוק של קטגוריה מסויימת לדוגמא ניתן לראות בC=1 דיוק של 0.65 על קטגוריה Coast מול 0.61 ב C=50 אך עליה קטנה בדיוק בForest .





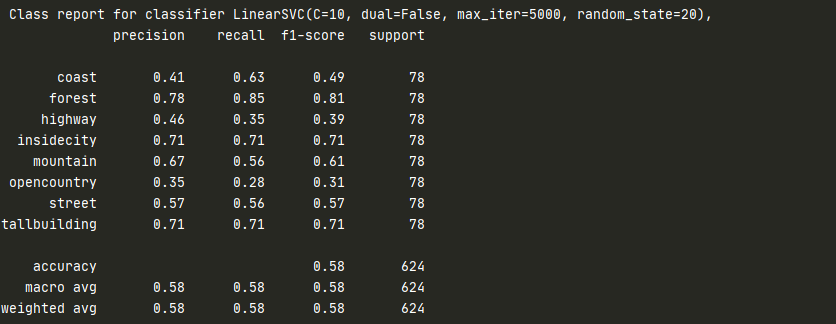




קיבלתי 0.60% דיוק בסיווג באמצעות שימוש באלגוריתם זה, כעת אבדוק את רטיו חלוקת מסד הנתונים לסט האימון והמבחן.

**חלוקה של 0.7 לאימון 0.3 למבחן :**

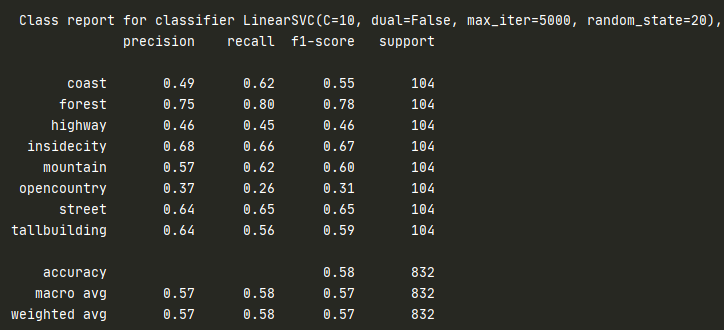
****

****

ניתן לראות כי הדיוק ירד במעט.

**חלוקה של 0.6 לאימון 0.4 למבחן :**

****



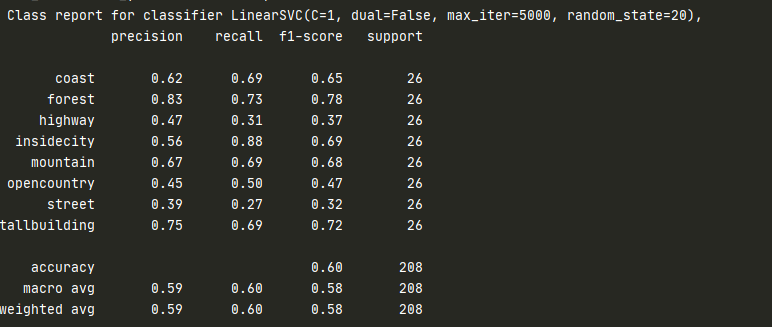
ניתן גם כאן לראות ירידה בדיוק.

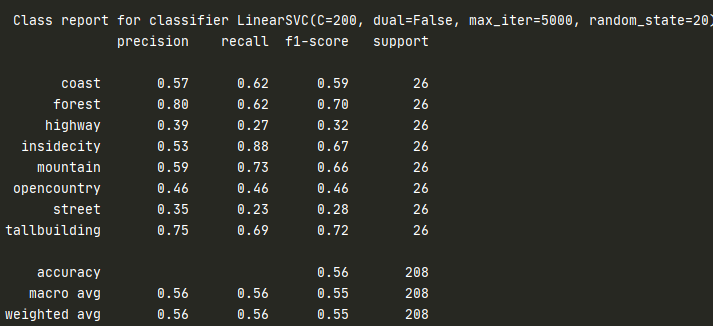
דבר זה אינו מפתיע כי מסד הנתונים שלי גם ככה מצומצם, צמצום סט האימון הינו דבר שרק מרע על דיוק המודל במשימה זו, אעבור להגדלת מסד האימון!

**חלוקה של 0.9 לאימון 0.1 למבחן :**



כאן קרה דווקא עניין מעניין, ככל שהפרמטר C גדל אז אחוז הדיוק של המודל קטן!





מכאן אסיק כי התוצאה האופטימלית של המסווג היא חלוקה ל0.8 למסד האימון ו0.2 למסד המבחן.

פרמטרים אלו העניקו לי דיוק של **0.60**% שזה לא דבר כזה רע יחסית למסד נתונים קטן כזה ומודל פשוט.

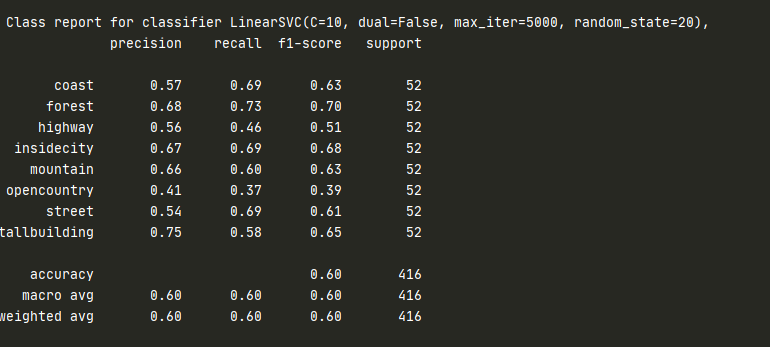
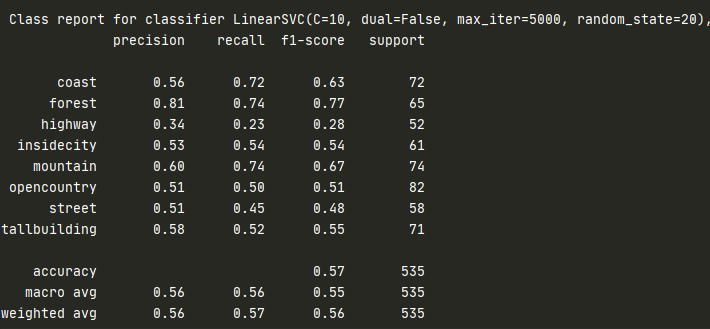
(מעניין מה הייתה התוצאה אם הייתי עושה Data Augmentation אך לא הגענו לזה בקורס עדיין)

אבדוק את גישה 2:

**התחלתי בחילוק של סט הנתונים וסט האימון ברטיו של 4:1 ,0.8 0.2**



כאן אני משתמש במלוא התמונות במסד הנתונים אך האם נקבל תוצאות טובות יותר?

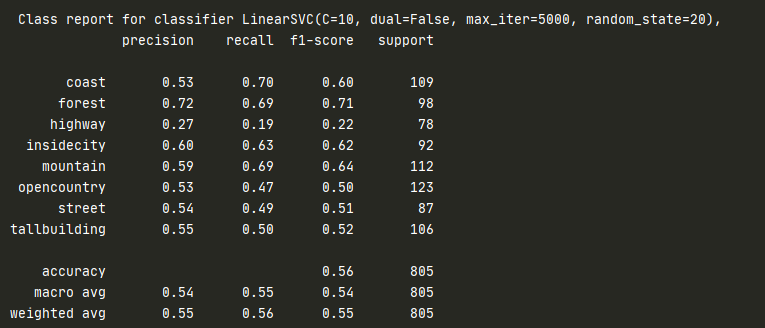


התשובה היא **לא** במקרה זה, ניתן לראות בתמונה הראשונה אשר הדיוק שלנו ירד ל0.57 לעומת ה0.6 במודל המאוזן, ניתן לראות הפרשים משמעותיים בדיוק הקטגוריה highway אשר כללה אך ורק 260 תמונות סה"כ!

במודל המאוזן הדיוק שלה היה 0.51 לעומת 0.28 במודל הלא מאוזן, מאי איזון זה אני חששתי לכן החלטתי לבנות שני סוגי מודלים ולבדוק את תיאורית איזון מסד הנתונים(אשר ברורה למדי בנושאי DL אך רציתי לבחון אותה גם כאן) .

אסיק לעת עתה כי המודל המאוזן הוא הטוב ביותר כרגע אך בכל זאת אבדוק את המודל הלא מאוזן בזריזות על חלוקה שונה של מסד הנתונים.

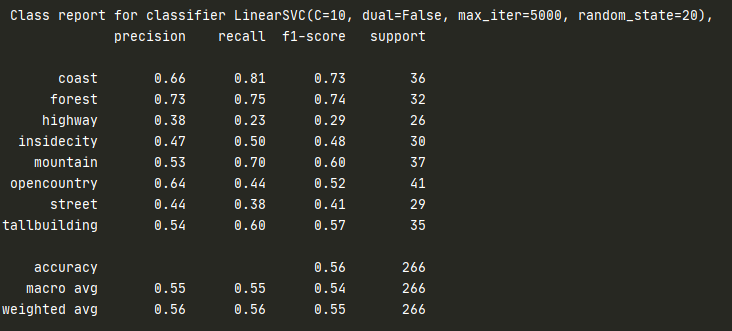
**חלוקה של 0.7 לאימון 0.3 למבחן :**

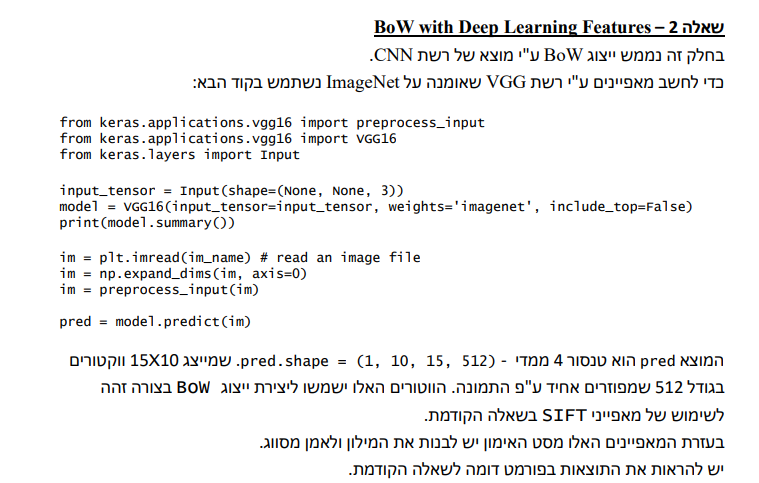
ירידה בדיוק נראית גם כאן.

**חלוקה של 0.9 לאימון 0.1 למבחן :**





גם כאן נראה ירידה בדיוק, אעבור לשאלה השניה בממן זה ואבדוק גם כאן את שני המודלים.



בשאלה זו לא השתנו דברים רבים מהשאלה הקודמת אך במקום להשתמש בSift בתור Feature Extractor אני משתמש בחלק של VGG16 לשם Feature Extractor.

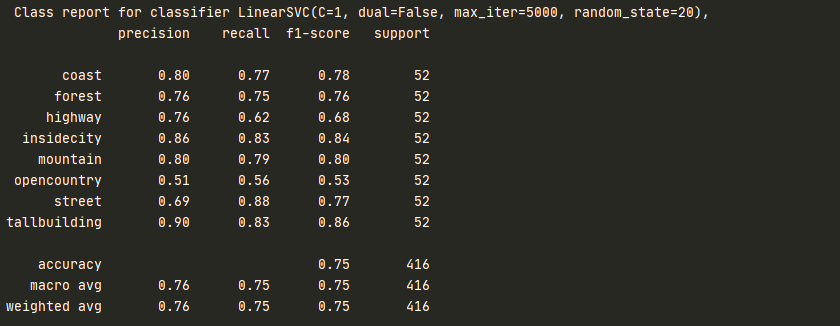
אני מקבל טנסור בגודל הבא: 

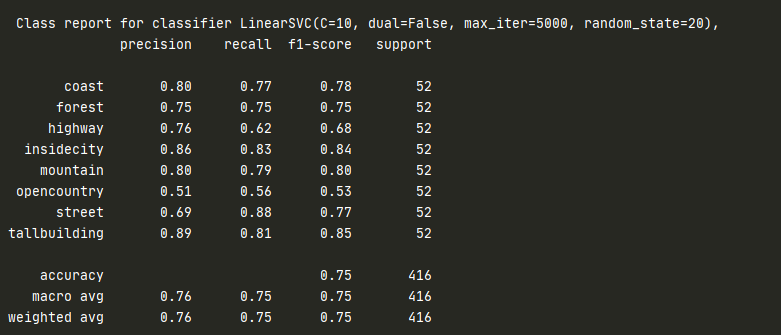
וממיר אותו לצורה - זו הצורה שהצלחתי להגיע בה לדיוק הכי גבוה של המודל.

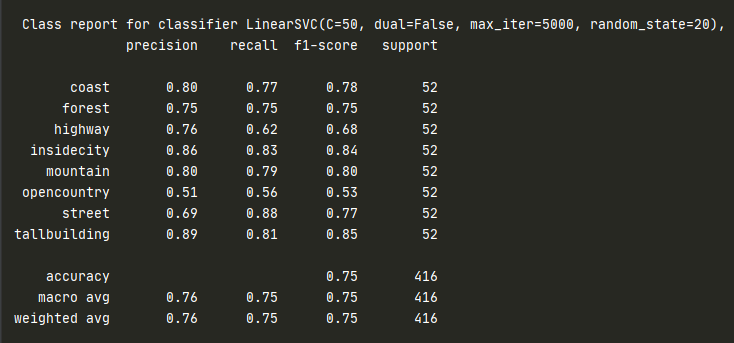
גם כאן אבדוק את שני הגישות שדיברתי עליהם מקודם.

אתחיל בגישה 1:

**התחלתי בחילוק של סט הנתונים וסט האימון ברטיו של 4:1 ,0.8 0.2**

****

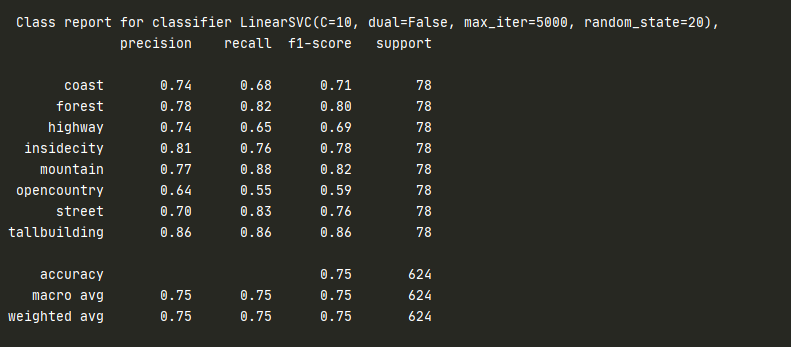




גם כאן פרמטר הרגולציה לא השפיע על הדיוק הסופי, אך קיבלתי דיוק של 0.75% ! זה הרבה יותר טוב מהדיוק של המודל הקודם שהשתמש בSIFT.

**חלוקה של 0.7/0.3**

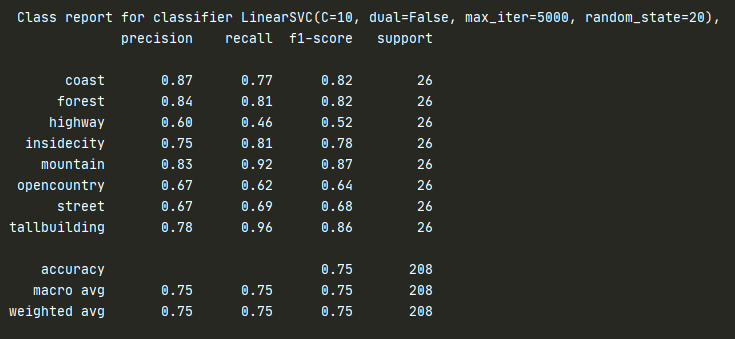
****

****

גם כאן הגעתי לדיוק של 0.75 מפתיע שלא הייתה ירידה באחוזי דיוק הפעם.

**חלוקה של 0.9/0.1**



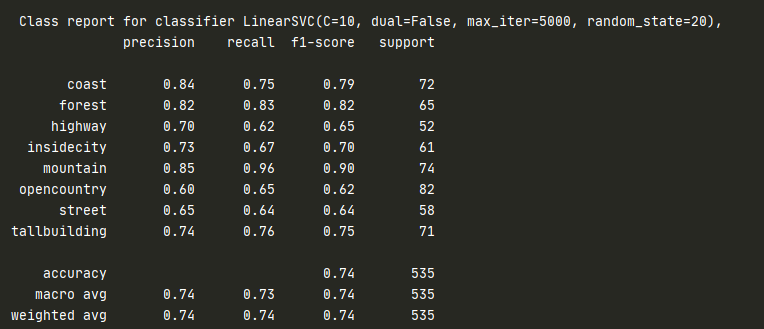
****

גם כאן הגענו לאותו דיוק, מעניין מאוד נראה כי המודל הזה אמיד יותר לשינויים בכמות הנתונים שעליו הוא מתאמן.

כעת אנסה את גישה 2 הלא מאוזנות ונראה את תוצאתיה על המודל המשולב עם VGG:

**חלוקה של 0.8/0.2 :**

****

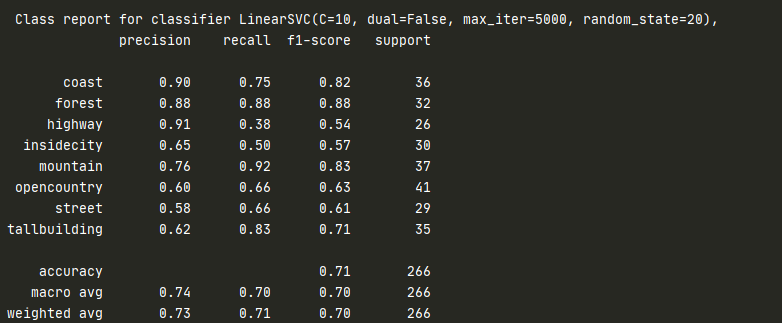


גם כאן ניכרת ירידה קטנה באחוז הדיוק אך הפגיעה לא משמעותית מדיי.

אנסה חלוקה נוספת.

**חלוקה של 0.9/0.1**

****

****

**חלוקה זו פגעה בדיוק בצורה משמעותית.**

**שימוש בטבלאות ROC ו AUC**

בסעיף זה אעבוד עם גישה 1 בלבד- הסט המאוזן ואבדוק גרפי ROC.

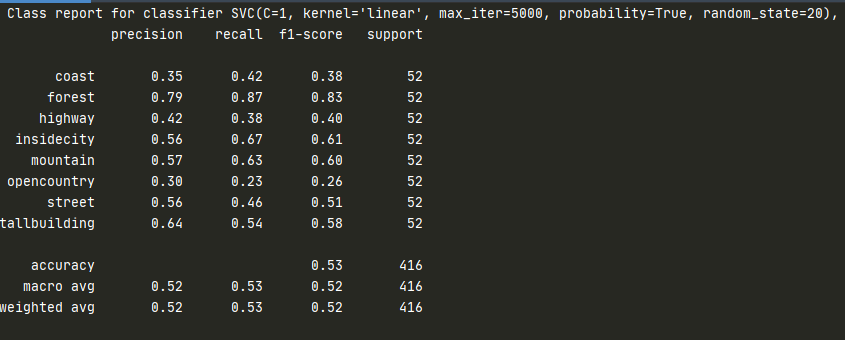
לשם כך הייתי צריך לשנות את פונקציית הSVM מ SVM.linearSVC אל SVM.SVC ושימוש בKernal ליניארי, החזרת פלט הסתברויות לכל סיווג במקום קטגוריה יחידה.

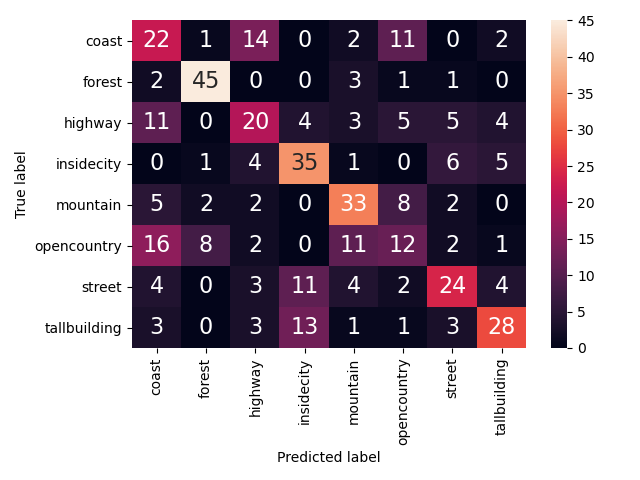
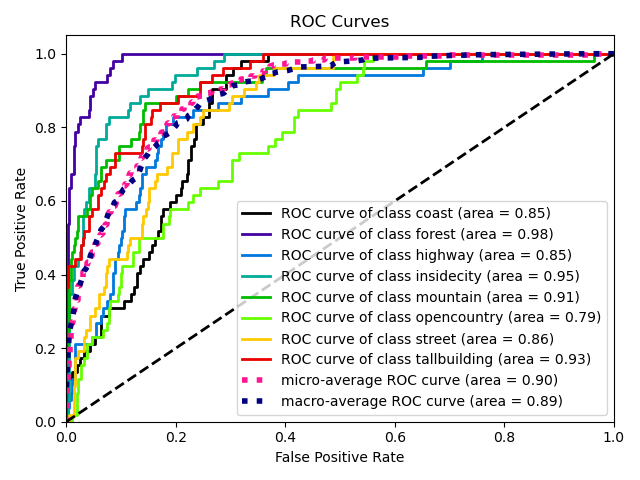
החלפת פונקציה זו פגע בציון המודל F1-score למרות שהמודל היה זהה וכביכול אמור להיות בעל מימוש זהה בSKlearn כאשר בוחרים kernel ליניארי.

השארתי בקוד אפשרות להחליף בשני המודלים בקובץ BagOfWords .

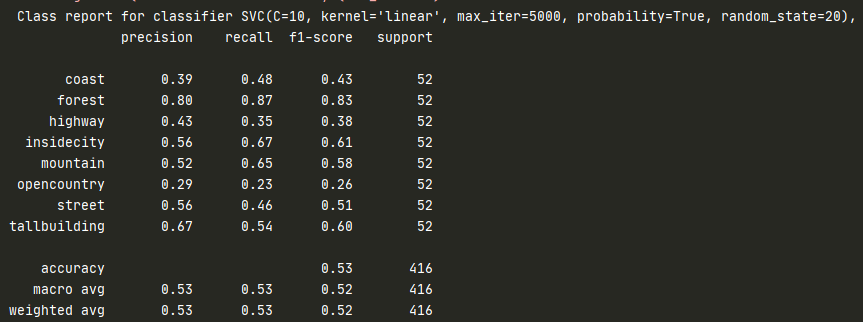
ניסיתי גם חלוקה שונה של בסיס הנתונים,כמות BINS שונים וכו כמו שתואר בהתחלת השאלה אך הנתונים האופטימלים נשארו זהים.

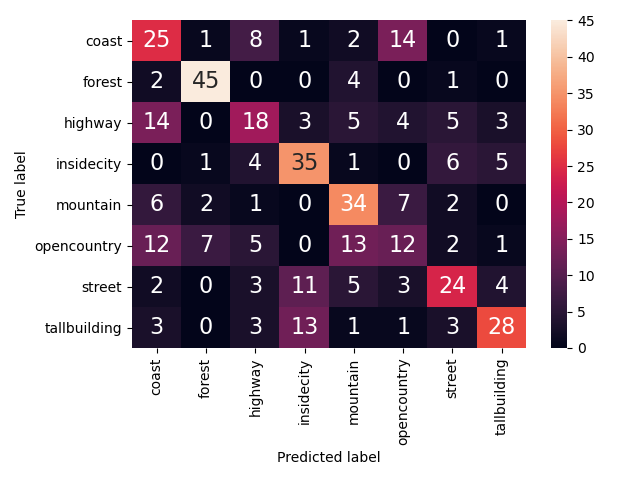
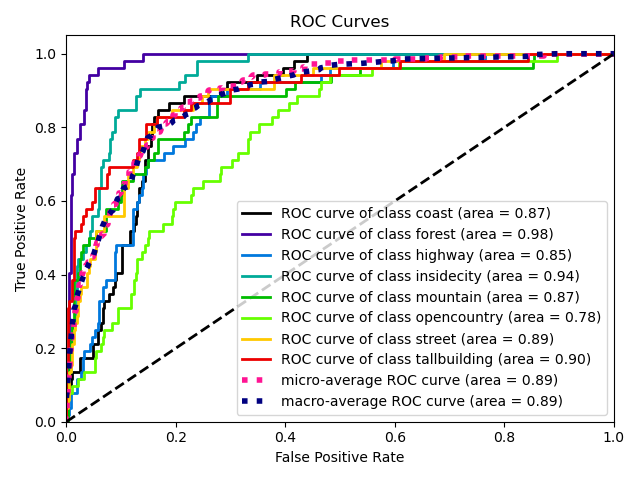
C=1



****

**C=10**

****

****

ניתן לראות כאן קורולציה בין F1-Score של קטגוריה מסוימת אל מול השטח AUC בגרף ה ROC.

לדוגמא opencountry קיבל שטח של 0.78 כאשר F1 שלו היה 0.26 לעומת Forest שקיבל שטח של 0.98 כאשר F1 היה 0.83.

זה בהחלט נראה כי הקורולציה קצת מוזרה אך לא לשכוח שגרף ROC בדרך כלל ממומש לסיווג בינארי (לא ממש הבנתי את הצורך שלו כאן ולדעתי הדרך היותר טובה לעמידת דיוק המודל כאן הייתה יכולה להיות Confusion Matrix)

בROC ננקטה הגישה של One-Vs-ALL לכן אני משער שזו הסיבה שההדיוק שם גבוה יותר.

בסוף הוספתי גם Confusion Matrix לשם תצוגה נוספת של הנתונים.

אם נסתכל על השורה של Opencountry נראה כי באמת הרבה תמונות תויגו לא נכון כ Coast ו Mountain

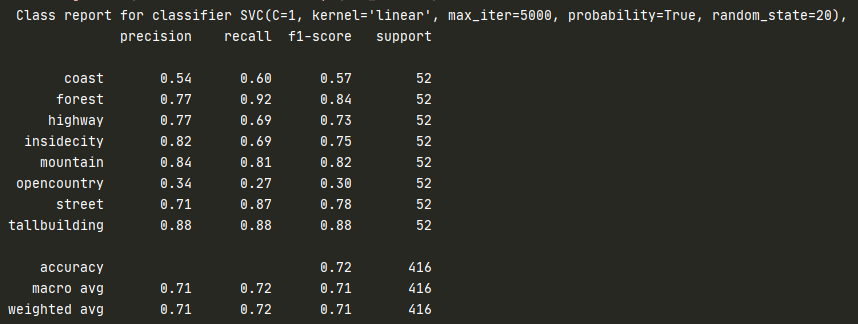
13 תמונות תויגו כ Mountain, לעומת 12 תמונות שבאמת תויגו נכונה כ Opencountry!

בעזרת טבלה זו באמת אפשר להבין את הנימוקים לאחוזי הדיוק של המודל על קטגוריות שונות!

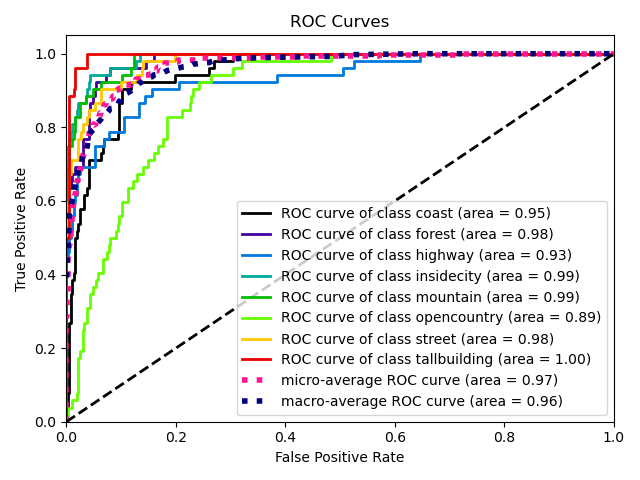
**מודל BOW VGG :**

גם כאן אנקוט אך ורק בגישה 1 ואבדוק את גרף הROC.

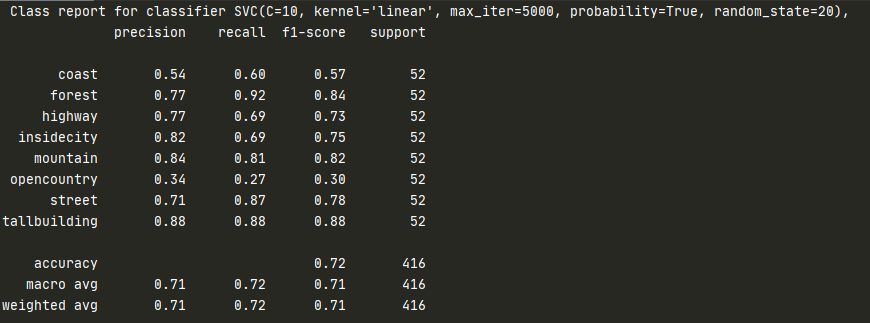
C=1

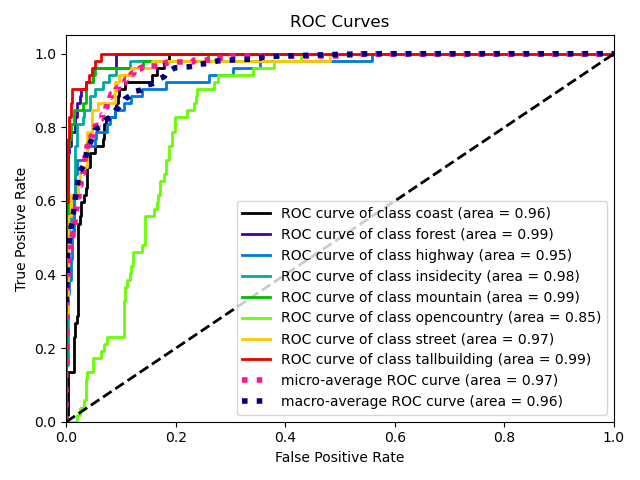


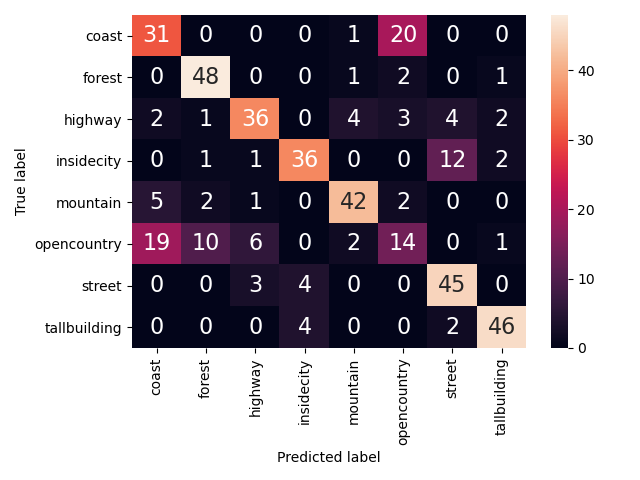
**נראה כאן כי הדיוק של המודל לא נפגע בצורה דרסטית כמו בSIFT**, יש ירידה של כ3 אחוז בדיוק.



**C=10**

****

****

****

כאן כבר ניתן לראות דיוקים גבוהיים הרבה יותר, רוב השטחים של הקטגוריות גבוהיים מאוד והנמוך ביותר הוא כמובן opencountry 0.85, כאשר הוא מקבל דיוק של f1 score 0.30.

ניתן לראות הפעם כי 19 תמונות של OpenCountry תויגו כCoast! לעומת רק 14 שתויגו נכון.

במקביל לכך 20 תמונות של Coast תויגו בOpenCountry.

ניתן להסיק מכך שהקטגוריה OpenCountry דומה לCoast לפי המודל שלנו.

הפעם ההצטברות המרכזית של התיוגים התרכזה באלכסון הראשי וזה בהחלט סימן משמח.

המסקנה המרכזית גם כאן הוא שהשימוש בVGG בהחלט מדוייק יותר מאשר שימוש בSIFT (לא פלא שתחום הלמידה עמוקה האיץ את התפתחות תחום הראייה ממוחשבת)

**לסיכום**

נראה כי גישה 1 בהחלט עדיפה כאן וכדי לשמור על איזון במספר התמונות הנלקחות לסט האימון, ברגע שלא שמרנו על איזון נראה כי המודל שלנו נטה לכיוון קטגוריות עם תמונות רבות וזה פגע באחוזי הדיוק של המודל.

* החלוקה הטובה ביותר של מסד הנתונים לשני המודלים הייתה 80-20.
* גישה 1 של חלוקה מאוזנת של קטגוריות הניבה תוצאות טובות יותר.
* מדד הרגולציה C לא השפיע יותר מדיי על הדיוק הכללי של המודל.
* המודל LinearSVC שהשתמש ברשת VGG16 הגיע ל0.75% דיוק לעומת 0.60% דיוק במודל SIFT.
* מודל ה SVC עם kernel ליניארי הגיע לדיוק נמוך יותר 0.72% לעומת 0.53% דיוק במודל SIFT .
* גרפי הROC נמצאים בקרולציה חיובית עם F1 Score של כל אחד מהקטגוריות.
* הוספתי גם טבלת Confusion Matrix אשר מסבירה לדעתי את סיווגי המודל בצורה טובה כבונוס.

היה ממן נחמד מאוד, לצערי נתקלתי בהרבה בעיות עם גרפי ROC אך בסוף הצלחתי להתגבר על הבעיות(הטכניות והרצון לעבוד עם Confusion Matrix) וליצור אותם.

😊