מבוא לראייה ממוחשבת (22928) – ממ"ן 12

מגיש: דותן אסלמן 301372975

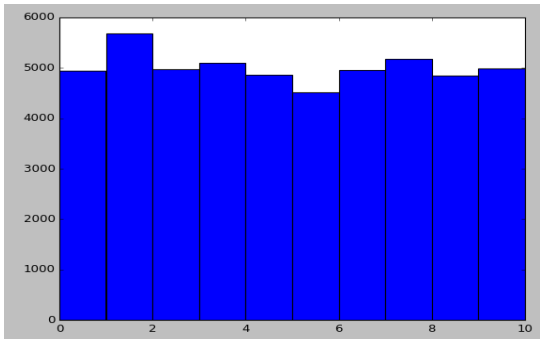
חלק 1 - MNIST

מצ"ב קוד לשאלה 1 בקובץ part1.py
יש לקרוא את הקוד החל מהערה ### START OF SCRIPT

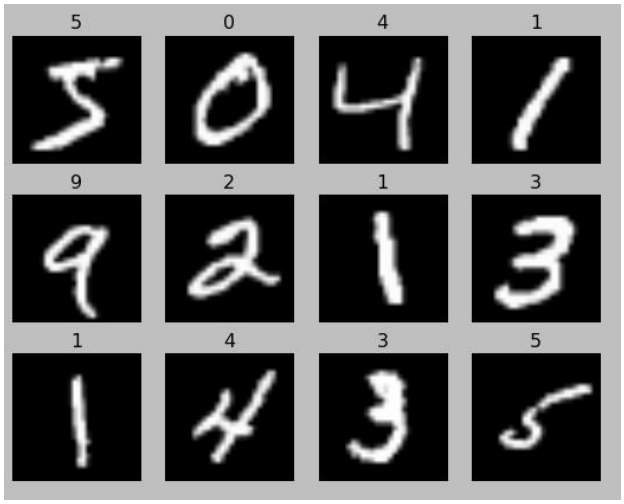
- *שאלות הקדמה:*
  - *כמה תמונות יש מכל ספרה?*

ספרה	מספר מופעים
0	4932
1	5678
2	4968
3	5101
4	4859
5	4506
6	4951
7	5175
8	4842
9	4988

או בהיסטוגרמה:

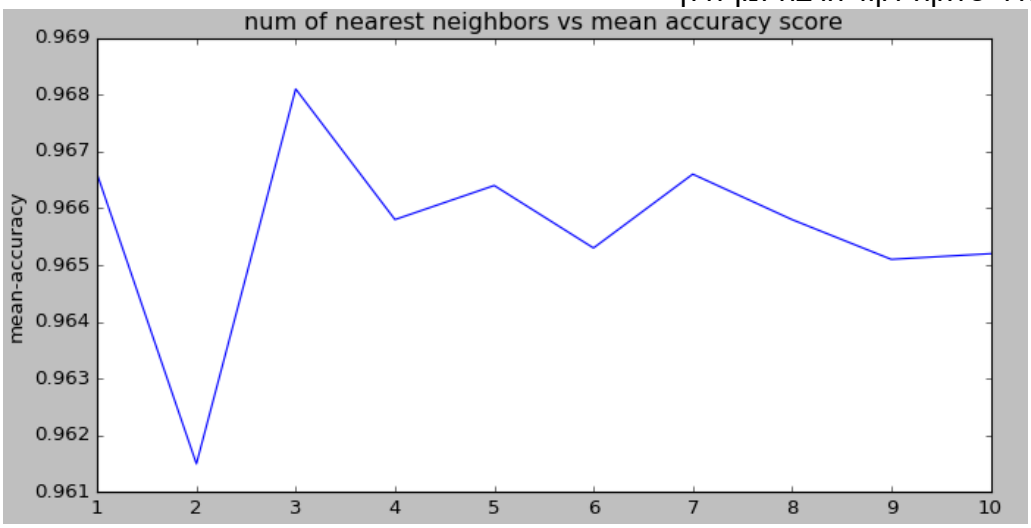


  - *הצג את 12 הספרות הראשונות מתוך ה*train *ב* figure *אחד*



a. חשב את ביצועי KNN עבור k=1..10 צייר את התוצאות על גרף:

*מזהיר שלוקח לקוד הרבה זמן לרוץ



- *מה התוצאה האופטימלית?*

בהיבטי accuracy מתקבל מהגרף ש- K=3 התוצאה הטובה ביותר

- *איך מתבצעת ההחלטה כאשר k זוגי?*

הנחתי שהכוונה בשאלה היא מקרה של תיקו בין 2 labels ולא בהכרח k זוגי, למשל גם מקרה של k=5 אבל יש בתוך 5 השכנים 2 מופעים של label אחד ו 2 של label אחר (ושלישי פעם אחת) אז יש תיקו בניהם.

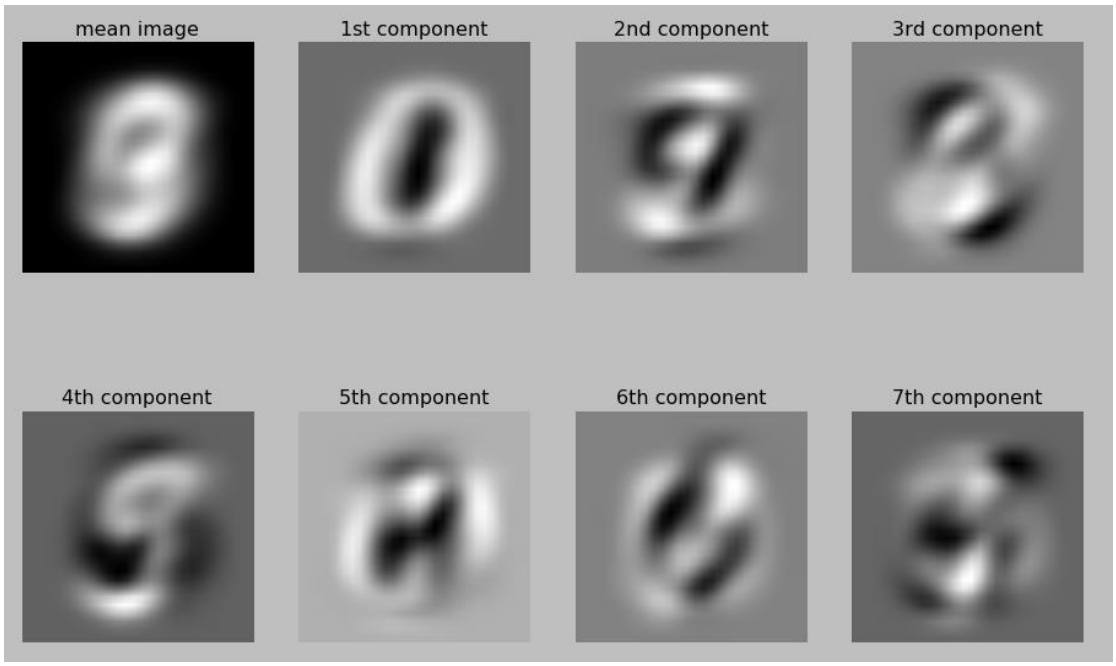
במקרה כזה ניתן להריץ את האלגוריתם עם "משקלים" שמתבססים על מרחקים וכך כל אחד מK השכנים הקרובים ביותר יקבל 'משמעות' לפי המרחק שלו ביחד לquery בהרבה מקרים זה יכול לשבור את התיקו, אך לא תמיד, ובכל מקרה במקרה סופי של תיקו תתרחש ההחלטה הבאה תוך התיעוד של sklearn :

Warning - Regarding the Nearest Neighbors algorithms, if it is found that two neighbors, neighbor k+1 and k, have identical distances but different labels, the results will depend on the ordering of the training data.

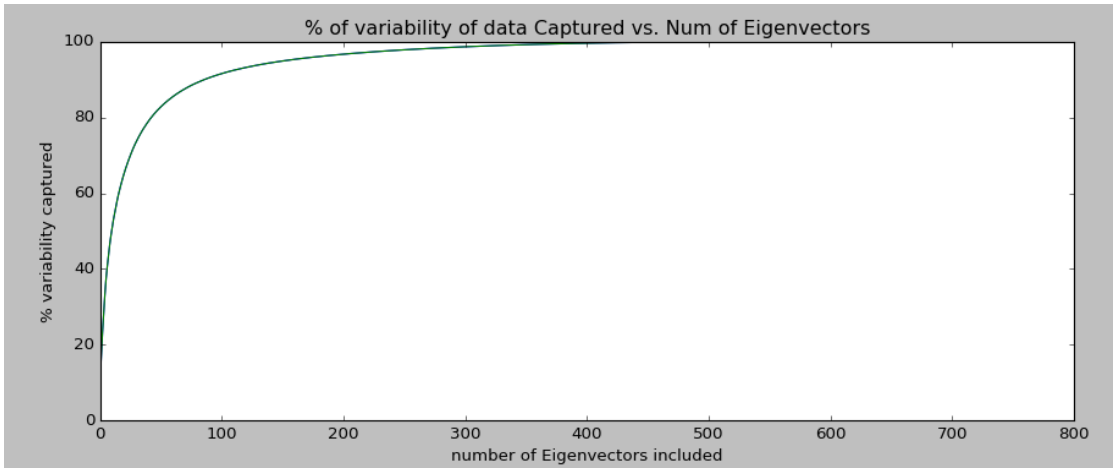
כלומר, במקרה של "תיקו" יבחר הlabel שהופיע קודם בtrain.

b. *חשב PCA על סט האימון, צייר את הספרה הממוצעת ו6 הpc הראשונים*

השתמשתי בספריית PCA של sklearn



c. *צייר גרף של הvariance הכולל כתלות בn הרכיבים הראשונים*



d. *כמה בסיסים צריך כדי להגיע ל variance 95% וכמה ל 80%*

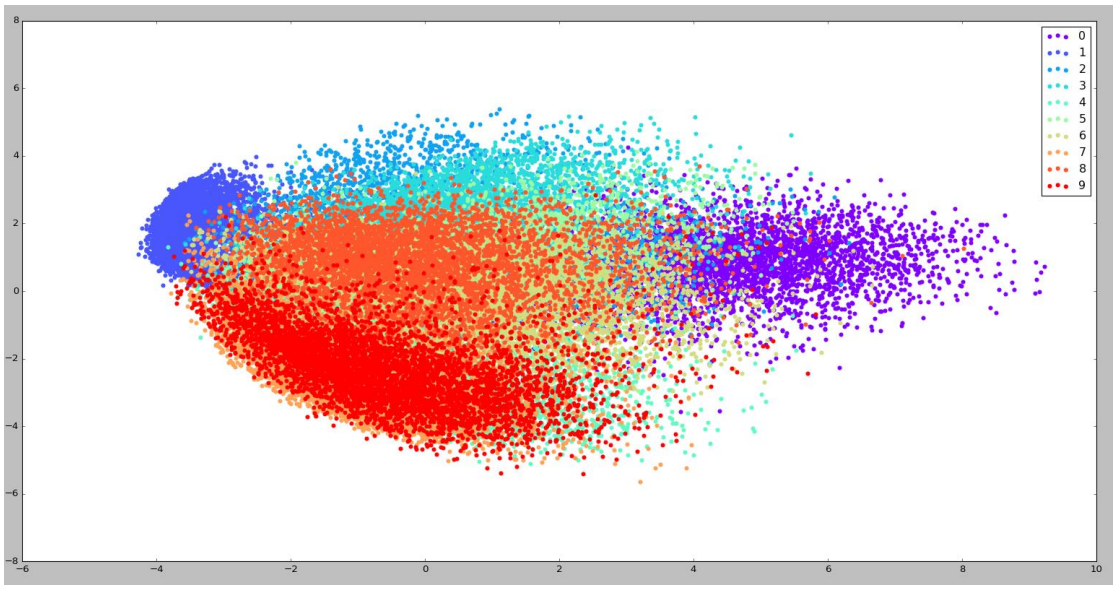
- עבור 80% צריך 43 בסיסים
- עבור 95% צריך 153 בסיסים

e. *הטל את הספרות למימד 2 וצייר את הוקטורים המתקבלים, כל ספרה בצבע אחר*

על מנת להטיל את הוקטורים למימד 2 יכולתי להגדיר PCA חדש ולקנפג אותו ל2 principle component בלבד אך זה היה גוזל לי זמן ריצה בחישוב הfit מחדש, במקום זה, חישבתי את ההטלה ע"פ הנוסחה הבאה (מתוך המצגת):

$$z' = P * (z - \mu)^T$$

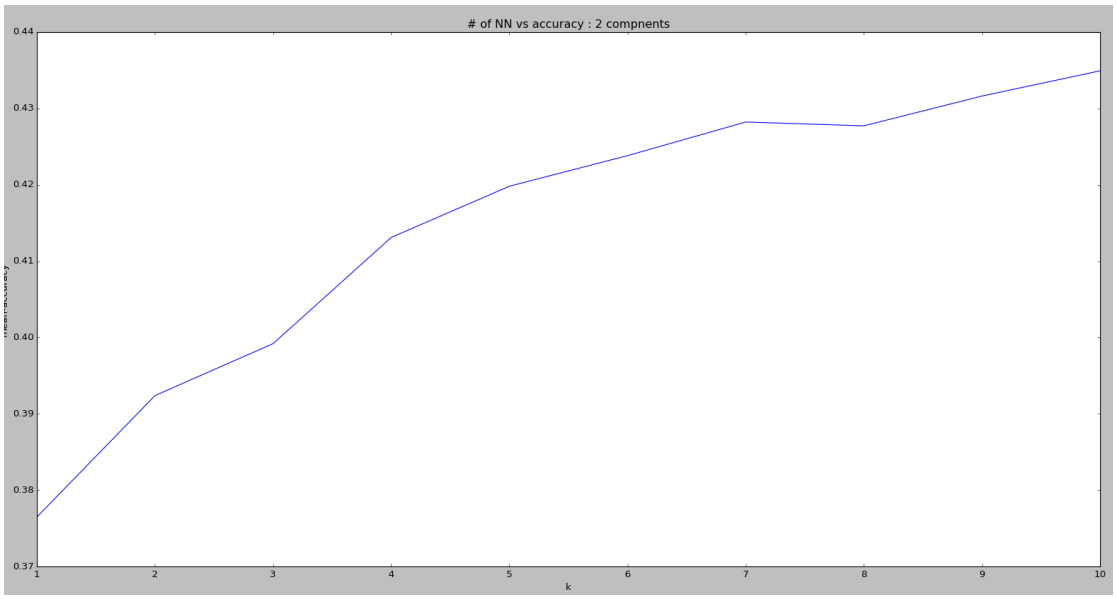
כאשר P הינו וקטור ה Principle components , z הינו וקטור התמונה וμ מסמן את וקטור התמונות הממוצע, במקרה של הטלה ל2 מימדים לקחתי רק את 2 האיברים הראשונים ב P.



f. *חזור על שאלה 1 כאשר כל ספרה מיוצגת ע"י ההטלה שלה למימד 2,10,20*

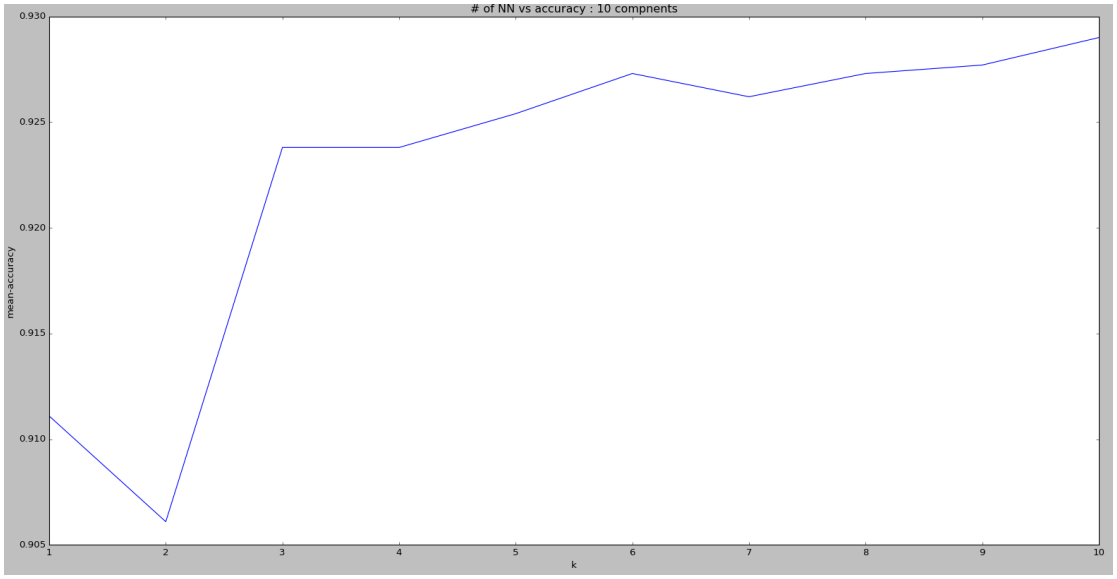
לצורך כך אני משתמש באותה פונקציה שהגדרתי בשאלה 1 (עם input אחר לאחר ההטלה) **run10Knn**

עבור 2 מימדים:



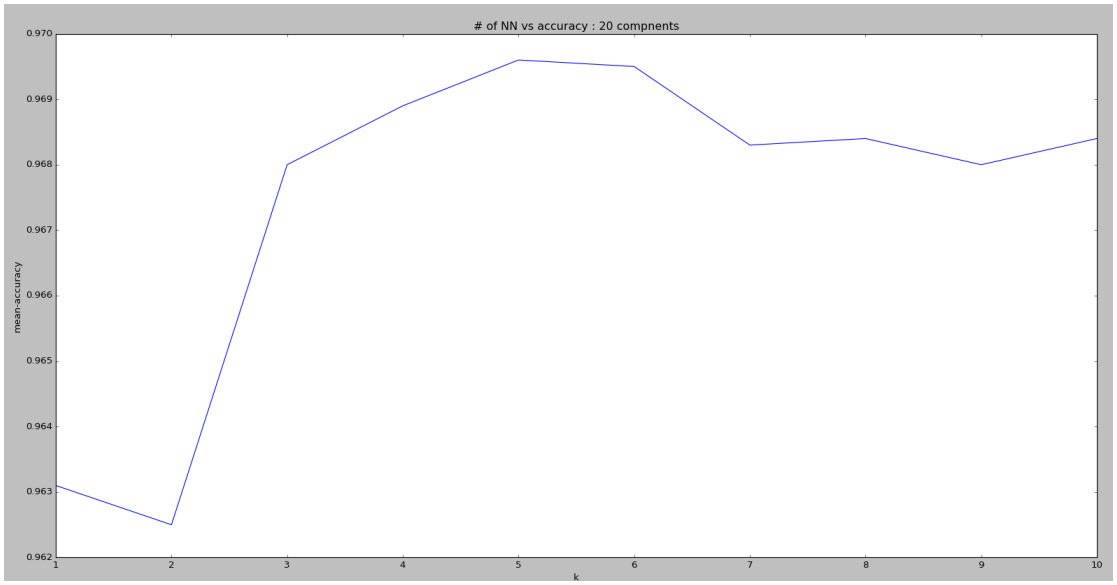
התוצאה הטובה ביותר מתקבלת כאשר k=10 ועושה רושם שהיה שווה לבדוק k אף גדול יותר.

עבור 10 מימדים:



תוצאה טובה ביותר כאשר k=10

עבור 20 מימדים:



תוצאה טובה ביותר כאשר k=5

g. *עבור ספרה כלשהי, הטל את הספרה למימד k והטל אותה בחזרה צייר את השחזור עבור k= 2,5,10,50,100,150 כתוב את הנוסחה שהשתמשת להטלה ולשחזור.*

הנוסחה שהשתמשתי בה להטלה הינה אותה נוסחה שהצגתי בסעיף e (מתוך המצגת):

$$z' = P * (z - \mu)^T$$

בפייתון זה נראה קצת שונה :

np.dot(vec- pca.mean_ , pca.components_[:n_comp].T)

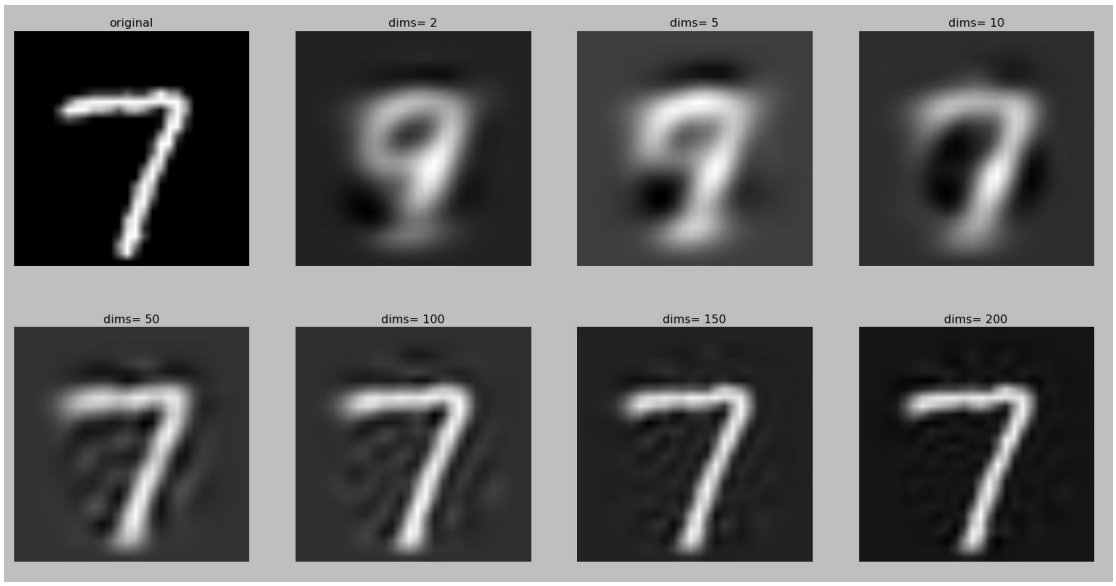
הנוסחה לשחזור הינה :

$$z = \mu + z' * P$$

בפייתון : _np.dot(vec, pca.components_[:n_comp]) + pca.mean

כאשר P - הprinciple components , z – וקטור המקור , μ הוקטור הממוצע.

הרעיון הוא שבגלל שהP הינה אורטוגונלית (נובע מהעובדה שpc הם unit vectors ובת"ל) כדי לבצע את הטרנפורמציה ההופכית שלה מספיק להשתמש בtranspose .



(הוספתי dims=200 בשביל הסימטריות)

*.h PCA מתקשה בתיאור dataset מורכב כמו במקרה שלנו. הפתירון הוא לחשב pca לכל מחלקה בנפרד. חשב מודל pca לכל ספרה בנפרד, הצג את 6 הPC הראשונים, מה ההבדל יחסית למודל של כל הספרות יחד? (**ניסוח מחודש שלי**) בנה classifier מבוסס על איכות השחזור באופן הבא: חשב מודל לכל ספרה, עבור כל ה test_set חשב הטלה לכל אחד מהמודלים ובצע שחזור מכל אחד מהמודלם, בדוק איזה מהמודלים ביצע את השחזור הטוב ביותר(ע"פ מרחק) ובחר את המודל ששיחזר באופן הטוב ביותר בתור הערך ה'נכון'. מה הביצועים של המסווג הזה?*

לצורך סעיף זה הגדרתי class בשם **PCAPredictor** בעל 2 פונקציות :fit – לתהליך ה'למידה' על הtrain data וpredict – לתהליך ההשוואה עבור הtest set.

נתחיל בלהציג את 6 ה PC הראשונים בכל מודל :

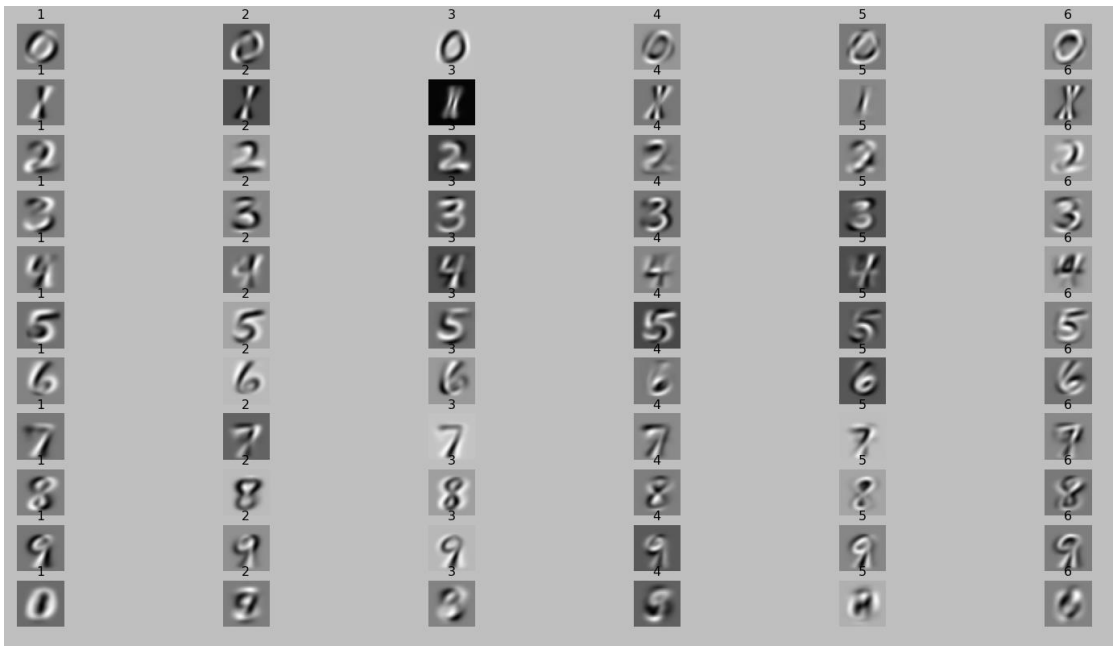


Figure 1- הרכיבים לפי סדר משמאל לימין(ראשון עד שישי)- כל שורה מבטא ספרה החל מ0 עד 9. שורה אחרונה הינה עבור PCA מאוחד לכל הtrain

לשם השוואה, צירפתי בשורה האחרונה (שורה 11) את הPC של המודל הכללי שמתייחס לכל התמונות. ניתן לראות בבירור שלעומת המודל הכללי כאשר מבצעים PCA עבור כל class בנפרד המאפיינים של כל class הרבה יותר ברורים.

את ה"מרחק" בין התמונה לשחזור חישבתי ע"י MSE.
על מנת להציג את ביצועי האלגוריתם המוצע השתמשתי בפונקציית classification_report של sklearn שהחזיר :

	precision	recall	f1-score	support
0	0.94	0.88	0.91	1047
1	0.91	0.89	0.90	1162
2	0.84	0.89	0.86	969
3	0.80	0.86	0.83	934
4	0.70	0.93	0.80	742
5	0.91	0.67	0.77	1203
6	0.79	0.95	0.86	800
7	0.73	0.90	0.81	832
8	0.72	0.78	0.75	900
9	0.87	0.62	0.73	1411
avg / total	0.83	0.82	0.82	10000

ו accuracy של 0.821

חלק 2 - סיווג ע"י BOW

מצ"ב קוד לשאלה בקובץ part2.py
יש לקרוא את הקוד החל מהערה ### START OF SCRIPT

א. **DATASET:**
בחרתי את הclasses : "opencountry" ו-"coast" . בחרתי את 2 אלה בכוונה כיוון שבאתר שמציע את הdataset הזה נראה שהם classes עם בלבול יחסית גבוה בניהם מה שעוזר לי להציג בצורה יותר אינפורמטיבית את התוצאות (כשיש מדי פעם גם שגיאות)
מתוך : http://people.csail.mit.edu/torralba/code/spatialenvelope/

ב. **PARAMETERS:**
השתמשתי בפרמטרים הבאים :

Name	Description	Default value
train_test_ratio	יחס בין כמות התמונות שהולכות לתהליך האימון לכמות התמונות שהולכות לשלב הבדיקה	0.8
dsift_step_size	גודל צעד בבחירת נק' עניין בdense sift	25
n_clusters	מספר המרכזים עבור KMEAN	120
regTerms	ערך הרגולריזציה(C) עבור ה SVM (בודקים מספר ערכים שונים)	[0.0001, 0.001, 0.01, 0.1, 1, 5, 10]
Kmeans_tol	Relative tolerance with regards to inertia to declare convergence	1e-3
Basedir	התיקייה בה נמצאות התמונות	Data

ג. **שלב טעינת המידע**
שלב טעינת המידע ממומש בפונקציה **load_data**.
בפונקציה אני טוען את התמונות מתוך תיקיית הbasedir טוען את כל התמונות וקובע את הסיווג בהתאם לשם הקובץ
לאחר מכן אני 'מערבב' את כל התמונות באמצעות הפונקציה shuffle כדי לקבל בכל הרצה חלוקה אחרת של train ו test ומפריד בניהם לפי פרמטר היחס.

מאחר ומדובר במסווג בוליאני התייחסתי לopencountry בתור הדוגמה ה'שלילית' 0 וcoast בתור הדוגמה החיובית – 1

ד. שלב האימון

כדי לבצע dense-sift עברתי על התמונה בצעדים בגודל שווה dsift_step_size ואספתי keypoints ללא התחשבות בתוכן התמונה. המימוש נמצא בפונק' calcSifts

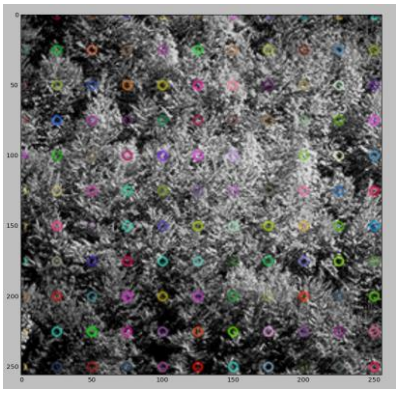


Figure 2 - תמונה כלשהי ועליה סימון של הKEYPOINTS עליהם רץ SIFT

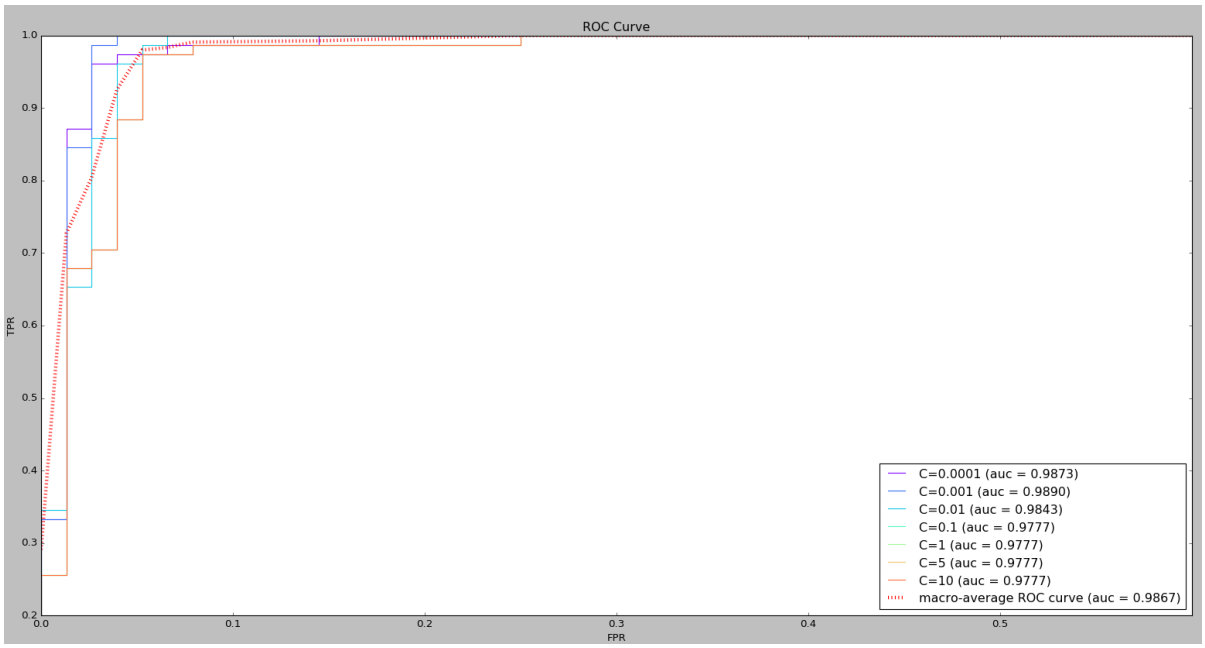
לאחר מכן חישבתי את הcodebook שלי באמצעות KMEANS, בחרתי כמות 'מילים לפי הפרמטר n_clusters. הגדרתי במיוחד n_init=1 שזה מספר הפעמים שkmeans ירוץ שוב ושוב כדי להימנע ממינימום לוקאלי לא מספיק טוב, זה אמור לפגוע בביצועים אבל הפך את זמן הריצה שלי לסביר. כמו כן ייצאתי את הפרמטר tol וגם אותו הגדלתי לערך יחסית גדול כדי לשפר את זמני הריצה על פני ביצועים.

לאחר חישוב כל ה'מילים' ביצעתי את תהליך חישוב הbow-descriptor ועבור כל תמונה מהtrain חישבתי את ההיסטוגרמה של מופעי ה'מילים' שלה מהcodebook. הפונק' לחישוב הdescriptor עבור תמונה בודדת נקראת calcBOWDescriptor

את מטריצת הדיסקריפטורים לכל תמונה יחד עם הlabels הכנסתי למסווג SVM ליניארי באופן הבא: SVC(kernel = 'linear',C=c) ויצרתי מספר מסווגים כ"א עם ערך רגולריזציה שונה (C) למטרות השוואה בהמשך.

ה. שלב הבדיקה

בשלב הבדיקה ביצעתי dense-sift באותו אופן שהוצג בסעיף הקודם. לאחר מכן חישבתי היסטוגרמת BOW עבור כל תמונה ולבסוף הבדיקה נעשתה על כל אחד ממודולי הSVM .
עבור כל מודל חישבתי את התוצאה באמצעות הפונק' decision_function שלsklearn
וחישבתי roc_curve ו auc . להלן התוצאות :

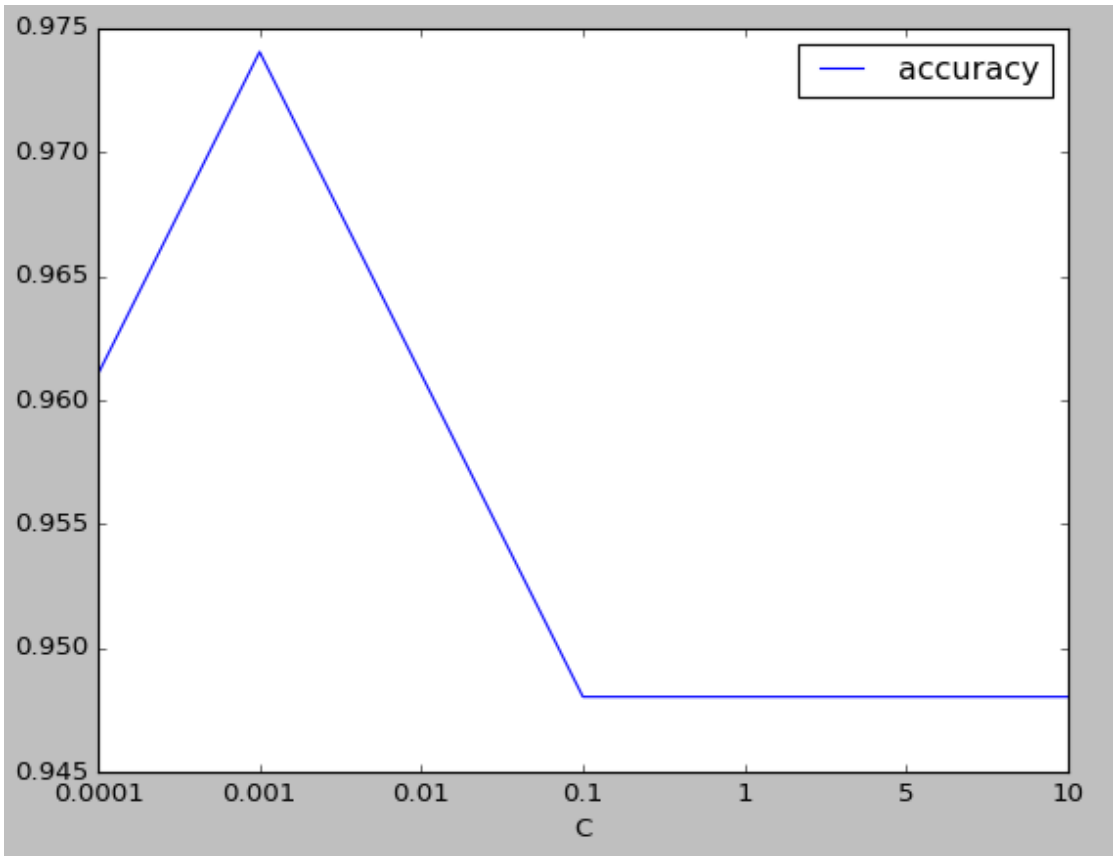


*שיחקתי קצת עם הגבולות של x ו y כדי שהגרף יראה יותר טוב (y מתחיל ב0.2 ו x נגמר ב0.6)

נראה שהתוצאות הטובות ביותר (בהיבטי AUC) הן כשC=0.001 , בנוסף – הקו המקווקו באדום הינו ה rocהממוצע (macro average)

ו. מה הפרמטרים האופטימליים של המסווג?

ביצעתי השוואה על מהו הערך C האופטימאלי ע"י חישובaccuracy-score עבור C-ים שונים:

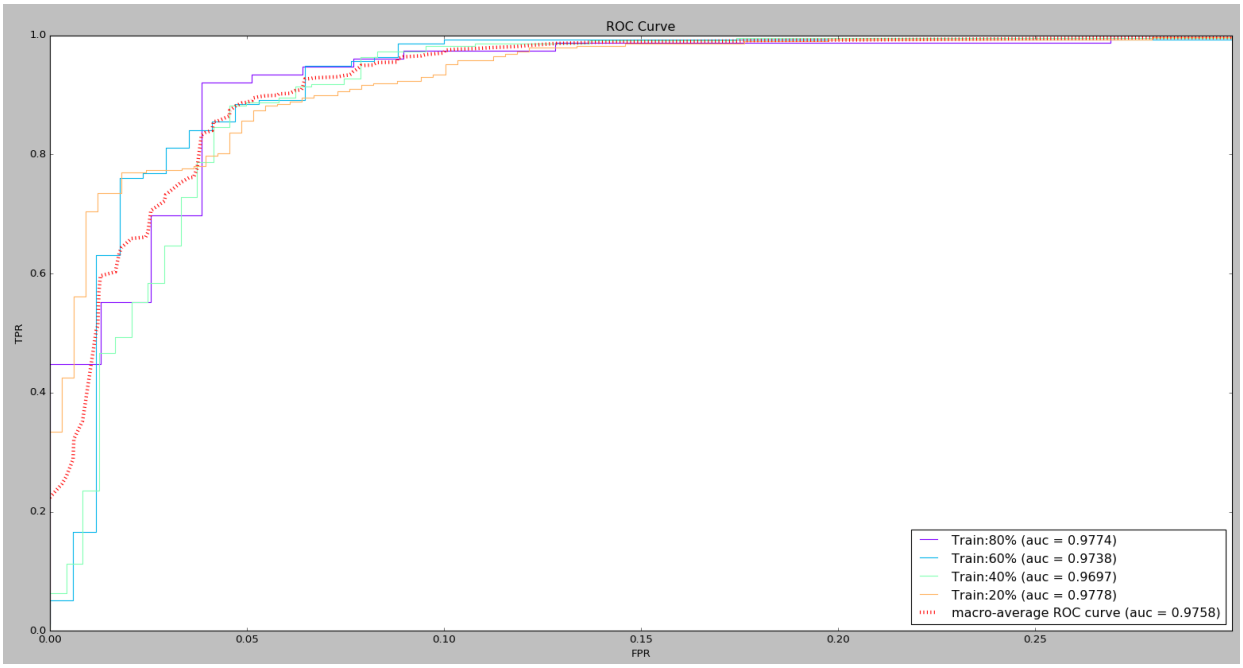


בהיבטי accuracy – הC האופטימלי הוא 0.001

ז. איך ישתנו התוצאות עבור חלוקות שונות של train ו test ? מה התוצאה הממוצעת?

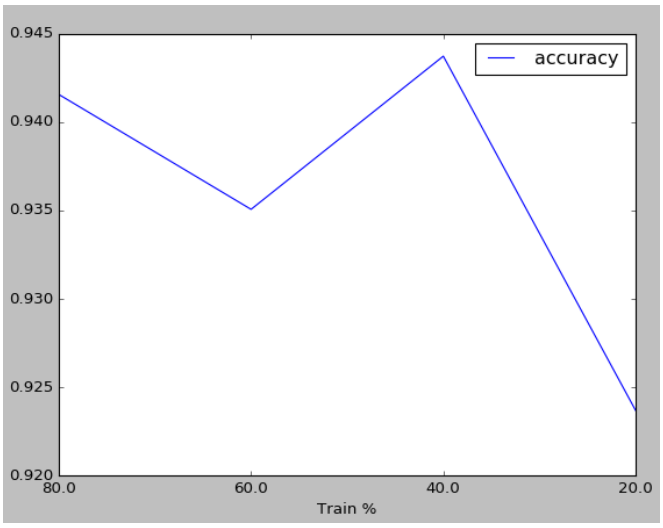
קיבעתי את הC לפרמטר האופטימלי והרצתי את האלגוריתם בחלוקות הבאות של TARIN-TEST: 0.8(המקור) , 0.6, 0.4, 0.2 וחילצתי ROC עבור כ"א מההרצות, בנוסף חישבתי MACRO-AVARAGE של התוצאות כדי לקבל את התוצאה הממוצעת.

קטע הקוד שכתבתי מעט לא יעיל מבחינת reuse של קוד באמצעות פונקציות משותפות בין 2 השלבים אבל עשיתי את זה על מנת שיהיה יעיל יותר מבחינת זמן ריצה (וככה אוכל להפעיל מחדש רק חלקים מפונקציית הTRAIN או הTEST כל פעם.



*xlim=0-0.3

כאשר הקו המקווקו באדום הינו הROC הממוצע. נתוני ה ACCURACY לכל הרצה :



נראה שבהרצה הנוכחית (כל הרצה טיפה שונה בגלל הדגימה הרנדומלית של TRAIN ו-TEST) מבחינת accuracy ההרצה הטובה ביותר היתה כשהיחס הוא 40% לTRAIN, קצת מפתיע אבל ההפרשים מאד קטנים וגודל הdataset הוא ממילא די קטן אז כל 'טעות' היא משמעותית והdataset כל פעם מוגרל מחדש אז כמובן שיכול לקרות.