**ממן 12**

**מגיש: ניצן קרני, 208939215**

**חלק 1**

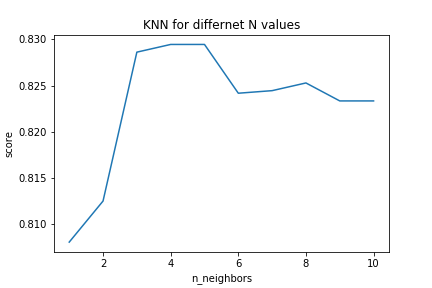
קודם כל לפני שהתחלנו לעבוד עם המידע, ייבאנו את המידע, והצגתי את 16 התמונות הראשונות בדאטה סט על גריד של

ארבע על ארבע בצבעי אפור.



שאלה 1

בחלק זה נממש מסווג מסוג KNN על מנת לסווג כל תמונות בסגמנט הבדיקה שלנו למחלקה המתאימה. חילקתי את המידע שלנו 80% אימון 20% בדיקה, לאחר מכן בדקתי 10 מודלים שונים של KNN שאומנו על אותו דאטה. עבור כל מודל ערך K מתאים מ1 ל10. כל התוצאות נכנסו לתוך מערך התוצאות והוצגו בגרף שלהלן:



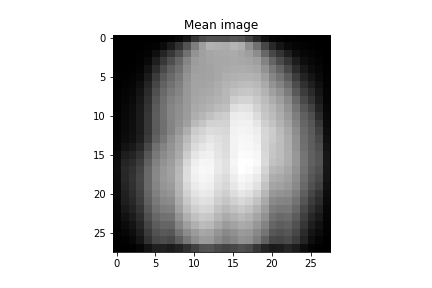
ניתן להבחין כי הערך K שנותן תוצאה אופטימלית הוא K=5

Maximum score is 0.8372222222222222, maximized by K=5

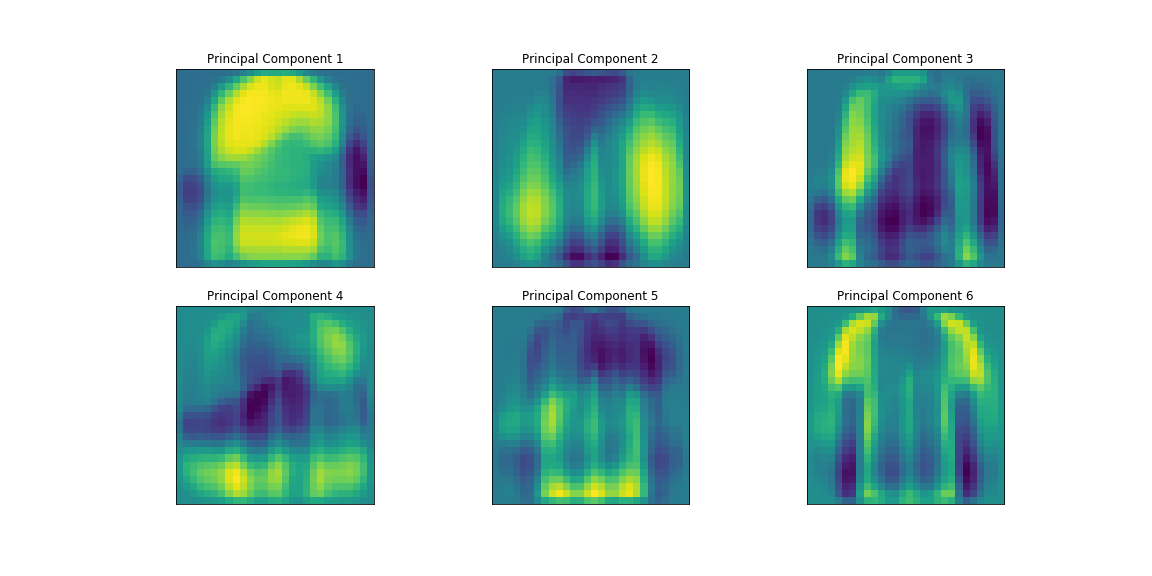
במקרה בו K זוגי, כאשר אין "תיקו" בהצבעות השכנים אז המודל יתנהג כמו באי זוגי. במידה וקיים תיקו בהצבעות, המודל יבחר את המחלקה שבתיקו של השכן הקרוב ביותר לנקודה.

שאלה 2

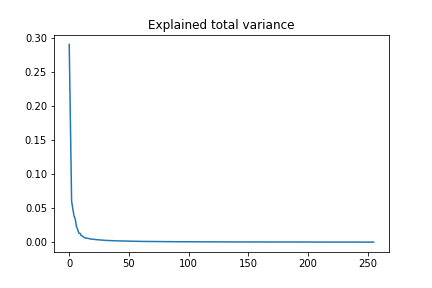
(b את טרנספורמציית PCA חישבתי על גבי 256 components, את הממוצע של התמונות הוצאתי ממודל הPCA שאימנתי על הנתונים.



אלו הם ששת הprincipal components הראשונים שחושבו מהמודל:



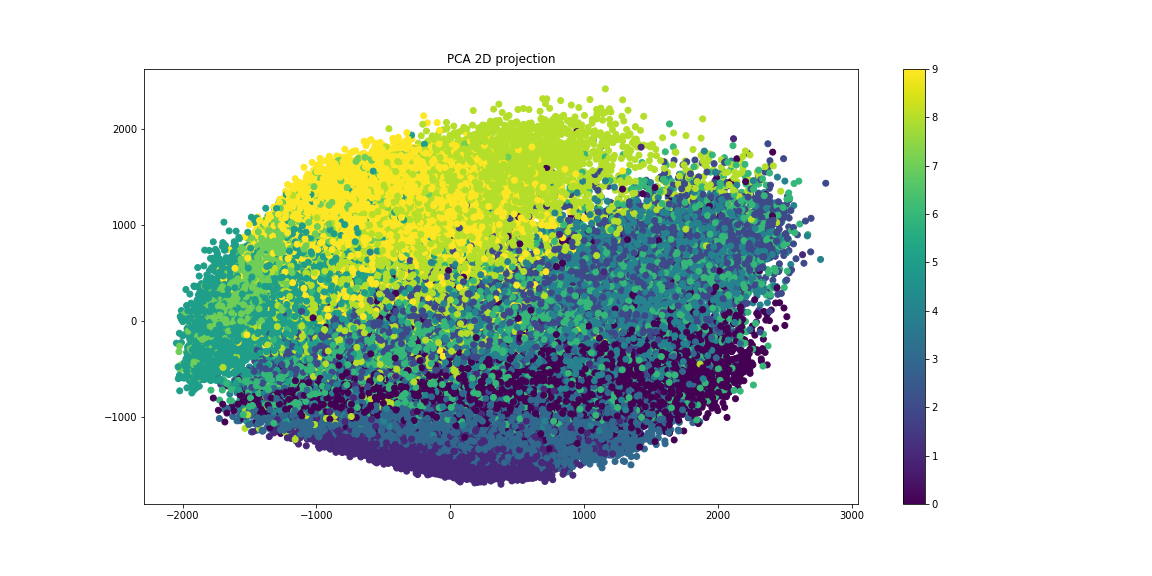
(c גרף השונות המוסברת על ידי n הרכיבים הראשיים מתואר להלן:



d) כדי להגיע ל95% שונות דרושים 189 בסיסים. על מנת להגיע ל80% שונות דרושים 25 בסיסים. חישבתי את מספר הבסיסים על פי השטח שמתחת לגרף של השונות המוסברת. כאשר עברתי את האחוז הרצוי ניתן להסתפק במספר הבסיסים שהובילו לכל אחוז רצוי.

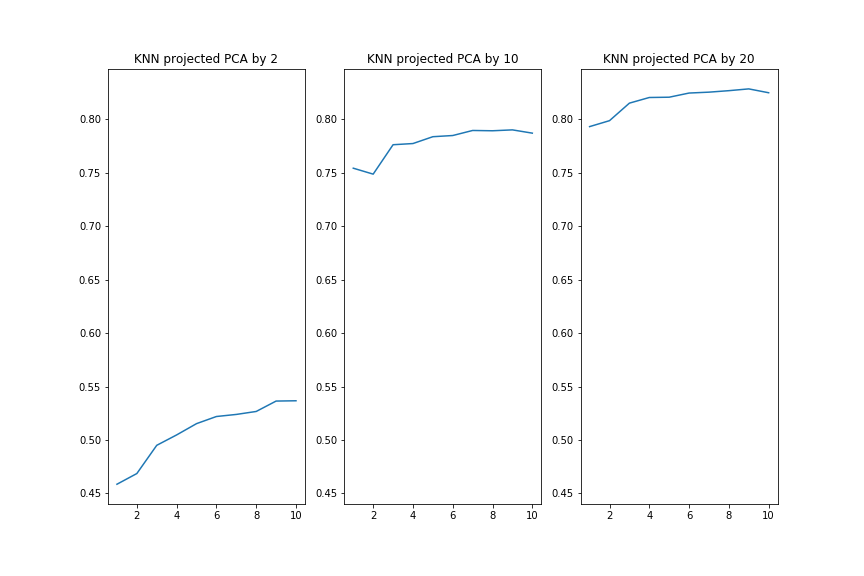
Number of bases needed to reach 80% variance: 25

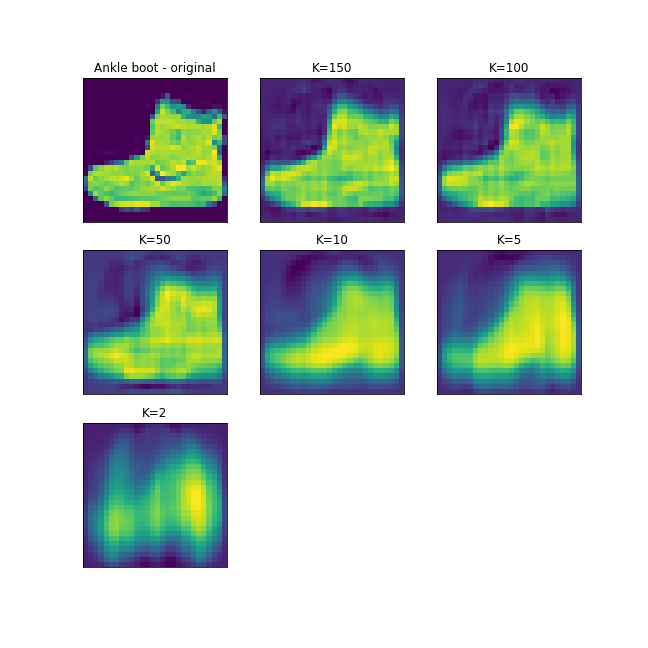
Number of bases needed to reach 95% variance: 189

e) ניצור מודל PCA עם שני רכיבים ראשיים. את המודל נפעיל על הנתונים ונטיל אותם למישור דו מימדי. לאחר מכן נציג את כל ההטלות של התמונות כגרף scatter כאשר כל מחלקה צבועה בצבע המתאים לה.

f) עבור כל נטיל את התמונות לפני האימון בKNN למימד המתאים. לאחר ההטלה ניתן להעביר את ההטלות במודל הKNN על מנת לסווגן למחלקות המתאימות. נראה שכדי להגיע לתוצאות מספקות מספיק להטיל את התמונות למימד 20 ובכך להאיץ את האימון בהרבה שכן כעת ישנם הרבה פחות features להתחשב בהם.

להלן גרף המציג את ההטלות השונות:



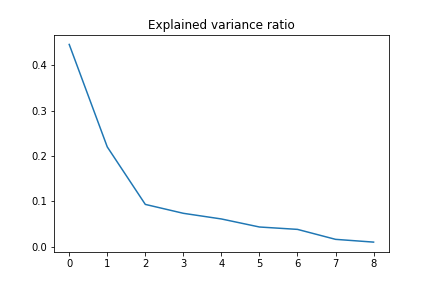
(g בחרתי את התמונה הראשונה בנתונים ( נעל קרסול ) עליה ביצעתי הטלה עבור ערכי K המתאימים ועבור כל הטלה לאחר מכן חישבתי את התמונה המקורית באמצעות פונקציית inverse transform, התמונות שחושבות מוצגות כאן להלן:

שאלה 3

c) יצרתי מודל מסוג LDA באמצעות המימוש של scikit-learn LinearDiscriminantAnalysis. הפעלתי אותו עבור 9 רכיבים.

אימנתי אותו על התמונות והמחלקות.

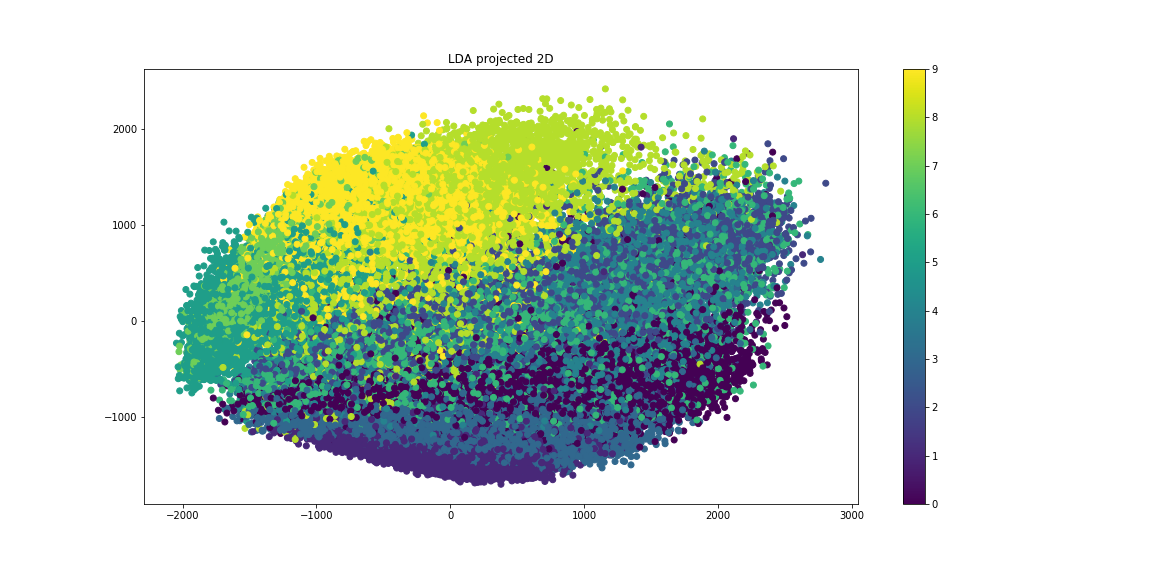
להלן מוצג גרף השונות המוסברת לאורך הבסיסים:



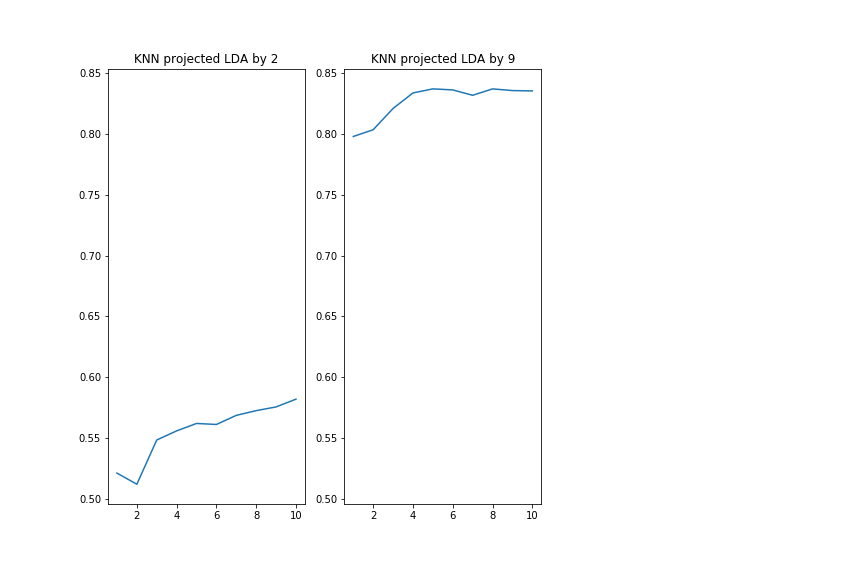
d)החישוב התבצע כמו בשאלה הקודמת. עבור 95% שונות דרושים 8 בסיסים. עבור 80% שונות דרושים 5 בסיסים.

Number of bases needed to reach 80% variance: 5

Number of bases needed to reach 95% variance: 8

e) כמו בשאלה הקודמת רק עבור המודל החדש של LDA. להלן ההטלה

F) מאחר ומדובר בLDA לא ניתן להטיל למימדים הגדולים או שווים למספר המחלקות. לכן בשאלה זו נשתמש בערכים 2=K וב9=K. את השאר נחשב בדומה לשאלה 2. ניתן להבחין כי כאן הביצועים הרבה יותר טובים משל PCA ואפילו עבור הטלה למימד נמוך יותר מה שמשפר את זמן הריצה.



**חלק 2**

שאלה 1

\*את המחלקות של התמונות שלפתי משמות התמונות לפני סימן \_

1. על מנת להשתמש בdenseSIFT כדי לחלץ את התכונות של כל תמונה בניתי גריד בגודל צפוף מספיק כדי לתפוס את כל התכונות האפשריות בתמונה. גודל כל צעד בגריד יהיה חזקה שמינית של שורש שלישי של אורך התמונה בפיקסלים. (ניסוי וטעייה ). עבור כל תמונה מהנתונים נחשב את המאפיינים והמתארים שלה באמצעות SIFT כאשר נוסיף לו כמסכה את הגריד שיצרנו. חישוב הSIFT נעשה באמצעות הפונקציה המובנית של OpenCV2.

את כל המתארים אספתי למטריצה בגודל של מערך התמונות על 1024x128 כאשר 1024 זה מספר המתארים של כל תמונה ו128 הוא הוקטור של אותו מתאר.

1. בכדי להזין את אלגוריתם kmeans שיחשב את המרכזים השונים למתארים אני צריך לפרוס את המתארים על גבי מטריצה דו מימדית. את מטריצת המתארים נפרוש כך שבכל שורה יופיע מתאר אחד בלבד, את וקטור המטרה כדי שנוכל להעביר אותו במודל נתאים לו מספרים לכל מחלקה לאחר מכן נתאים אותו כך שלכל מתאר תותאם לו המטרה שלו. חילקתי את הנתונים ל80% אימון ו20% בדיקה. יצרתי מודל kmeans עם 128 מקבצים ואימנתי אותו על סט האימון.
2. עבור כל תמונה במטריצה של המתארים ( לפני הפרישה ) נמצא את הקלאסטרים של המתארים שלה. ניצור היסטוגרמה של הקלאסטרים עבור כל תמונה ונוסיף אותה למערך ההיסטוגרמות שלנו. כל היסטוגרמה כזאת בעצם מייצגת את "שק המילים" של אותה תמונה והיא מאפיינת את התמונה.
3. כדי שנוכל לחשב את השטח מתחת לעקומת ROC נאלץ לממש את הSVM באמצעות גישת OneVsRest לצורך כך נצטרך להעביר את וקטור המחלקות ( target ) בינריזציה. כלומר עבור כל ערך בוקטור תיווצר שורה באורך מספר המחלקות כך שהסיבית במקום בשורה שמסמל את מספר המחלקה תהינה דלוקה והשאר כבויות.

כדי למצוא את הפרמטרים האופטימליים בדקתי עבור החלוקות הבאות של המידע: 90%, 80%, 70%, 60%

עבור כל חלוקה נפצל את המידע לסט אימון וסט בדיקה. נבצע נורמליזציה לנתונים באמצעות StandardScaler של scikitlearn זאת כדי להקל על החישוב של הנתונים בSVM ולא לתת משקל יתר לנתונים בעלי ערכים גבוהים.

לאחר מכן נבדוק את הפרמטרים של C : 0.001, 0.02, 1, 100, 1000

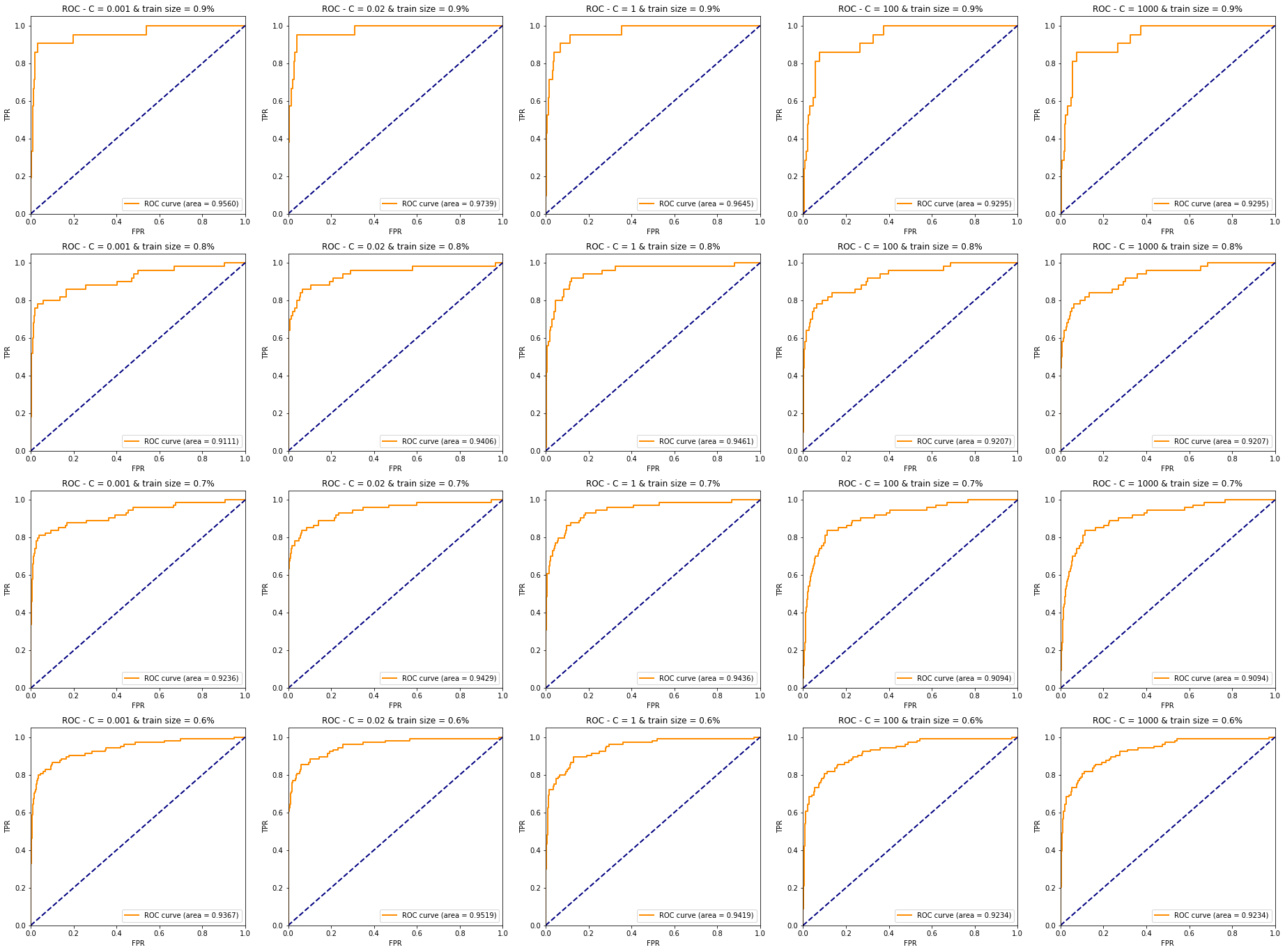
עבור כל פרמטר ניצור מודל SVM לינארי עם C מתאים, נגדיר max\_iter להיות 1000 כדי שנוכל לסיים את הריצה בזמן סביר. נאמן את המודל על המידע שלנו.

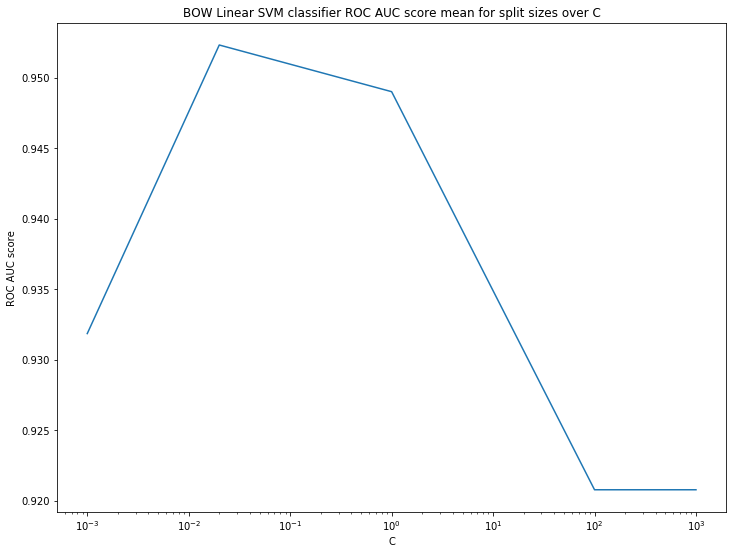
**שלב הבדיקה**

את סעיפים 1,2,3 ביצענו בפעמים הקודמות. כעת נותר להשתמש בסט הבדיקה שלנו.

נסווג את סט הבדיקה שלנו למחלקות המתאימות. לאחר מכן נחשב את עקומות הROC עבור כל אחד מהפיצולים ופרמטרי C

בנוסף עבור כל עקומה נחשב את השטח שמתחת לעקומה.





הפרמטרים האופטימליים עבור המסווג הם 1=C

התוצאה הממוצעת היא: 0.9278

80%/20% split, The C parameter that maximize the roc auc score is: 1.0

80%/20% split, The mean of ROC AUC score for all the different C parameters is: 0.927844262295082

שאלה 2

ניצור את רשת VGG באמצעות הקוד המופיע בממן.

נחשב את המתארים של כל תמונה באמצעות העברה של מטריצת התמונות בפעולת הכנת הנתונים ולאחר מכן נעביר את התמונות ברשת על מנת להשיג את המתארים שלהם.

לכל תמונה נוצרו 64 מתארים, נפרוש את כל המתארים במטריצה דו מימדית. כמו בשאלה 1 נתאים את וקטור הtarget לעבודה בmulticlass.

מכאן התהליך זהה לחלוטין לתהליך שעברנו בשאלה 1.

הפרמטרים האופטימליים עבור המסווג הם 0.02 = C

התוצאה הממוצעת היא: 0.9915

קל לראות שהביצועים הפעם טובים בהרבה מביצועי מתארי הSIFT.

80%/20% split, The C parameter that maximize the roc auc score is: 0.02

80%/20% split, The mean of ROC AUC score for all the different C parameters is: 0.9915573770491803

