תרגיל בית 3 מבוא למערכות לומדות

דוח עבודה

מגישים:

אביב כספי – 311136691

יקיר יהודה - 205710528

<u>שלבי עבודה:</u>

1. **Pre-processing:**

דבר ראשון שעשינו היה ביצוע עיבוד ראשוני לדאטה שלנו, לאחר שקיבלנו את הפיצ'רים המתאימים.
מימוש חלק זה נמצא בקובץ preprocessing.py.

א. המרה לone-hot של הפיצ'רים הנומינליים ללא חשיבות לסדר :
מתוך הפיצ'רים שנבחרו היה עלינו לשנות רק את 'Most_Important_Issue' ל-one-hot.

ב. המרת כל הפיצ'רים הקטגוריאליים למספריים:
על מנת שנוכל לעבוד על הדאטה, עלינו לשנות את כל הפיצ'רים שיהיו מספריים.

ג. תיקון ערכים שליליים בפיצ'רים:
בדומה לתרגיל קודם, גם כעת ישנו פיצ'ר בעל ערכים שליליים למרות שלא ייתכן כי הערכים של הפיצ'ר יהיו שליליים.
בצורה דומה לתרגיל הקודם, התמודדנו עם בעיה זו על ידי ביצוע abs על הפיצ'ר הנ"ל. (כמו בתרגיל קודם, זיהינו כי התפלגות הדאטה השלילי מתאימה בדיוק להתפלגות הפיצ'ר בערכים החיובים, לכן החלטנו כי ייתכן כי בזמן הדגימה שונה הסימן בטעות, לכן החלטנו רק לבצע ערך מוחלט על הדאטה הנ"ל)

ד. לאחר מכן ביצענו פיצול לדאטה שלנו:
בדומה לתרגיל הקודם בחרנו את הפיצול הבא – train(70%), val(15%), test(15%)

ה. מחיקת outliers:
בשלב זה מצאנו את הדוגמאות בהן ערך הפיצ'ר במרחק גדול מ4.5 סטיות תקן מהממוצע של הפיצ'ר, והצבנו NAN במקומות אלה.
פעולה זו ביצענו רק על סט האימון, והשתמשנו בלייבלים של הדאטה על מנת למצוא outliers יחסית לליבל.

ו. Imputation :
השלמת ערכים חסרים ביצענו בנפרד לסט האימון ולסט הולידציה והבדיקה -
עבור סט האימון בחרנו בשיטה הנקראת Bootstraping שלמדנו בכיתה.
בשיטה זו, עבור כל ערך חסר, השלמנו על ידי כך שדגמנו מתוך הדאטה שלנו עם חשיבות ללייבל, ערך חדש.
עבור סט הולידציה והבדיה בחרנו לטפל בצורה שונה.
תחילה לקחנו את סט האימון, ועבור כל פיצ'ר חישבנו את ממוצע הפיצ'ר עבור ערכים נומריים ואת הערך הנפוץ ביותר עבור ערכים נומינליים (ללא חשיבות ללייבל).
לאחר מכן, השלמנו את סט הולידציה והבדיקה בעזרת הערכים שחושבו מסט האימון.

ז. Scaling:
עבור פיצ'רים יוניפורמים ביצענו scaling לטווח [1 1-]
עבור פיצ'רים נורמלים ביצענו נורמליזציה עם ממוצע 0 וסטיית תקן 1.

2. **מציאת ערכי hyperparameters עבור כל מודל:**
מימוש חלקים 2-4 נמצא בקובץ main.py.

השלב הבא היה לאמן מודלים על סלט האימון ולבחור את המודל הטוב ביותר, על מנת לבצע זאת תחילה חיפשנו עבור כל מודל את ההיפר-פרמטרים הטובים ביותר עבורו על ידי ביצוע cross validation .
מודלים שבחרנו לבדוק הינם:
RandomForestClassifier, KNN, SVC, DecisionTreeClasifier, GaussianNB

בחרנו להתמקד במודלים שלמדנו בכיתה וכאלה שהביאו ביצועים טובים בתרגיל בית הקודם.
בנוסף רצינו להוסיף מודל לינארי ומודל לא לינארי על מנת לכסות את האפשרויות של פילוג הדאטה.
היפר הפרמטרים הטובים ביותר עבור כל מודל שמצאנו הינם:

RandomForestClassifier: n_estimators: 220, min_samples_split: 10
KNeighborsClassifier: n_neighbors: 3
SVC: kernel: 'rbf'
DecisionTreeClassifier: min_samples_split: 4
GaussianNB:

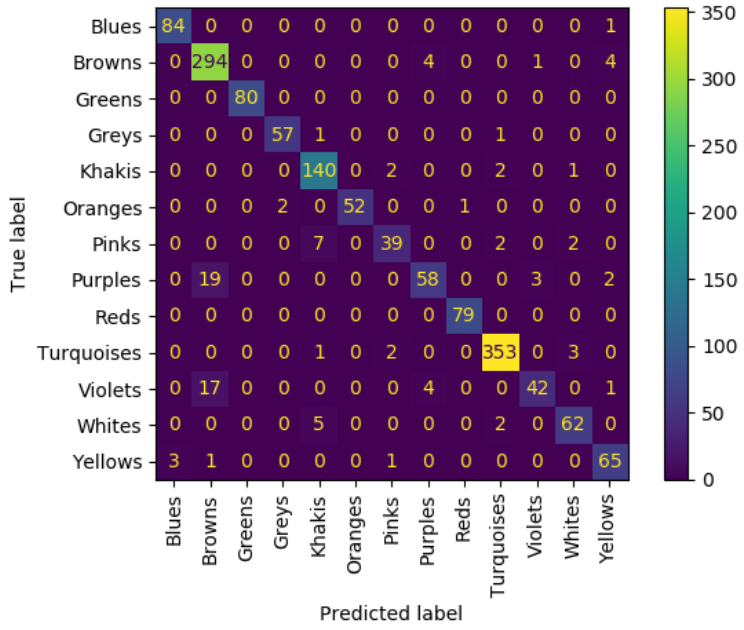
3. **אימון המודלים והערכת ביצועים:**

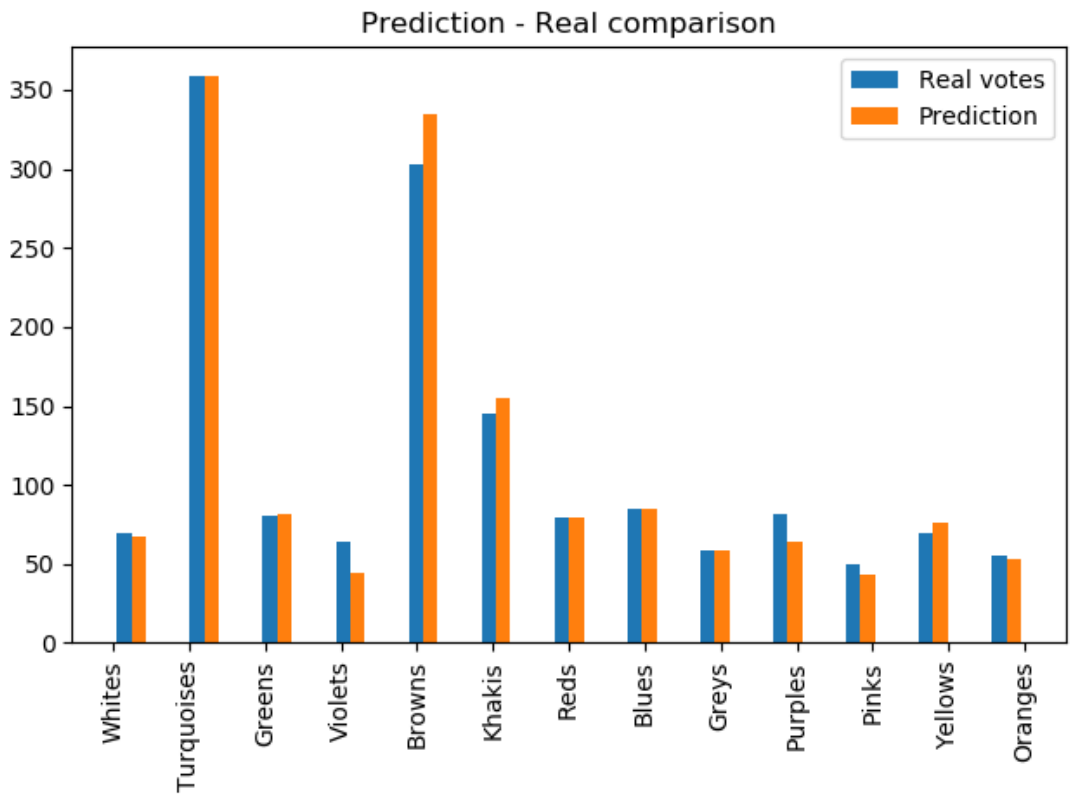
בשלב זה ביצענו אימון לכל אחד מהמודלים שציינו, עם ההיפר פרמטרים שנמצאו בסעיף הקודם.
כל מודל אימנו על סט האימון, וביצענו הערכת ביצועים על סט הולידציה לפי מספר מדדים. להוסיף איזה מדדים
לבסוף הסתכלנו על כל התוצאות שקיבלנו ובחרנו בצורה ידנית את המודל המוצלח ביותר.(בהמשך בבונוס הראשון ממימשנו פונקציה המבצעת זאת בצורה אוטומטית על ידי שקלול התוצאות)
המודל שקיבל את הביצועים הטובים ביותר היה המודל RandomForestClassifier עם הפרמטרים הבאים : n_estimators: 220, min_samples_split: 10

4. **אימון מודל נבחר וחיזוי המשימות:**

לאחר בחירת המודל המוצלח ביותר, ביצענו אימון שלו בעזרת סט האימון והולידציה יחדיו (בחרנו להשתמש בסט הולידציה לאימון מפני שכעת אין צורך לבצע ולידציה נוספת).
כעת עברנו לביצוע משימות החיזוי:

א. חיזוי הצבעה של כל אדם בסט הבדיקה, והצגת confusion matrix:





ניתן לראות בגרף את ההבדל בין חיזוי המודל שלנו לבין הערכים האמיתיים של ההצבעות.

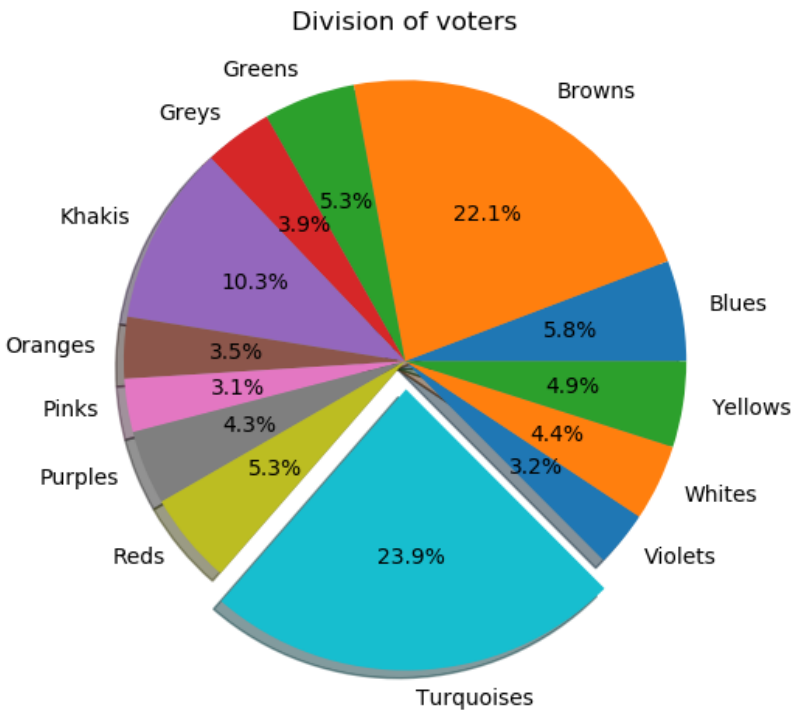
אחוז שגיאה כולל של המודל:

Test Error : 6.5%

Train Error : 1.18%

ב. חיזוי המפלגה המנצחת לפי סט הבדיקה:**Turquoises**

ג. חיזוי פילוג ההצבעה לכל מפלגה:



ניתן לראות בגרף את פילוג המצביעים לכל מפלגה בסט הבדיקה.
בנוסף הינה הנתונים שיצרו את הגרף :
Blues': 5.8, 'Browns': 22.2, 'Greens': 5.267, 'Greys': 4.0, 'Khakis': 10.333, '
'Oranges': 3.533, 'Pinks': 3.0, 'Purples': 4.2, 'Reds': 5.267, 'Turquoises':
23.933, 'Violets': 3.2, 'Whites': 4.333, 'Yellows': 4.933

ניתן לראות כי טורקיז ניצח עם כמעט 24% מהקולות.

ד. רשימת מצביעים עבור כל מפלגה בשביל שירותי ההסעה:
על מנת ליצור את רשימת המצביעים לכל מפלגה, החלטנו להשתמש בפוקנציה Predict_proba של המסווג RandomForest , על מנת לקבל עבור כל מצביע את ההסתברות שיצביע לכל אחת מהמפלגות.
לאחר מכן הפעלנו פעולת סף עם ערך 0.6 וקבענו כי כל המצביעים בעלי הסתברות הצבעה למפלגה מסויימת מעל 0.6 נכניס אותם לרשימת ההסעות (מפני שיש סיכוי גדול מאד שיבחרו במפלגה).
את רשימת המצביעים שהרכבנו לכל מפלגה ניתן למצוא בקובץ indices_list.csv
כאשר עבור כל מפלגה קיימת שורה בקובץ המכילה את האינדקסים של המצביעים לאותה מפלגה בסט הבדיקה.

בונוס ראשון:

A. חלק זה ממומש בקובץ Auto_model_selection

התבקשנו לממש בחירת מודל אוטומטי, ביצענו חלק זה על ידי פונקציה בקובץ הנל אשר לפי הדאטה ששמור לנו (train and validation) ובחירת כמה אלגוריתמי סיווג, לכל אלגוריתם אנו נותנים טווח של פרמטרים בהתאם לאלגוריתם, יש פונקציה אשר מחפשת לכל אלגוריתם את הפרמטרים הטובים ביותר (נמדד על ידי cross_val_score) ,לאחר מכן בודקים לכל האלגוריתמים לפי הפרמטרים בהתאם מי מקבל את הציון הגבוה ביותר על סט הוולידציה, בודקים על יד מספר פונקציות ציון למשל accuracy_score ועוד . לכל פונקציית ציון יש משקל אשר מי שינצח בציון על פונקציה זו יצבור את המשקל והאלגוריתם שיצבור הכי הרבה משקל ינצח ויוחזר על ידי הפונקציה.

B. חלק זה ממומש בקובץ one_size_doesnt_fit_all.py

בסעיף זה נתבקשנו למצוא את המודלים הכי טובים עבור כל משימת חיזוי ספציפית.
על מנת לבצע זאת, יצרנו 3 פונקציות ניקוד חדשות, אחת לכל משימה, כאשר פונקציה ניקוד הינה פונקציה אשר מקבלת את המסווג, דאטה ולייבלים ולדאטה:
Party winner prediction: עבור משימה זו בחרנו לממש פונקציה, אשר מחשבת את המפלגה המנצחת האמיתית (לפי הלייבלים שהתקבלו), לאחר מכן מבצעת חיזוי להצבעות של הדאטה שקיבלנו, ומחשבת את התפלגות ההצבעות לכל מפלגה.
כעת, לאחר שיש לנו את תוצאות הפילוג והערך האמיתי, החזרנו את אינדקס המפלגה שניצחה ברשימת פילוג ההצבעות, כאשר רשימה זו ממוינת מקטן לגדול לפי מספר המצביעים, כלומר ככל שיש יותר מצביעים האינדקס יותר גדול (כאשר אם המפלגה ניצחה האינדקס שלה יהיה 12).
כלומר אנחנו נותנים ציון לפי המיקום של המפלגה המנצחת בחיזוי.

Division of voters: עבור משימה זו בחרנו לממש פונקציה, אשר מחשבת מרחק אבסולוטי בין פילוג ההצבעות האמיתי לבין הפילוג שחזינו.
כלומר תחילה הפונקציה מחשבת את ההתפלגות האמיתית של הדאטה (מספר מצביעים לכל מפלגה), לאחר מכן מתבצע חיזוי הצבעה לכל מצביע בדאטה, וחישוב חדש של חיזוי פילוג ההצבעה לכל מפלגה. לבסוף עבור כל מפלגה מחושב המרחק האבסולוטי בין מספר המצביעים האמיתי לבין מספר המצביעים החזוי.
הפונקציה מחזירה את מספר הדוגמאות (מרחק מקסימלי) פחות סכום המרחקים שחושבו (על מנת שערך גבוה יינתן לחיזוי קרוב יותר למציאות – מרחק מינימלי)

Transportation services: עבור משימה זו בחרנו לממש פונקציה, אשר מחשבת את דיוק המסווג.
החלטנו כי ככל שדיוק המסווג גדול יותר כך המצביעים אשר נתן כרשימה לשירותי הסעה יהיו מדויקים יותר, בנוסף נציין כי התעלמנו ממסווגים אשר לא מאפשרים לקבל הסתברות החלטה לפי הפונקציה predict_proba .
הפונקציה מבצעת חיזוי הצבעה לכל מצביע בדאטה ומחזירה את accuracy_score של המסווג.
דיוק גבוה יותר מקבל ציון גבוה יותר.

לאחר יצירת פונקציות אלה, ביצענו בחירת פרמטרים ואימון ובחירת מודל אופטימלי כמו שביצענו במשימת חובה, אך כעת עבור כל משימה השתמשנו בפונקציית הניקוד המתאימה.
(נזכיר, כי בזמן מציאת פרמטרים ואימון ובחירת מודל, ביצענו השוואה לפי פונקציית ניקוד שהתקבלה בפונקציה, כלומר גם במקרה שלנו נמצא את הפרמטרים האופטימליים והמודל האופטימלי עבור המשימה הנוכחית)

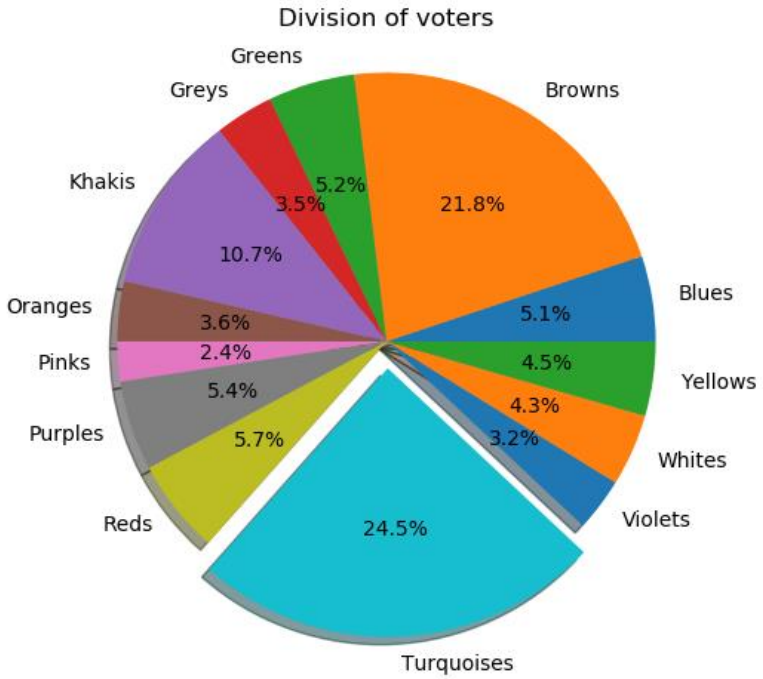
תוצאות:
Party winner prediction : המסווג המנצח הינו RandomForestClassifier עם הפרמטרים n_estimators': 60, 'min_samples_split': 2'

Division of voters: המסווג המנצח הינו KNeighborsClassifier עם הפרמטר :'n_neighbors'

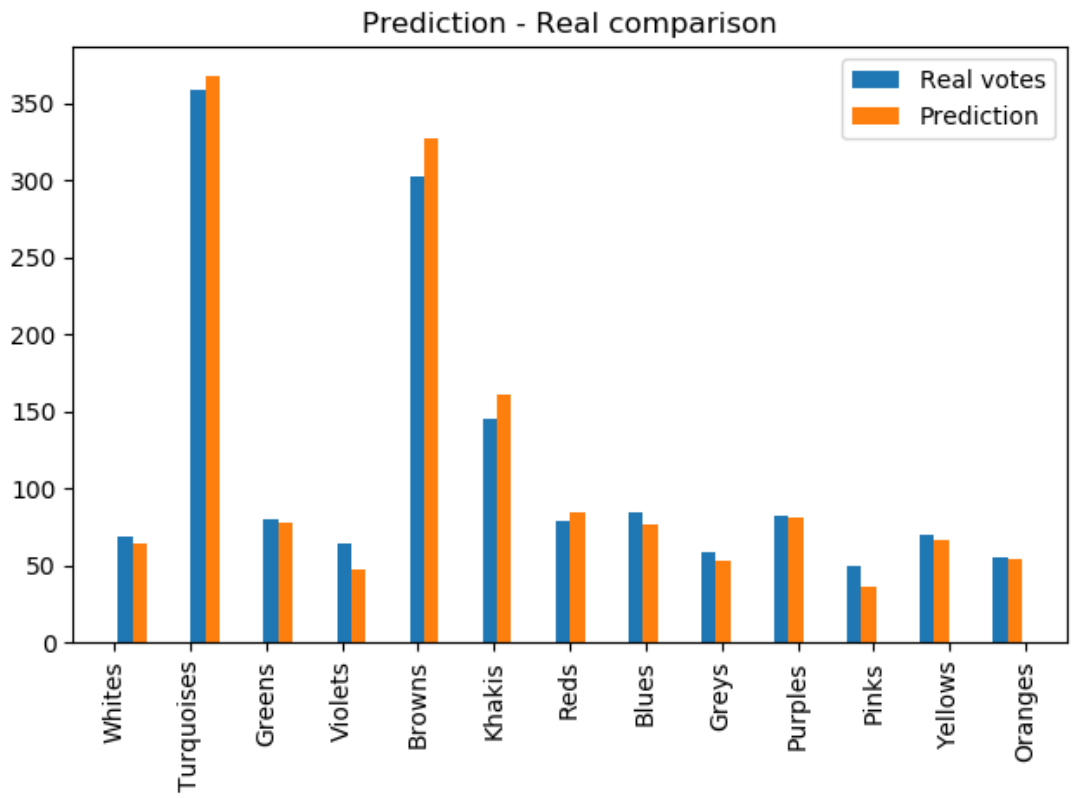
Transportation services: המסווג המנצח הינו RandomForestClassifier עם הפרמטרים
n_estimators: 100, 'min_samples_split': 2'

דיוק המסווג עבור כל אחד מהמסווגים שמצאנו: (דיוק – accuracy_score)
Accuracy for clf of each task : {'party_winner': 0.9306666666666666,
'division_of_voters': 0.8446666666666667, 'transportation': 0.938}

התפלגות הצבעות עבור מסווג עם פילוג טוב ביותר:



הבדל בין חיזוי לאמת:



C. חלק זה נמצא בקובץ fourth_prediction.

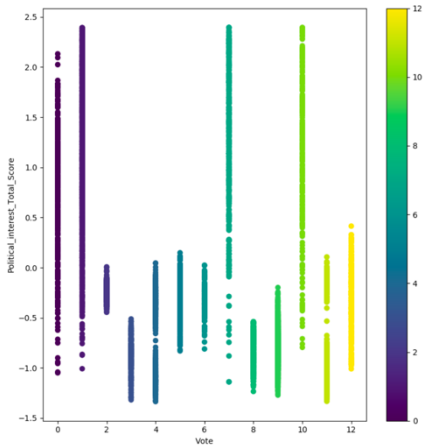
בחלק זה התייחסנו לחלוקת הדאטה ל train ו test .

ראשית הסתכלנו על השכבות הראשונות של עץ החלטה וראינו מי מהתכונות בעלי משקל גדול על תוצאות המודל והסתכלנו על התכונות איך הן משפיעות על ההצבעה ואיך הן מפולגות.
לכל תכונה בנפרד שינינו את קבוצת ה test עבור אותה תכונה כדי להשפיע על התוצאות של המנצח על ידי שהמסווג יטעה בגלל תכונה זו בסיווג.שינינו כל תכונה ולאחר שינוי קבוצת הtest בדקנו את המודל (שמאומן על train תקין) ואת המנצח.

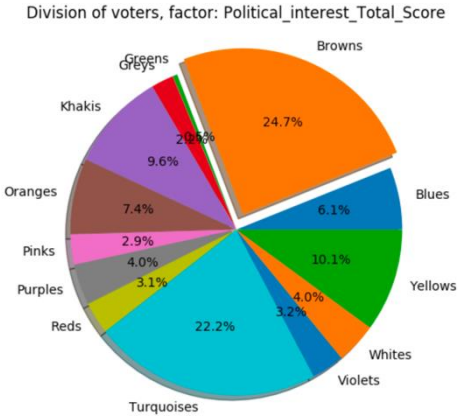
בנוסף ראינו מסעיפים קודמים שהמנצח הוא Turquoises(9) ומקום שני הוא Brown (1) וקיבלנו את התכונות הבאות כקריטיות עבור תוצאות הסיווג

Political_interest_Total_Score : ניתן לראות כי המצביעים של 1 ו-9 מפולגים בצורה כזאת שעבור 1 יש בוחרים בטווח גדול יותר ובאזור הגבוהה של הערכים והטווח עבור 9 קטן יותר ולמטה.

פילוג של train data על תכונה זו ביחס להצבעות:

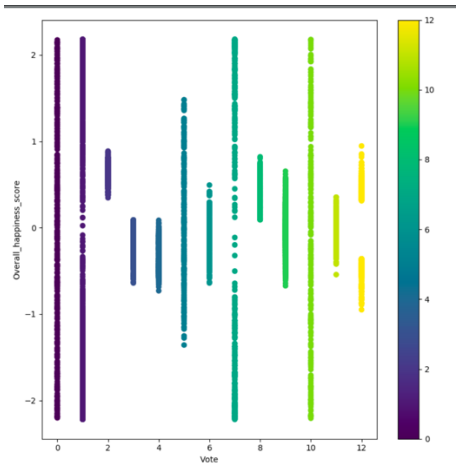


brown מנצח לאחר שינוי תכונה זו בtest:

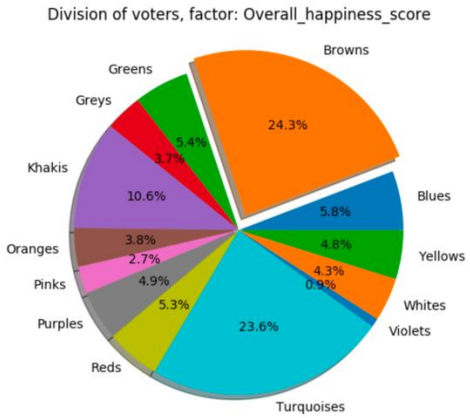


Overall_happiness_score : ניתן לראות כי המצביעים של 1 ו-9 מפולגים בצורה כזאת שעבור 1 יש בוחרים בטווח גדול יותר בכל הטווח מלבד האמצע ב והטווח עבור 9 קטן יותר ובאמצע.

פילוג של train data על תכונה זו ביחס להצבעות:

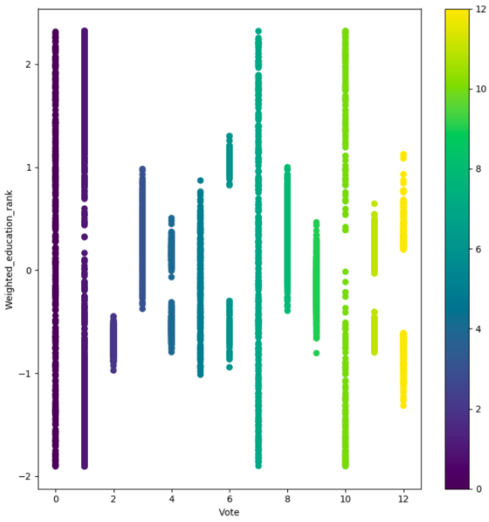


brown מנצח לאחר שינוי תכונה זו בtest:

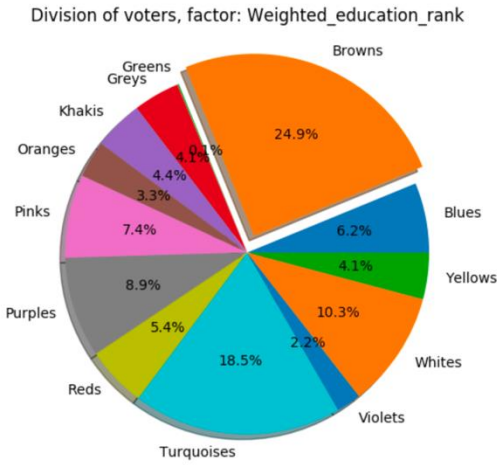


Weighted_education_ran: ניתן לראות כי המצביעים של 1 ו-9 מפולגים בצורה כזאת שעבור 1 יש בוחרים בטווח גדול יותר בכל הטווח מלבד האמצע והטווח עבור 9 קטן יותר ובאמצע.

פילוג של train data על תכונה זו ביחס להצבעות:

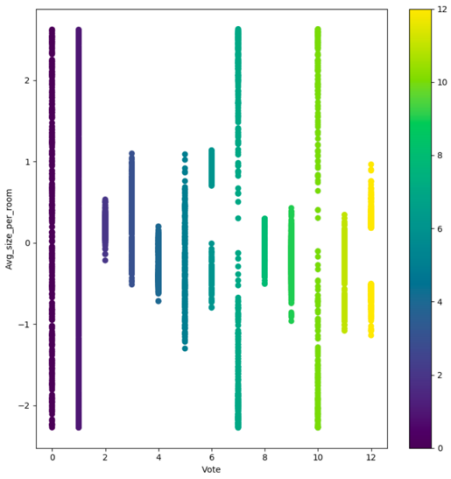


brown מנצח לאחר שינוי תכונה זו בtest:

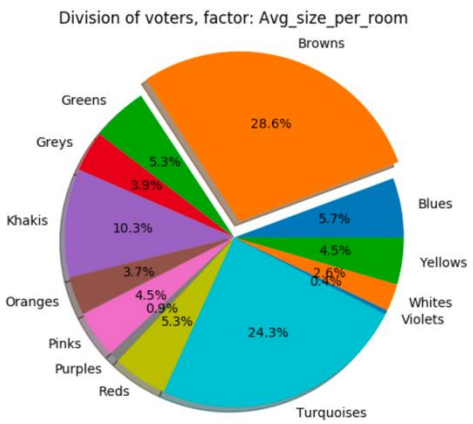


Size per room : ניתן לראות כי המצביעים של 1 ו-9 מפולגים בצורה כזאת שעבור 1 יש בוחרים בטווח גדול יותר בכל הטווח והטווח עבור 9 קטן יותר ובאמצע.

פילוג של train data על תכונה זו ביחס להצבעות:



brown מנצח לאחר שינוי תכונה זו בtest:



בונוס שני מימוש ADALINE:

תחילה נציין כי בחרנו לממש את גירסאת SGD של Adaline על מנת שההשוואה תהיה הוגנת ל perceptron , כלומר בכל שלב מבצעים צעד עבור כל סמפל בסט.

1. השוואת ביצועים על דאטה סט iris ו- digits:

עבור כל אחד מהסטים, ביצענו תחילה פיצול עבור train-test ודאגנו לבצע כל פעם השוואה כאשר בחרנו קלאס יחיד ופיצלנו אותו משאר הקלאסים.

לדוגמא עבור הסט של איריס, בחרנו כל פעם קלאס אחד , קבענו אותו כ-1 ואת שאר הקלאסים קבענו כ- 1- וביצענו את ההשוואה בין המודלים.

להלן התוצאות עבור כל אחד מהמודלים וכל אחד מפיצולי הקלאסים:

		Adaline		**Perceptron**	
		train	test	train	test
Iris	**setosa vs all**	1.0	1.0	1.0	1.0
	versicolor vs all	0.74	0.696	0.732	0.652
	virginica vs all	0.929	0.913	0.961	1.0
Digits	**0 vs all**	0.996	1.0	1.0	0.996
	1 vs all	0.975	0.97	0.986	0.963
	2 vs all	0.992	0.981	0.999	0.996
	3 vs all	0.978	0.974	0.984	0.959
	4 vs all	0.993	0.996	0.998	0.993
	5 vs all	0.991	0.992	0.999	0.989
	6 vs all	0.991	0.996	0.996	0.963
	7 vs all	0.991	0.996	0.996	0.993
	8 vs all	0.962	0.963	0.959	0.959
	9 vs all	0.972	0.956	0.99	0.981

עבור סט iris הרצנו את האימונים עם lr = 1e-4, num_iter = 1000

עבור סט digits הרצנו את האימונים עם lr = 1e-5, num_iter = 1000

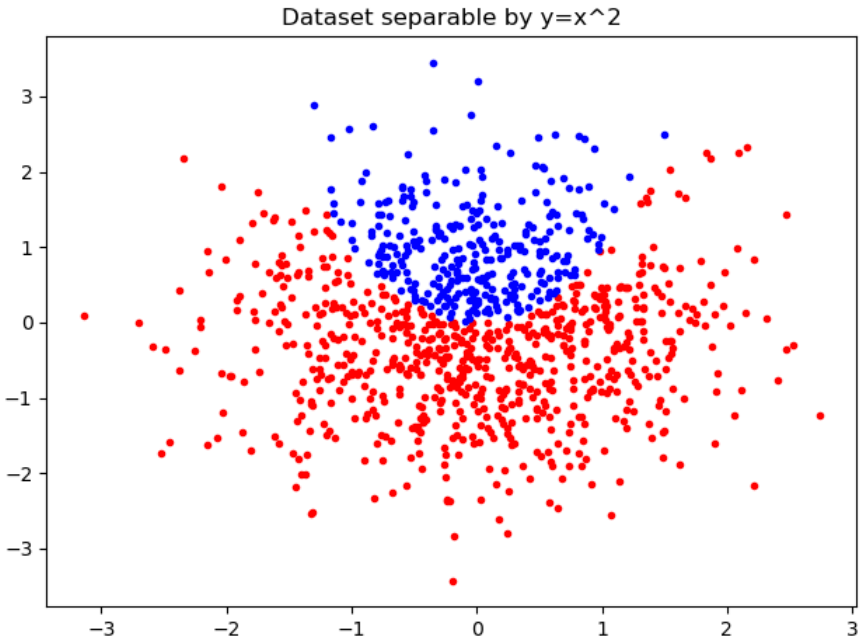
2. בחלק זה יצרנו שני סטים של נקודות, כאשר עבור הסט הראשון המודל של ADALINE עבד בצורה טובה יותר (מהר יותר והצליח להתכנס), לעומת הסט השני בו Perceptron עבד בצורה מהירה יותר.

סט ראשון:

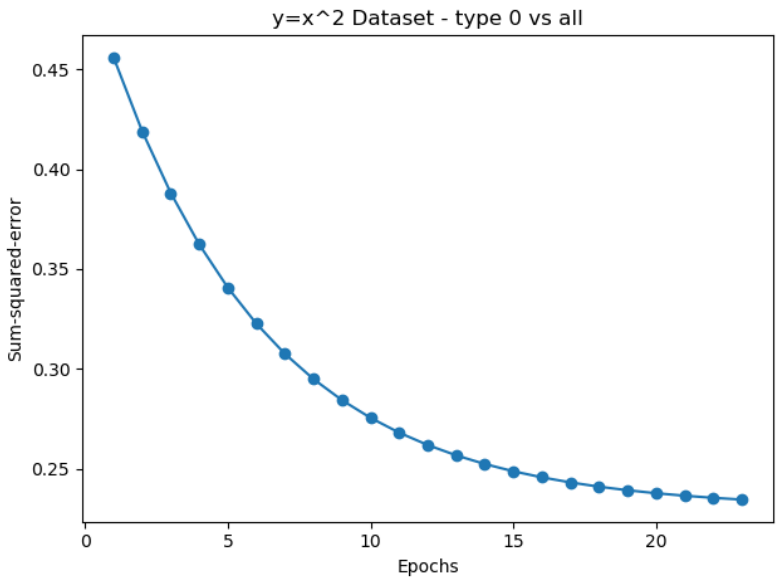
עבור סט זה בחרנו ליצור דאטה אשר אינו ניתן להפרדה לינארית, כלומר לא קיים קו אשר מפריד בצורה מושלמת בין הקלאסים השונים.

עבור סט זה מודל הAdaline הצליח להתכנס לישר מסויים בצורה מהירה, לעומתו מודל perceptron לא הצליח להתכנס כלל, וכמו שנציג בגרף, ערכי הloss שלו משתנים מקצה לקצה כל איטרציה.

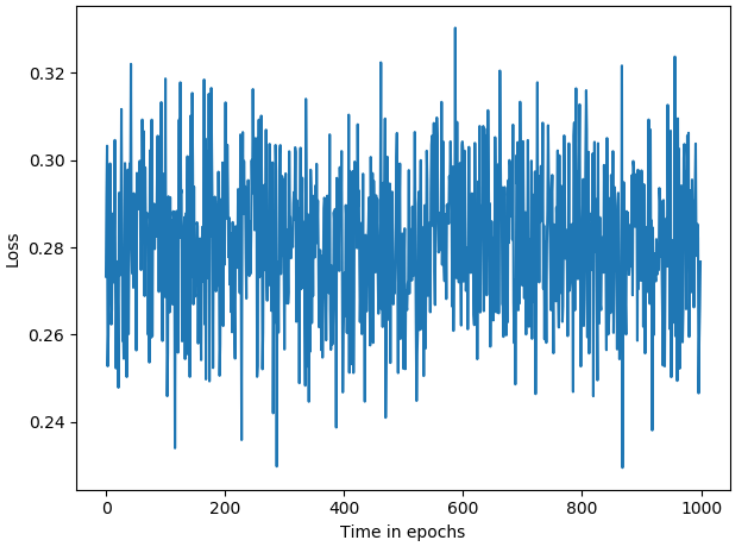
הדאטה סט שיצרנו מופרד על ידי הפונקציה $y = x^2$:



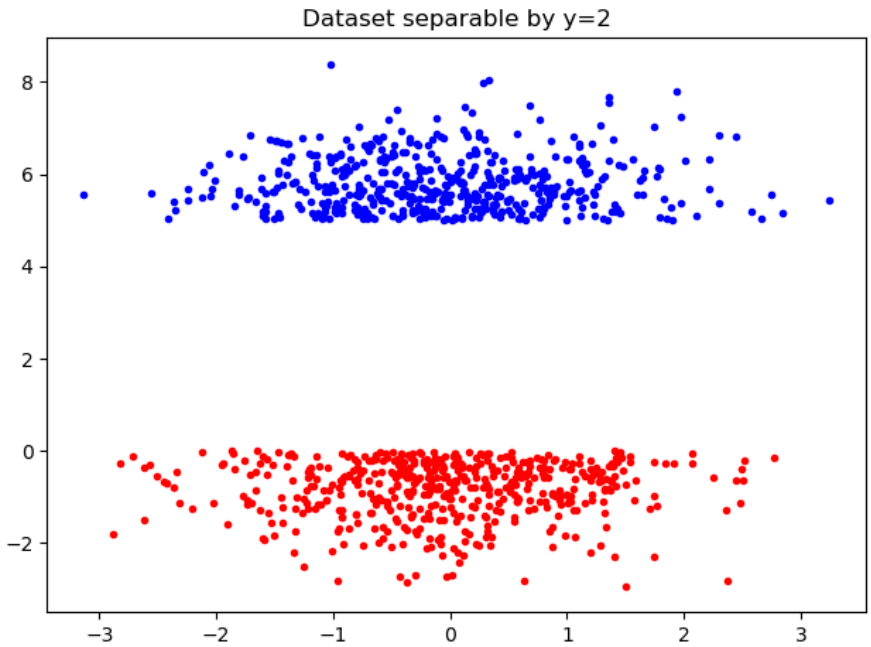
Loss של כל אחד מהאלגוריתם וזמן התכנסות:
Adaline:22 epochs



Perceptron:1000 epochs (did not converge)

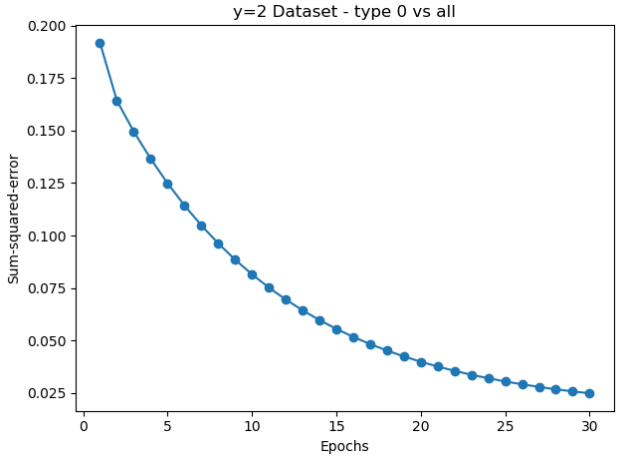


סט שני:
דאטה סט זה נוצר קח שניתן להפריד אותו על ידי קו לינארי
דאטה סט המופרד על ידי ישר y=2:

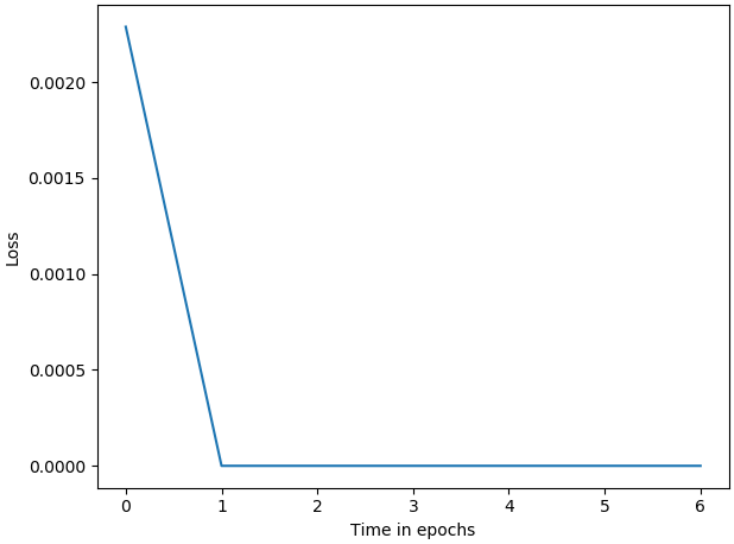


עבור סט זה, שני המודלים הצליחו להתכנס, כאשר perceptron הצליח להתכנס יותר מהר מאשר Adaline

Loss של כל אחד מהמודלים וזמן התכנסות:
Adaline:29 epochs



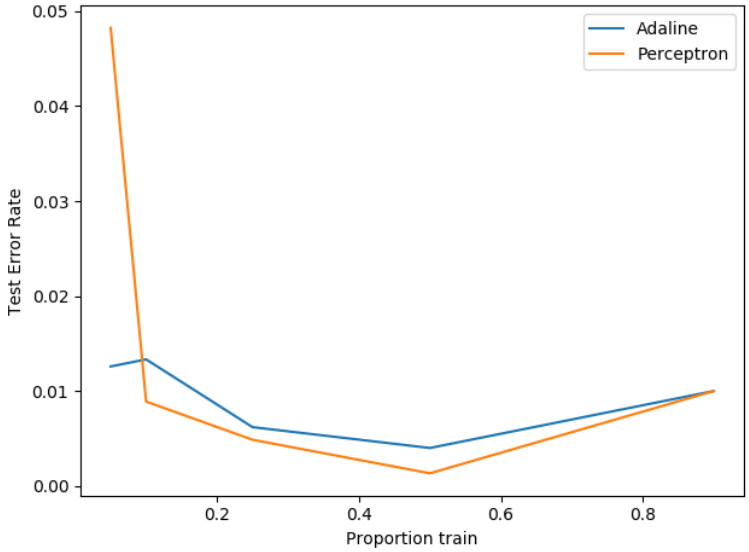
Perceptron: 1 epoch



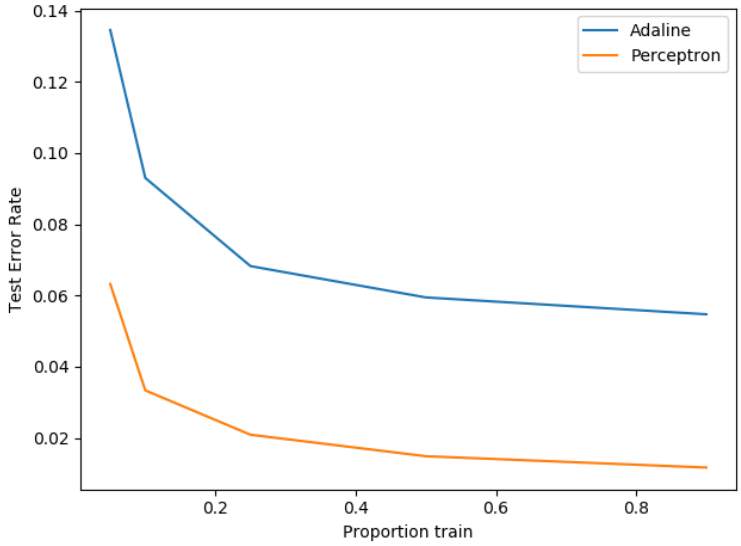
ניתן לראות כי כאשר קיימת הפרדה לינארית, הפרספטרון מתכנס מהר יותר מאשר אדאלין, אך כאשר לא ניתן להפריד לינארית את הדאטה, הפרספטרון לא מצליח להתכנס כלל, לעומת אדאלין שכן.
הבדל זה נובע מצעד העדכון של כל אלגוריתם.
עבור הperceptron צעד העדכון הינו $w_i = \begin{cases} w_{i-1}, & \hat{y} = y \\ w_{i-1} + lr * x * y, & \hat{y} \neq y \end{cases}$
נשים לב כי בגלל שהדאטה אינו ניתן להפרדה לינארית, תמיד נבצע עדכון בגודל קבוע למשקולות, כלומר עבור כל שגיאה בחיזוי של המודל נבצע צעד בגודל קבוע לכיוון אותה דוגמא.
כלומר כל עוד יש שגיאות נמשיך לשנות את המשקולות שלנו (תמיד יהיה שגיאות כי אין הפרדה לינארית)
לעומת זאת ב Adaline צעד העדכון הינו $w_i = w_{i-1} + lr * (\hat{y} - y) * x$
כלומר ניתן לראות כי יש התייחסות לחיזוי המודל, כלומר ככל שנתקדם, צעד העדכון יקטן מפני שהחיזוי יתקרב לערך האמיתי, עד אשר נגיע לנק' מינימום בה לא ניתן לשפר יותר את החיזוי.
לכן אם נסתכל על גרף ההפסד של מודל זה, נשים לב שהוא מונוטוני יורד עד אשר מתכנס לעומת perceptron אשר קופץ מקצה לקצה ולא מתכנס.

3. כעת ביצענו השוואה בצורה דומה בsklearn של שני האלגוריתמים, עבור שני הדאטה סטים של איריס וספרות:

Iris dataset setosa vs all



Digits dataset 5 vs all



בשני מקרים אלה, הדאטה פריד לינארית, וניתן לראות כי Perceptron מתכנס יותר מהר מאשר Adaline .