



3

68

30/05/19

עידו יחזקאל

אמיר אביבי

idoye

saavivi

2 0 4 3 9 7 3 6 8

3 0 5 1 8 3 8 7 3

מבוא למערכות לומדות

236756

סמסטר אביב תשע"ט

Mandatory Part – Modeling:

ראשית כמתבקש טענו את הקובץ הדוגמאות שוב, והחלנו עליו את מניפולציות העיבוד המקדים להכנת המידע כפי שביצענו בתרגיל הקודם, מניפולציות אלה כוללות:

1. חלוקת סט הדוגמאות הכולל ( על די StratifiedShuffleSplit) לשלושה סטים:

|  |  |
| --- | --- |
| Percentage from Original Data Set | Data Set |
| 65% | Train set |
| 10% | Validating set |
| 25% | Test set |

1. בחירת סט הפיצ'רים הנכון כפי שנאמר בתרגיל הנוכחי.
2. הוצאות ערכים שהינם outliers, לדוגמא ערכים שליליים.
3. השלמת ערכים חסרים לפי השיטות המקובלות: closest fit, feature correlation, mean and majority.
4. ביצוע נורמליזציה לערכים קטיגוריאליים וZ-scale לערכים נומינליים.
5. ייצוא המידע ל 3X2 קבצי CSV לפני ואחרי השנויים.

לאחר מכן, ניגשנו למשימת החיזוי כאשר בחלק החובה נדרשנו לחזות:

* מה היא המפלגה המנצחת?
* חלוקת הקולות בין המצביעים
* אספקת שירותי הסעה למצביעים של כל מפלגה.

על מנת להתמודד עם משימות אלו נרצה למצוא את המסווג הטוב ביותר מבין קבוצת מסווגים ולכן ביצענו את התהליך הבא:

1. בחירת הפרמרטים הטוביםן ביורת עבור כל סוג מסווג שאני מכירים, בהערכת המודל נעשת באמצעות k- fold cross balstion על סט האימון.
2. בחירת המסווג הטוב ביותר על פני כולם וזאת על ידי הערכת המודל על סט הוולידציה.
3. חיזוה המשימות בעזרת המסווג הטוב ביותר.

First Bonus – Relief Algorithm:

כל המימושים והמסקנות לבונוס זה מסתמכים על המאמר:

1. זיהוי קשרים בין תכונות לבין סיווג, בהינתן כי הסיווג ממופה למספרים הבאים:

'Blues'**:** 0**,**

'Browns'**:** 1**,**

'Greens'**:** 2**,**

'Greys'**:** 3**,**

'Khakis'**:** 4**,**

'Oranges'**:** 5**,**

'Pinks'**:** 6**,**

'Purples'**:** 7**,**

'Reds'**:** 8**,**

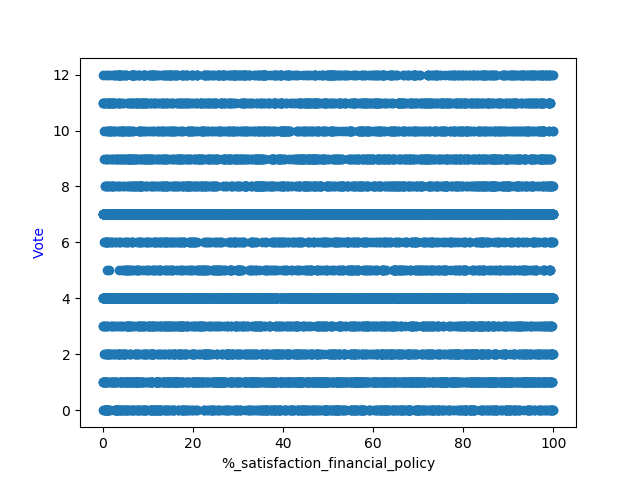
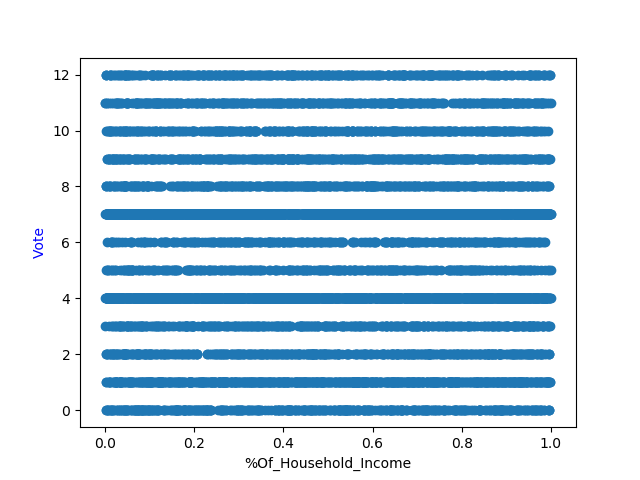
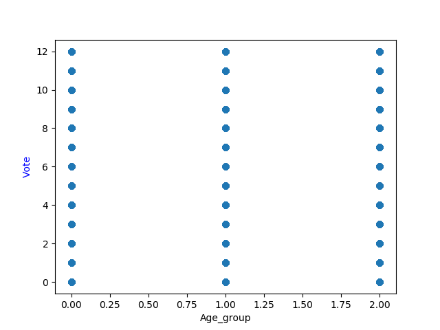
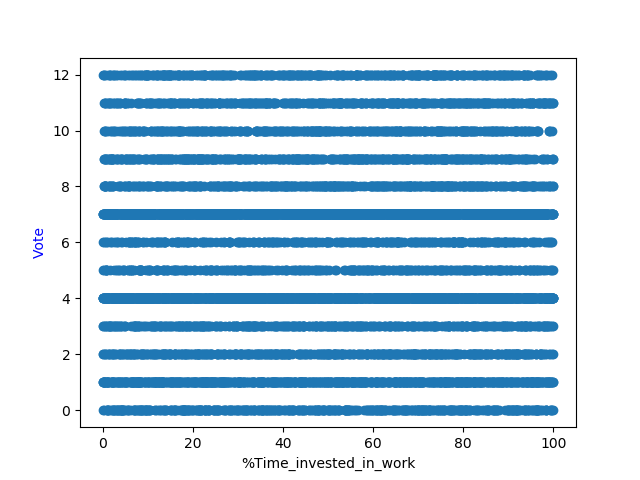
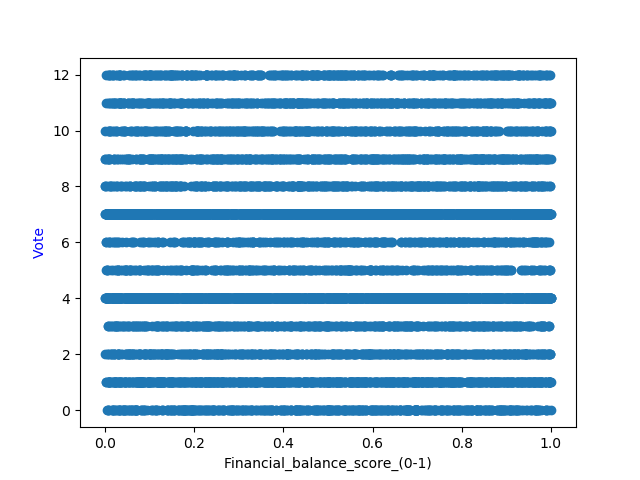
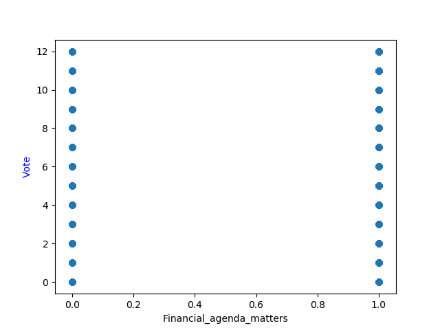
'Turquoises'**:** 9**,**

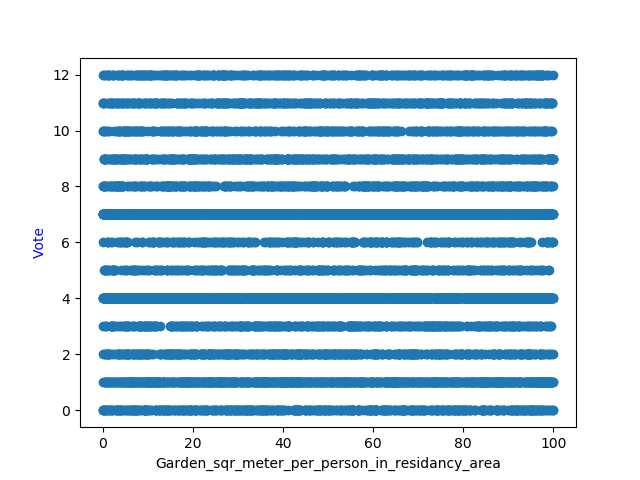
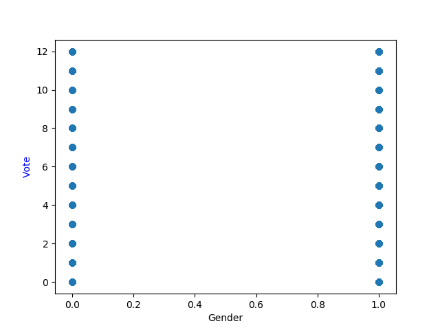
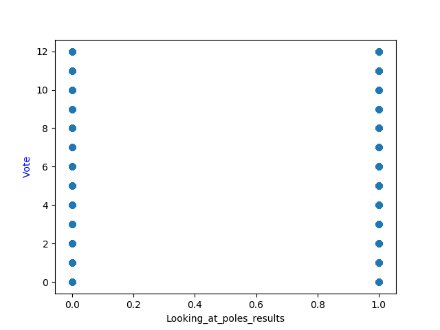
'Violets'**:** 10**,**

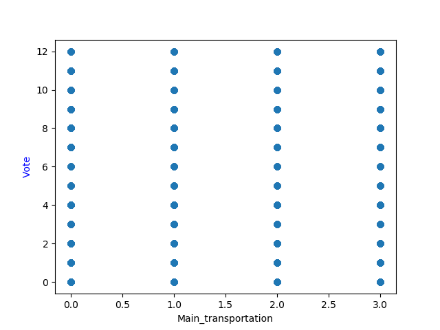
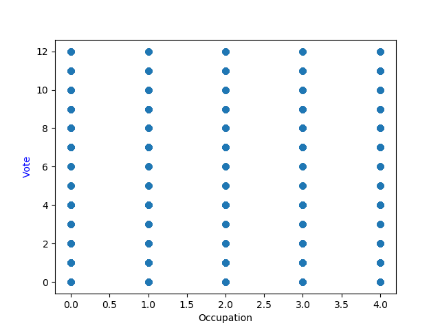
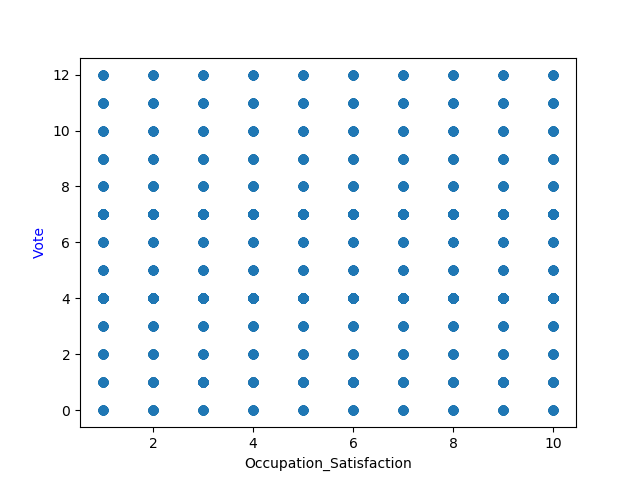
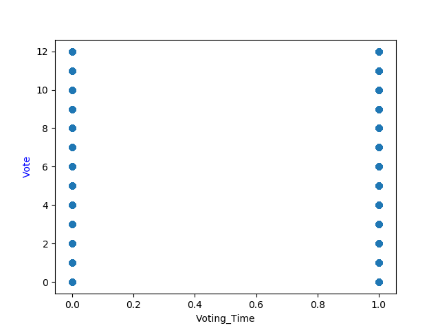
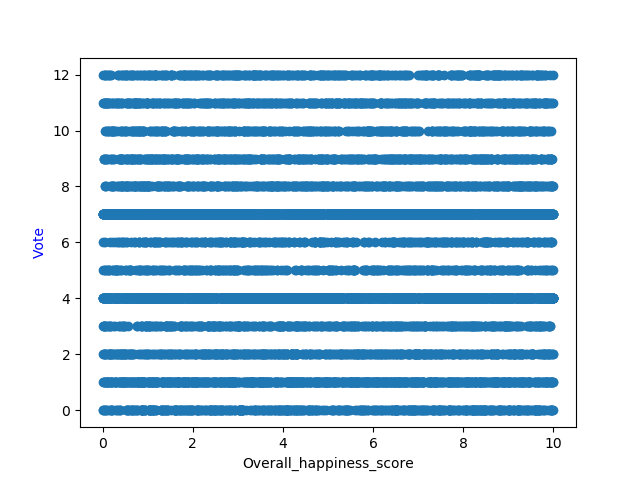
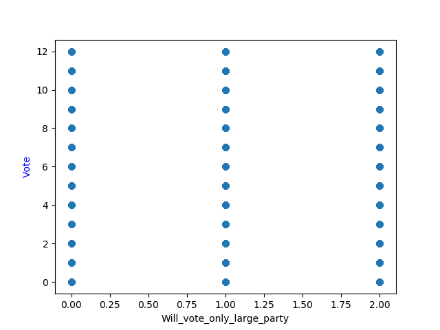
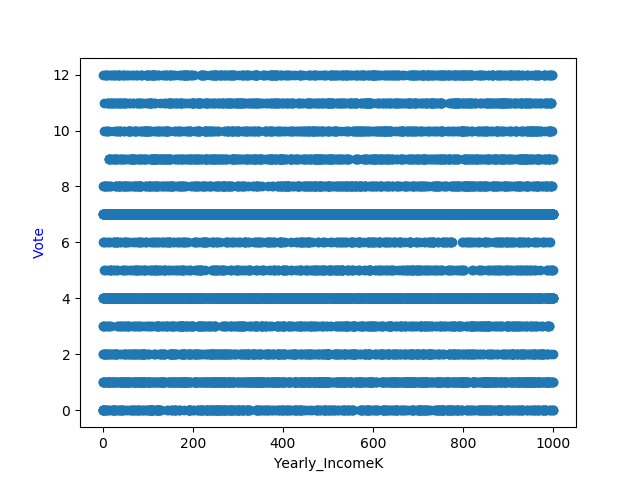
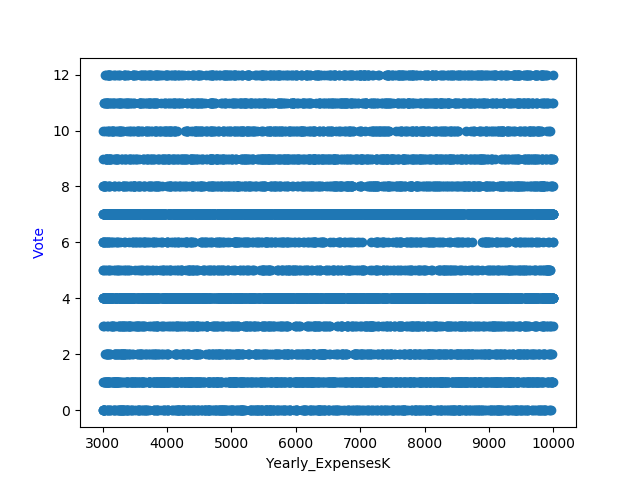
'Whites'**:** 11**,**

'Yellows'**:** 12

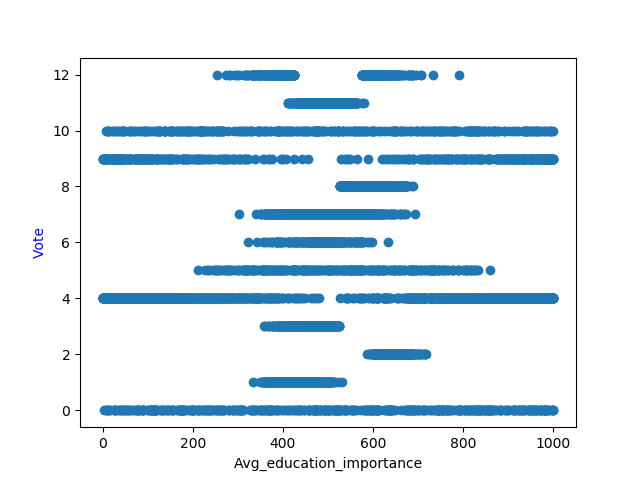
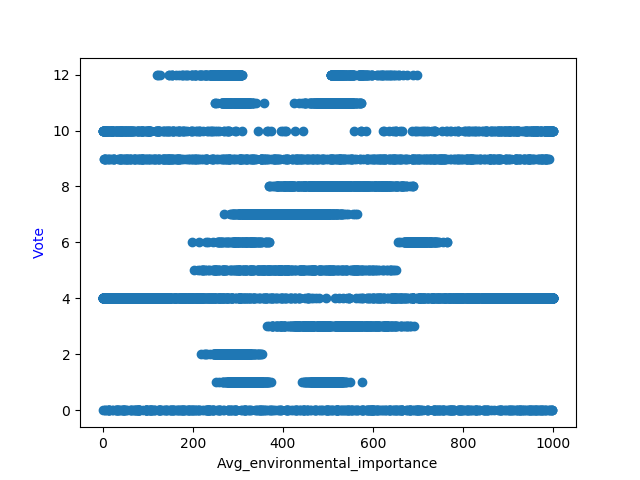
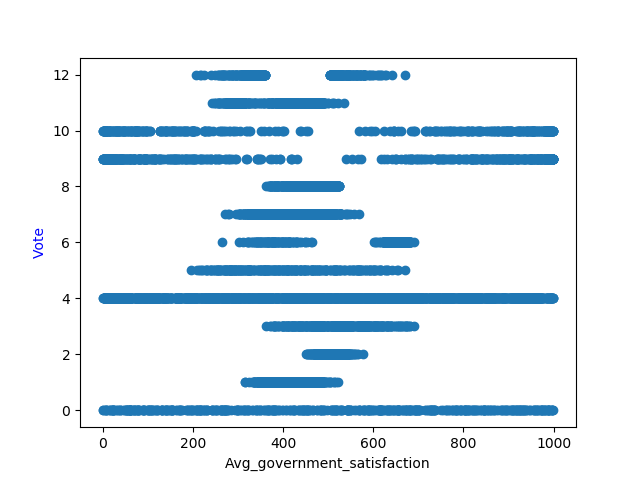
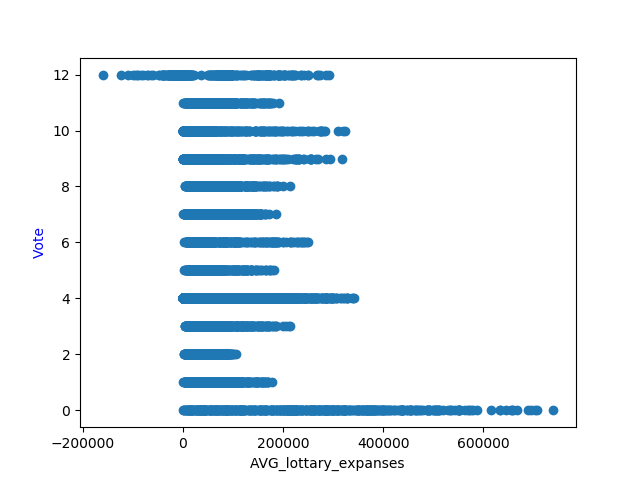
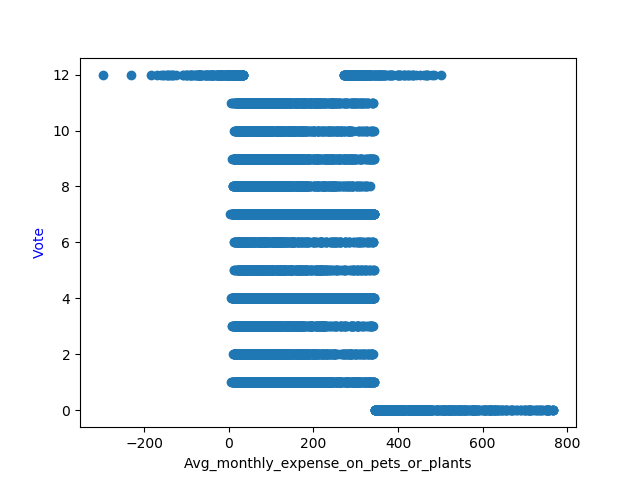
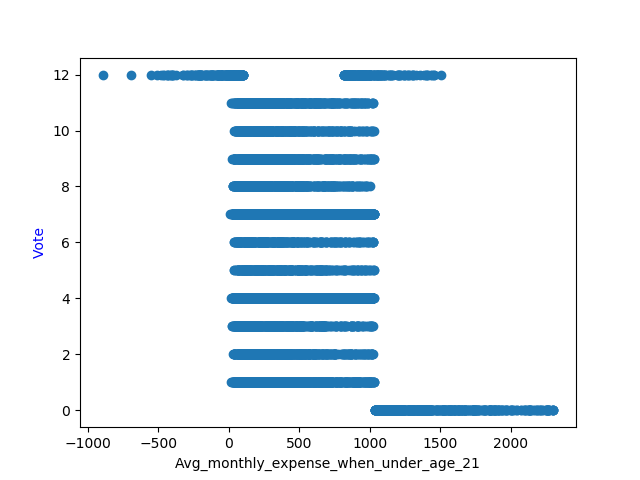
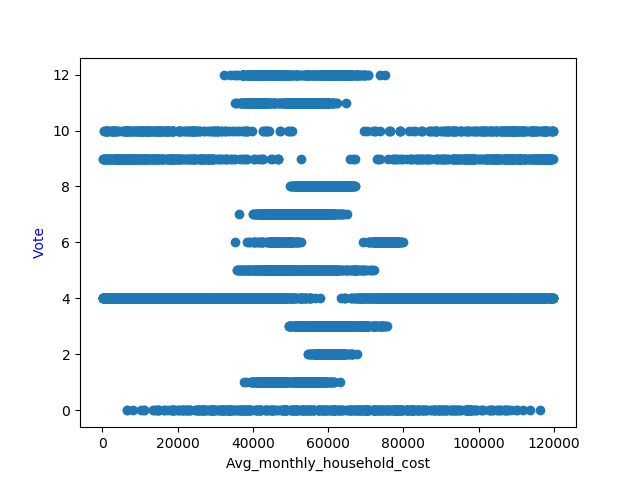
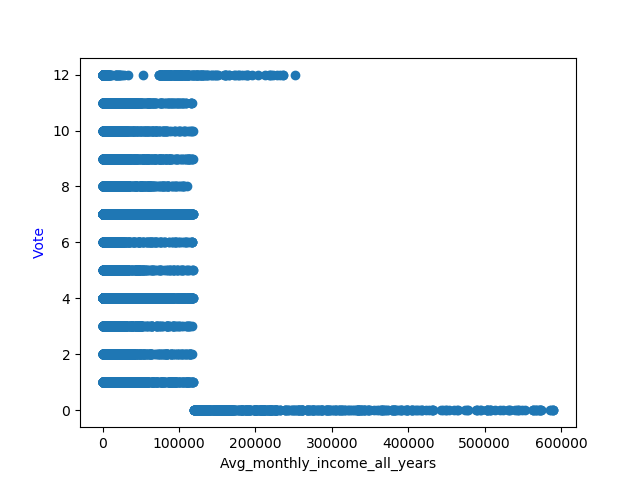
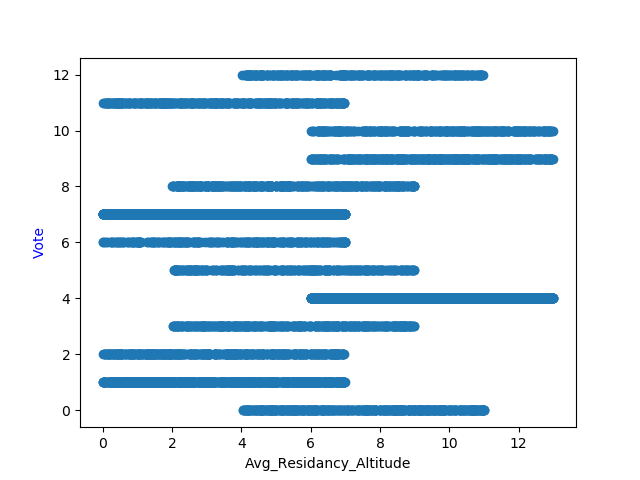
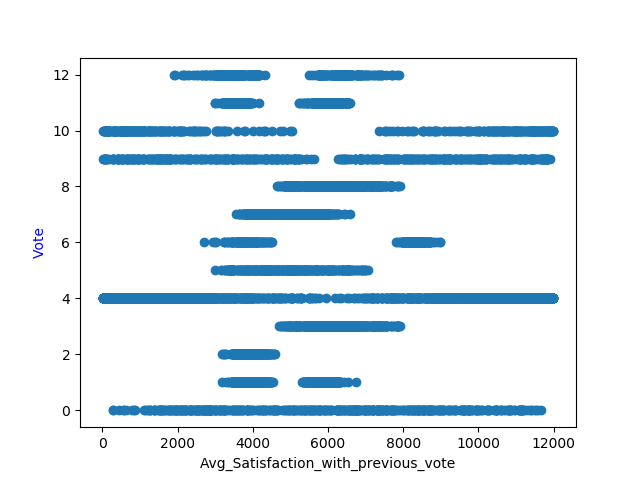
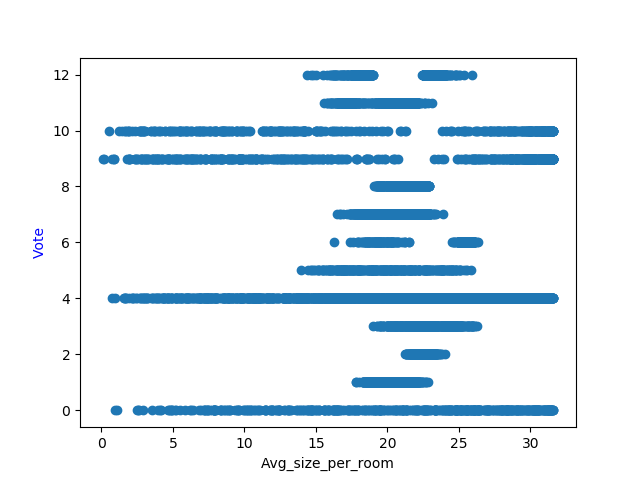
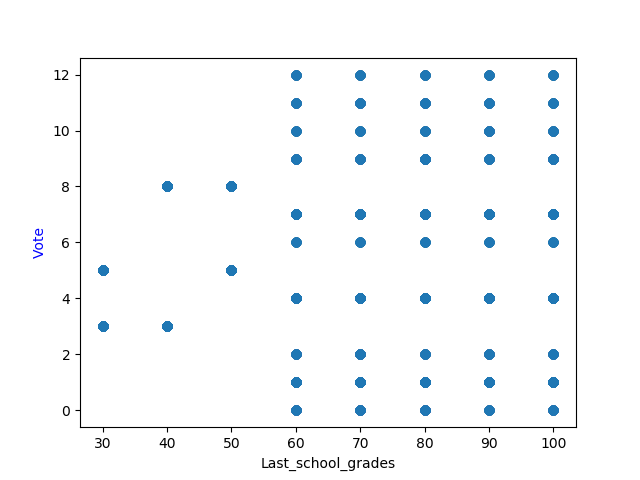
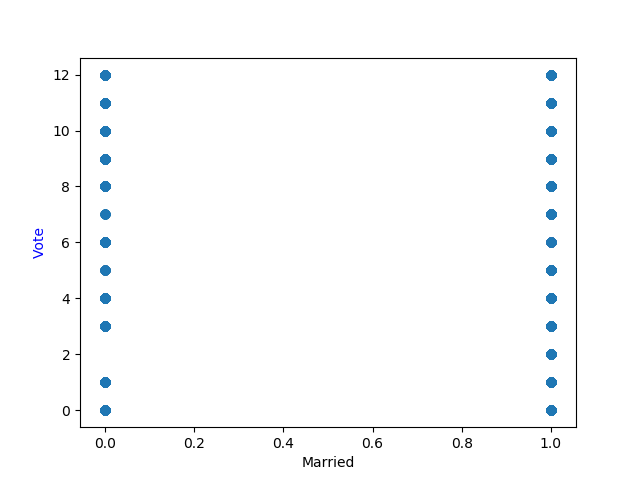
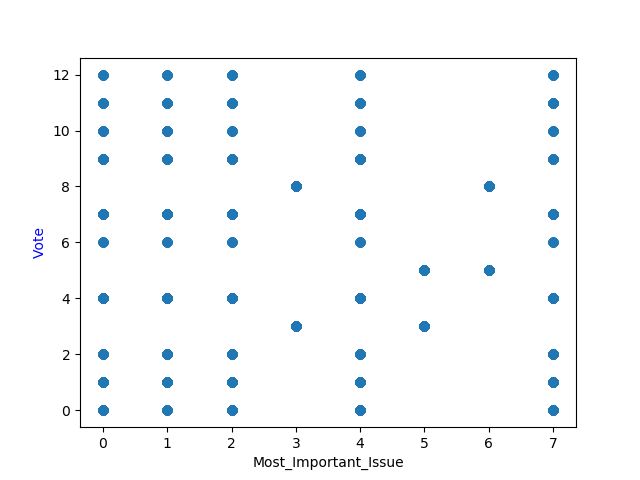
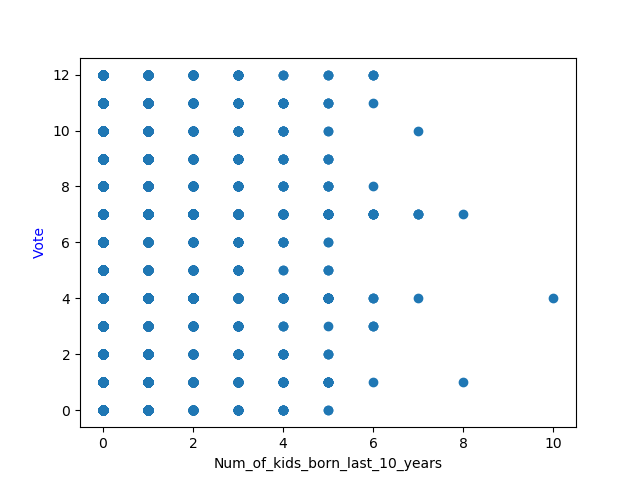
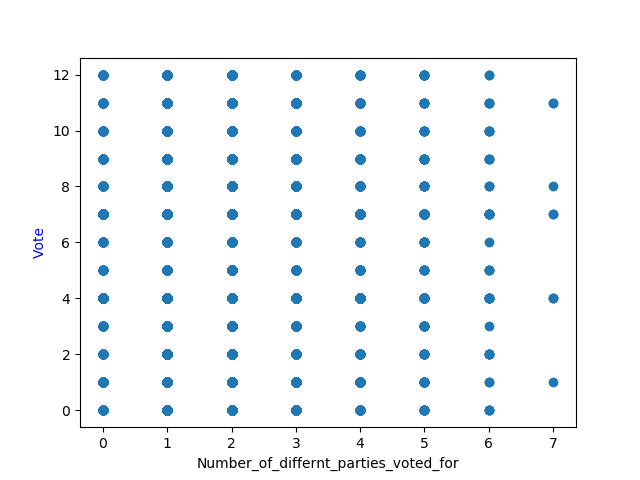
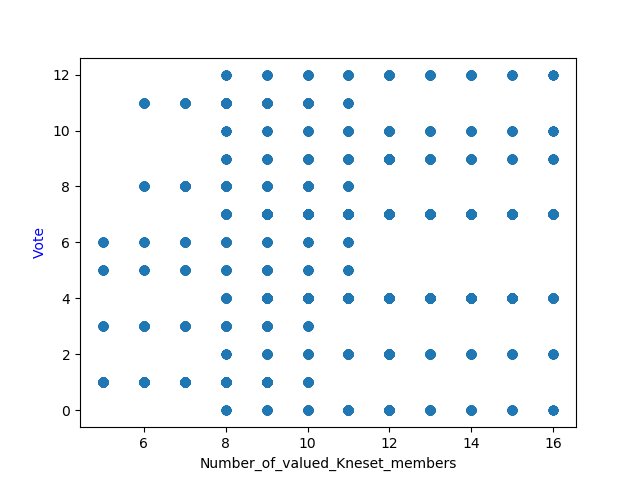
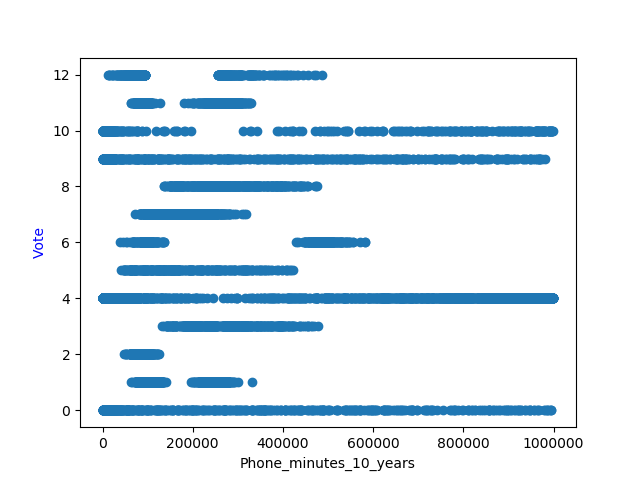
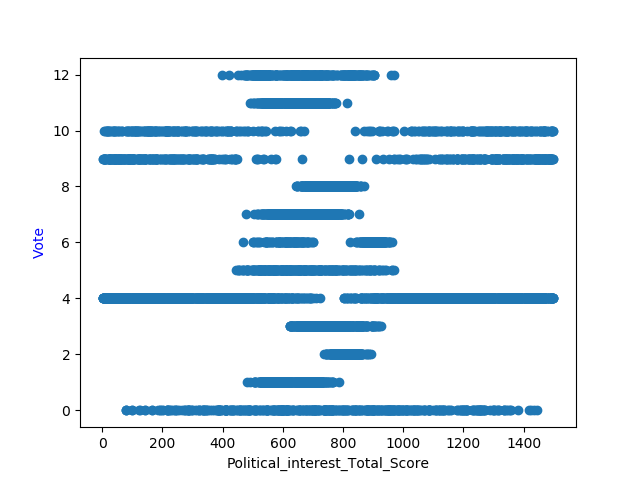
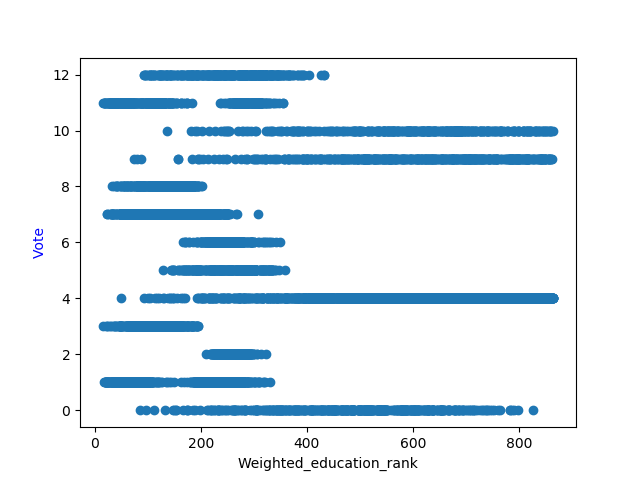
תכונות אשר לא הצלחנו לייחס להם קשר להן בעלי גרף המתפזר באופן אחיד על פני הסיווג לדוגמא:

לעומת זאת לתכונות הבאות מצאנו קשרים לסיווג:

* Avg\_education\_importance- הצבעים כחול, חאקי, טורקיז וסיגל נפוצים אצל כולם אמנם שאר הצבעים נפוצים רק אצל ערכים ממוצעים:  
  
* Avg\_environmental\_importance- הצבעים כחול, חאקי, טורקיז וסיגל נפוצים אצל כולם אמנם שאר הצבעים נפוצים רק אצל ערכים ממוצעים:  
  
* Avg\_government\_satisfaction- הצבעים כחול, חאקי, טורקיז וסיגל נפוצים אצל כולם אמנם שאר הצבעים נפוצים רק אצל ערכים ממוצעים:  
  
* AVG\_lottary\_expanses – בעלי הוצאה גבוהה נוטים לבחור כחול ובעלי הוצאה נמוכה נוטים לבחור צהוב:  
  
* Avg\_monthly\_expense\_on\_pets\_or\_plants- בעלי הוצאה גבוהה נוטים לבחור כחול ובעלי הוצאה נמוכה נוטים לבחור צהוב:  
  
* Avg\_monthly\_expense\_when\_under\_age\_21- בעלי הוצאה גבוהה נוטים לבחור כחול ובעלי הוצאה נמוכה נוטים לבחור צהוב:  
  
* Avg\_monthly\_household\_cost- הצבעים כחול, חאקי, טורקיז וסיגל נפוצים אצל רוב הערכים אמנם שאר הצבעים נפוצים רק אצל ערכים ממוצעים:  
  
* Avg\_monthly\_income\_all\_years- בעלי הכנסה גבוהה בוחרים רק כחול:  
  
* Avg\_Residancy\_Altitude- מגמה מעורבת בעיקר לכל קבוצת גבהים קיימת קבוצת צבעים:  
  
* Avg\_Satisfaction\_with\_previous\_vote – הצבעים כחול, חאקי, טורקיז וסיגל נפוצים אצל כולם אמנם שאר הצבעים נפוצים רק אצל ערכים ממוצעים:  
  
* Avg\_size\_per\_room- הצבעים כחול, חאקי, טורקיז וסיגל נפוצים אצל כולם ובעלי ערכים קטנים בוחרים רק בהם:  
  
* Last\_school\_grades- בעלי ערכים קטנים בוחרים רק באפור, כתום ואדום:  
  
* Married- רווקים (לא נשואים) לא בוחרים בירוק (בניגוד למציאות 😉):  
  
* Most\_Important\_Issue- הצבעים אפור, כתום ואדום נפוצים רק אצל ערכים מסוימים:  
  
* Num\_of\_kids\_born\_last\_10\_years- ערכים גבוהים נוטים לבחור בחום, חאקי, סגול וסיגל:  
  
* Number\_of\_differnt\_parties\_voted\_for- ערכים מקסימליים נוטים לבחור בחום, חאקי, סגול, אדום ולבן:  
  
* Number\_of\_valued\_Kneset\_members- מגמה מעורבת בערכים נמוכים וגבוהים:  
  
* Phone\_minutes\_10\_years- ערכים גבוהים בוחרים רק בצבעים כחול, חאקי, טורקיז וסיגל, שאר הצבעים נפוצים בעיקר אצל ערכים נמוכים:  
  
* Political\_interest\_Total\_Score- הצבעים כחול, חאקי, טורקיז וסיגל נפוצים אצל רוב הערכים אמנם שאר הצבעים נפוצים רק אצל ערכים ממוצעים:  
  
* Weighted\_education\_rank- ערכים גבוהים בוחרים רק בצבעים כחול, חאקי, טורקיז וסיגל, שאר הצבעים נפוצים בעיקר אצל ערכים נמוכים:  
  

1. האלגוריתם מומש בקובץ *bonus\_relief.py* בקובץ זה קיימת הפונקציה *relief* אשר מקבלת:

* x\_train: train data frame
* y\_train: train labels data frame
* local\_nominal\_feature: the nominal features to examine
* local\_numerical\_features: the numerical features to examine
* 𝑁 , num\_of\_iter: number of iterations
* 𝜏 , threshold: threshold

ומממשת את האלגוריתם ולבסוף מחזירה את התכונות הנבחרות.

עבור 𝑁=1,000,𝜏=1 התקבלו התכונות הבאות:

'Looking\_at\_poles\_results'

'Gender'

'Overall\_happiness\_score'

'Avg\_Residancy\_Altitude'

'Yearly\_ExpensesK'

'%Time\_invested\_in\_work'

'Phone\_minutes\_10\_years'

'%\_satisfaction\_financial\_policy'

'Avg\_monthly\_income\_all\_years'

'Number\_of\_differnt\_parties\_voted\_for'

'Number\_of\_valued\_Kneset\_members'

'Main\_transportation'

'Occupation'

'Financial\_agenda\_matters'

יתרונות:

* קל למימוש.
* מתמודד עם מספר גדול של דוגמאות.
* ניתן לשלוט בבחירת תכונות על ידי קביעת סף.
* משתמש בניתוח סטטיסטי בלבד
* יעילות זמן ריצה פולינומיאלית  
   )   
  נובעת מ:  
  1. לא מחפשים במרחב תתי הסטים של התכונות בצורה מפורשת

2. ויתור על מינימאליות תת סט התכונות המוחזרות.

* יחסית חסין לרעש
* לא מושפע מתלויית בין הפיצ'רים.

חסרונות:

* דרוש מספר איטרציות גדול על מנת לקבל תוצאות ברורות.
* לא תמיד ברור איזה סף לקבוע על מנת לקבל תת קבוצה טובה.
* לעומת SFS לא בוחן מסווג ספציפי ולכן נוכל לקבל תוצאות שונות על מסווגים שונים.
* מושפע מהחסרונות שנובעים מכל חישוב על בסיס "מרחק", לדוגמא:  
  1. תכונות נומנליות שיכולות להיות חסרות משמעות משפיעות באופן דרסטי על המרחק ל  
   .

2. מעניק לכל התכונות משקל שווה בחישוב המרחק(כתלות בבעיה, תכונה זו יכולה להיות מועילה מאוד, או מזיקה מאוד).

First Bonus – SFS Algorithm:

1. האלגוריתם מומש בקובץ *bonus\_sfs.py* בקובץ זה קיימת הפונקציה *sfs\_algo* אשר מקבלת:

* x\_train: train data frame
* y\_train: train labels data frame
* clf: classifier to examine
* subset\_size: user required subset size not mandatory

מממשת את האלגוריתם על ידי ניקוד של k-cross fold validation (k=3) ולבסוף מחזירה את קבוצת התכונות הנבחרות, גודל הקבוצה נקבע על ידי פרמטר של המשתמש או אם לא ניתן אז כאשר אף תכונה לא משפרת את הסיווג.  
  
בחנו את האלגוריתם על ידי שני מסווגים שונים:

* SGDClassifier תוצאות:

SVM Classifier accuracy score before SFS is: 0.7723601652376123

SVM Classifier selected features are: [

'Weighted\_education\_rank', 'Avg\_monthly\_income\_all\_years', 'Last\_school\_grades', 'Number\_of\_valued\_Kneset\_members', 'Phone\_minutes\_10\_years', 'Avg\_education\_importance', 'Avg\_government\_satisfaction', 'Married', 'AVG\_lottary\_expanses', 'Avg\_Residancy\_Altitude', 'Yearly\_ExpensesK', 'Avg\_size\_per\_room']

SVM Classifier accuracy score after SFS is: 0.8070339800786065

* KNN תוצאות:

K Neighbors Classifier accuracy score before SFS is: 0.5826105372698062

K Neighbors Classifier selected features are: [

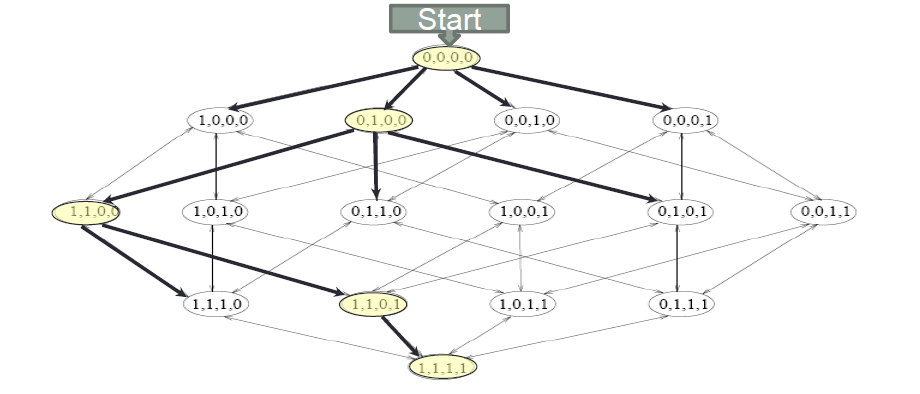
'Avg\_environmental\_importance', 'Avg\_government\_satisfaction', 'Avg\_education\_importance', 'Avg\_Satisfaction\_with\_previous\_vote', 'Last\_school\_grades']

K Neighbors Classifier accuracy score after SFS is: 0.8313137509347758

יתרונות:

* קל למימוש.
* מתמודד עם מספר גדול של דוגמאות.
* ניתן לשלוט בגודל תת קבוצת התכונות הנבחרות.
* התכונות הנבחרות מותאמות לאלגוריתם למידה ספציפי.
* החלטתיו נקבעות על ידי המדד שמוערך אלגוריתם הלמידה ולא בצורה מוחלטת, כל מתכנת יכול לבחור את המדד אותו הוא רוצה למקסם.
* סקלבילי – ניתן לייצר סט תוצאות(תת סט של פיצ'רים) הולכות וגדולות ע"י שימוש באלגוריתמי קלסיפיקציה שונים, ובכך לקבל תמונה מציאותית יותר לגבי אופן התנהגות הבעיה.

חסרונות:

* זמן ריצה תלוי במספר התכונות ולכן נעדיף להפעילו בהינתן מספר קטן של תכונות.
* החסרון העיקרי - **לא ניתן לחזור אחורה ולבטל בחירה של תכונה גם אם היא redundant לחלוטין.**
* סכנה לoverfit- מכיוון שהתכונות הנבחרות נבחרות ביחס לאחוזי הדיוק על סט הולדיצה, ישנה סכנה ברורה שהתכונות הנבחרות יהיו רגישות יותר ל overfit.
* חמדני – בוחר להכניס \ לא להכניס את התכונה הבאה רק אם משפרת את יכולת החיזוי ולכן לא לוקח בחשבון תלויות בין תכונות( בעיה בה זה יכול לבוא לידי ביטוי היא לדוגמא בעיית ה XOR)
* עוד חסרון שנובע מחמדנות- מחפש במרחב תתי התכונות באופן אינקרמנטלי(עולה), ולכן סביר להניח שבמרחב תכונות גדול יפספס תתי סטים של תכונות בעלות אחוזי דיוק גבוהים יותר.