

למידה חישובית – תרגיל 3

מיכל אברמוב, 301834297
בוריס בורשבסקי, 311898746

מה מצורף

1. מחברת המפרטת בשלבים את מה שעשינו (modeling_notebook.ipynb)
2. קובץ modeling.py המכיל את כל הקוד של התרגיל
3. קובץ html שמאפשר צפייה נוחה יותר במחברת
4. קבצי datan.
5. קובץ test_predictions.csv עם תוצאות הפרדיקציה

התרגיל

התרגיל מחולק ל-3 שלבים עיקריים:

1. בחינה בסיסית של מספר מודלים
2. בחינה מעמיקה של 3 מודלים בולטים
3. הפעלה ופרדיקציה בעזרת המודל שנבחר

בחינה בסיסית

במהלך הבחינה הבסיסית רצנו על מספר רב של מודלים על datan של הtrain וחישבנו עליהם cross_val_score על מנת להעריך את הפרדיקציה, את כולם ניתן לראות במחברת, אנו נפרט פה על כמה עיקריים:

- ראינו ש LinearSVC ו SVC נתנו לנו ניקוד באזור ה 0.87 ולכן לא בחרנו אותם
- ראינו ש OneVsOneClassifier נתן ניקוד של 0.905737.
- ראינו ש GaussianNB נתן ניקוד של 0.86, גם Perceptron ו LinearDiscriminantAnalysis נתנו ציונים יחסית נמוכים
- עבור **KNeighborsClassifier** רצנו על כל מאפשרויות מ-2 עד 20 ע"מ למצוא את ה-א האופטימלי, מבחינותינו ה-א האופטימלי היה 3 והוא נתן רק ניקוד של 0.918114

```

minimum k neighbors = 2, score = 0.902213
minimum k_neighbors = 3, score = 0.918114
minimum k_neighbors = 4, score = 0.907826
minimum k_neighbors = 5, score = 0.916752
minimum k_neighbors = 6, score = 0.913081
minimum k_neighbors = 7, score = 0.912299
minimum k_neighbors = 8, score = 0.909593
minimum k_neighbors = 9, score = 0.910556
minimum k_neighbors = 10, score = 0.908232
minimum k_neighbors = 11, score = 0.906881
minimum k_neighbors = 12, score = 0.905926
minimum k_neighbors = 13, score = 0.903800
minimum k_neighbors = 14, score = 0.901478
minimum k_neighbors = 15, score = 0.900868
minimum k_neighbors = 16, score = 0.896038
minimum k_neighbors = 17, score = 0.896219
minimum k_neighbors = 18, score = 0.893895
minimum k_neighbors = 19, score = 0.890801
Best n_neighbors size: 3
KNeighborsClassifier with best N param: 0.918114

```

- עבור **DecisionTreeClassifier** רצנו על כל האפשרויות של עצים עם גודל split משתנה וראינו שעבור split מינימאלי בגודל 8 אנו מקבלים ציון של 0.933549

```

minimum splitter = 2, score = 0.931029
minimum splitter = 3, score = 0.931037
minimum splitter = 4, score = 0.929864
minimum splitter = 5, score = 0.931403
minimum splitter = 6, score = 0.932374
minimum splitter = 7, score = 0.933361
minimum splitter = 8, score = 0.933549
minimum splitter = 9, score = 0.933537
minimum splitter = 10, score = 0.932958
minimum splitter = 11, score = 0.932960
minimum splitter = 12, score = 0.933149
minimum splitter = 13, score = 0.933525
minimum splitter = 14, score = 0.933131
minimum splitter = 15, score = 0.933141
minimum splitter = 16, score = 0.931596
minimum splitter = 17, score = 0.931994
minimum splitter = 18, score = 0.933155
minimum splitter = 19, score = 0.931619
Best Splitter size: 8
DecisionTreeClassifier with best splitter: 0.933549
DecisionTreeClassifier Default score: 0.931029

```

- ניסינו אותו דבר עם **RandomForestClassifier** וראינו שעבורו אנו מקבלים ציונים מעל 9.4 כאשר עבוד splitter בגודל 4 מקבלים אפילו 9.5

```
minimum splitter = 2, score = 0.949628
minimum splitter = 3, score = 0.950242
minimum splitter = 4, score = 0.951995
minimum splitter = 5, score = 0.949627
minimum splitter = 6, score = 0.949261
minimum splitter = 7, score = 0.949468
minimum splitter = 8, score = 0.949853
minimum splitter = 9, score = 0.947952
minimum splitter = 10, score = 0.948108
minimum splitter = 11, score = 0.950255
minimum splitter = 12, score = 0.946556
minimum splitter = 13, score = 0.946952
minimum splitter = 14, score = 0.944035
minimum splitter = 15, score = 0.946767
minimum splitter = 16, score = 0.946762
minimum splitter = 17, score = 0.948711
minimum splitter = 18, score = 0.947346
minimum splitter = 19, score = 0.947522
Best Splitter size: 4
RandomForestClassifier with best splitter: 0.951995
```

בחינה מעמיקה של 3 מודלים בולטים

- בשלב זה בחרנו 3 מודלים שאותם נבחן ע"י פרדיקציה על הtrain והצגה של classification report:

- RandomForestClassifier(min_samples_split=4)

```
***** RandomForestClassifier *****
              precision    recall  f1-score   support

   Blues      0.86667      0.46429      0.60465         28
   Browns      0.90378      0.97317      0.93719        1081
   Greens      0.99579      0.99475      0.99527         952
   Greys       0.96429      0.97297      0.96861         333
   Oranges      0.94481      0.91509      0.92971         318
   Pinks       0.94889      0.85060      0.89706         502
   Purples      0.97828      0.97828      0.97828        1197
   Reds        0.94753      0.97152      0.95938         316
   Whites      0.89809      0.74603      0.81503         189
   Yellows      0.93893      0.99194      0.96471         248

 avg / total      0.95278      0.95256      0.95149        5164
```

- KNeighborsClassifier(n_neighbors=3)

```
***** KNeighborsClassifier *****
              precision    recall  f1-score   support

   Blues      0.26087      0.21429      0.23529         28
   Browns      0.88003      0.97040      0.92301        1081
   Greens      0.99260      0.98634      0.98946         952
   Greys       0.85373      0.85886      0.85629         333
   Oranges      0.85507      0.74214      0.79461         318
   Pinks       0.95561      0.81474      0.87957         502
   Purples      0.96787      0.98162      0.97470        1197
   Reds        0.82303      0.92722      0.87202         316
   Whites      0.90780      0.67725      0.77576         189
   Yellows      0.91304      0.93145      0.92216         248

 avg / total      0.92102      0.92022      0.91853        5164
```

- DecisionTreeClassifier(min_samples_split=8)

```
***** DecisionTreeClassifier *****
              precision    recall  f1-score   support

   Blues      0.46875      0.53571      0.50000         28
   Browns      0.90331      0.93340      0.91811        1081
   Greens      0.99473      0.99160      0.99316         952
   Greys       0.96970      0.96096      0.96531         333
   Oranges      0.89969      0.90252      0.90110         318
   Pinks       0.85972      0.85458      0.85714         502
   Purples      0.95703      0.94904      0.95302        1197
   Reds        0.93103      0.93987      0.93543         316
   Whites      0.78107      0.69841      0.73743         189
   Yellows      0.95473      0.93548      0.94501         248

 avg / total      0.92977      0.92971      0.92958        5164
```

ניתן לראות ש**RandomForestClassifier** מציג תוצאות טובות משמעותית משני הclassifiers האחרים מבחינת המדדים, אחריו ה**DecisionTreeClassifier** ורק לבסוף **KNeighborsClassifier**. בשלב זה בכל זאת בחרנו להתקדם עם שלושת המודלים ע"מ למדוד את הביצועים שלהם.

הפעלה ופרדיקציה בעזרת המודלים שנבחרו

עבור

`RandomForestClassifier(min_samples_split=4)`

אימנו את המודל ה train data והרצנו פרדיקציה על הטסט:

ע"פ הפרדיקציה במודל זה קיבלנו שהמפלגה הזוכה היא ה **סגולה** בפער קטן מהחומים כמו שניתן לראות בהתפלגות:

```
Vote distribution
Blues, 3.000000, 0.175850%
Browns, 399.000000, 23.388042%
Greens, 315.000000, 18.464244%
Greys, 102.000000, 5.978898%
Oranges, 95.000000, 5.568581%
Pinks, 144.000000, 8.440797%
Purples, 410.000000, 24.032825%
Reds, 118.000000, 6.916764%
Whites, 47.000000, 2.754982%
Yellows, 73.000000, 4.279015%
```

בשלב זה ברצנו בדיקה של ה classification שלנו מול המידע test המקורי וראינו דיוק של 94% - למעשה טעינו 100 מתוך 1706 ניסויים.

Confusion matrix:

```
[ 'Blues', 'Browns', 'Greens', 'Greys', 'Oranges', 'Pinks', 'Purples', 'Reds', 'Whites', 'Yellows' ]
array([[ 3,  0,  0,  0,  0,  0,  0,  0,  0,  3],
       [ 0, 355,  0,  0,  0,  2,  3,  0,  5,  0],
       [ 0,  0, 312,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 95,  7,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  7, 83,  0,  0,  3,  0,  0],
       [ 0, 19,  0,  0,  0, 134,  5,  0,  0,  0],
       [ 0,  2,  3,  0,  0,  3, 398,  0,  0,  0],
       [ 0,  0,  0,  0,  5,  0,  0, 114,  0,  0],
       [ 0, 23,  0,  0,  0,  5,  4,  0, 42,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 70]])
```

כשהסתכלנו על ה confusion matrix ראינו שבאופן יחסי יש הרבה טעויות לטובת החומים מה שעלול להסיט את התוצאות, נבדוק את המודל השני.

KNeighborsClassifier(n_neighbors=3)

אימנו את המודל ה train data והרצנו פרדיקציה על הטסט.

גם ע"פ הפרדיקציה במודל הזה קיבלנו שהמפלגה הזוכה היא ה **סגולה** הפעם בפער טיפה גדול יותר מהחומים כמו שניתן לראות בהתפלגות:

```
Vote distribution
Blues, 5.000000, 0.293083%
Browns, 393.000000, 23.036342%
Greens, 312.000000, 18.288394%
Greys, 112.000000, 6.565064%
Oranges, 73.000000, 4.279015%
Pinks, 143.000000, 8.382181%
Purples, 410.000000, 24.032825%
Reds, 130.000000, 7.620164%
Whites, 57.000000, 3.341149%
Yellows, 71.000000, 4.161782%
```

בשלב זה ברצנו בדיקה של ה classification שלנו מול המידע test המקורי וראינו דיוק של 92% - למעשה טעינו ב139 מתוך 1706 ניסויים, 39 יותר מבמודל של Random Forest.
גם פה בדקנו את ה confusion matrix וראינו הרבה טעויות לטובת החומים:

```
[ 'Blues', 'Browns', 'Greens', 'Greys', 'Oranges', 'Pinks', 'Purples', 'Reds', 'Whites', 'Yellows' ]
array([[ 1,  0,  0,  0,  0,  0,  0,  0,  0,  5],
       [ 0, 351,  0,  0,  0,  3,  6,  0,  5,  0],
       [ 0,  0, 307,  0,  0,  0,  5,  0,  0,  0],
       [ 0,  0,  0, 91,  6,  0,  0,  6,  0,  0],
       [ 0,  0,  0, 17, 60,  0,  0, 16,  0,  0],
       [ 0, 19,  0,  0,  0, 136,  2,  0,  1,  0],
       [ 0,  2,  5,  0,  0,  1, 397,  0,  1,  0],
       [ 0,  0,  0,  4,  7,  0,  0, 108,  0,  0],
       [ 0, 21,  0,  0,  0,  3,  0,  0, 50,  0],
       [ 4,  0,  0,  0,  0,  0,  0,  0,  0, 66]])
```

DecisionTreeClassifier(min_samples_split=8)

אימנו את המודל ה train data והרצנו פרדיקציה על הטסט.

גם ע"פ הפרדיקציה במודל הזה קיבלנו שהמפלגה הזוכה היא ה **סגולה** הפעם בפער משמעותי.

Vote distribution

Blues, 11.000000, 0.644783%

Browns, 366.000000, 21.453693%

Greens, 316.000000, 18.522860%

Greys, 104.000000, 6.096131%

Oranges, 94.000000, 5.509965%

Pinks, 163.000000, 9.554513%

Purples, 406.000000, 23.798359%

Reds, 118.000000, 6.916764%

Whites, 64.000000, 3.751465%

Yellows, 64.000000, 3.751465%

בדקנו את ה confusion matrix והפעם ראינו שהטעויות לטובת החומים מתקזזות יחסית ואין מפלגה שטועים לטובתה בצורה משמעותית:

```
[ 'Blues', 'Browns', 'Greens', 'Greys', 'Oranges', 'Pinks', 'Purples', 'Reds', 'Whites', 'Yellows' ]
array([[ 3,  0,  0,  1,  0,  0,  0,  0,  0,  2],
       [ 0, 329,  2,  0,  0, 12,  7,  0, 15,  0],
       [ 0,  0, 311,  0,  0,  0,  1,  0,  0,  0],
       [ 0,  0,  0, 97,  6,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  6, 81,  0,  0,  6,  0,  0],
       [ 0, 14,  0,  0,  0, 138,  6,  0,  0,  0],
       [ 0,  7,  3,  0,  0, 10, 384,  0,  2,  0],
       [ 0,  0,  0,  0,  7,  0,  0, 112,  0,  0],
       [ 0, 16,  0,  0,  0,  3,  8,  0, 47,  0],
       [ 8,  0,  0,  0,  0,  0,  0,  0,  0, 62]])
```

בשיטה זו קיבלנו 142 ו 91.676436% אך הטעויות מתפלגות באופן שיותר מתאים לנתונים.

לבסוף החלטנו שעבור המשימה של לחזות מה כל מועמד יצביע עדיף את **RandomForestClassifier** בגלל אחוז הדיוק שלו והתחזית שלו נמצאת בקובץ test_predictions.csv

ראינו כי עבור המשימה של לבחור מפלגה מנצחת עדיף את **DecisionTreeClassifier** אך לא הורדנו את התחזית שלו לקובץ.

Bonus "one size doesn't fit all"

We created a script called pred_with_dec_tree.py which predicts with the insights from here, its predictions better likely to choose the better party, but has more errors than Random Forest Classifier, the explanation was above in this document.