סמינר בהנדסת תוכנה

Divorce Predictors

מאגר המידע [לינק]

מאגר המידע מכיל נתונים של מחקר שבו נשאלו זוגות נשואים וגרושים 54 שאלות בנושאים הקשורים למערכת היחסים שלהם, כך שכל תשובה היא מספר בין 1-5 המשקף את מידת הסכמתם עם הנאמר.

במחקר זה, שנערך בסוף שנת 2018, מבין המשתתפים 84 (49%) התגרשו ו- 86 (51%) זוגות נשואים. בקבוצת המחקר היו 84 גברים (49%) ו- 86 נשים (51%). גילאי המשתתפים נעו בין 20 ל- 63 ($\bar{X}$ = 36.04, SD = 9.34).

למרות שנתוני המחקר נאספו משבעה אזורים שונים בטורקיה, הנתונים היו בעיקר מאזור הים השחור (n = 79). מבין המשתתפים 74 (43.5%) נישאו מאהבה, ו- 96 (56.5%) נישאו בנישואין מסודרים (שידוך). בעוד ל-127 (74.7%) מהמשתתפים היו ילדים, ל-43 (25.3%) לא היו ילדים.

בנוסף, 18 (10.58%) מהמשתתפים היו בוגרי בית ספר יסודי, 15 (8.8%) היו בוגרי בית ספר תיכון, 33 (19.41%) היו בוגרי תיכון, 88 (51.76%) היו בוגרי מכללות ו -15 (8.8%) היו בעלי תואר שני.

ההכנסה החודשית של המשתתפים הייתה כדלקמן: 34 (20%) אנשים הרוויחו מתחת ל- 4000 ש"ח, 54 (31.76%) השתכרו בין 4000-6000 ש"ח, 28 (16.47%) בין 6000-8000 ש"ח ו- 54 (31.76%) היו בעלי הכנסה חודשית של מעל 8000 ש"ח.

התשובות לשאלות שבמחקר נאספו בראיון פנים מול פנים. המשתתפים שבזוגות הגרושים ענו על השאלות בהתבסס על הזוגיות שלהם כשהיו נשואים. ומתוך הזוגות הנשואים שהשתתפו במחקר, נכללו רק אלה עם נשואים מאושרים, ללא אף מחשבה על גירושים.

תיאור הפרמטרים שמשמעותיים לעבודה המחקר שלנו:

- טיב היחסים בין בני הזוג.
- היכרות אישית של תחומי העניין של בן הזוג האחר.
- זמן איכות, תחביבים ותחומי חיים משותפים של בני הזוג.
- דעות והשקפות עולם דומות בנושאים שונים.
- כנות ופתיחות בין בני הזוג.

תיאור הפרמטרים שפחות משמעותיים לעבודה המחקר שלנו:

כל דבר שאינו קשור באופן ישיר למערכת היחסים של בני הזוג (כמו גיל, הכנסה, ילדים וכו').

ההיפותזה של המחקר:

במחקר שלנו נציג היפותזות לגבי השפעת פרמטרים שונים על סיכויי הגירושים של בני זוג. אנו נבחן מאפיינים שונים במספר תחומים של מערכות יחסים של זוגות נשואים, כיצד הם באים לידי ביטוי בשיעורי הגירושים של הזוגות והאם ניתן לנבא על פיהם גירושים עתידיים.
אנו צופות כי השקפות חברתיות ופוליטיות מסוימות משפיעות על בריאות ואורך הנישואין. אלה מאפיינים יאפשרו לנו לחזות מי הסיכוי הטוב ביותר להתגרש או להישאר נשואים.
שאלת המחקר: אילו מאפיינים או השקפות אישיות קשורים ביותר לגירושין?

המטרה מאחורי ההיפותזה של המחקר:

מטרתנו בעבודה זו היא לחקור אספקטים שונים במערכות יחסים של זוגות נשואים וכאלה שהיו נשואים והתגרשו, אשר יש להם השפעה על הסיכויים של זוגות להתגרש. אנו נשאף לצמצם ככל האפשר את מספר הגורמים בעלי המשקל הגדול ביותר על סיכויי גירושים, מבלי לפגוע בשיעורי ההצלחה של החיזוי.

האלגוריתמים

סוג אחד של מודל נפוץ ב-ML הוא:

- Classification - מידע מחולק למספר סופי וידוע של תוויות (מחלקות).

הבעיה בפרויקט שלנו היא בעיית קלסיפיקציה (סיווג).

סיווג מידע היא פעולה נפוצה בלמידת מכונה - בהינתן אוסף נתונים אשר צריך להפריד אותם לשתי קבוצות על סמך ניסיון קודם כאשר המטרה היא להחליט לאיזו משתי הקבוצות לשייך את המידע החדש. כל רשומת מידע באוסף הנתונים מיוצגת על ידי וקטור בגודל קבוע.

מטרת הלמידה המונחית היא ללמוד על ההתפלגות המשותפת, ומתוך כך לנבא עבור אלמנט את התיוג שלו. האלגוריתם "לומד" כלל המקשר בין תיאור הדוגמה לתווית המתאימה לה, וכך יפתור בעיה שבה המטרה היא להצמיד תווית לכל דוגמה. האלגוריתם יופעל על דוגמאות חדשות, שהתווית שלהן לא ידועה, ויחזה מהי התווית הנכונה לכל דוגמה.

הגדרת בעיית הסיווג

נגדיר תחילה את המרכיבים הבסיסים של בעיית הסיווג.

X - מרחב הקלט. אלמנט טיפוסי במרחב הקלט הוא $\underline{x}=(x_1,...x_d)\in X$, כאשר $x_i \in \Re$.

וקטור x נקרא גם וקטור המאפיינים של הקלט.

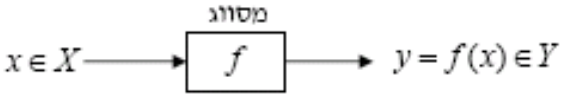
Ω - אוסף סופי של מחלקות שונות. אנו מעוניינים לשייך כל אלמנט בקלט לאחת המחלקות.

בפרויקט שלנו יש שתי מחלקות בלבד (נשואים וגרושים).

איבר אופייני יסומן כ- $\omega_i \in \Omega$.

Y - אוסף סופי של החלטות (או פעולות) אפשריות. זהו מרחב הפלט של המסווג. כל תוצאה ב-Y נרצה להתאים למחלקה כלשהי $\omega_i \in \Omega$.

את המסווג נוכל לתאר כך:



$f : X \to Y$

את הלימוד נשיג על ידי הזנת המערכת באוסף של n דוגמאות מסווגות $\{x_k, d_k\}_{k=1}^{n}$, כאשר עבור כניסה x_k הסיווג הנכון הוא $d_k \in \Omega$. בסוף תהליך הלמידה של המערכת נרצה שעל סמך הדוגמאות שהזנו, המערכת תדע לסווג לבדה כל אלמנט קלט שתקבל.

את התוצאות המתקבלות נציג בעזרת confusion matrix.

במטריצת הבלבול (confusion matrix) ישנם 4 סוגי תוצאות:

1. זוגות נשואים שהאלגוריתם סיווג נכון (true negatives).
2. זוגות נשואים שהאלגוריתם טעה בסיווג שלהם וסיווג אותם כזוג גרוש (false positives).
3. זוגות גרושים שהאלגוריתם טעה בסיווג שלהם וסיווג אותם כזוג נשוי (false negatives).
4. זוגות גרושים שהאלגוריתם סיווג נכון (true positives).

אלגוריתם ראשון:

Random Forest

עץ החלטה (Decision Tree) הינו אלגוריתם שניתן ליישם גם ברגרסיה וגם בקלסיפיקציה.
עץ נבנה על ידי פיצול המידע לתתי קבוצות, כל פעם בהסתמך על שדה אחד של המידע. תהליך זה מבוצע באופן רקורסיבי עד שכל תת קבוצה מכילה רק איברים מאותו הסוג, או עד שפיצול נוסף לא יביא לתוצאות טובות יותר .אחת הבעיות שיש לעצי החלטה היא overfitting, התאמה חזקה מדי למידע שאיתן התאמן המודל, אך יכולת פחות טובה לתת תשובה טובה עבור מידע חדש. לתהליך זה קוראים recursive partitioning.

אלגוריתם עץ ההחלטה קובע תחילה את צומת השורש (Root Node) האופטימלי של העץ באמצעות קריטריון "מקסום הרווח מהמידע". חשיבותו של מאפיין הינו הרווח מהמידע הצפוי ממנו (Expected Information Gain), כאשר רווח זה נמדד על ידי הירידה באי הוודאות הצפויה אשר תתרחש, ככל ותתרחש, עם קבלת מידע אודות אותו מאפיין. לאחר מכן, אלגוריתם עץ ההחלטה ממשיך לעשות אותו הדבר עבור הצמתים העוקבים. הקצוות של הענפים הסופיים של העץ מכונים צמתי עלים (Leaf Nodes). כאשר עץ ההחלטה משמש לסיווג, או אז צמתי העלים כוללים בחובם את ההסתברויות של כל אחת מהקטגוריות להיות הקטגוריה הנכונה. כאשר עץ ההחלטה משמש לחיזוי ערך נומרי, או אז צמתי העלים מספקים את ערך התוחלת של היעד (Target). הגיאומטריה של העץ נקבעת באמצעות סט האימון (Training Set), אך הסטטיסטיקה שעוסקת ברמת הדיוק של העץ צריכה, כמו תמיד בלמידת מכונה, לבוא מתוך סט הבדיקה (Test Set) ולא מתוך סט האימון.

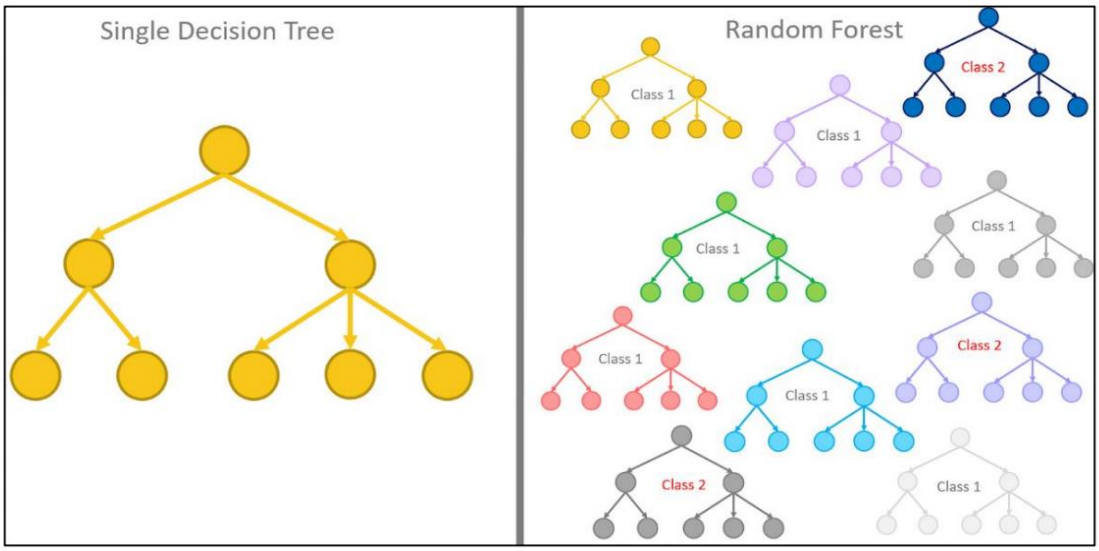
אחת השיטות הנפוצות לקחת את היכולת חיזוי של עץ החלטה (ושל מודל בכללי) ולשפר אותה היא, חיבור של כמה מודלים יחדיו. שיטות של חיבור של כמה מודלים יחד נקראות learning ensemble והם מאוד נפוצות ב-ML, במיוחד בתחום של עצים.
ישנם כל מיני שיטות לקחת מספר מודלים על מנת ליצור מודל כללי טוב יותר. אנחנו נתמקד באחד בשם bagging שנותן משקל שווה לכל המודלים שהוא מאמן. הוא מאמן כל אחד מהמודלים בעזרת חלק רנדומלי, בדרך כלל שווה בגודלו של המידע.
גישה נוספת ליצירת יער אקראי היא להגריל באופן מקרי את רמות הסף (Thresholds) המשמשות עבור המאפיינים, במקום לחפש את רמת הסף האפשרית הטובה ביותר. זה יכול להיות יעיל מבחינת חישובית שכן מציאת רמת הסף האופטימלית של המאפיין בכל צומת עלולה לקחת זמן רב.

יער הוא שילוב של כמה עצים על מנת לקבל תוצאה טובה יותר שפחות פגיעה ל-overfitting. נגיד עבור בעיית classification, במקום לאמן עץ אחד נצטרך לאמן כמה. כמובן שאם נאמן כמה עצים עם מידע זהה נקבל עצים זהים, מכאן נובע שעלינו לפצל את המידע בצורה כלשהי.

יער אקראי, כפי ששמו מרמז עליו, מורכב ממספר עצי החלטה, כאשר המידע נבחר רנדומלית. העצים נוצרים על ידי דגימה מתוך המאפיינים או מתוך התצפיות (כלומר, באמצעות גישת ה-bagging). כל אחד מהעצים נותן תוצאה לא-אופטימאלית (Suboptimal) אך באופן כללי, על פי רוב החיזוי בדרך זו משתפר.

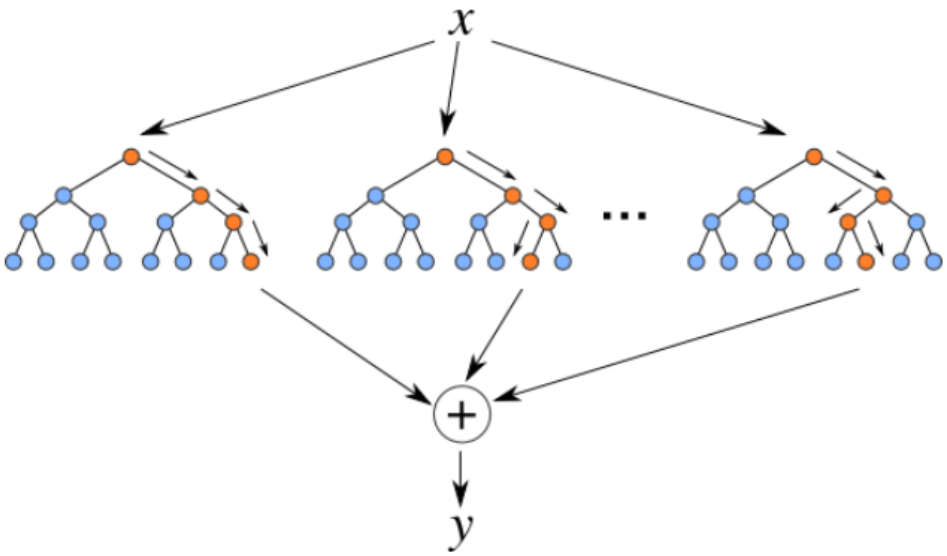
המודל מקבל כפרמטר כמה עצים הוא רוצה (N) לפי כך הוא בוחר רנדומלית כל פעם 1/N מכלל המידע ומאמן את העץ על המידע הזה. לאחר שכל העצים מוכנים, נחזה בעזרת כל אחד מהם מה הוא הפתרון. נצטרך לבחור גם באיזה דרך מחשבים פתרון מהתשובות של כל העצים.

בבעיית classification ניתן לקחת את הפתרון הנפוץ ביותר (ברגרסיה יש שיטות שונות).



Random forest (יער אקראי) הוא מטה-מסווג אשר מורכב ממספר עצי סיווג אקראיים. כל עץ סיווג נבנה תוך שימוש בתת רשימה של מאפיינים שנבחרו באקראי מתוך כלל המאפיינים. הבחירה האקראית מונעת מצבי התאמת יתר ומאפשרת בניית מודלים שמכלילים יותר טוב על נתוני זמן אמת. בשלב החיזוי התוצאה מתקבל על ידי מיצוע של התחזיות עבור כל עץ סיווג אקראי.

דוגמה ליער אקראי שהתקבל משלושה עצי סיווג אקראיים:



ניתן לחשב את חשיבותו של כל אחד מהמאפיינים השונים ביער אקראי באמצעות הרווח הממוצע המשוקלל מהמידע (אשר נמדד על בסיס מדד אנטרופי או על פי מדד ג'יני) עם משקולות פרופורציונאליות למספר התצפיות שנלקח בצומת מסוים.

הפעלת האלגוריתם על הנתונים

Random Forest Classifier

In [13]:
```
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)
```

Out[13]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

תוצאות

1. תוצאות ההרצה על כל הפרמטרים המקוריים

בחישוב המלא, על כל 54 הפרמטרים שנחקרו במחקר, הרצנו את האלגוריתם random forest מספר פעמים, עם שינויים ביחס החלוקה של הנתונים לאימון ולמבחן.

עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 40-60 בהתאמה, האלגוריתם מניב את התוצאות הבאות:

```
In [14]:  pred_rfc = rfc.predict(X_test)
          rfc.score(X_test,y_test)

Out[14]:  0.9705882352941176
```

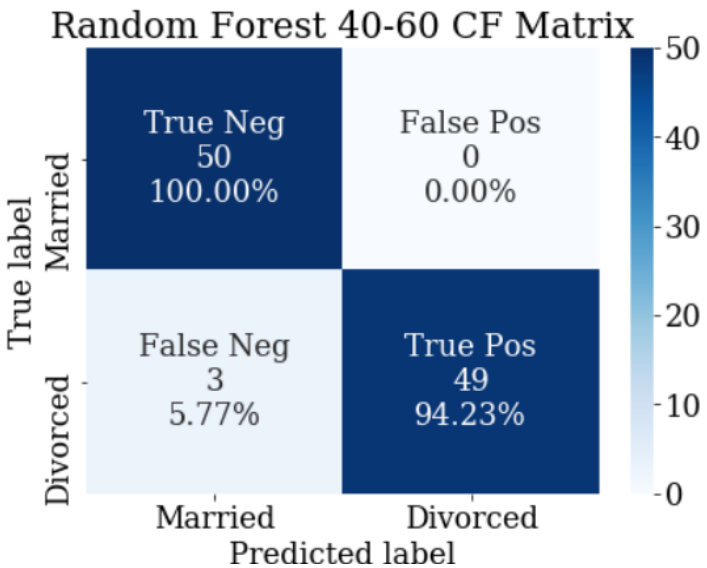
```
In [15]:  # classification report
          print(classification_report(y_test, pred_rfc))

              precision    recall  f1-score   support

           0       0.94      1.00      0.97        50
           1       1.00      0.94      0.97        52

    accuracy                           0.97       102
   macro avg       0.97      0.97      0.97       102
weighted avg       0.97      0.97      0.97       102
```

```
In [16]:  rf_cf_matrix = confusion_matrix(y_test, pred_rfc)
          make_confusion_matrix(rf_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                                title='Random Forest 40-60 CF Matrix', sum_stats=False)
```



עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 20-80 בהתאמה, האלגוריתם מניב את התוצאות הבאות:

```
In [14]:  pred_rfc = rfc.predict(X_test)
          rfc.score(X_test,y_test)

Out[14]:  0.9705882352941176
```

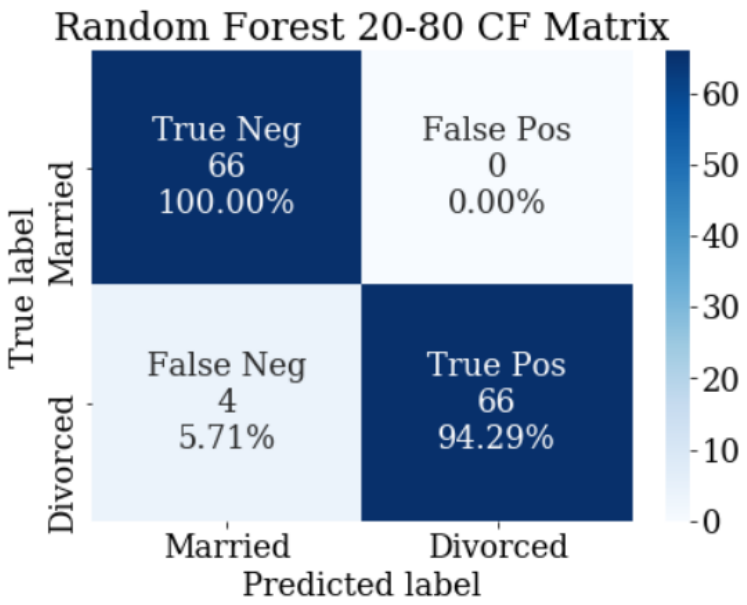
```
In [15]:  # classification report
          print(classification_report(y_test, pred_rfc))

              precision    recall  f1-score   support

           0       0.94      1.00      0.97        66
           1       1.00      0.94      0.97        70

    accuracy                           0.97       136
   macro avg       0.97      0.97      0.97       136
weighted avg       0.97      0.97      0.97       136
```

```
In [16]:  rf_cf_matrix = confusion_matrix(y_test, pred_rfc)
          make_confusion_matrix(rf_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                                title='Random Forest 20-80 CF Matrix', sum_stats=False)
```



2. תוצאות ההרצה על מספר מצומצם של הפרמטרים המקוריים

בחישוב המצומצם, על שישה פרמטרים בלבד שנמצאו להיות בעלי המשקל הגבוה ביותר בחיזוי התוצאות במחקר, הרצנו את האלגוריתם random forest מספר פעמים, עם שינויים ביחס החלוקה של הנתונים לאימון ולמבחן.

עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 40-60 בהתאמה, האלגוריתם מניב את התוצאות הבאות:

```
In [14]:  pred_rfc = rfc.predict(X_test)
          rfc.score(X_test,y_test)

Out[14]: 0.9803921568627451
```

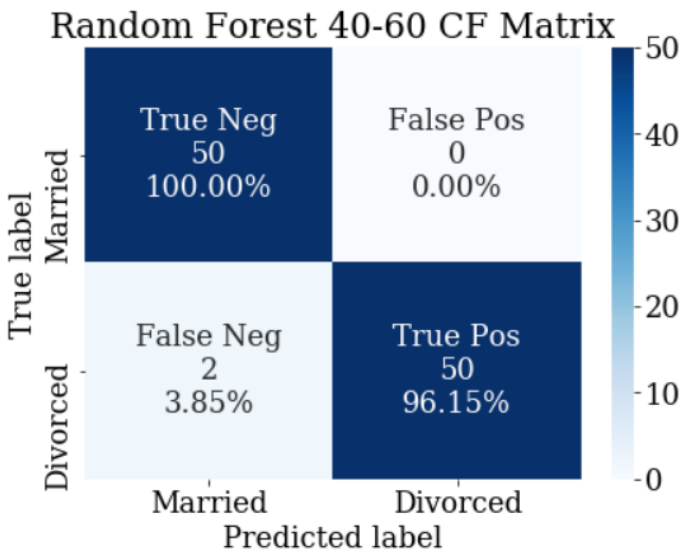
```
In [15]:  # classification report
          print(classification_report(y_test, pred_rfc))

              precision    recall  f1-score   support

           0       0.96      1.00      0.98        50
           1       1.00      0.96      0.98        52

    accuracy                           0.98       102
   macro avg       0.98      0.98      0.98       102
weighted avg       0.98      0.98      0.98       102
```

```
In [16]:  rf_cf_matrix = confusion_matrix(y_test, pred_rfc)
          make_confusion_matrix(rf_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                                title='Random Forest 40-60 CF Matrix', sum_stats=False)
```



עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 20-80 בהתאמה, עם מספר מצומצם של פרמטרים (6 בלבד) האלגוריתם מניב את התוצאות הבאות:

In [14]:
```
pred_rfc = rfc.predict(X_test)
rfc.score(X_test,y_test)
```

Out[14]: 0.9705882352941176

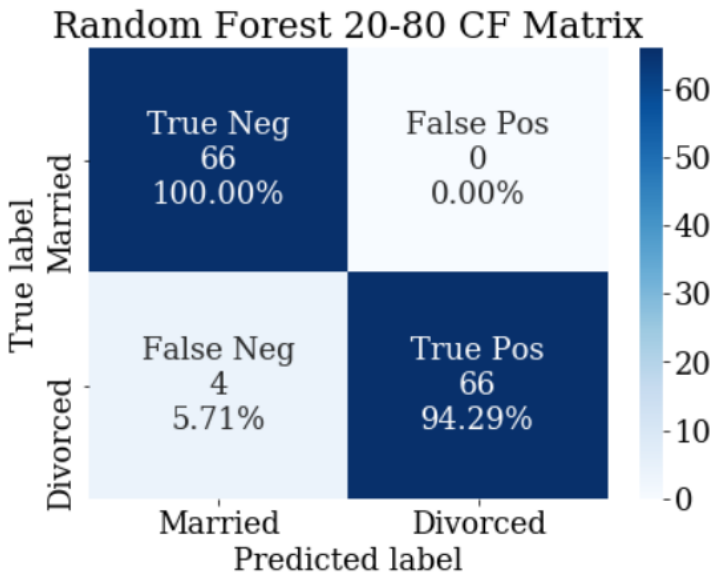
In [15]:
```
# classification report
print(classification_report(y_test, pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        66
           1       1.00      0.94      0.97        70

    accuracy                           0.97       136
   macro avg       0.97      0.97      0.97       136
weighted avg       0.97      0.97      0.97       136
```

In [16]:
```
rf_cf_matrix = confusion_matrix(y_test, pred_rfc)
make_confusion_matrix(rf_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                      title='Random Forest 20-80 CF Matrix', sum_stats=False)
```



אלגוריתם שני:

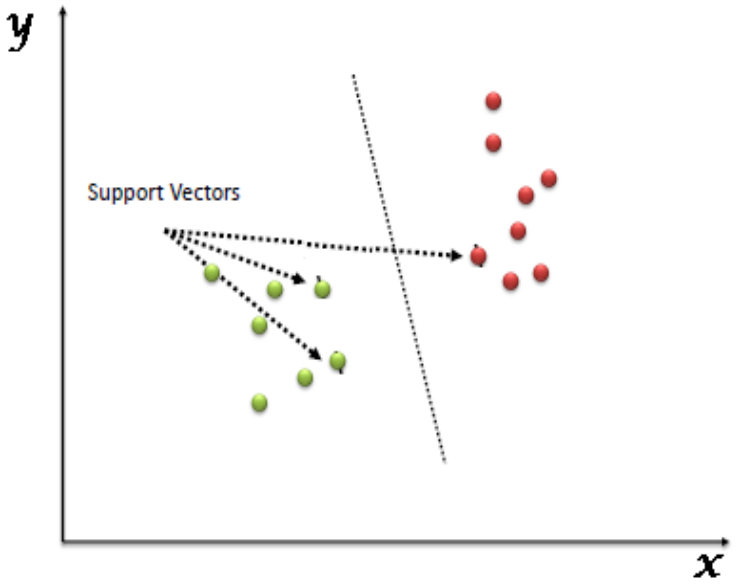
SVM – Support Vector Machine

SVM הוא אלגוריתם מסוג למידה מונחית - Supervised learning, אשר משמש לניתוח נתונים לסיווג ולרגרסיה.

האלגוריתם SVM הוא למעשה מודל של רגרסיה, שעל פיו ניתן ללמוד כיצד ערכו של המשתנה התלוי משתנה כאשר חל שינוי בערכו של אחד המשתנים הבלתי תלויים, וערכי שאר המשתנים הבלתי תלויים נשארים קבועים.

מודלים של רגרסיה משמשים לעיתים קרובות לצורך חיזוי ערכים של המשתנה התלוי במצבים שונים הנקבעים על ידי ערכי המשתנים הבלתי תלויים. SVM מאפשר סיווג של המשתנה התלוי למספר קבוצות על ידי למידה מונחית.

באלגוריתם זה, אנו מתווים כל פריט נתונים כנקודה בחלל n ממדי (n הוא מספר התכונות שיש), כאשר הערך של כל תכונה הוא הערך של קואורדינטות מסוימות. לאחר מכן, אנו מבצעים סיווג על ידי מציאת מסווג בין שני סיווגים.



בגישה של SVM מחלקים את הנקודות שקיבלנו כקלט במרחב הווקטורי, באמצעות על מישור (מרחב n-1 ממדי בתוך מרחב -n ממדי) כך שהמרחק בין אותו על-מישור המחלק את הנקודות לבין הנקודות הממוקמות הכי קרוב אליו, יהיה המרחק המקסימלי האפשרי. כלומר, ה- SVM בונה על מישור שהוא המפריד הלינארי.

מטרת ה - SVM היא להגדיל את המרחק בין המפריד הלינארי (מרחק שוליים), ולמעשה למצוא את המפריד בעל השוליים הרחבים ביותר. דוגמאות האימון המתלכדות עם מישורי השוליים נקראות וקטורים תומכים, ומכאן נגזר שם האלגוריתם.

יתרונות SVM

ביצועים מצוינים מבחינה אמפירית (בדיקות באמצעות תצפיות של חושים פיזיים או של מכשירים המרחיבים את החושים, ומבוסס על תוצאות תצפיות אלה): יישומים מוצלחים בתחומים רבים (ביו-אינפורמטיקה - חקר המידע הביולוגי באמצעות מחשב, זיהוי טקסט, זיהוי תמונות).

השימוש ב- SVM בקוד

SVM Classifier

In [17]:
```
clf = svm.SVC(kernel='rbf', class_weight='balanced')
clf.fit(X_train, y_train)
```

Out[17]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight='balanced', coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

תוצאות - SVM

1. תוצאות ההרצה על כל הפרמטרים המקוריים

בחישוב המלא, על כל 54 הפרמטרים שנחקרו במחקר, הרצנו את האלגוריתם SVM מספר פעמים, עם שינויים ביחס החלוקה של הנתונים לאימון ולמבחן.

עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 40-60 בהתאמה, האלגוריתם מניב את התוצאות הבאות:

```
In [18]:  pred_clf = clf.predict(X_test)
          clf.score(X_test,y_test)

Out[18]:  0.9705882352941176
```

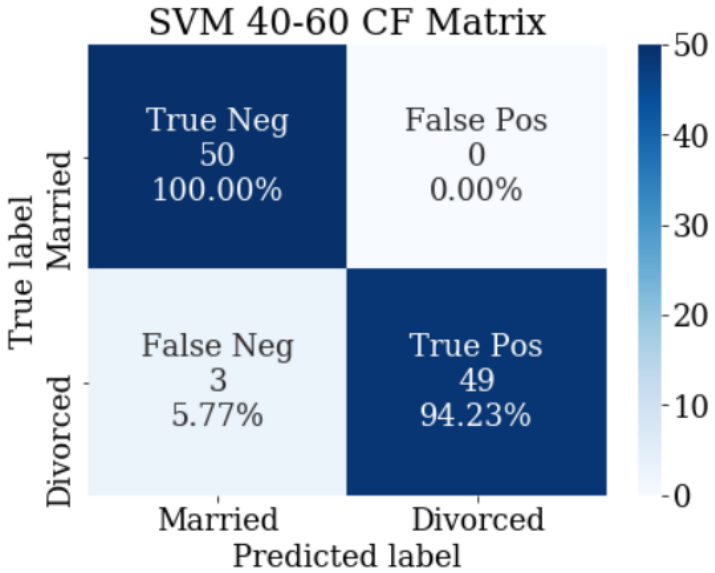
```
In [19]:  print(classification_report(y_test, pred_clf))

              precision    recall  f1-score   support

           0       0.94      1.00      0.97        50
           1       1.00      0.94      0.97        52

    accuracy                           0.97       102
   macro avg       0.97      0.97      0.97       102
weighted avg       0.97      0.97      0.97       102
```

```
In [20]:  svm_cf_matrix = confusion_matrix(y_test, pred_clf)
          make_confusion_matrix(svm_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                                title='SVM 40-60 CF Matrix', sum_stats=False)
```



עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 20-80 בהתאמה, האלגוריתם מניב את התוצאות הבאות:

```
In [18]:  pred_clf = clf.predict(X_test)
          clf.score(X_test,y_test)

Out[18]:  0.9779411764705882
```

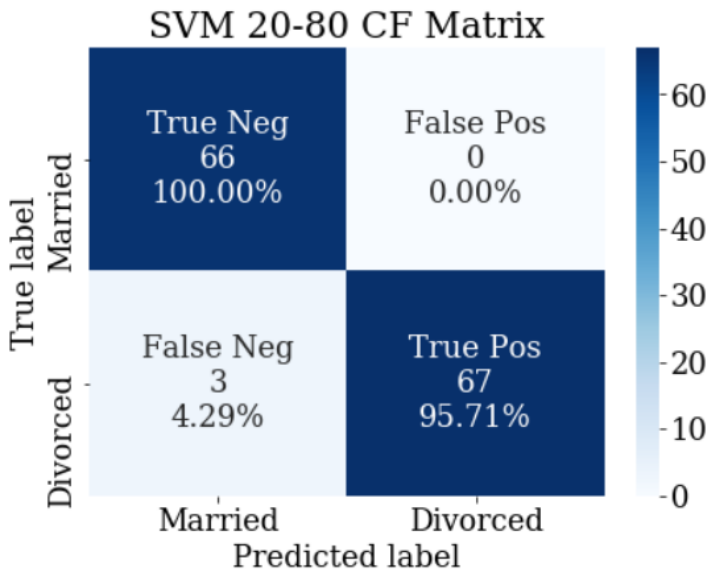
```
In [19]:  print(classification_report(y_test, pred_clf))

              precision    recall  f1-score   support

           0       0.96      1.00      0.98        66
           1       1.00      0.96      0.98        70

    accuracy                           0.98       136
   macro avg       0.98      0.98      0.98       136
weighted avg       0.98      0.98      0.98       136
```

```
In [20]:  svm_cf_matrix = confusion_matrix(y_test, pred_clf)
          make_confusion_matrix(svm_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                                title='SVM 20-80 CF Matrix', sum_stats=False)
```



2. תוצאות ההרצה על מספר מצומצם של הפרמטרים המקוריים

בחישוב המצומצם, על שישה פרמטרים בלבד שנמצאו להיות בעלי המשקל הגבוה ביותר בחיזוי התוצאות במחקר, הרצנו את האלגוריתם SVM מספר פעמים, עם שינויים ביחס החלוקה של הנתונים לאימון ולמבחן.

עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 40-60 בהתאמה, האלגוריתם מניב את התוצאות הבאות:

```
In [18]:  pred_clf = clf.predict(X_test2)
          clf.score(X_test2,y_test2)

Out[18]:  0.9803921568627451
```

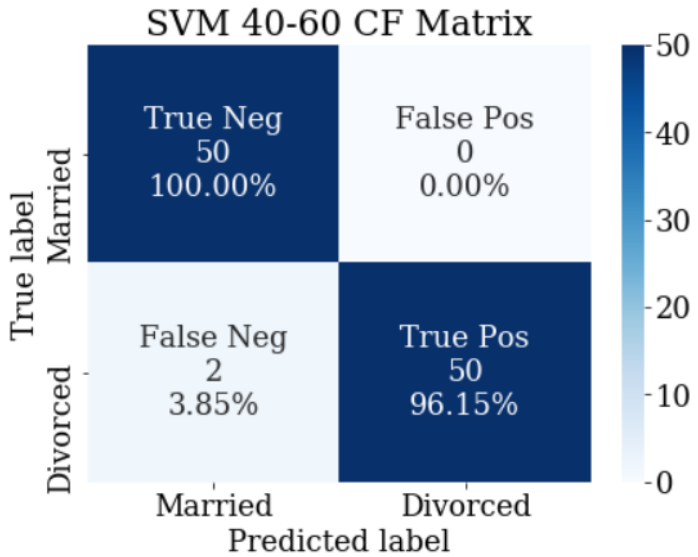
```
In [19]:  print(classification_report(y_test2, pred_clf))

              precision    recall  f1-score   support

           0       0.96      1.00      0.98        50
           1       1.00      0.96      0.98        52

    accuracy                           0.98       102
   macro avg       0.98      0.98      0.98       102
weighted avg       0.98      0.98      0.98       102
```

```
In [20]:  svm_cf_matrix = confusion_matrix(y_test2, pred_clf)
          make_confusion_matrix(svm_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                                title='SVM 40-60 CF Matrix', sum_stats=False)
```



עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 80-20 בהתאמה, עם מספר מצומצם של פרמטרים (6 בלבד) האלגוריתם מניב את התוצאות הבאות:

```
In [18]:  pred_clf = clf.predict(X_test)
          clf.score(X_test,y_test)

Out[18]:  0.9705882352941176
```

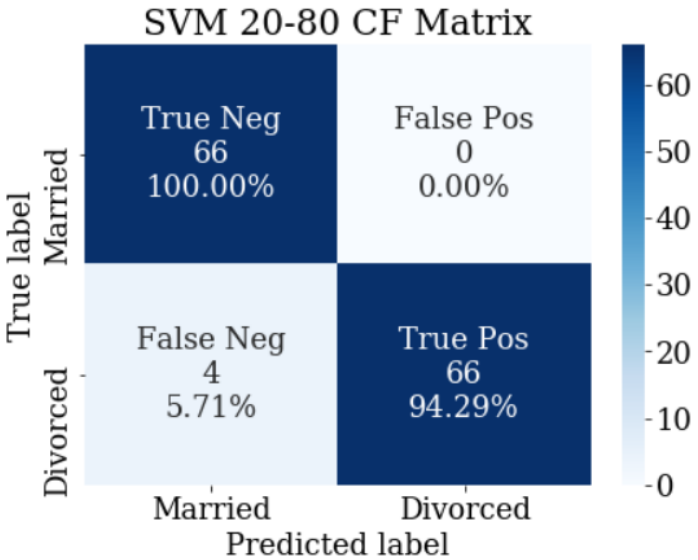
```
In [19]:  print(classification_report(y_test, pred_clf))

              precision    recall  f1-score   support

           0       0.94      1.00      0.97        66
           1       1.00      0.94      0.97        70

    accuracy                           0.97       136
   macro avg       0.97      0.97      0.97       136
weighted avg       0.97      0.97      0.97       136
```

```
In [20]:  svm_cf_matrix = confusion_matrix(y_test, pred_clf)
          make_confusion_matrix(svm_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                                title='SVM 20-80 CF Matrix', sum_stats=False)
```



אלגוריתם שלישי:

Backpropagation algorithm

אלגוריתמים זה קשור לרשת עצבית מלאכותית או רשת נוירונים הוא מודל מתמטי חישובי שפותח בהשראת תהליכים מוחיים או קוגניטיביים המתרחשים ברשת עצבית טבעית ומשמש במסגרת למידת מכונה.

רשת נוירונים לכשעצמה היא אוסף פעולות מתמטיות יחסית פשוטות. הרשת מכילה משקלים שהם למעשה הפרמטרים שקובעים מהם אותם פעולות מתמטיות. ז"א שינוי משקלי הרשת ישנה את הפעולות המתמטיות שמבצעת הרשת.

רשת מסוג זה מכילה בדרך כלל מספר רב של יחידות מידע (קלט ופלט) המקושרות זו לזו, קשרים שלעיתים קרובות עוברים דרך יחידות מידע "חבויות". צורת הקישור בין היחידות, המכילה מידע על חוזק הקשר, מדמה את אופן חיבור הנוירונים במוח.

ניתן למשל לאמן רשת נוירונים לפתרון בעיית סיווג (classification): בעיה שבה יש להחליט האם הקלט נופל תחת קטגוריה (מחלקה) מסוימת או לא.

הבעיה שבה עוסקת העבודה שלנו היא בעיית חיזוי גירושים: עבור סט נתונים המכילים מידע של אחד מבני הזוג עבור שאלות המחקר, יש לקבוע האם הוא גרוש או לא.

המשקלים (הפרמטרים) של הרשת, שבהתחלה נבחרים באקראי לאט "מתכוננים" לבצע את המשימה. זה קורה ע"י כך שבכל איטרצית אימון אנחנו נותנים לרשת פידבק חיובי אם צדקה או שלילי אם טעתה. ולאחר מספר רב של איטרציות אימון, הרשת תדע בעצמה לסווג מידע, גם כזה שלא נתקלה בו בעבר.

כלומר, אותם משקלים אקראיים משתנים להיות משקלים כאלו המתאימים לבעיית הסיווג הנתונה.

Backpropagation (חלחול או פעפוע לאחור) היא שיטה שהייתה ידועה בכל מיני גרסאות שונות עוד משנות השישים, ובשנות השמונים הייתה בשימוש בעולם רשתות הנוירונים.

האלגוריתם משווה את הפלט המתקבל לפלט הרצוי בכך שהוא מחשב את השגיאה, גוזרים אותה לפי כל משקל ומוסיפים את הנגזרת לכל משקל לקראת האיטרציה הבאה. הדרך היעילה לחשב את הנגזרות היא לחשב אותן מהמשקלים בשכבה האחרונה וללכת אחורה עד לשכבה הראשונה.

אלגוריתם Gradient descent מביא את השגיאה למינימום בכל איטרציה באמצעות מציאת מינימום מקומי לפונקציית השגיאה, ומשתמש ב-backpropagation על מנת לחשב את הנגזרות בצורה יעילה.

במהלך תהליך האימון אנו נותנים לרשת ערך הקובע עד כמה מוצלחת הרשת בתוצאותיה אשר מחושב בעזרת נקראת פונקציית ההפסד או פונקציית ההפסד (loss function) . תהליך האימון הינו למעשה שינוי המשקלים (הפרמטרים) של הרשת כך שפונקציית ההפסד תהיה מינימאלית.
כדי להביא את פונקציית ההפסד למינימום משתמשים באלגוריתם Backpropagation. הרעיון הוא לעדכן את המשקלים, בכל הרצה של סט האימון לרשת הנוירונים. עדכון המשקלים נעשה ביחס לשגיאות המחושבות עבור כל אחת מהדוגמאות בסט האימון .
הממוצע של שינויי המשקל הללו תוך כדי מעבר על סט האימון הוא הערכה של השינוי האמתי שיקרה כתוצאה משינוי משקלים המבוסס על הבאה למינימום של פונקציית ההפסד.
בכל איטרצית אימון בוחרים את השינוי המזערי המוצלח מכולם. ממשיכים כך עד שלא מצליחים לשפר יותר את ביצועי הרשת.
תהליך אופטימיזציה זה נעשה באמצעות Gradient Descent שמתואר ע"י המשוואה הבאה:

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

המשמעות של תהליך זה היא שבודקים איך כל השינויים האפשריים בכל אחד מאותם משקלים משפיעים על פונקציית ההפסד. ועושים לכל אחד מהם את אותו השינוי שהכי יקטין את פונקציית ההפסד.
כוחן הגדול של רשתות נוירונים מקורו ביכולת הלמידה וההסתגלות שלהן לסביבה.
הרעיון הכללי הוא להזין לרשת כמה שיותר דוגמאות קונקרטיות, המרכיבות את סט האימון, כאשר בכל פעם שנותנים לרשת "ללמוד" דוגמה, המשקלים של הרשת משתנים על מנת שהפלט של יתאים כמה שיותר לדוגמאות שנלמדו עד עכשיו. הדרך לעשות זו הינה להגדיר "פונקציית הפסד" אשר נמצאת ביחס ישר עם השגיאה של הרשת, כלומר כמה הרשת עובדת "רע", ולמצוא את המינימום שלה וע"י כך לגרום לרשת לעבוד "טוב" במידת האפשר.

רשת נוירונים מכילה מספר רב של יחידות מידע המקושרות זו לזו. כל שכבה בעלת מספר יחידות מידע מקבלת קלט, וקטור, כופל כל איבר שלו במשקל (תחילה רנדומלי), מבצע פונקציית אקטיבציה (סיגמואיד – מוצגת בהמשך) שמגדירה את הפלט שיכול להיכנס לשכבה הבאה בהיררכיה. ככל שיש יותר שכבות כך התוצאה תהיה מדויקת יותר. כל שכבה ברשת העצבית היא רגרסיה לוגיסטית.
במחקר שלנו הקלט יהיה התשובות של אחד מבני הזוג. הפלט המתקבל יהיה וקטור שהקואורדינטה בעלת הערך הגבוה ביותר תייצג את המחלקה (גרוש או נשוי) בו נמצא בן הזוג.

מהי רשת נוירונים

רשת נוירונים היא כלי בסיסי בתחום הלמידה הממוחשבת. עקרון הלמידה מנסה לחקות את אופן הלימוד בתוך המוח האנושי, למרות שסודות המוח האנושי עדיין לא ידועים לגמרי.

רשת נוירונים היא צירוף של רכיבי חישוב פשוטים (הנוירונים), בעלי אופי אנלוגי, אשר יוצרים מיפוי בין משתני הכניסה ליציאה.

לרשתות אלה מספר יתרונות:

- מבנה מודולרי המאפשר יכולת גידול ויצירת מיפויים חדשים.
- חישובים מקביליים המאפשרים זמן חישוב מהיר.
- יכולת לימוד.

רשתות הנוירונים המלאכותיות הנפוצות ביותר מכונות רשתות פרספטרון רב שכבתי.

שלב אימון הרשת

האימון מתבצע בצורה הבאה:

הרשת נבנית טרם תחילתו של האימון. קלט ללא סיווג מוזן לרשת ועובר בכל השכבות שציינו למעלה. בסוף התהליך מתקבלת במטריצת המוצא החלטה מה הוא סוג האובייקט ומיקומו.

לאחר הבדיקה מול הקלט המסווג המקורי, הרשת מבצעת את תהליך הלמידה- תיקון כלל המשקלים עבור כל נוירון בכל שכבה. הרשת חוזרת על תהליך זה מספר רב של פעמים עד אשר השגיאה בין התמונה שנכנסה להחלטת הרשת יחסית קטנה.

לאחר מספר קלטים רב שהזנו לרשת נקבל התכנסות לערך שגיאה מסוים.

1. פעולות העברת הקלט ברשת- forwarding

תהליך זה הוא תהליך העברת הקלט בכל שכבות הרשת עד אשר מתקבלת תוצאת החיזוי ממטריצת המוצא. את תוצאת מטריצת המוצא משווים מול הקלט המקורי שהזנו לרשת ומתקבלת השגיאה.

2. תהליך עדכון המשקלים לאחור-back propagation.

תהליך זה נועד על מנת לתקן את המשקלים של כל הנוירונים בכל השכבות. הוא מתבצע על ידי חישוב פונקציית ההפסד-loss function עבור כל נוירון מהשכבה האחרונה עד לשכבת הקונבולוציה הראשונה.

הפרספטרון

הפרספטרון הוא נוירון (רכיב חישובי בסיסי) המקיים את הקשר הבא:

$$y = \varphi\left(\sum_{i=1}^{d} w_i x_i + w_0\right)$$

כאשר:

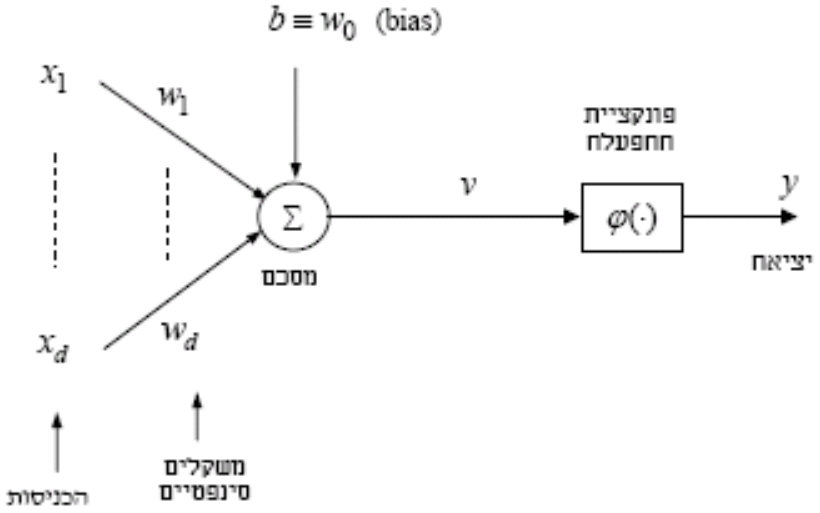
$x_1,.....,x_d$ - משתני כניסה.

y – יציאה.

$w_1,.....,w_d$ - פרמטרים של הפרספטרון, נקראים גם משקלים סינפטיים.

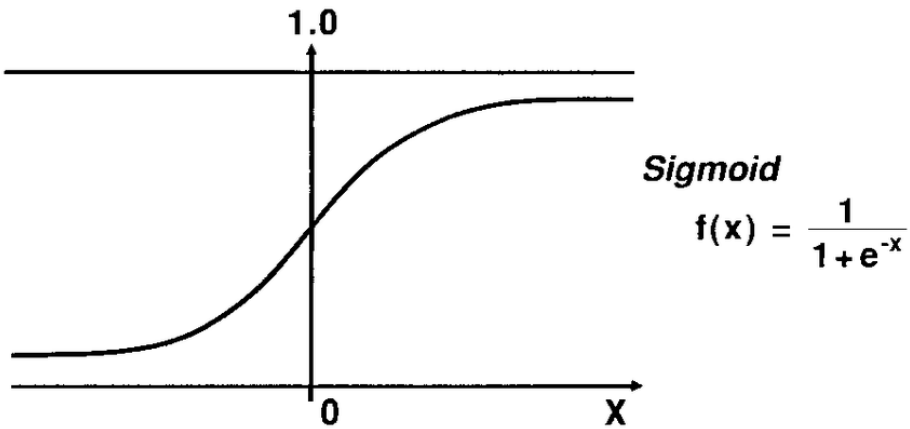
w_0 - פרמטר הטיה (bias).

φ - פונקציה לא ליניארית, נקראת פונקציית ההפעלה של הפרספטרון.



פונקציית ההפעלה φ היא פונקציה לא ליניארית, מונוטונית עולה ובעלת טווח חסום.

אנו בחרנו להשתמש בפונקציה sigmoid:



נגדיר את וקטור הפרמטרים של הנוירון כ- $\underline{w} = (w_0, w_1,..., w_d)^T$. על ידי שינוי פרמטרים אלו, נגדיר את פונקציית הנוירון.

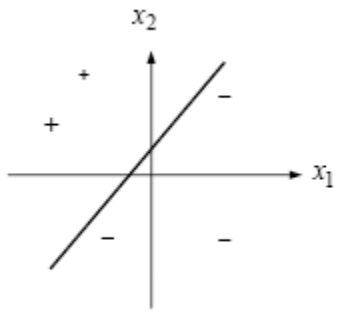
אם נגדיר את הווקטור $\underline{x} = (1, x_1, ..., x_d)$ כווקטור משתני הכניסה, נוכל לרשום את פעולת הנוירון ברישום וקטורי מקוצר כ - $y = \varphi(\underline{w}^T \underline{x})$.

לפי הגדרת הפרספטרון, הוא מוגבל בקרוב לפונקציה לינארית (עד כדי העיוות של פונקציית ההפעלה הלא לינארית).

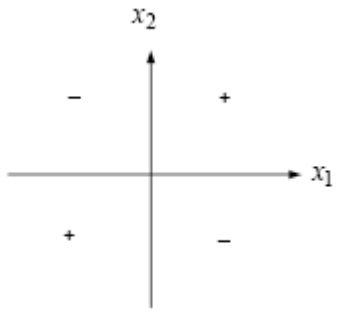
בפרויקט שלנו אנו רוצות לסווג קלט מסוים לשתי מחלקות: c_1, c_2 . נקבע את חוק ההחלטה לקביעת המחלקה לפי מוצא המסווג כ-

$$\begin{cases} y > \alpha & \Rightarrow \quad c_1 \\ y < \alpha & \Rightarrow \quad c_2 \end{cases}$$

נניח לדוגמה, כי ממד הקלט הוא 2. הפרספטרון יוכל להפריד בין המחלקות רק אם הן נמצאות משני צידי המישור.



אולם, לא תמיד נוכל לקבל תוצאה כזו מפרספטרון יחיד, לדוגמה, עבור הפונקציה XOR:



במקרה כזה נעזר בצירוף של מספר פרספטרונים – רשת נוירונים.

אלגוריתם הגרדיאנט לכוונון פרמטרי הפרספטרון

נניח כי נתונות לנו n דוגמאות מסווגות $\{x_k, d_k\}_{k=1}^{n}$ כאשר $x_k \in \Re^d$ ו d_k הוא הסיווג הנכון של x_k .

נגדיר את פונקציית המחיר הריבועית:

$$E(w)=\frac{1}{2}\sum_{k=1}^{n}e_k^{\,2}$$

כאשר $e_k=d_k-y_k=d_k-\varphi\left(\underline{w}^T\underline{x}_k\right)$.

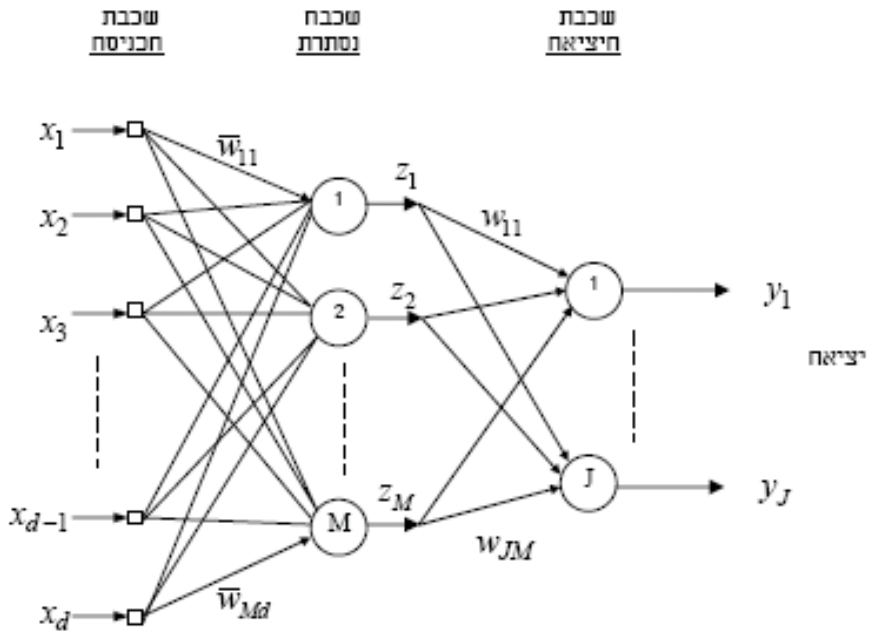
נרצה למצוא וקטור פרמטרים $\underline{w}$ שיביא למינימום את פונקציית ההפסד.

פרספטרון רב שכבתי

נדון כעת בצירוף של מספר פרספטרונים המחוברים במבנה היוצר מספר שכבות.

רשתות כאלה נקראות גם רשתות היזון קדמי (Feedforward Networks), כל שכבה מזינה את השכבה הבאה אחריה.

מבנה אופייני של רשת כזאת מתואר באיור הבא:



שכבת הביניים כוללת שכבת פרספטרונים המוזנים על ידי משתני הכניסה $(x_1,...,x_d)$.

יציאת הפרספטרון ה- m בשכבת הביניים נתונה על ידי:

$$z_m=\varphi_m\left(\overline{w}_{m0}+\sum_{i-1}^{d}\overline{w}_{mi}x_i\right)=\varphi_m\left(\overline{w}_m{}^T X\right)$$

כאשר $\overline{w}_m=\left(\overline{w}_{m0},...,\overline{w}_{md}\right)^T$ הוא וקטור המשקולות עבור פרספטרון זה, ו- $X=(1,x_1,...x_d)^T$

פונקציית ההפעלה. φ_m יכולה להשתנות מנוירון לנוירון אך בדרך כלל היא תהיה זהה בכל שכבה.

שכבת היציאה מוזנת מיציאות הפרספטרונים בשכבת הביניים. יציאת הפרספטרון ה-j בשכבת היציאה נתונה על ידי:

$$y_j = \varphi_j\left(w_{j0} + \sum_{m=1}^{M} w_{jm} z_m\right) = \varphi_j\left(w_j^{\ T} Z\right)$$

כאשר $w_j = \left(w_{j0},...,w_{jM}\right)^T$ הוא וקטור המשקולות עבור פרספטרון זה, ו- $Z = \left(1, z_1,...,z_M\right)^T$

רשת כללית לא מוגבלת לשלוש שכבות כמו בדוגמה, ייתכן מספר אחר של שכבות נסתרות.

אלגוריתם Back Propagation

כעת אנו רוצים לקבוע את ערכי פרמטרי הרשת. האלגוריתם הנפוץ ביותר הוא אלגוריתם Back Propagation. אלגוריתם זה מבוסס על אלגוריתם הגרדיאנט שהזכרנו קודם.

נניח כי נתונים לנו סדרה של n דוגמאות מסווגות $\{x_k, d_k\}_{k=1}^n$, כאשר התווית $d_k \in \Re^J$ מציינת את הערך הרצוי עבור הקלט x_k. נגדיר את θ כווקטור הפרמטרים של כל הנוירונים ברשת.

השגיאה הריבועית תוגדר על ידי

$$E(\theta) = \sum_{k=1}^{n} E_k(\theta),$$

כאשר

$$E_k(\theta) = \frac{1}{2}\left\|d_k - y_k\right\|^2 = \frac{1}{2}\left\|d_k - f(x_k)\right\|^2 = \frac{1}{2}\sum_{j=1}^{J}\left(d_{kj} - y_{kj}\right)^2$$

נחשב את הגרדיאנט לפי השכבות השונות.

שכבת היציאה:

$$\frac{\partial E_k(\theta)}{\partial w_{jm}} = -\sum_{i=1}^{J}\left(d_{ki} - y_{ki}\right)\frac{\partial}{\partial w_{jm}}\varphi_i\left(w_i^{\ T} Z_k\right)$$

עבור $i \neq j$ נקבל כי $\frac{\partial}{\partial w_{jm}}\varphi_i\left(w_i^{\ T} Z_k\right) = 0$, ולכן נקבל $\frac{\partial E_k(\theta)}{\partial w_{jm}} = -\left(d_{kj} - y_{kj}\right)\varphi_j'\left(w_j^{\ T} Z_k\right) z_{km}$.

נגדיר $\delta_{kj} = \left(d_{kj} - y_{kj}\right)\varphi_j'\left(w_j^{\ T} Z_k\right)$.

לבסוף נקבל כי $\frac{\partial E_k(\theta)}{\partial w_{jm}} = -\delta_{kj} z_{km}$

השכבה הנסתרת:

בדרך דומה ניתן להראות כי מקבלים $\frac{\partial E_k(\theta)}{\partial \overline{w}_{mi}} = -\overline{\delta}_{km} x_{ki}$

כאשר $\overline{\delta}_{km} = \varphi_m{}'\left(\overline{w}_m{}^T X_k\right)\sum_{j=1}^{J} w_{jm}\delta_{kj}$.

כעת נציב את הגרדיאנטים במשוואת עדכון הפרמטרים $\theta_{t+1} = \theta_t - \eta \frac{\partial E(\theta)}{\partial \theta}$ ונקבל משוואות לקבלת הפרמטרים עבור כל שכבה:

$$\begin{cases} w_{jm} = w_{jm} + \eta \sum_{k=1}^{n} \delta_{kj} z_{km} \\ \overline{w}_{mi} = \overline{w}_{mi} + \eta \sum_{k=1}^{n} \overline{\delta}_{km} x_{ki} \end{cases}$$

הפרמטר η נקרא גודל הצעד (זהו למעשה הגרדיאנט – הנגזרת הכיוונית). בחירה בגודל צעד גדול מדי יביא לאי יציבות של האלגוריתם, אך צעד קטן מדי יגרום לאיטיות של ההתכנסות, ויגדיל את הסיכוי להיעצר על מינימום מקומי.

לסיכום, שלבי הפעולה הסטנדרטיים באימון רשת נוירונים הם:

- בשלב הראשון עוברים forward על הרשת להוצאת הניחוש.
- אחר כך מחשבים את השגיאה של הניחוש.
- גוזרים את השגיאה לפי הפלטים.
- גוזרים בצורה איטרטיבית משכבה לשכבה אחורה, את השגיאה לפי המשקלים בכל שכבה . הגזירה נעשית לפי כלל השרשרת, והשימוש בו עוזר להעביר את הנגזרות "אחורה" ברשת.
- השלב האחרון לאחר קבלת הנגזרות לפי כל משתנה הוא העדכון, לפי descent gradient.

תוצאות

1. תוצאות ההרצה על כל הפרמטרים המקוריים

בחישוב המלא, על כל 54 הפרמטרים שנחקרו במחקר, הרצנו את האלגוריתם backpropagation מספר פעמים, עם שינויים ביחס החלוקה של הנתונים לאימון ולמבחן.

עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 40-60 בהתאמה, האלגוריתם מניב את התוצאות הבאות:

```
In [22]:  pred_mlpc = mlpc.predict(X_test)
          accuracy_score(y_test, pred_mlpc)

Out[22]:  0.9705882352941176
```

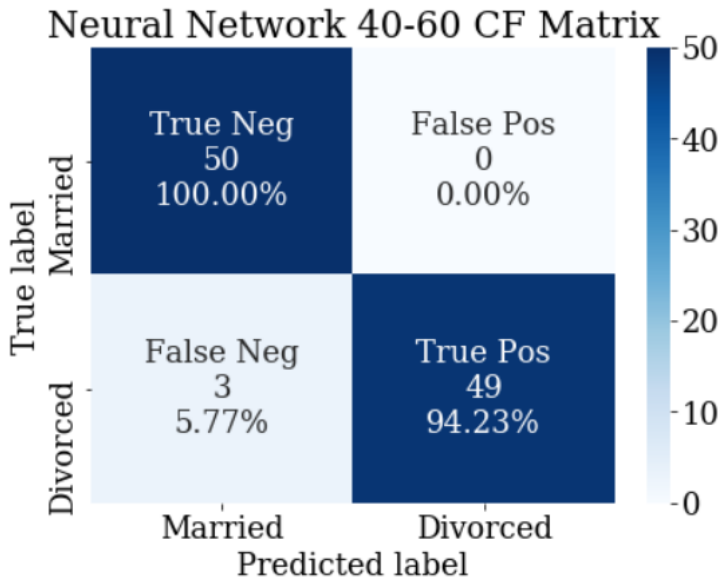
```
In [23]:  print(classification_report(y_test, pred_mlpc))

              precision    recall  f1-score   support

           0       0.94      1.00      0.97        50
           1       1.00      0.94      0.97        52

    accuracy                           0.97       102
   macro avg       0.97      0.97      0.97       102
weighted avg       0.97      0.97      0.97       102
```

```
In [24]:  nn_cf_matrix = confusion_matrix(y_test, pred_mlpc)
          make_confusion_matrix(nn_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                                title='Neural Network 40-60 CF Matrix', sum_stats=False)
```



עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 20-80 בהתאמה, האלגוריתם מניב את התוצאות הבאות:

```
In [22]:  pred_mlpc = mlpc.predict(X_test)
          accuracy_score(y_test, pred_mlpc)

Out[22]:  0.9705882352941176
```

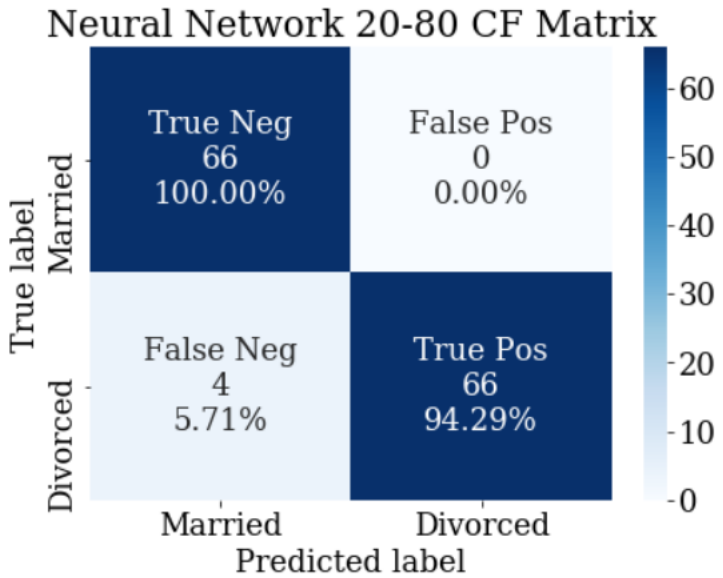
```
In [23]:  print(classification_report(y_test, pred_mlpc))

              precision    recall  f1-score   support

           0       0.94      1.00      0.97        66
           1       1.00      0.94      0.97        70

    accuracy                           0.97       136
   macro avg       0.97      0.97      0.97       136
weighted avg       0.97      0.97      0.97       136
```

```
In [24]:  nn_cf_matrix = confusion_matrix(y_test, pred_mlpc)
          make_confusion_matrix(nn_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                                title='Neural Network 20-80 CF Matrix', sum_stats=False)
```



2. תוצאות ההרצה על מספר מצומצם של הפרמטרים המקוריים

בחישוב המצומצם, על שישה פרמטרים בלבד שנמצאו להיות בעלי המשקל הגבוה ביותר בחיזוי התוצאות במחקר, הרצנו את האלגוריתם backpropagation מספר פעמים, עם שינויים ביחס החלוקה של הנתונים לאימון ולמבחן.

עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 40-60 בהתאמה, האלגוריתם מניב את התוצאות הבאות:

In [22]:
```
pred_mlpc = mlpc.predict(X_test3)
mlpc.score(X_test3, y_test3)
accuracy_score(y_test3, pred_mlpc)
```

Out[22]:
```
0.9803921568627451
```

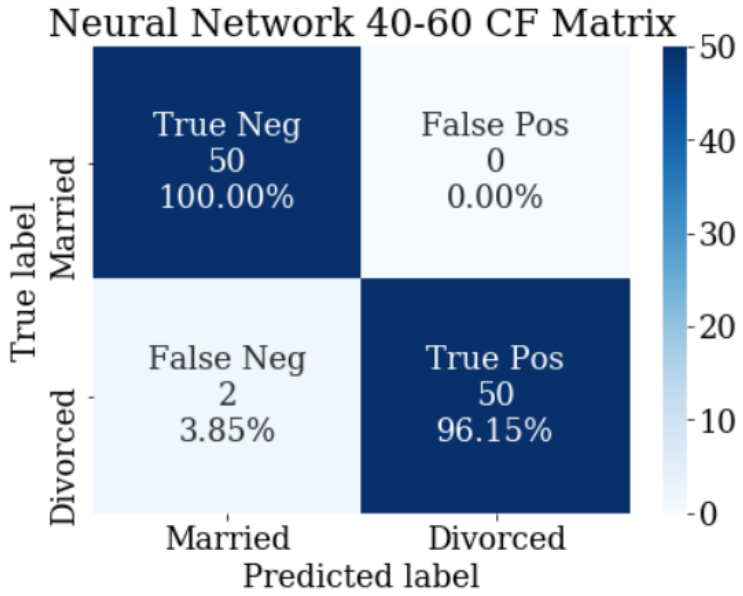
In [23]:
```
print(classification_report(y_test3, pred_mlpc))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        50
           1       1.00      0.96      0.98        52

    accuracy                           0.98       102
   macro avg       0.98      0.98      0.98       102
weighted avg       0.98      0.98      0.98       102
```

In [24]:
```
nn_cf_matrix = confusion_matrix(y_test3, pred_mlpc)
make_confusion_matrix(nn_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                      title='Neural Network 40-60 CF Matrix', sum_stats=False)
```



עבור חלוקת נתוני האימון ונתוני המבחן ביחס של 20-80 בהתאמה, עם מספר מצומצם של פרמטרים (6 בלבד) האלגוריתם מניב את התוצאות הבאות:

```
In [22]:  pred_mlpc = mlpc.predict(X_test)
          accuracy_score(y_test, pred_mlpc)

Out[22]:  0.9779411764705882
```

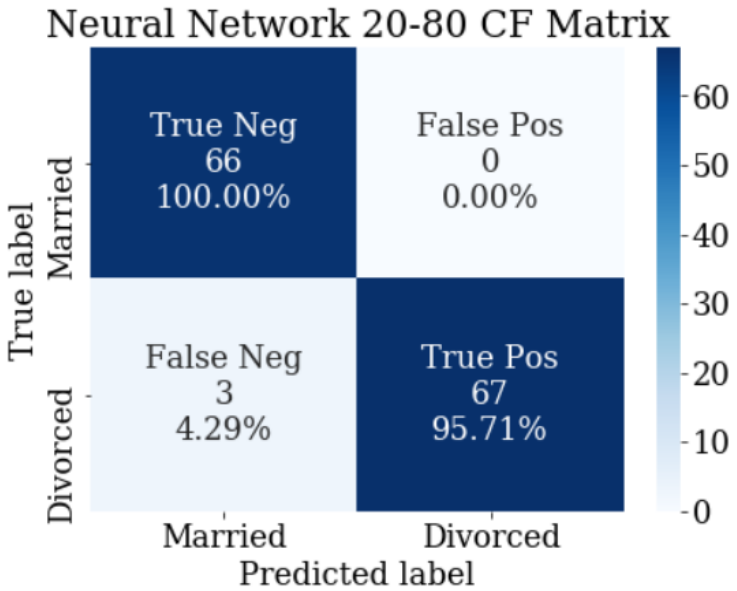
```
In [23]:  print(classification_report(y_test, pred_mlpc))

              precision    recall  f1-score   support

           0       0.96      1.00      0.98        66
           1       1.00      0.96      0.98        70

    accuracy                           0.98       136
   macro avg       0.98      0.98      0.98       136
weighted avg       0.98      0.98      0.98       136
```

```
In [24]:  nn_cf_matrix = confusion_matrix(y_test, pred_mlpc)
          make_confusion_matrix(nn_cf_matrix, group_names=LABELS, categories=CATEGORIES,
                                title='Neural Network 20-80 CF Matrix', sum_stats=False)
```



מסקנות

המחקר נעשה על זוגות נשואים וגרושים החיים בטורקיה, המדינה במקום ה-12 בדירוג המדינות בעלות שיעור הגירושים הגבוה ביותר (מתוך 33 מדינות).

המחקר שנעשה השתמש בסקאלה לחיזוי גירושים (Divorce Predictors Scale) המבוסס על תיאוריית "התרפיה לזוגות" של גוטמן (2012-2014). לפי תיאוריה זו, ארבעת הפרמטרים החשובים ביותר לחיזוי גירושים הם: ביקורת, זלזול, חסימה ריגשת והתגוננות. גוטמן גילה שהשילוב של ארבעת פרמטרים אלה וזה גירושים עם שיעור הצלחה של כ-90%. לפי גוטמן, פרמטר נוסף שחוזה גירושים הוא כישלון של ניסיונות לתיקון.

6 הפרמטרים בעלי ההשפעה הגדולה ביותר על חיזוי הסיכוי לגירושים הם:

1. I know we can ignore our differences, even if things get hard sometimes.
2. We don't have time at home as partners.
3. I think that one day in the future, when I look back, I see that my wife and I are in harmony with each other.
4. My wife and I have similar ideas about how marriage should be.
5. I know my wife's basic concerns.
6. We're just starting a fight before I know what's going on.

כאשר אנו בוחנות את 6 הפרמטרים הללו, ניתן לראות כי הם קשורים ליצירת משמעות משותפת וניסיונות כושלים לתקן, לאמונה בזוגיות ולהתנהגויות קונפליקט שליליות.

בהשוואה לתוצאות שהתקבלו במחקר המקורי שנעשה, הצלחנו להשתוות אליהם בתוצאות החיזוי המוצלחים ואף להגיע לתוצאות טובות יותר, בחלק מהמקרים.

צמצום מספר הפרמטרים שנבחנו שיפר את תוצאות החיזוי עבור שלושת האלגוריתמים השונים, בשיעורים שונים.

כמו כן, ההבדלים בתוצאות של האלגוריתמים השונים הם כמעט ולא ניכרים כלל. עם זאת מצאנו שהאלגוריתם בעלי יכולות החיזוי המוצלחות ביותר הוא אלגוריתם רשת הנוירונים.

ההיבט האנושי

בתחילת העבודה כאשר בחרנו את הנושא, לא חשבנו שנצליח להגיע לתוצאות סופיות ברורות וחד משמעיות כפי שהצלחנו להשיג.
בעידן הטכנולוגי של ימינו, אפילו את התחום האישי והריגשי ביותר לא ניתן להפריד מעולם הבינה המלאכותית ולמידת מכונה. כבר היום ניתן לראות באפליקציות היכרויות פופולריות רבות את הניסיון לחזות באמצעות שאלות אישיות את סיכויי ההצלחה של הקשר בין שני בני הזוג הפוטנציאליים.
בחרנו בנושא הרלוונטי ביותר כנראה עבור פרק זמן זה של החיים (במידה ואנחנו לא נשואים כבר).
ולכן אין זה מפתיע כי אנחנו שומעים מסביבנו בכל מקום על זוגות שהכירו באפליקציות למטרה זו והיום הם כבר נשואים. תוצאות המחקר יכולות להוות פריצת דרך בתחום האפליקציות להיכרויות.

בנוסף, הפרויקט שבחרנו יכול להשפיע על מערכות יחסים בחיים האמיתיים, תלוי באופן השימוש בתוצאות, במיוחד אם יועצי נישואין יכללו זאת ככלי בעבודתם. למרבה הצער, מישהו עשוי להיות מושפע לשקול גירושין שלא היו צריכים או שבני זוג אולי לא יתחתנו שחשבו על זה, אם התחזית מצביעה על כך שנישואיהם לא סבירים שיימשך. מצד שני, זה יכול לספק תמיכה בשירותי הכרויות המספקים את מי שמחפש קשרים ארוכי טווח על ידי זיהוי גורמים אישיים התורמים לאריכות החיים של הנישואין.

הניסוי אותו הדגמנו בסמינר התבצע בטורקיה.
לשם השוואה נראה סטטיסטיקות ונתונים מישראל בנושא המחקר בסמינר על מנת להציג תמונה רחבה של הנושא.
על פי נתוני הלשכה המרכזית לסטטיסטיקה (הלמ"ס) שיעור הגירושין בישראל נע בין 26% -27% בהשוואה למדינות האיחוד האירופי ששם שיעור הגירושים עומד על כ-35%.
ניתן לראות הבדל במספרים בין האוכלוסיות השונות החיות בישראל. (לכל אלף תושבים - 1.9% יהודים, 1.1% מוסלמים, 0.5% נוצרים -ו% 0.8 דרוזים)
ישנה השפעה של התרבות והדת על אחוזי הגירושין בקרב האוכלוסיות הללו.

מחקר שבחן משתנים המשפיעים על הסיכויים של זוג להתגרש מעלה כי השכלת האישה, ותק בנישואין, בעלות על דירה ונישואין של בני זוג ממוצא זהה מקטינים את הסיכויים של 10 נישואין להסתיים בגירושין.
נראה גם אחוזי גירושין בספרד בהתאם לתרבות וההגבלות שהיו בשנים מסוימות.
למשל, עד שנת 1981 גירושין בספרד לא היו אפשריים כלל. החל מ-1981 הגירושין בספרד היו חוקיים, אך עם מגבלות שונות. החל משנת 2005 הוסרו המגבלות על גירושין בספרד, וכתוצאה מכך שיעור הגירושין במדינה עלה בצורה ניכרת.

נוסף על כך, בתחילת שנות ה-90 עלו שיעורי הגירושין במזרח אירופה לאחר נפילת המשטרים הקומוניסטיים.

ניתן לומר באופן כללי (ובהתאם למשתנים שונים) כי ישנה עליה באחוזי הגירושין באירופה במהלך השנים (וזאת לצד ירידה משמעותית באחוזי הנישואין באירופה).

מקורות

1. https://dergipark.org.tr/en/download/article-file/748448
2. https://dergipark.org.tr/en/pub/nevsosbilen/issue/46568/549416
3. https://www.edureka.co/blog/artificial-intelligence-algorithms/
4. https://data-flair.training/blogs/ai-and-machine-learning/
5. https://www.upgrad.com/blog/types-of-artificial-intelligence-algorithms/
6. https://dzone.com/articles/data-science-vs-artificial-intelligence-vs-machine?fromrel=true
7. https://en.wikipedia.org/wiki/Support-vector_machine
8. https://www.gottman.com/blog/the-four-horsemen-stonewalling/
9. https://scikit-learn.org/stable/modules/neural_networks_supervised.html