# פרוייקט גמר

# למה בחרנו את הדאטה: בחרנו בדאטה של מדד האוניברסיטאות כדי לגלות מה המדדים המשמעותיים ביחס לדירוג האוניברסיטאות בעולם.

ננסה לחזות את דירוג האוניברסיטה ע"פ הנתונים בשנת 2016, וע"י כך נוכל ללמוד מה הם המדדים שהכי השפיעו על הדירוג.

הנושא מעניין כמובן גם בראי אוניברסיטת בר אילן.

תיאור הדאטה:

הדאטה מורכב מטבלה בעלת 800 שורות, ו-13 עמודות. בכל שורה נמצאת מדידה של אוניברסיטה אחרת בשנת 2016. בעמודות נמצאים המדדים, ובעמודה הראשונה נמצא הדירוג הכולל.

עיבוד הדאטה:

* לאחר סינון תאים ריקים ותאים שהכילו מקף - מספר השורות ירד ל 702.
* חלק מהעמודות הכילו ערכים שלא הועברו לערך מספרי ישירות, כמו 73% ו- "1,223", לכן העברנו אותם ידנית לערכים מספריים

הסבר תוכן העמודות:

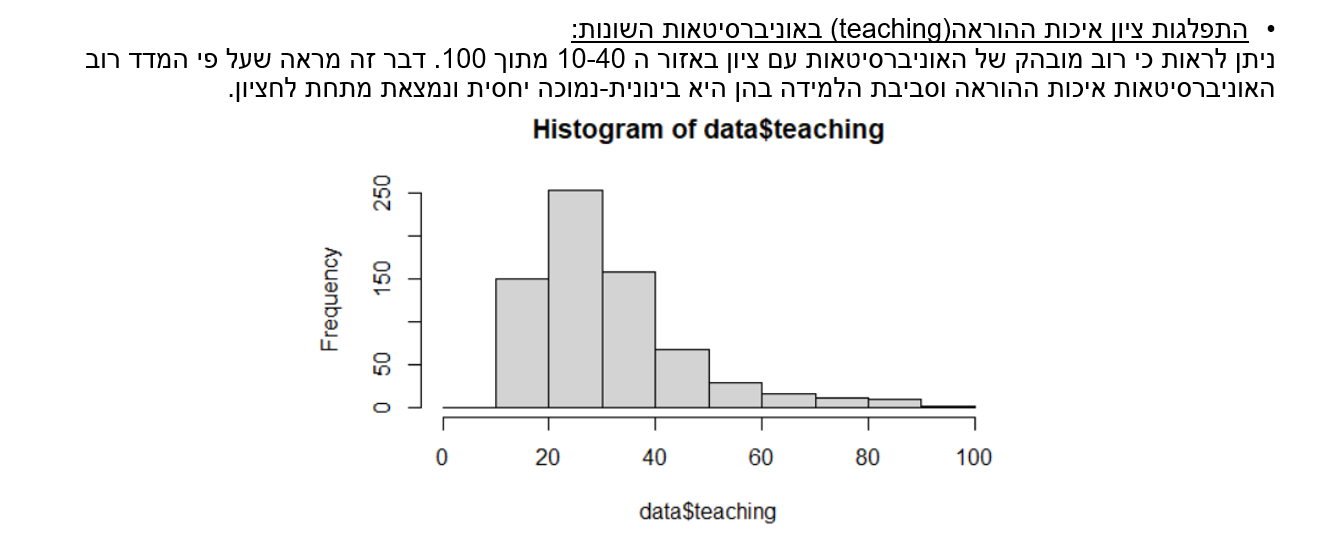
1. world\_rank = מדד עולמי של האוניברסיטה
2. university\_name = שם האוניברסיטה
3. country = המדינה בה האוניברסיטה ממוקמת
4. international = ציון עבור גיוון של אנשים ממדינות שונות (סטודנטים, אנשי סגל ומחקר)
5. teaching =ציון עבור איכות למידה (סביבת הלמידה)
6. research = ציון עבור רמת מחקר(נפח, תקצוב ומוניטין)
7. citations = ציון עבור עד כמה מחקרי האוניברסיטה מצוטטים(השפעת מחקרי האוניברסיטה על העולם)
8. income = ציון האוניברסיטה להכנסה בתעשייה(העברת ידע)
9. num\_students = מספר הסטודנטים באוניברסיטה
10. student\_staff\_ratio = יחס בין מספר הסטודנטים למספר אנשי הסגל
11. international\_students = (אחוז הסטודנטים הבינלאומיים (שהיגרו ממדינה אחרת
12. female\_male\_ratio = יחס בין מספר הנשים למספר הגברים באוניברסיטה
13. year = שנת המדידה

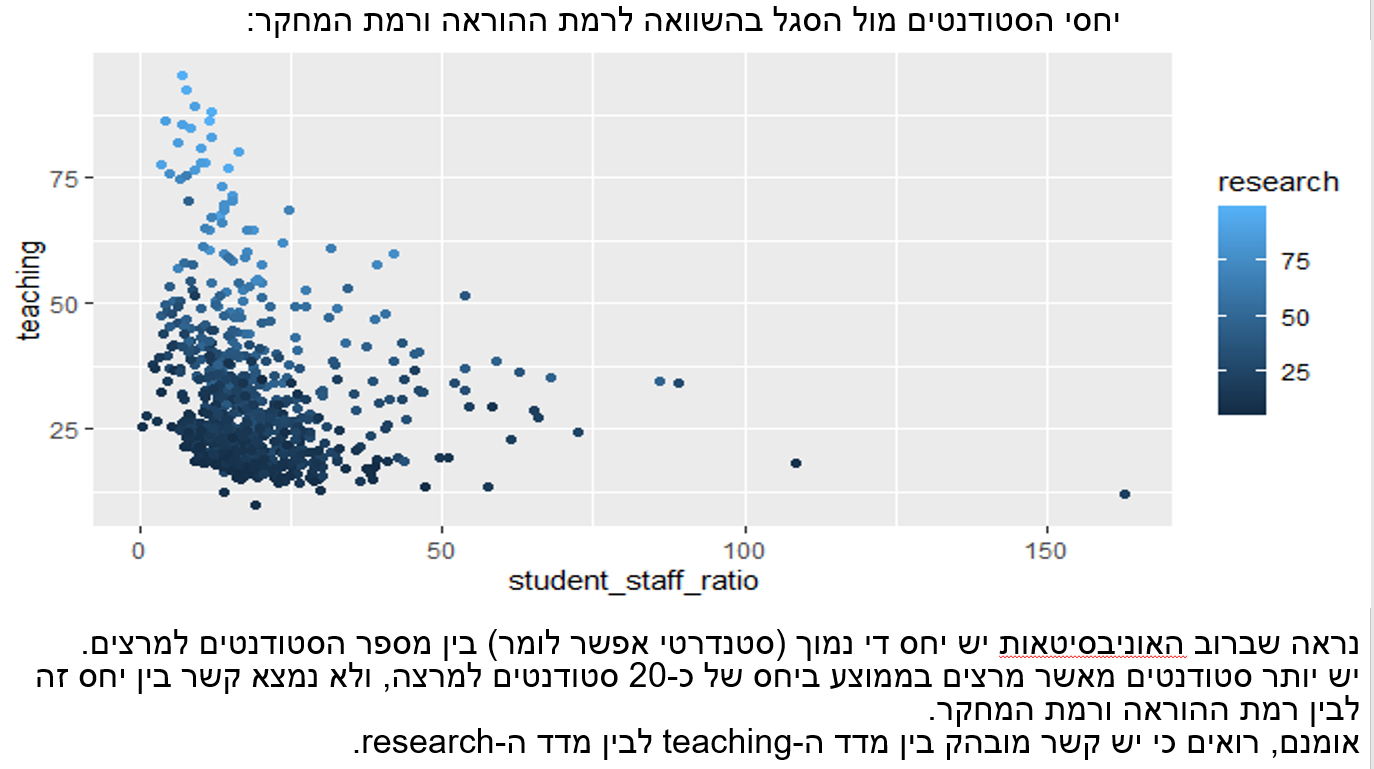
**EDA**

נתונים וגרפים על הדאטה:

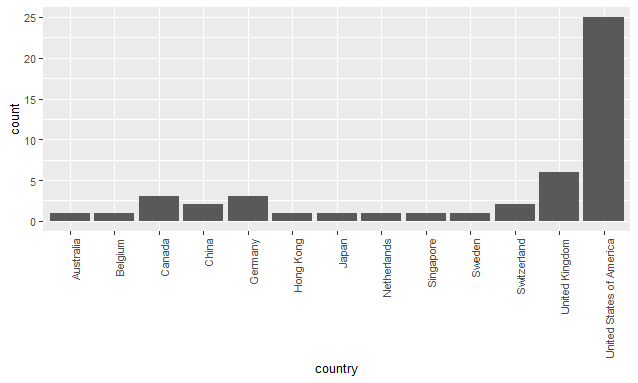
התפלגות ציון איכות ההוראה באוניברסיטאות השונות:







מאילו מדינות מגיעות 50 האוניברסיטאות הכי טובות?



אלגוריתמים:

**עיבוד מקדים:** לשם ה-classification ויצירת labels, ביצענו חלוקה לקבוצות שונות לפי דירוג כל אוניברסיטה בצורה הבאה (השתמשנו בצורת הדירוג הזו לשאר סוגי האלגוריתמים להלן):

יצירת ה-labels:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| דירוג: | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| טווח: | 1 - 10 | 11 - 50 | 51 - 100 | 101 - 150 | 151 - 200 | 201 - 253 | 254 - 302 | 303 - 350 | 351 - 401 | 402 - 600 | 601 - 800 |

(טווח - כל האוניברסיטאות בטווח הדירוג 1-10 קיבלו label 10, המייצג את 10 האוני' הטובות ביותר)

אחרי שניקינו, הרצנו את אלגוריתמים שונים על ה-data בניסיון לחזות את ה- rank, תוצאות האלגוריתמים:

Decision Tree:

**עיבוד מקדים:**

ביצעתי את הלמידה ללא העמודות של ה-world\_rank (אותה חוזים, אשר על כן לא רלוונטי ללמוד עליה),  
עמודת ה-year (זאת השנה שאנו חוזים בה את הנתונים),  
וכן עמודת ה-female\_male\_ratio, כיוון שיש בעיה בלא הצלחנו להתגבר עליה נכון לעכשיו (המספרים שם היו בפורמט שגרם ל-excel לחשוב שמדובר בתאריך, אחרי הרבה זמן שעבדתי כדי לשנות את זה למספר המקורי, המספרים שם נשארו כ-chars ולא כ-int, ניסיונות ארוכים להמרת העמודה\המרת היחס לתוכן העמודה העלו חרס בשלב זה).

התוצאות של עץ ההחלטה היו טובות ביותר עבור עומק 13 (העומק המקסימלי).

**אסטרטגיית הלמידה**:

בוצעו 300 אפוקים של למידה, בכל אפוק שהייתה תוצאה טובה יותר מה-current\_best, נשמרו המשקלים והדיוק החזויים במקומם.

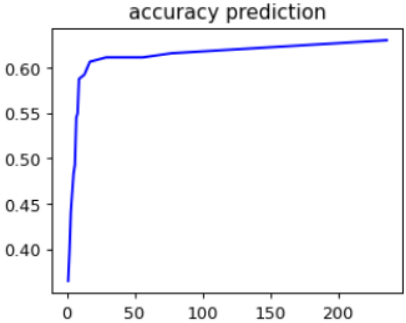
**התוצאות:**

Accuracy: 0.6303317535545023

Weights: [0.01947872 0.01354113 **0.16065597** 0.02119478 **0.28136656** **0.37019189**

0.04502295 0.01782069 0.03941616 0.03131115]

משקלי העמודות לעיל בהתאמה משמאל לימין:

university\_name, country, **teaching**, international, **research**, **citation**, income, num\_students, student\_staff\_ratio, international\_students.

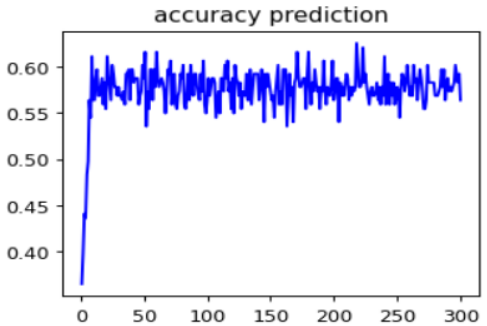
**ויזואליזציות:**

מצורך משמאל גרף למידה, ציר Y הוא ה-accuracy, ציר X זה מספר האפוקים:

גרף הלמידה המוצג הוא עבור שיפורים בלבד, לכן רואים רק עלייה.

גרף עבור כלל ה accuracy בתהליך הלמידה:

**דיון:**

כפי שניתן לראות, מודגש המדדים המשפיעים ביותר על הדירוג, התוצאות בקורלציה טובה מאוד עם ה heat-map שבוצע על הדאטה. נציין כי נתון גבוה יחסית מהשאר הוא מספר הסטודנטים, שמראה כי אוניברסיטאות טובות יותר מושכות אליהן יותר סטודנטים, דבר המעיד על הרמת האוניברסיטה כ-"אפקט העדר".

SVM:

**עיבוד מקדים:**

המרתי את תוכן העמודות של ה country וה-university name מ-chars לייצוג מספרי.

הורדתי מהלמידה את עמודות:

female\_male\_ratio (מכיוון שלא הצלחתי להמיר לייצוג מספרי כפי שהוסבר באלגוריתם decision tree, ולכן האלגוריתם לא הצליח לרוץ)

עמודת ה-year, שגם בה אין צורך.

ה-world\_rank כרגיל הוא ה-class החזוי ב-KSVM.

כהכנה נוספת ללמידה, ביצעתי ערבוב (shuffle) של השורות ב-data.

**אסטרטגיית הלמידה**:

חלוקת הdata נעשתה לאחר shuffle כאמור, ה-test קיבל 116 רצפים, האימון את כל השאר (584).

נבנו שני מודלים של SVM, הראשון עבורו הקרנל הוגדר vanilladot, מפריד לינארי,

לשני rbfdot, לא מפריד לינארי (לכן נצפה להצלחה גבוהה יותר).

אימנתי את שניהם, וקיבלתי data\_classifier, בו השתמשתי כדי ליצור את טבלת ההסכמה של חיזוי world\_rank של ה-test מול ה-world\_rank האמיתי.

**התוצאות:**

**עבור vanilladot**:

טבלת הסכמה:

FALSE TRUE

43 74

באחוזים:

FALSE TRUE

0.3675214 0.6324786

**עבור rbfdot**:

טבלת ההסכמה:

FALSE TRUE

22 95

באחוזים:

FALSE TRUE

0.1880342 0.8119658

לאחר הצלחה יחסית טובה במודל השני, ניסיתי בkernel להשתמש ב"radial" וקיבלתי תוצאות טובות מאוד:

טבלת ההסכמה:

FALSE TRUE

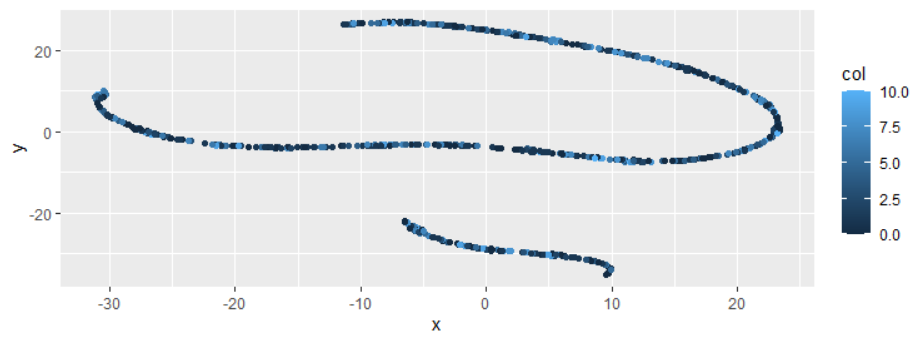
7 110

באחוזים:

FALSE TRUE

0.05982906 0.94017094

**דיון:** המודל הלא לינארי, הצליח בשיעור ניכר הרבה יותר מאשר הלינארי, דבר המצביע על כך שהדאטה פריד בצורה טובה יותר משמעותית לא לינארית.



אכן, ניתן לראות זאת בפלט של אלגוריתם T-sne בתמונה הבאה:

אמנם נוכל להגיע להפרדה מסוימת בקו לינארי כפי שהתוצאות הראו, אך בקו לא לינארי נצליח יותר. מבין ה-kernels הלא לינאריים, ה-"radial" הצליח מאוד, דבר שמראה שהדאטה סה"כ פריד בצורה לא לינארית בצורה טובה.

Random forest

יישמנו אלגוריתם Random forest בעל 1000 עצים, כדי לחזות את ציון האוניברסיטאות.  
**עיבוד מקדים:**

ביצענו shuffling לדאטא לפני שביצענו split לאימון וtest.**.** הדאטא חולק על ידי פונקציית initial\_split ללא העמודות הבאות:  
,-university\_name, -country,-female\_male\_ratio, -year  
הסיבה לכך היא שלמשל university\_name וcountry הן עמודות שלא נכנסות לחישוב הציון (למירב ידיעתינו ועל פי בדיקה מקדימה באינטרנט של אופן חישוב הציון במדד זה). הסרנו את עמודת female\_male\_ratio משום שהיוותה בעיה במהלך הריצה של האלגוריתמים, זוהתה כטיפוס char במקום int למרות שניסינו לתקן זאת, ותיקנו את הערכים(המרנו לעשרוני) - ועדיין זה לא עבד. כך או כך, עמודה זו לא העמודה העיקרית בחישוב הציון גם, אז החלטנו להסירה מהחישוב באלגוריתם.

כמו בשיטות הקודמות, הדאטא סונן לשנת 2016 וסיננו מהדאטא NA's. world\_rank - הקלאס שנרצה לחזות.

**אסטרטגיית למידה:**  
קבענו 1000 עצים עבור המודל שלנו על ידי שימוש בחבילת Parsnip:

parsnip model object

Fit time: 571ms

Ranger result

Call:

ranger::ranger(x = maybe\_data\_frame(x), y = y, num.trees = ~1000, num.threads = 1, verbose = FALSE, seed = sample.int(10^5, 1), probability = TRUE)

Type: Probability estimation

Number of trees: 1000

Sample size: 525

Number of independent variables: 8

Mtry: 2

Target node size: 10

Variable importance mode: none

Splitrule: gini

OOB prediction error (Brier s.): 0.328784

לאחר מכן, בדקנו את הביצועים של האימון במודל על פי מדדי ROC וAccuracy והשתמשנו בחבילת yardstick בכדי לחשב את מדדים אלה:

# A tibble: 1 x 3

.metric .estimator .estimate

<chr> <chr> <dbl>

roc\_auc hand\_till 1

# A tibble: 1 x 3

.metric .estimator .estimate

<chr> <chr> <dbl>

accuracy multiclass 0.998

תוצאות ביצועי האימון יצאו overfitting. ניסינו להוריד את כמות העצים כדי לנסות להוריד את הסבירות לoverfit אך הניסיון לא צלח. ככל הנראה הסיבה לכך היא שכמות הדוגמאות שלנו עליה האלגוריתם התאמן מעטה מידי(בכל זאת יש גבול למספר האוניברסיטאות בעולם). בנוסף, סיבה מהותית גם כן היא שהאלגוריתם עצמו יכולים לזכור בקלות יותר את דאטא בסט האימון ולכן אם ניתן לו לחזות על פי הדאטא בסט האימון אז סיכוי גדול שנקבל overfit.  
נבדוק את הביצועים על סט הtest על פי אותם מדדים:

# A tibble: 1 x 3

.metric .estimator .estimate

<chr> <chr> <dbl>

roc\_auc hand\_till 0.961

# A tibble: 1 x 3

.metric .estimator .estimate

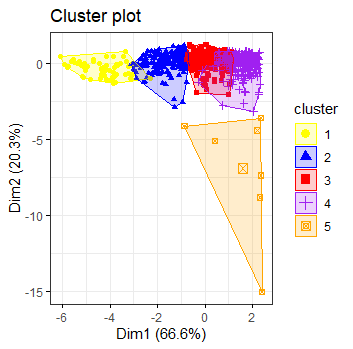
<chr> <chr> <dbl>

accuracy multiclass 0.734

K-means

בשיטה זו בדקנו כיצד הדאטא מתחלק והאם הקשר הינו הדוק בין העמודות שנחשבים כקריטריונים לחישוב הדירוג העולמי של האוניברסיטאות על פי המדד. לשם כך, בחרנו ב5 עמודות מתוך הדאטא, 3 מתוכן הן הקריטריונים שנאמר על ידי מדד Times שבהן משתמשים לחישוב הדירוג והעמודה החמישית היא עמודה שבחרנו שרירותית מתוך אלה שלא נאמר שנבחרת להיות חלק מהחישוב:

1. "world\_rank"
2. "teaching"
3. "research"
4. "citations"
5. "num\_students"



כפי שניתן לראות, אכן 3 העמודות הראשונות משפיעות הכי הרבה על העמודה הראשונה (הדירוג העולמי) והעמודה החמישית אינה מצומדת אל שאר העמודות.

## KNN

נריץ את אלגוריתם **KNN**- k-nearest neighbors. האלגוריתם מקבל סט אימון וסט מבחן. עבור על דוגמה בסט מבחן, מתבצעת בדיקה מי הם K השכנים הקרובים ביותר, ועל פיהם הדוגמה מסווגת.

stratify train test split:

#remove labels, normalize and return labels  
world\_rank\_col = data$world\_rank  
data = data %>% select(-world\_rank)  
data <- as.data.frame(scale(data))  
data$world\_rank = world\_rank\_col  
  
cell\_split <- initial\_split(data, strata = world\_rank)  
cell\_train <- training(cell\_split)  
cell\_test <- testing(cell\_split)

extract train labels:

train\_labels = cell\_train%>% select(world\_rank)  
train\_labels = train\_labels[,1]

extract test labels

test\_labels = cell\_test%>% select(world\_rank)  
test\_labels = test\_labels[,1]

remove labels from data

cell\_train = cell\_train %>% select(-world\_rank)  
cell\_test = cell\_test %>% select(-world\_rank)

אחרי שבנינו את הדאטה, ננסה להריץ את האלגוריתם עם K=1:

university\_test\_pred <- knn(train = cell\_train, test = cell\_test, cl = train\_labels, k=1)

#print accuracy  
tab <- table(university\_test\_pred,test\_labels)  
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}  
print(accuracy(tab))

## [1] 61.05263

הדיוק הוא 61%. אבל מכיוון שבחרנו שרירותית K=1, נוכל לנסות לשפר את התוצאה ע"י חיפוש K אופטימלי:

#find best k  
x <- c(1,3,5,7,9,11)  
for (val in x) {  
 university\_test\_pred <- knn(train = cell\_train, test = cell\_test, cl = train\_labels, k=val)  
   
 tab <- table(university\_test\_pred,test\_labels)  
 accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}  
 cat("K=", val,",accuracy=",accuracy(tab),"\n")  
}

## K= 1 ,accuracy= 61.05263   
## K= 3 ,accuracy= 63.68421   
## K= 5 ,accuracy= 63.15789   
## K= 7 ,accuracy= 59.47368   
## K= 9 ,accuracy= 59.47368   
## K= 11 ,accuracy= 62.10526

כפי שניתן לראות, החיפוש לא נתן לנו K טוב יותר. עכשיו ננסה שיטה אחרת לשיפור הדיוק- Sequential Forward Selection. מטרת השיטה היא למצוא תת קבוצה אופטימלית של פיצ'רים, כך שהדיוק יהיה מקסימלי:

library(FSinR)

evaluator <- wrapperEvaluator("knn")  
searcher <- searchAlgorithm('sequentialForwardSelection')  
results <- featureSelection(data, 'world\_rank', searcher, evaluator)

The best features are:

results$bestFeatures

## teaching international research citations income num\_students  
## [1,] 1 1 1 1 0 0  
## student\_staff\_ratio international\_students  
## [1,] 0 0

קיבלו תת קבוצה של הפיצ'רים. נבנה את הדאטה מחדש וננסה שוב את האלגוריתם:

new\_data = data %>% select(-income ,-num\_students ,-student\_staff\_ratio ,-international\_students)  
  
#train test split  
cell\_split <- initial\_split(new\_data, strata = world\_rank)  
cell\_train <- training(cell\_split)  
cell\_test <- testing(cell\_split)  
  
#extract train labels  
train\_labels = cell\_train%>% select(world\_rank)  
train\_labels = train\_labels[,1]  
  
#extract test labels  
test\_labels = cell\_test%>% select(world\_rank)  
test\_labels = test\_labels[,1]  
  
#remove labels from data  
cell\_train = cell\_train %>% select(-world\_rank)  
cell\_test = cell\_test %>% select(-world\_rank)  
  
#find best k  
x <- c(1,3,5,7,9,11)  
for (val in x) {  
 university\_test\_pred <- knn(train = cell\_train, test = cell\_test, cl = train\_labels, k=val)  
 tab <- table(university\_test\_pred,test\_labels)  
 accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}  
 cat("K=", val,",accuracy=",accuracy(tab),"\n")  
}

## K= 1 ,accuracy= 80   
## K= 3 ,accuracy= 78.94737   
## K= 5 ,accuracy= 75.26316   
## K= 7 ,accuracy= 75.26316   
## K= 9 ,accuracy= 73.68421   
## K= 11 ,accuracy= 74.21053

קיבלנו דיוק גבוה בהרבה- 80%! מה שאומר לנו שישנם פיצ'רים שמפריעים ללמידה של KNN.

## AdaBoost:

אלגוריתם AdaBoost הוא אלגוריתם שמשתמש במספר גדול של עצי החלטה, בדומה ל Random Forest, אך עם שיטה שנקראת Boosting לשיפור האלגוריתם. מכיוון שהקוד לאלגוריתם לא עבד בשפת R, נאצלנו להשתמש בשפת פייתון:

data = pd.read\_csv('C:\\Users\\avish\\CLionProjects\\sol\_test.csv')  
  
y = data['world\_rank']  
data = data.drop(columns=['world\_rank'])  
clf = AdaBoostClassifier(n\_estimators=100, random\_state=0)  
  
(trainX, testX, trainY, testY) = train\_test\_split(data, y,  
 test\_size=0.25, stratify = y)  
  
clf = clf.fit(trainX, trainY)  
  
clf.score(testX, testY)

## 0.4126984126984127

## LDA

נשתמש ב- LDA בשביל להוריד ממדים:

library(MASS)  
library(ggord)

#train test split  
cell\_split <- initial\_split(new\_data, strata = world\_rank)  
cell\_train <- training(cell\_split)  
cell\_test <- testing(cell\_split)  
linear <- lda(world\_rank~., cell\_train)

linear

## Coefficients of linear discriminants:

## LD1 LD2 LD3 LD4

## teaching 1.6518657 -0.5734547 2.36911982 0.09622907

## international 0.5246117 0.1638108 0.22659642 1.14875619

## research 2.2978830 -0.7356920 -2.46175226 -0.10217332

## citations 2.7635988 1.2746896 0.07518398 -0.71444198

## Proportion of trace:

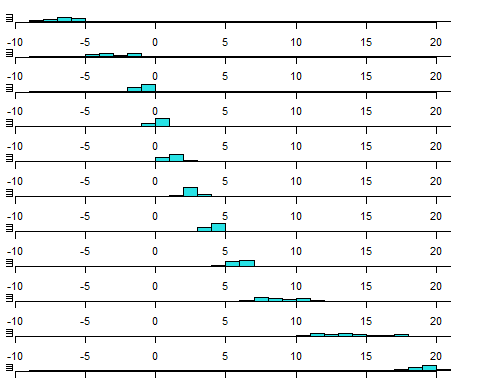
## LD1 LD2 LD3 LD4

## 0.9875 0.0111 0.0008 0.0006

זה אומר לנו ש- LDA1 מסביר לנו את רוב השונות בדאטה. נסתכל על LDA1:

LDA 1:

p <- predict(linear, cell\_train)  
par(mar=c(1,1,1,1))  
ldahist(data = p$x[,1], g = cell\_train$world\_rank)



ישנה הפרדה בLDA 1 , אך גם קצת חפיפות בין הקלאסים.

LDA 2:

par(mar=c(1,1,1,1))  
ldahist(data = p$x[,2], g = cell\_train$world\_rank)

תמונה שמכילה טקסט, ארון

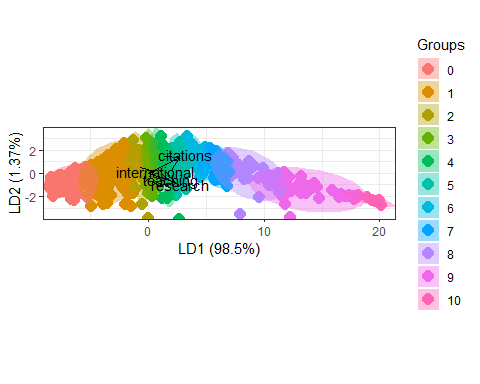
התיאור נוצר באופן אוטומטי

כפי שציפינו, החפיפה בLDA 2 היא גדולה.

עכשיו נשרטט את ה- Bi-Plot, כדי לראות את פיזור הדאטה ע"פ LDA:

Bi-Plot:

library(ggord)  
ggord(linear, as.factor(cell\_train$world\_rank))



גם כאן אפשר לראות הפרדה טובה, אך מעט חפיפות.

כעת, נשתמש ב- LDA בשביל פרדיקציה:

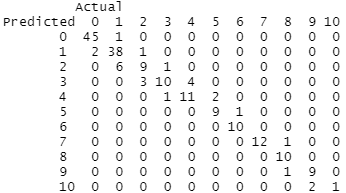
p2 <- predict(linear, cell\_test)$class

tab1 <- table(Predicted = p2, Actual = cell\_test$world\_rank)

tab1

sum(diag(tab1))/sum(tab1)   
## 0.8631579

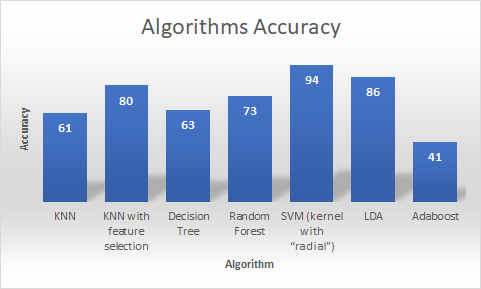
האלגוריתם נתן תוצאות מצוינות- 86% דיוק.



**סיכום ומסקנות:**

טבלה מסכמת:

|  |  |
| --- | --- |
| **אלגוריתם** | **דיוק** |
| KNN | 61 |
| KNN with feature selection | 80 |
| Decision Tree | 63 |
| Random Forest | 73 |
| SVM (kernel with “radial”) | 94 |
| LDA | 86 |
| Adaboost | 41 |



דיון בתוצאות:

* מהטבלה אנו רואים כי האלגוריתם שהצליח הכי טוב הוא SVM, ההצלחה שלו בראש ובראשונה תלויה בכך שלא לינארי, כי כשניסינו לעשות עבורו kernel לא לינארי הגענו לאחוזי הצלחה באזור 60.
* עם זאת, ה LDA מוריד מימדים, ומסווג לינארית, ולכן ציפינו שהציון שלו יהיה נמוך יחסית, אך הוא הגיע לציון כמעט הטוב ביותר.  
  איך זה מסתדר? ככל הנראה, הורדת המימדים של ה LDA עוזרת לבצע בכל זאת סיווג לינארי.
* אלגוריתם ה Adaboost היה נמוך בהרבה מ-Random forest, שזה מפתיע בהתחשב בעובדה ש Adaboost הוא שיפור שלו. חיפשנו וראינו שהסיבה יכולה להיות dataset קטן יחסית, שאיתו אלגוריתמי ה- boosting פחות טובים בסיווג שלהם.
* עשינו feature selection רק עם אלגוריתם KNN. מכיון שהוא שיפר מאוד את התוצאות, אפשר לומר שישנם עמודות שמפריעות לחיזוי טוב. יכול להיות שאם היינו עושים feature selection לשאר האלגוריתמים, היינו מקבלים תוצאות טובות יותר.
* ניתן לומר שהצלחת ה-decision tree לא כ"כ טובה בגלל שה-split על ה-data שהוא עשה לא הצליחה להפריד בצורה טובה את כל הפיצ'רים כדי ליצור קלסיפיקציה טובה ועל בסיסה חיזוי טוב.
* כצפוי, Random forest הצליח יותר טוב מ Decision trees, מכיון ש RF מתבסס על הרבה עצי החלטה, ולכן מקבל החלטה טובה יותר.

**ביבליוגרפיה:**

מקור הנתונים: <https://www.kaggle.com/mylesoneill/world-university-rankings>

**קוד:**

טעינת ה-data:

|  |
| --- |
| install.packages("tidyverse") library (tidyverse) data <- read\_csv("C:\\Users\\Israel\\Desktop\\ML & application to biological data analysis\\Final project\\archive\\timesData - UsedWithLabels.csv") |

סינון:

|  |
| --- |
| data = data %>% filter(year == 2016) data[data==""]<-NA  data[data=="-"]<-NA data<-data[complete.cases(data),] |

קוד הגרפים:

1. hist(data$teaching) # plot histograma type
2. hist(data$research)
3. ggplot(data = data, mapping = aes( x=student\_staff\_ratio,y = teaching)) +geom\_point(mapping = aes(x=student\_staff\_ratio, y = teaching))+geom\_point(mapping = aes(colour = research))
4. ggplot(data = data, mapping = aes(x = income, y=research))+geom\_smooth(method="lm")+geom\_point(mapping = aes(x = income, y=research))+geom\_point(mapping = aes(colour = teaching))
5. ggplot(data = data, mapping = aes( x=research,y = teaching)) + geom\_smooth(method="lm") +geom\_point(mapping = aes(x=research, y = teaching))
6. ggplot(data = data, mapping = aes(x = income, y=research))+ geom\_smooth(method="lm") +geom\_point(mapping = aes(x = income, y=research))+geom\_point(mapping = aes(colour = female\_male\_ratio))
7. ggplot(data = data, mapping = aes(y = research, x=world\_rank))+ geom\_smooth(method="lm") +geom\_point(mapping = aes(y = research, x=world\_rank))+geom\_point(mapping = aes(colour = female\_male\_ratio))
8. best\_uni = filter(data,world\_rank >=9)

ggplot(data = new\_data2) + geom\_bar(mapping = aes(x = country))+ theme(text = element\_text(size=10), axis.text.x = element\_text(angle=90, hjust=1))

**קוד עבור האלגוריתמים: -supplementary**

|  |
| --- |
| Decision Tree: (python) #py\_install("pandas") #py\_install("matplotlib") #py\_install("seaborn") #py\_install("scikit-learn") import numpy as np import pandas as pd import matplotlib.pyplot as plt from sklearn import tree from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import LabelEncoder  df = pd.read\_csv("C:/Users/Israel/Desktop/ML & application to biological data analysis/Final project/archive/timesData - UsedWithLabels.csv") df = df.dropna() #drop nans rows LE = LabelEncoder() df['country'] = LE.fit\_transform(df['country']) df['university\_name'] = LE.fit\_transform(df['university\_name']) features0 = df.drop(['world\_rank','female\_male\_ratio','year'], axis=1 ) features = np.array([features0]) labels0 = df.pop("world\_rank") labels = np.array([labels0]) X\_train, X\_test, y\_train, y\_test = train\_test\_split(  features0, labels0,  test\_size=0.3,  random\_state=42, ) max\_pred = 0.0001 best\_depth = 0 pred\_array = [] i\_lst = [] pred\_lst = [] i\_for\_depth = 0.0 accuracy = 0.0 j=0 for i in range(300):  j=i+1  print (j)  clf = tree.DecisionTreeClassifier(criterion='gini', splitter='best', max\_depth=j, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features=None, random\_state=None, max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, class\_weight=None, ccp\_alpha=0.0)    clf.fit(X = X\_train, y = y\_train)  clf.feature\_importances\_ # [ 1., 0., 0.]  accuracy = clf.score(X=X\_test, y=y\_test)   if (max\_pred <= accuracy):  best\_depth = clf.get\_depth()  pred\_array = clf.predict(X\_test)  max\_pred = accuracy # 1.0  print("better accuracy: ",max\_pred)  best\_wight = clf.feature\_importances\_ # [ 1., 0., 0.]  pred\_lst.append(accuracy)  i\_lst.append(j)  print("best\_depth:",best\_depth, "pred\_array:",pred\_array,"max\_pred:",max\_pred,"best\_wight:",best\_wight)  fig = plt.figure(figsize=(10,5)) ax = fig.add\_subplot(111) plt.plot(i\_lst,pred\_lst,"-b");  ax.set\_title('accuracy prediction') ax.set\_xlabel("index") ax.set\_ylabel("accuracy") plt.show() plt.close() |

1. SVM:

|  |
| --- |
| #R code:  library (tidyverse) data <- read\_csv("C:\\Users\\Israel\\Desktop\\ML & application to biological data analysis\\Final project\\archive\\timesData - UsedWithLabels.csv")   data = data %>% filter(year == 2016) data[data==""]<-NA data[data=="-"]<-NA data<-data[complete.cases(data),]  #changin country & university name to numeric representation: data$country <- as.factor(data$country) #convert the string from character to factor data$university\_name <- as.factor(data$university\_name) #convert the string from character to factor #data = data[,-2] #remove country #if factor above doesnt work somewhy #data = data[,-2] #REMOVE UNI' NAME  #do twice,rmv female male ration as its a chr col, 2T to remove year as well data = data[,-12]  data = data[,-12] #else do only once: #data = data[,-13] #REMOVE year  #shuffling the data by rows: data = data[sample(1:nrow(data)),]  #Doing scaling: #scaleddf <- as.data.frame(sapply(data, function(i) if(is.numeric(i)) scale(i) else i)) #the algo will do it as well, so meybe not needed #data.frame(round(scales::rescale(-scaleddf$world\_rank, to = c(1, 100)))) #rescale 1 column #need to check rescaling all column altogether #data <- data.frame(round(scales::rescale(-scaleddf$count, to = c(1, 100)))) #rescale <- function(x) (x-min(x))/(max(x) - min(x)) \* 100 #data <- rescale(data) #data #install.packages("kernlab") #install.packages("ISLR") library(tidyverse) # data manipulation and visualization library(kernlab) # SVM methodology library(e1071) # SVM methodology library(ISLR) # contains example data set "Khan" library(RColorBrewer) # customized coloring of plots  #make any negative num to 0 data[data < 0] <- 0  #splitting to train and test data\_train <- data[118:702, ] data\_test <- data[1:117, ]  #Building the actual model: data\_classifier <- ksvm(world\_rank ~ ., data = data\_train, kernel = "vanilladot") #data\_classifier <- ksvm(world\_rank ~ ., data = data\_train, kernel = "rbfdot")  data\_classifier  data\_predictions <- floor(predict(data\_classifier, data\_test)+0.5) #added floor #data\_predictions <- ceiling(predict(data\_classifier, data\_test)-0.5) #added ceiling #data\_predictions <- trunc(predict(data\_classifier, data\_test)) #added ceiling  #make any negative num to 0 data\_predictions[data\_predictions < 0] <- 0 head(data\_predictions) table(data\_predictions, data\_test$world\_rank) agreement <- data\_predictions == data\_test$world\_rank table(agreement) #in percentage prop.table(table(agreement)) #Avg 60% success rate. Lets move on to improving the model performance: #There are many kernal functions for SVM, but a standard Gaussian Radial basis function (RBF) kernal is a good place to start. #We'll use the ksvm() function here:  WR\_classifier\_rbf <- ksvm(world\_rank ~ ., data = data\_train, kernel = "rbfdot") #NOT STREIGHT LINE #WR\_classifier\_rbf <- ksvm(world\_rank ~ ., data = data\_train, kernel = "vanilladot") #STREIGHT LINE #Create a prediction for this new model: #WR\_predictions\_rbf <- floor(predict(WR\_classifier\_rbf, data\_test)+0.5) WR\_predictions\_rbf <- ceiling(predict(WR\_classifier\_rbf, data\_test)-0.5)  #And compare it to the previous model: agreement\_rbf <- WR\_predictions\_rbf == data\_test$world\_rank table(agreement\_rbf) prop.table(table(agreement\_rbf))  WR\_classifier\_rbf <- ksvm(world\_rank ~ ., data = data\_train, kernel = "radial")  #Create a prediction for this new model:  #WR\_predictions\_rbf <- floor(predict(WR\_classifier\_rbf, data\_test)+0.5) WR\_predictions\_rbf <- ceiling(predict(WR\_classifier\_rbf, data\_test)-0.5)   #And compare it to the previous model: agreement\_rbf <- WR\_predictions\_rbf == data\_test$world\_rank table(agreement\_rbf) prop.table(table(agreement\_rbf)) |

**T-sne:**

|  |
| --- |
| #R: #install.packages("Rtsne") library(ggplot2) library(Rtsne) tsne\_out <- Rtsne(data) tsne\_plot <- data.frame(x = tsne\_out$Y[,1], y = tsne\_out$Y[,2], col = data$world\_rank) ggplot(tsne\_plot) + geom\_point(aes(x=x, y=y, color=col))  Random Forest: ### Importing libraries ### library (tidyverse) library(ggplot2) library(tidymodels)  library(modeldata)  library(vip)  library(yardstick)  ### Loading data ### data <- read.csv(file = "./timesData - UsedWithLabels-fixed FMRatio.csv") ###Preprocessing the data### #Removing NA's data[data==""]<-NA data[data=="-"]<-NA data<-data[complete.cases(data),] #Filtering for year 2016 data = data%>%filter(year == 2016)   data %>%   count(world\_rank) %>%   mutate(prop = n/sum(n))  #Shuffling the data by rows: data = data[sample(1:nrow(data)),]   ###Splitting the data###  #Selecting to remove the columns: university\_name, country,female\_male\_ratio and year: data\_split <- initial\_split(data %>% select(-university\_name, -country,-female\_male\_ratio,-year), strata = world\_rank)  data\_train <- training(data\_split) data\_test <- testing(data\_split)  #Training set proportions by world\_rank data\_train %>%   count(world\_rank) %>%   mutate(prop = n/sum(n))  #Test set proportions by world\_rank data\_test %>%   count(world\_rank) %>%   mutate(prop = n/sum(n))  ### Random Forest Modelling ###  rf\_mod <-   rand\_forest(trees = 1000) %>%   set\_engine("ranger") %>%   set\_mode("classification")  #set.seed(234) rf\_fit <-   rf\_mod %>%   fit(factor(world\_rank) ~ . , data = data\_train) rf\_fit   ### Estimating performance ### rf\_training\_pred <-   predict(rf\_fit, data\_train) %>%   bind\_cols(predict(rf\_fit, data\_train, type = "prob")) %>%   # Add the true outcome data back in  bind\_cols(data\_train %>%   select(world\_rank))  rf\_training\_pred %>% # training set predictions  roc\_auc(truth = factor(world\_rank), c(.pred\_0,.pred\_1,.pred\_2,.pred\_3,.pred\_4,  .pred\_5,.pred\_6,.pred\_7,.pred\_8,.pred\_9,.pred\_10))   rf\_training\_pred %>% # training set predictions  accuracy(truth = factor(world\_rank), .pred\_class)   ## Lets see how the model performs on the test data:   rf\_testing\_pred <-   predict(rf\_fit, data\_test) %>%   bind\_cols(predict(rf\_fit, data\_test, type = "prob")) %>%   bind\_cols(data\_test %>% select(world\_rank))  rf\_testing\_pred %>% # test set predictions  roc\_auc(truth = factor(world\_rank), c(.pred\_0,.pred\_1,.pred\_2,.pred\_3,.pred\_4,  .pred\_5,.pred\_6,.pred\_7,.pred\_8,.pred\_9,.pred\_10))  rf\_testing\_pred %>% # test set predictions  accuracy(truth = factor(world\_rank), .pred\_class) |

|  |
| --- |
| 5.K-means: ### Importing libraries ### library(dplyr) library(stats) library(ggpubr) library(factoextra) ### Data Loading ### data <- read.csv(file = "./timesData - UsedWithLabels-fixed FMRatio.csv")  ### Filtering NA's ### data[data==""]<-NA data[data=="-"]<-NA data<-data[complete.cases(data),]  ### Filtering to year 2016 ### data = data%>%filter(year == 2016)  ### Creating dataframe from the data with only numeric columns ### world\_ranks\_z <- as.data.frame(lapply(data[,c("world\_rank","teaching","research","citations","num\_students")], scale))  ### Clustering the data to 5 clusters ### worldrank\_clusters <- kmeans(scale(data[,c("world\_rank","teaching","research","citations","num\_students")]), 5, nstart = 1)   ### Plotting the clusters ###  fviz\_cluster(worldrank\_clusters, data = world\_ranks\_z,  palette = c("yellow","blue","red","purple","orange"),   geom = "point",  ellipse.type = "convex",   ggtheme = theme\_bw() ) |