פרויקט ביג דאטה

רקע:

כדורסל הוא משחק כדור בו מתחרות זו בזו שתי קבוצות, בנות חמישה שחקנים כל אחת, הצוברות נקודות באמצעות השחלת כדור דרך חישוק הסל של הקבוצה היריבה (מתוך ויקיפדיה).

הכדורסל הוא אחד מענפי הספורט הפופולריים בעולם (מקום שני לפי totalsportek) ואחת הסיבות שהוא מאוד פופולרי הינה שקשה מאוד לחזות את זהות המנצחת.

עם התפתחות הפופולריות של המשחק, גדל גם הביקוש למידע מדיוק על הפועלות שקורות במגרש וניתוח של מידע זה. בעוד בעבר, התמקדו אך ורק בנקודות, עבירות ובאסיסטים (מסירה שיוצרת סל) עם הזמן הוסיפו מדדים נוספים, עקיפים יותר, שיכולים להשפיע על הניצחון וההפסד במשחק (ריבאונדים, הגנה והתקפה, חטיפות, איבודי כדור וחסימות).

בגלל פופולריות המשחק, ישנו פוטנציאל גדול למי שמצליח לדעת מי תנצח את המשחק עוד לפני שהוא מתרחש. בין אם זה חברות הימורים, כדי שיוכלו להרוויח כסף. מנהלי קבוצות שרוצים להביא את קבוצתם לאליפות. מאמנים, שיוכלו להחליט על איזה מדדים ופונקציות של השחקנים כדאי להשקיע את זמן האימונים. שחקנים, בשביל לדעת מה לשפר וכיצד להעלות את הסיכויים לקבל חוזה עתידי. וכמובן גם אוהדים של המשחק שרוצים להרגיש נעלים על חבריהם בכך שהם צדקו בזהות המנצחת.

בשנים האחרונות, ישנה התפתחות בתחום הניתוח הסטטיסטי של המדדים השונים הנמדדים במשחק, רבים מנסים למצוא מהו הפרט הסטטיסטי אשר יעזור להם לנבא את תוצאת המשחק.

אנו ננסה לבדוק האם ישנה אפשרות לדעת את זהות המנצחת על ידי מבט בסטטיסטיקה בלבד, במטרה שבעתיד נוכל להשתמש בידע הזה בשביל להבין מה דרוש מקבוצה לעשות על מנת לנצח את המשחק הבא.

שאלת מחקר – איזה סטטיסטיקות מנצחות משחק כדורסל.

בעבודה זו נחקור את האפשרות שישנם סטטיסטיקות מסוימות שאם ננתח אותם נוכל לדעת מה תהיה תוצאת המשחק.

תיאור המידע:

את המידע בו עשינו שימוש אספנו מהאתר: www.stats.nba.com - אתר המכיל סטטיסטיקות על אינספור מדדים.
המידע שנאסף מכיל נתונים על 4 עונות (2013-2017), הנתונים כוללים:

מדד	פירוט
TEAM_NAME	שם הקבוצה.
FG3M	מספר שלשות שנקלעו.
FG3A	מספר שלשות שנזרקו.
FG3_PCT	אחוז קליעת שלשות.
FTM	זריקות חופשיות שנקלעו.
FTA	זריקות חופשיות שנזרקו.
FT_PCT	אחוז קליעת זריקות חופשיות.
OREB	ריבאונד מתקיף.
DREB	ריבאונד מגן.
REB	מספר ריבאונדים.
AST	אסיסטים.
STL	גניבת כדור.

BLK	בלוקים.
TO	איבוד כדור.
PF	פאול אישי.
PTS	נקודות
FG2M	מספר 2 נקודות שנקלעו.
FG2A	מספר 2 נקודות שנזרקו.
FG2_PCT	אחוז קליעת 2 נקודות.
OUTCOME	תוצאה.
יתר העמודות הן אחוז המדדים לעומת הקבוצה השנייה.	

יש לציין שחלק מהנתונים לא היו זמינים באופן ישיר ונדרשנו לחשב אותם בעזרת הנתונים הקיימים.

תיאור כללי של הצעדים בביצוע הפרויקט:

איסוף מידע:

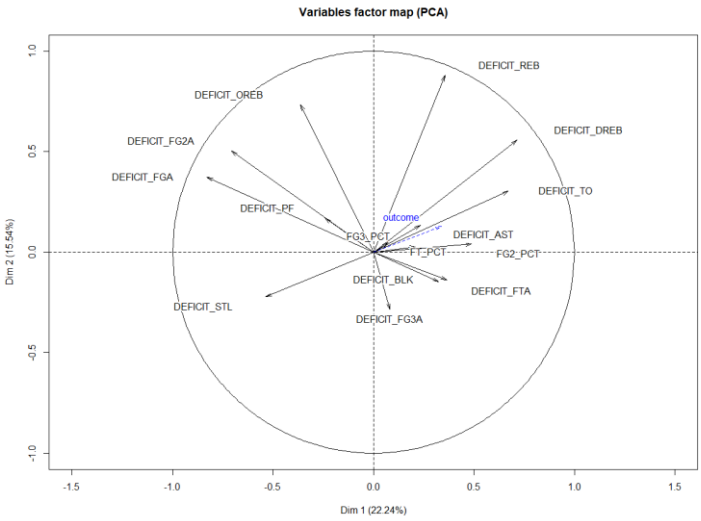
בשלב זה נעזרנו ב API המאפשר שליפת סטטיסטיקות מאתר ה- NBA. הכלי נכתב בשפת Python ולכן הקוד לאיסוף המידע שלנו נכתב גם הוא ב Python.
התוכנית שכתבנו מקבלת כארגומנט טווח של שנים ואוספת את המידע הרלוונטי לעונות בשנים האלה.
כאשר האיסוף מסתיים התוכנית מסדרת את המידע שנאסף בקובץ csv שיתאים לעבודה עם R.

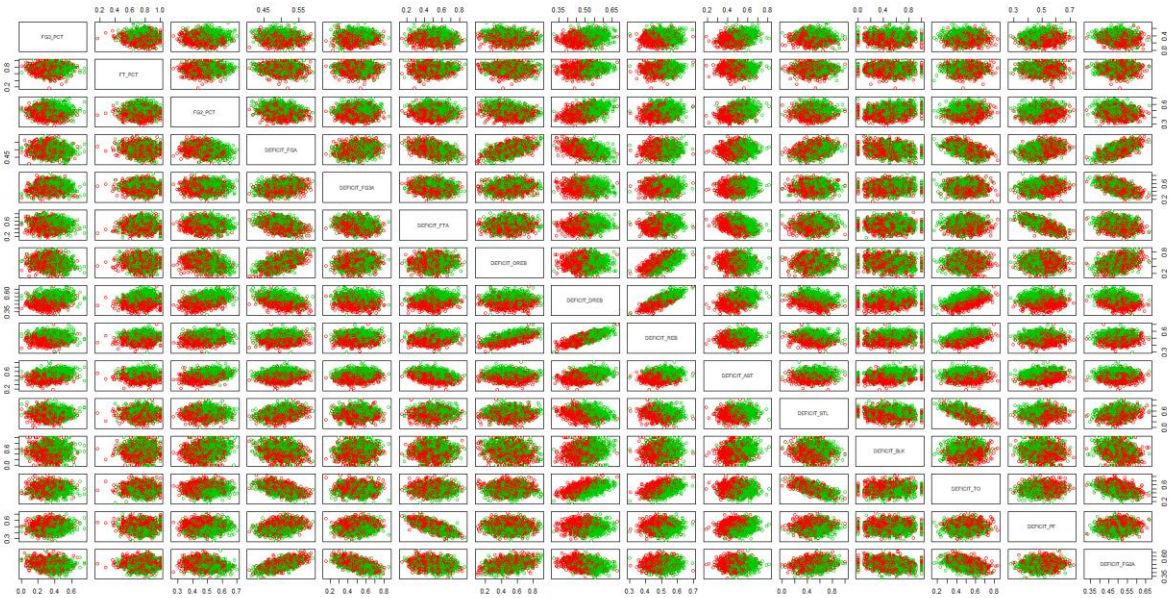
זיקוק מדדים והתמקדות במדדים הכי חזקים:

על מנת לבחון אילו מדדים הם הממדים החזקים ביותר שכדאי להשתמש בהם, השתמשנו בשיטת ה PCA (Principle Component Analysis).

ניתן לראות כי המדד – DEFICIT_DREB, הוא המדד עם הכי הרבה קורלציה.
כעת נרצה לראות הקורלציה של כל המדדים בזוגות:

	correlation	p.value
DEFICIT_DREB	0.71458012	0.000000e+00
DEFICIT_TO	0.67083400	1.017135e-288
FG2_PCT	0.48760305	3.094784e-132
DEFICIT_BLK	0.36506558	1.414876e-70
DEFICIT_REB	0.35557342	8.469140e-67
OUTCOME	0.33694743	9.386679e-60
DEFICIT_FTA	0.32433632	2.981061e-55
DEFICIT_AST	0.23253686	1.702077e-28
FG3_PCT	0.20748163	6.766563e-23
DEFICIT_FG3A	0.08211446	1.120738e-04
FT_PCT	0.07375866	5.231101e-04
DEFICIT_PF	-0.24143315	1.188063e-30
DEFICIT_OREB	-0.36268766	1.285373e-69
DEFICIT_STL	-0.53628857	1.124676e-164
DEFICIT_FG2A	-0.70768165	0.000000e+00
DEFICIT_FGA	-0.82796346	0.000000e+0





ניתן לראות כי DREB אכן יכול לסייע בסיווג המידע.

כעת בעזרת אלגוריתם למציאת אשכולות Kmeans, מצאנו אילו מדדים מנבאים הכי טוב בזוגות את תוצאת המשחק של הקבוצה.
כעת מצאנו את קבוצת המדדים אשר ביניהם נמצאים המדדים עם יכולת הניבוי הטובה ביותר:

- DEFICIT_DREB
- DEFICIT_AST
- FG3_PCT

על מנת לזקק ולמצוא את אלו המשפיעים ביותר, השתמשנו שוב ב Kmeans רק שכעת הפעלנו את האלגוריתם על כל הקומבינציות האפשריות של הקבוצה הנבחרת. גילינו שהקבוצה אשר משפיע הכי הרבה על תוצאת המשחק היא שלושת המדדים גם יחד.

ניבוי:

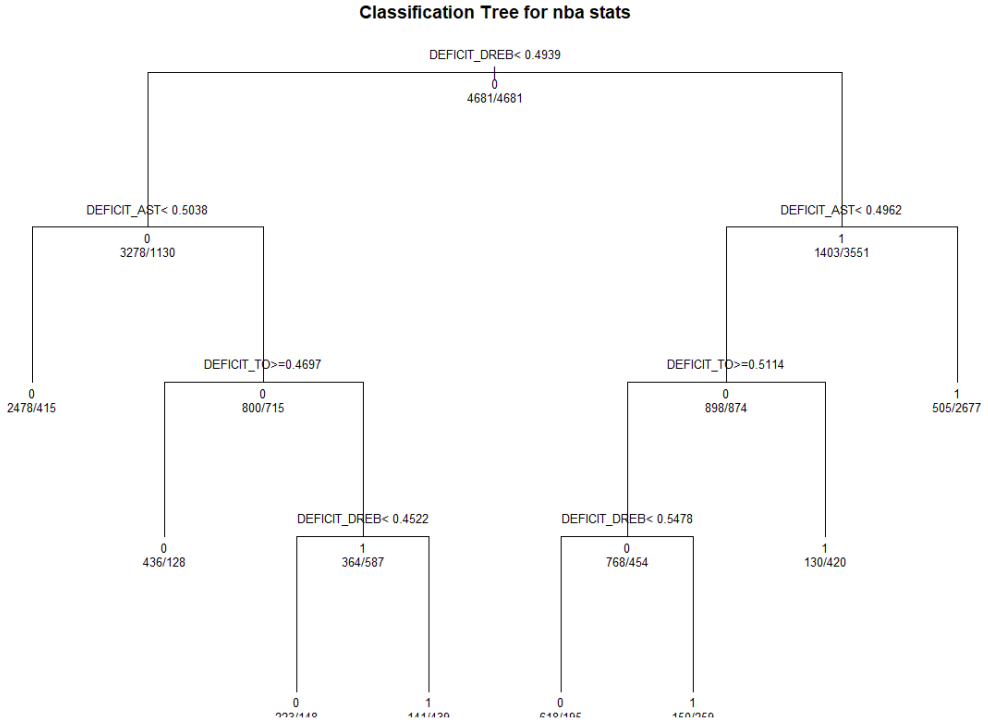
בשלב זה, השתמשנו באלגוריתם decision tree זאת מפני שאופם פעולת האלגוריתם מתאימה לסוג הניבוי אותו אנו מבקשים לבצע.
בדומה למדדים שמצאנו קודם, אך עם שינוי קל, האלגוריתם בחר במדדים:

- DEFICIT_DREB
- DEFICIT_AST
- DEFICIT_TO

האלגוריתם הגיע לרמת דיוק של 75% כאשר טבלת הפרדיקציה כדלהלן:

Reference / Prediction	0	1
0	1164	443
1	240	961

ועץ ההחלטות להלן:



תיאור תוצאות:

הצלחנו למצוא קשר בין סטטיסטיקה לבין תוצאת המשחק.

מצאנו שלמרות שיש המון מדדים סטטיסטים למשחק נתון, אפשר בעזרת חלק קטן מהמדדים ביניהם, ריבאונד הגנה ואסיסטים, לשער מי הקבוצה אשר ניצחה את המשחק.
הצלחנו להגיע לרמת דיוק של 75% בעזרת שלושה מדדים, כאשר כל מדד נוסף, פגע ביכולת החיזוי.

דיון בתוצאות:

כאשר ניגשנו למחקר, היינו סקפטיים לגבי האפשרות של חיזוי תוצאת המשחק בעזרת התבוננות בסטטיסטיקה. ההתבוננות בסטטיסטיקה נעשתה ללא התחשבות במדדים שקשורים לניקוד הקבוצה במשחק, סלי שלוש, סלי שדה וסלי עונשין. אך גם העלנו השערות לגבי איזה מדדים עלולים להוות גורם מכריע בתוצאת המשחק, במידה ואפשר לחזות זאת מהסטטיסטיקה.

ציפינו כי המדדים שישפיעו יותר על תוצאת המשחק יהיו בהתקפה, כגון ריבאונד התקפה ואחוזי קליעה, אך להפתעתנו גילינו כי דווקא המדד החזק ביותר שגילינו היה מדד הגנתי, ריבאונד ההגנה.

נראה כי הצלחנו להבין איזה הם הגורמים המשפיעים ביותר על תוצאת המשחק, אך אין די בהם על מנת לנבא באופן מדויק את תוצאת המשחק. למרות זאת, נראה כי הצלחה במדדים אלו כן עוזרת ב-3 מתוך 4 משחקים, לכן כדאי לנסות להביא את הקבוצה למצב בו היא משפרת את המדדים האלו.

מן המחקר שעשינו, אין להסיק כיצד מדדים אלו השפיעו על תוצאת המשחק ואיך על קבוצה לנהוג על מנת להביא את עצמה למצב בו היא שולטת במדדים אלו.

הקוד שהיה בשימוש:

קובץ 1:

```
library(data.table)
library(dplyr)
library(ggplot2)
library(mclust)
library(FactoMineR)

# read all csv files into one data.table
read.files <- function(){
  files <- list.files(pattern = '\\.csv')
  tables <- lapply(files, read.csv, header = TRUE)
  combined.df <- do.call(rbind , tables)
  return(data.table(combined.df))
}

match = read.files()
match = data.table(match[complete.cases(match)])
match[,DEFICIT_REB:=NULL]
# organize data
# outcome - The outcome of a match
# deficty table - each row contains data of a team in a certain matc
h.
outcome=match$OUTCOME
deficity_table = match[ ,(grepl('(?:DEFICIT|PCT).*(?<!M)$', names(ma
tch), perl = T)), with=F]

# Plotting data in pairs, trying to find correlation between feature
s.
# plot(deficity_table, col=outcome+2
pca = PCA(cbind(deficity_table, outcome), quanti.sup = ncol(deficity
_table)+1)

pca_desc = dimdesc(pca,1)
print(pca_desc)

# scale data.
scaled_data = scale(deficity_table)

# picked what apeers to be the strongest feature:
# DEFICIT_DREB - Defensieve rebound.
# Finding most correlated features with DEFICIT_DREB

pair.correlation = function(tb, feature, thresh){
  len = ncol(tb)
  correlated = vector()
  correlated[1] = feature
  rand.index = vector()
  i = 1
  for ( m in 1:len ) {
    if (m == feature ){
       next()
    }
```

```
    clust = kmeans(scaled_data[, c(feature, m)], 2)
    rand.index[i] = adjustedRandIndex(outcome, clust$cluster)
    if( rand.index[i] > thresh){
      correlated[i+1] = m
      i = i + 1
    }
  }
  return(list(correlated, mean(rand.index)))
}

# Checking that truly DREB is the most corrlated feature.
max.corr.feat = 0
max.rand.index= -Inf
for (i in c(1:ncol(scaled_data))){
  pairs = pair.correlation(scaled_data, i, 0.20)
  if ( max.rand.index < unlist(pairs[2]) ){
    max.rand.index = pairs[2]
    max.corr.feat = i
    feat.group = unlist(pairs[1])
  }
}

print(paste0("Correlated feature: ", max.corr.feat))

print(feat.group)

# Find the best combinations from picked features, using kmeans clus
tering and rand index.

brute.force.F.S <- function(dt, true.lables, chosen_features)
{
  if ( length(chosen_features) == 1 ){
    print("Only one feature. No work here.")
    return(chosen_features)
  }
  selected.features = vector()
  max.rand.index = -Inf
  for (m in c(1:length(chosen_features))) {
    combs = combn(chosen_features, m) # for each Combination.
    for (i in c(1:ncol(combs))) {
      features = combs[, i]
      clust = kmeans(dt[, features] , 2)
      current.rand.index = adjustedRandIndex(true.lables, clust$clus
ter)
      if (current.rand.index > max.rand.index) {
        max.rand.index = current.rand.index
        selected.features = features
      }
    }
  }
  print('for')
  print(selected.features)
  print(paste0("rand index: ", max.rand.index))
  print('*****************************************')
```

```
  return(selected.features)
}


selected_features = brute.force.F.S(scaled_data, outcome, feat.group
)

# find out how many teams whom have greater features actually won th
e game for each combination
# of features.
outcome.to.featurs = function(dt, true.lables, chosen_features ){
  num_feat = length(chosen_features)
  dt = dt[,chosen_features, with = F]
  dt[,OUTCOME:=true.lables]
  diff_dt=dt[seq(2,nrow(dt),2),]-dt[seq(1,nrow(dt)-1,2),]
  diff_dt=sign(diff_dt)


  diff_dt[,SUM_ALL:=rowSums(diff_dt)][,SUM_FEATURES:=rowSums(diff_dt
[,1:num_feat, with = F])]
  diff_dt=diff_dt[abs(SUM_FEATURES)==(num_feat)]
  won_features = nrow(diff_dt)
  diff_dt=diff_dt[abs(SUM_ALL)==(num_feat+1)]
  won_games = nrow(diff_dt)
  print("present data:")
  print(chosen_features)
  print(paste0("won features: ", won_features, " won games: ", won_g
ames))
  print(paste0("score: ", won_games / won_features))
  print('----------------------------------------')
}

for (m in c(1:length(selected_features))) {
  combs = combn(selected_features, m) # for each Combination.
  for (i in c(1:ncol(combs))) {
    features = combs[, i]
    outcome.to.featurs(as.data.table(scaled_data), outcome, features
)
  }
}
```

קובץ 2:

```r
library(dplyr)
library(data.table)
library(caret)
library(rpart)


# read all csv files into one data.table
read.files <- function(){
  files <- list.files(pattern = '\\.csv')
  tables <- lapply(files, read.csv, header = TRUE)
  combined.df <- do.call(rbind , tables)
  return(data.table(combined.df))
}

# prepare data table with only DEFICIT, PCT and OUTCOME statistics.
data <- read.files()
outcome=data$OUTCOME
data = data[ ,(grepl('(?:DEFICIT|PCT|OUTCOME).*(?<!M)$', names(data)
, perl = T)), with=F]
data = data[complete.cases(data)]
data[,DEFICIT_REB:=NULL]

data$OUTCOME = as.factor(data$OUTCOME)
# generate data partition for training and testing
inTrain = createDataPartition(data$OUTCOME, p=0.7, list = FALSE)
training = data[inTrain,]
testing = data[-inTrain,]

# train the algoritham on genrated data.
dtree = train(OUTCOME~., method="rpart", data=training, na.action =
na.pass)
print(dtree$ finalModel)

# predict using decision tree.
prediction = predict(dtree, newdata = testing)
print(prediction)



# check using confusion matrix how good is the prediction.
cm = confusionMatrix(prediction, testing$OUTCOME)
print(cm)

# grow a decision tree for visualization.
fit <- rpart(OUTCOME~., method = "class", data=data)

printcp(fit)

plotcp(fit)

summary(fit)
plot(fit, uniform = TRUE, main="Classification Tree for nba stats")
text(fit, use.n = TRUE, all = TRUE, cex=.8)
```