NFL combine positionsanalyse - R analyse

Link til colab: https://colab.research.google.com/drive/1xeA_oGZhXaQ0eCSZd92YK8BF6zDHUPkZ

Meningen med denne analyse er, at prøve at finde en sammenhæng mellem en spillers fysiske karakteristika og d
forskellige pladser med deres egne notationer. Udfra et datasæt om combine data (Den fysiske evaluering før en
er hovedmålet at se, om en spiller faktisk spiller på den rigtige position i forhold til hvad algoritmen fortæller pers
dataet, i form af, at flere positioner ligner hinanden (Center og Guard, Wide receiver og Corner back, Tight end og

Der laves 2 beskrevne analyser og en analyse for nysgerrighedens skyld, en på baggrund af de normale positione
kombinationer af positioner. Det er sådan, at datasættet indeholder "underpositioner" af allerede kendte posiitone
der er en undergruppering af DE, og NT der er en undergruppering af DT, her kombineres disse posiitoner, så, når
observationer for disse undergrupper. Til sidst laves en lille analyse som inddeler positionerne efter om de er "tur
medium drenge med lidt vægt som stadig er semi-mobile samt de lette drenge som har høj mobilitet men ingen v

Den største fejlkilde vi har i projektet er vores prediction test og dets konklusion igennem en confusionmatrix. Vo
spiller skulle spille, på vegne af spillerens fysiske mål, hvilket, desværre, også kan inkludere om spilleren spiller p
godt spille på en position som spilleren ikke er "fysisk egnet" til, i forhold til de resterende spillere.

Vi installere pakkerne:

```
#install.packages("FactoMineR")
#install.packages("factoextra")
#library(devtools)
#install_github("vqv/ggbiplot")
#install.packages("GGally")
#install.packages("caret")
#install.packages("recipes")
#install.packages("rpart")
#install.packages('e1071', dependencies=TRUE)
#install.packages("ranger")
#install.packages("kernlab")
#install.packages("gbm")
#install.packages("MLmetrics")
#install.packages("kknn")
```

Vi benytter pakkerne:

```
library(tidyverse)
library(magrittr)
library(lubridate)
library(FactoMineR)
library(factoextra)
library(ggbiplot)
library(GGally)
library(caret)
library(recipes)
library(rpart)
library(e1071)
library(ranger)
library(kernlab)
library(gbm)
library(MLmetrics)
library(kknn)
```

Nu indlæses vores data, der fjernes variable vi ikke har brug for, samt konvertere variablerne til metric units i sted

Der dannes en vars_num som der arbejdes med løbende igennem projektet. Disse er variablerne vi vil inkludere i

dataframen "data" er vores første analyse, uden kombination, hvor vi subsetter nogen positioner med for få obse

dataframen "data2" er vores anden analyse, med kombination, hvor vi kombinere positioner af ligende stilling og

Grunden til at data2 er defineret før data er, at vi vil lave data2 på baggrund af det rå datasæt, hvor vi overskriver

```
data <- read_csv("https://github.com/bande15/sds/raw/master/combine_data.csv")

data %>% head()
count(data$Pos)


#Data hvor kombinatiooner er kombineret. EDGE er inkluderet i DE. OLB og ILB er inklu
data2 <- data %>% select(-Player, -Year, -Team, -AV, -Round, -Pick, -Pfr_ID)
data2 %<>% mutate(data2, Pos = fct_recode(Pos, "DE" = "EDGE")) %>% mutate(data2, Pos =
nrow(data2)
count(data2$Pos)

#Normal analyse, hvor vi har fjernet LS, QB, LB, OL, S, G og EDGE grundet lave antal
data %<>% select(-Player, -Year, -Team, -AV, -Round, -Pick, -Pfr_ID) %>% drop_na() %>%
nrow(data)
count(data$Pos)

vars_num <- c("Ht", "Wt", "Forty", "Vertical", "BenchReps", "BroadJump", "Cone", "Shut

data$Wt <- data$Wt*0.45
data$Ht <- data$Ht*2.54
data$Vertical <- data$Vertical*2.54
data$BroadJump <- data$BroadJump*2.54
```

```
Parsed with column specification:
cols(
  Player = col_character(),
  Pos = col_character(),
  Ht = col_double(),
  Wt = col_double(),
  Forty = col_double(),
  Vertical = col_double(),
  BenchReps = col_double(),
  BroadJump = col_double(),
  Cone = col_double(),
  Shuttle = col_double(),
  Year = col_double(),
  Pfr_ID = col_character(),
  AV = col_double(),
  Team = col_character(),
  Round = col_double(),
  Pick = col_double()
)
```

A tibble: 6 × 16

Player	Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle	Year	
<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
John Abraham	OLB	76	252	4.55	NA	NA	NA	NA	NA	2000	A
Shaun Alexander	RB	72	218	4.58	NA	NA	NA	NA	NA	2000	A
Darnell Alford	OT	76	334	5.56	25	23	94	8.48	4.98	2000	A
Kyle Allamon	TE	74	253	4.97	29	NA	104	7.29	4.49	2000	N
Rashard Anderson	CB	74	206	4.55	34	NA	123	7.18	4.15	2000	A
Jake Arians	K	70	202	NA	NA	NA	NA	NA	NA	2000	ar

A data.frame: 25 × 2

x	freq
<fct>	<int>
C	171
CB	630
DB	2
DE	487
DT	463
EDGE	23
FB	117
FS	229
G	14
ILB	276
K	85
LB	3
LS	20
NT	3
OG	365
OL	2
OLB	424
OT	460
P	120
QB	350
RB	540

S	27
SS	213
TE	337
WR	857

2884

A data.frame: 14 × 2

x	freq
<fct>	<int>
C	115
CB	311
DE	287
DT	253
FB	77
S	237
G	10
OL	371
OG	224
OT	273
QB	12
RB	245
TE	194
WR	275

2846

A data.frame: 14 × 2

x	freq
<fct>	<int>
C	115
CB	311
DE	279
DT	253
FB	77
FS	123
ILB	130
OG	224
OLB	240
OT	273
RB	245
SS	107
TE	194
WR	275

DEL 1.1 - Unsurpervised analyse af datasæt, UDEN komb

Her laves en unsupervised analyse af datasættet, uden kombination af positioner.

Vi anser dataet, her observerer vi hurtigt at de numeriske variabler og variablen Pos er inkluderet, deres modus sa
at alle positioner med få værdier er filtreret fra.

```
data %>% glimpse()
data %>% head()
sapply(data, mode)

count(data$Pos)
```

```
Observations: 2,846
Variables: 9
$ Pos       <chr> "OT", "OLB", "CB", "OT", "FS", "CB", "FS", "OG", "ILB", "DE…
$ Ht        <dbl> 193.04, 182.88, 175.26, 198.12, 182.88, 177.80, 185.42, 193…
$ Wt        <dbl> 150.30, 106.65, 78.75, 140.40, 93.60, 89.55, 93.15, 135.90,…
$ Forty     <dbl> 5.56, 4.72, 4.44, 5.34, 4.62, 4.44, 4.62, 5.07, 4.78, 5.09,…
$ Vertical  <dbl> 63.50, 78.74, 88.90, 71.12, 88.90, 95.25, 100.33, 80.01, 80…
$ BenchReps <dbl> 23, 21, 17, 20, 10, 16, 15, 17, 21, 26, 12, 28, 14, 14, 28,…
$ BroadJump <dbl> 238.76, 284.48, 302.26, 243.84, 289.56, 294.64, 302.26, 261…
$ Cone      <dbl> 8.48, 7.96, 7.03, 7.72, 6.92, 6.81, 6.48, 7.76, 7.17, 7.68,…
$ Shuttle   <dbl> 4.98, 4.39, 4.14, 4.73, 4.32, 4.04, 4.29, 4.58, 4.33, 4.49,…
```

A tibble: 6 × 9

Pos	Ht	Wt	Forty	Vertical	BenchReps	BroadJump	Cone	Shuttle
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
OT	193.04	150.30	5.56	63.50	23	238.76	8.48	4.98
OLB	182.88	106.65	4.72	78.74	21	284.48	7.96	4.39
CB	175.26	78.75	4.44	88.90	17	302.26	7.03	4.14
OT	198.12	140.40	5.34	71.12	20	243.84	7.72	4.73
FS	182.88	93.60	4.62	88.90	10	289.56	6.92	4.32
CB	177.80	89.55	4.44	95.25	16	294.64	6.81	4.04

```
Pos
	'character'
Ht
	'numeric'
Wt
	'numeric'
Forty
	'numeric'
Vertical
	'numeric'
BenchReps
	'numeric'
BroadJump
	'numeric'
Cone
	'numeric'
Shuttle
	'numeric'
```

A data.frame: 14 × 2

x	freq
<fct>	<int>
C	115
CB	311
DE	279
DT	253
FB	77
FS	123
ILB	130
OG	224
OLB	240
OT	273
RB	245
SS	107

SS 107
TE 194
WR 275

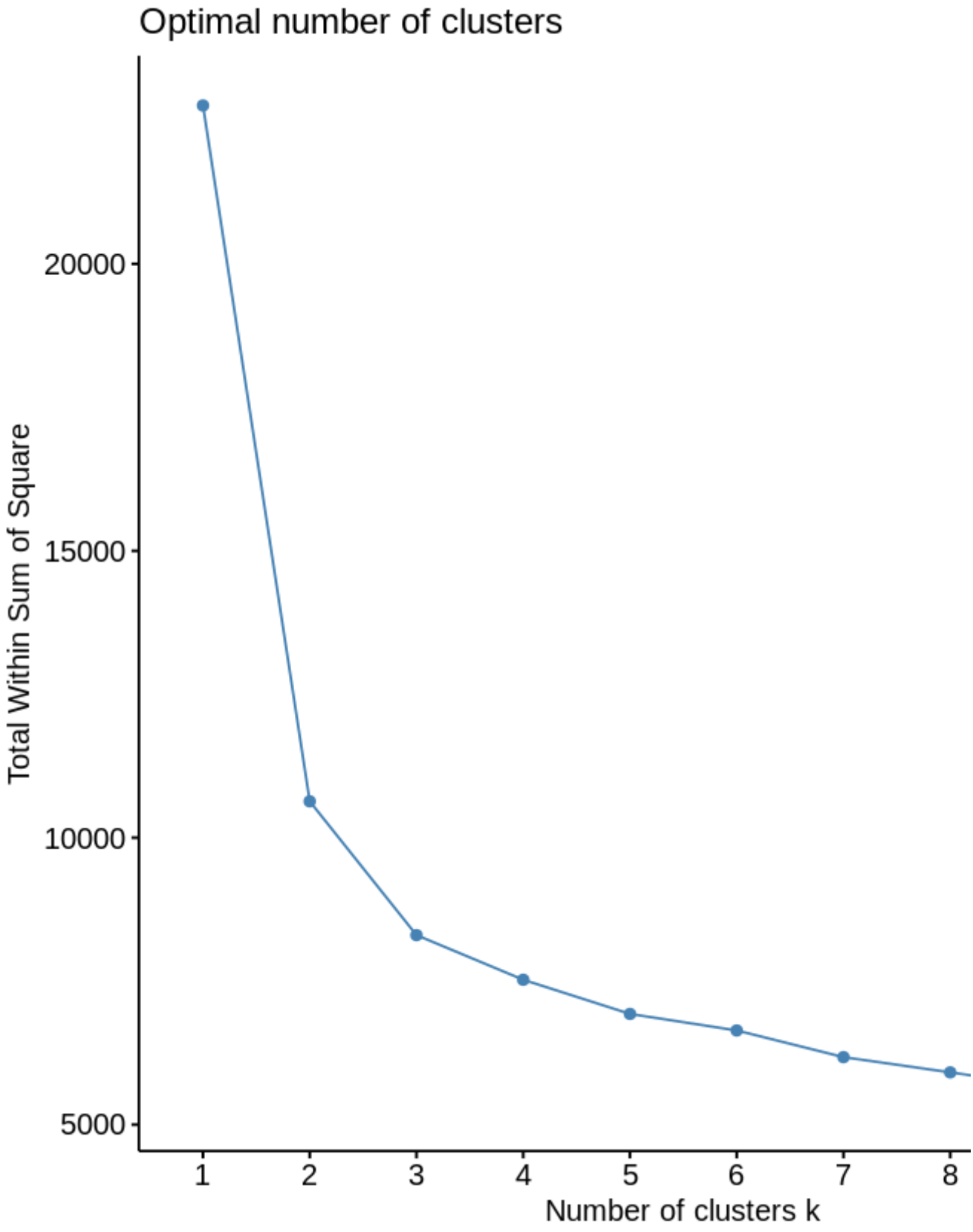
Vi laver nu en PCA analyse og bestemmer optimale antal clusters, hvilket vi får til 2:

```
data.pca <- prcomp(data[,vars_num], center = TRUE,scale. = TRUE)
summary(data.pca)

data[,vars_num] %>%
  scale() %>%
  fviz_nbclust(kmeans, method = "wss")

```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5    PC6     PC7
Standard deviation     2.3682 0.9223 0.78874 0.61423 0.46865 0.3721 0.34729
Proportion of Variance 0.7011 0.1063 0.07776 0.04716 0.02745 0.0173 0.01508
Cumulative Proportion  0.7011 0.8074 0.88515 0.93231 0.95976 0.9771 0.99214
                           PC8
Standard deviation     0.25073
Proportion of Variance 0.00786
Cumulative Proportion  1.00000
```



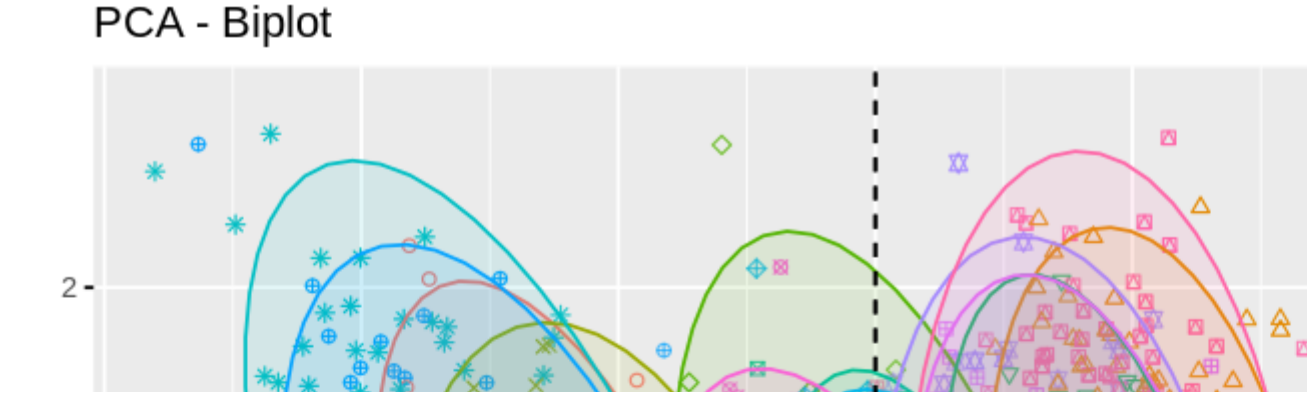
Nu laves en hurtig k-means analyse, hvor vi illustrere resultatet igennem et biplot med Pos grupperingen vist.

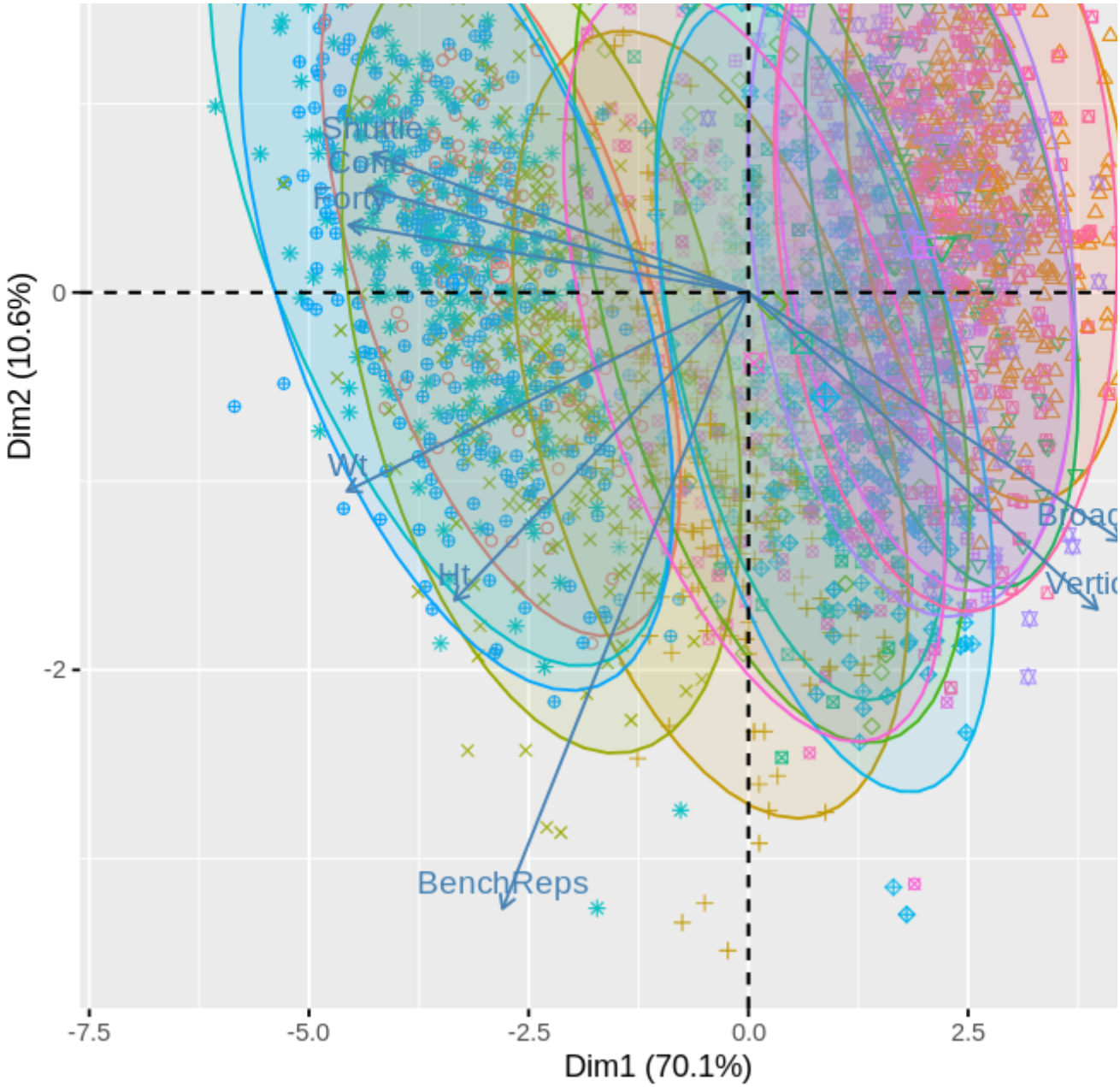
```r
km <- data[,vars_num] %>%
  scale() %>%
  kmeans( centers = 2, nstart = 2)

km %>%
  fviz_cluster(data = data[,vars_num],
               ggtheme = theme_gray())

data.pca %>%
  fviz_pca_biplot(alpha.ind = "cos2",
                  geom = "point",
                  habillage = data$Pos %>% factor(),
                  addEllipses = TRUE,
                  ggtheme = theme_gray())
```


Cluster plot
Dim2 (10.6%)
Dim1 (70.1%)
2
0
-2
-7.5
-5.0
-2.5
0.0
2.5


PCA - Biplot
2



Noter, Forty, Shuttle og Cone er adræt og opmærksomhedstest som er målt i sekunder, jo flere sekunder jo langs
Dvs. at selvom man kunne få opfattelsen af at Forty, Cone og Shuttle hører til blandt de venstre grupperinger, så e
grupperinger, altså WR, TE, CB etc.

DEL 1.2 - Supervised ML analyse af datasæt, UDEN komb

Her laves en supervised ML analyse, uden kombination af positioner. Her laves noget preprocessing i form af at s
er med fuld belæg. Den normale fordeling er 75%/25% eller 80%/20%. Der vil gerne laves en model der er så nøja
observationer for at verficere vores model, herved laves en 90%/10% analyse.

Efter splittet set vi på antal observationer i training og test datasættet.

```
data %<>% drop_na %>% select(Pos, everything())

index <- createDataPartition(y = data$Pos, p = 0.90, list = FALSE) # 75% to 25% split

```

```
training <- data[index,]
test <- data[-index,]

nrow(training)
nrow(test)

reci <- recipe(Pos ~ ., data = training) %>%
  step_center(all_numeric(), -all_outcomes()) %>% # Centers all numeric variables to r
  step_scale(all_numeric(), -all_outcomes()) %>% # scales all numeric variables to sd
  step_zv(all_predictors())  # Removed predictors with zero variance


# knn inputation of missing values
reci %<>%
  step_knnimpute(all_predictors())

# Recipe in the end has to be prepared on the training data
reci %<>%
  prep(data = training)

reci
```

```
2567
279
Data Recipe

Inputs:

      role #variables
   outcome          1
 predictor          8

Training data contained 2567 data points and no missing data.

Operations:

Centering for Ht, Wt, Forty, Vertical, BenchReps, ... [trained]
Scaling for Ht, Wt, Forty, Vertical, BenchReps, ... [trained]
Zero variance filter removed no terms [trained]
K-nearest neighbor imputation for Wt, Forty, Vertical, BenchReps, ... [trained
```

Nu definerer vi kontrol features, er er valgt 2-fold crossvalidation, grundet datasættets størrelse. Herefter bages d
datasæt og ser på antal levels i Pos variablen, for at være sikker på, at training og test har lige mange levels. Teor
med i det ene datasæt grundet fordelingen. Her verificeres at begge datasæt har samme antal positioner (levels)

```
x_train <- bake(reci, new_data = training) %>% select(-Pos)
y_train <- pull(training, Pos) %>% as.factor()

# test: split in y and x
x_test <- bake(reci, new_data = test) %>% select(-Pos)
y_test  <- pull(test, Pos) %>% as.factor()


ctrl <- trainControl(method = "cv", # repeatedcv, boot, cv, LOOCV, timeslice OR adapti
                     number = 2, # Number of CV's
                     classProbs = TRUE, # Include probability of class prediction
                     savePredictions = TRUE, # Save the prediction results
                     summaryFunction = multiClassSummary, # Which type of summary stat
                     verboseIter = FALSE,
                     # add adaptive resampling for hyperparamether search
                     adaptive = list(min = 3,
                                     alpha = 0.05,
                                     method = "gls",
                                     complete = TRUE),
                     search = "random" )
```

```
21
22 metric <- "Accuracy" # Which metric should be optimized (more on that later)
23 n_tune = 2 # number of tuning rounds
24
25 #x_train$Type1 <- as.character(x_train$Type1)
26
27 x_train <- cbind(Number = rownames(x_train), x_train)
28 x_train$Number <- as.numeric(x_train$Number)
29
30
31 x_test <- cbind(Number = rownames(x_test), x_test)
32 x_test$Number <- as.numeric(x_test$Number)
33
34 x_train %>% head()
35 y_train %>% head()
36 x_test %>% head()
37 y_test %>% head()
```


Nu begynder fitningen af den supervised ML analyse, her fittes i k-nearest neighbor classification fit, et decision t baggrund af deres multiklassifikation og validitet.

Vi fitter nu k-nearest neighbor classification fittet og viser den igennem en confusionmatrix. Confusionmatrix er v

```
fit_knnn <- train(x = x_train,
                  y = y_train,
                  trControl = ctrl,
                  metric = metric,
                  method = "kknn")

fit_knnn
```

```
pred_knnn <- predict(fit_knnn, newdata = x_test)
confusionMatrix(pred_knnn, y_test, positive = "Yes")
```

Modellens nøjagtighed er 45.16%, hvilket vil sige, at 45.16% af prediktionerne er korrekte. Dog ses det at mange a
såsom cornerback og widereceiver, DE og tightend samt OLB og ILB. Som "praktisk set" kræver samme fysiske u
form, igennem analyse 2.

Vi fitter nu en decision tree model og predikter den, det vises igennem et confusion matrix.

```
fit_dt <- train(x = x_train,
                y = y_train,
                trControl = ctrl,
                metric = metric,
                tuneLength = n_tune,
                method = "rpart")

fit_dt
```

```
pred_dt <- predict(fit_dt, newdata = x_test)
confusionMatrix(pred_dt, y_test, positive = "Yes")
```

Her får modellen en nøjagtighed på 52.33%. Samme problem som sidst i form af at positioner af samme fysiske

Vi fitter nu en random forest model, og viser prediktionen igennem et confusionmatrix.

```
fit_rf <- train(x = x_train,
                y = y_train,
                trControl = ctrl,
                metric = metric,
                tuneLength = n_tune,
                method = "ranger",
                importance = "impurity", # To define how to measure variable important
                num.trees = 25
                )

fit_rf
```

```
pred_rf <- predict(fit_rf, newdata = x_test)
confusionMatrix(pred_rf, y_test, positive = "Yes")
```

Her får modellen en nøjagtighed på 55.56% men udviser stadig samme fejltyper som før, dog er denne markant b
fejl, men også hvilken type fejl der forekommer. Se ses mindre spredning i fejl-prediktionen end de forhenværend

Her kan vi nu konkludere på datasættet.

Modellernes "nøjagtighed" er som følgende: Logaritme model: 52.33% Decision tree: 50.18% Random forest: 58.7

DEL 2.1 - Unsurpervised analyse af datasæt, MED kombi

Samme analyse som forrige afsnit, dog med det kombinerede datasæt. Derfor beskrives der ikke ligeså intensivt
FS og SS i S, og NS i G.

Vi anser datasættet:

```
data2 %>% glimpse()
data2 %>% head()
sapply(data2, mode)

count(data2$Pos)
```

Vi laver nu en PCA analyse hvor vi finder optimale antal clusters, dette er igen 2:

```
data2.pca <- prcomp(data2[,vars_num], center = TRUE,scale. = TRUE)
summary(data2.pca)

data2[,vars_num] %>%
  scale() %>%
  fviz_nbclust(kmeans, method = "wss")
```

Kmeans plot med og uden gruppering af Pos.

```
km <- data2[,vars_num] %>%
  scale() %>%
  kmeans( centers = 2, nstart = 2)

km %>%
  fviz_cluster(data = data2[,vars_num],
               ggtheme = theme_gray())

data2.pca %>%
  fviz_pca_biplot(alpha.ind = "cos2",
                  geom = "point",
                  habillage = data2$Pos %>% factor(),
                  addEllipses = TRUE,
                  ggtheme = theme_gray())
```

DEL 2.2 - Surpervised analyse af datasæt, MED kombinat

Vi laver nu noget preprocessing og splitter dataet op. Under samme grundlag som forrige analyse splittes datasæ

```
data2 %<>% drop_na %>% select(Pos, everything())
data2 <- droplevels(data2)

index <- createDataPartition(y = data2$Pos, p = 0.90, list = FALSE) # 75% to 25% split

training <- data2[index,]
test <- data2[-index,]

nrow(training)
nrow(test)

reci <- recipe(Pos ~ ., data = training) %>%
  step_center(all_numeric(), -all_outcomes()) %>% # Centers all numeric variables to m
  step_scale(all_numeric(), -all_outcomes()) %>% # scales all numeric variables to sd
  step_zv(all_predictors())  # Removed predictors with zero variance


# knn inputation of missing values
reci %<>%
  step_knnimpute(all_predictors())

# Recipe in the end has to be prepared on the training data
reci %<>%
  prep(data = training)

reci
```

Nu definerer vi kontrolfeatures og 2-fold crossvalidation, samt bager og træner datasættet.

```
x_train <- bake(reci, new_data = training) %>% select(-Pos)
y_train <- pull(training, Pos) %>% as.factor()

# test: split in y and x
x_test <- bake(reci, new_data = test) %>% select(-Pos)
y_test  <- pull(test, Pos) %>% as.factor()


ctrl <- trainControl(method = "cv", # repeatedcv, boot, cv, LOOCV, timeslice OR adapti
                     number = 2, # Number of CV's
                     classProbs = TRUE, # Include probability of class prediction
                     savePredictions = TRUE, # Save the prediction results
                     summaryFunction = multiClassSummary, # Which type of summary stat
                     verboseIter = FALSE,
                     # add adaptive resampling for hyperparamether search
                     adaptive = list(min = 3,
                                     alpha = 0.05,
                                     method = "gls",
                                     complete = TRUE),
```

```
                    search = "random" )

metric <- "Accuracy" # Which metric should be optimized (more on that later)
n_tune = 2 # number of tuning rounds

#x_train$Type1 <- as.character(x_train$Type1)

x_train <- cbind(Number = rownames(x_train), x_train)
x_train$Number <- as.numeric(x_train$Number)


x_test <- cbind(Number = rownames(x_test), x_test)
x_test$Number <- as.numeric(x_test$Number)

x_train %>% head()
y_train %>% head()
x_test %>% head()
y_test %>% head()
```

Vi fitter igennem de samme test som forrige analysen, kknn, decision tree og random forest.

Her fittes kknn modellen.

```
fit_knn <- train(x = x_train,
                 y = y_train,
                 trControl = ctrl,
                 metric = metric,
                 method = "kknn")

fit_knn
```


Nu laves kknn confusionmatrix.

```
pred_knn <- predict(fit_knn, newdata = x_test)
confusionMatrix(pred_knn, y_test, positive = "Yes")
```

Her får modellen en nøjagtighed på 57.95% udviser stadig fejl dog er spredningen af fejlene mindre end analyse

Her fittes decision tree modellen.

```
fit_dt <- train(x = x_train,
                y = y_train,
                trControl = ctrl,
                metric = metric,
                tuneLength = n_tune,
                method = "rpart")

fit_dt
```

Confusionmatrix laves for decision tree.

```
pred_dt <- predict(fit_dt, newdata = x_test)
confusionMatrix(pred_dt, y_test, positive = "Yes")
```

Her får modellen en nøjagtighed på 57.95% udviser stadig fejl dog er spredningen af fejlene mindre end analyse

Random forest fit:

```
fit_rf <- train(x = x_train,
                y = y_train,
                trControl = ctrl,
                metric = metric,
                tuneLength = n_tune,
                method = "ranger",
                importance = "impurity", # To define how to measure variable important
                num.trees = 25
                )

fit_rf
```

Igen laves der en random forest confusionmatrix.

```
pred_rf <- predict(fit_rf, newdata = x_test)
confusionMatrix(pred_rf, y_test, positive = "Yes")
```

Her får modellen en nøjagtighed på 60.78% udviser stadig fejl dog er spredningen af fejlene mindre end analyse 1 end forrige analyse.

Vi kan nu konkludere på datasættet:

Modellernes "nøjagtighed" er som følgende: K-nearest: % 57.95% Decision tree: 57.95% Random forest: 60.78%

Fra første analyse, uden kombination, blev der fundet:

Modellernes "nøjagtighed" er som følgende: K-nearest: 45.16% Decision tree: 52.33% Random forest: 55.56%

Her ses altså et markant bedre resultat end forrige analyse

Analyse 3 - Gruppering af position i 3 hovedgrupperinger

Af ren nysgerrighed vil vi se modellernes supervised performance, hvis grupperingen af Pos var mindre, og opdel defineret i databladet som "tunge, medium og lette" spillere. Alt efter om man er stor og langsom, medium stor o ment i vægt.

Her laves en hurtig analyse af denne gruppering. Med mest fokus på supervised.

```
data_gruppering <- read_csv("https://github.com/bande15/sds/raw/master/combine_data.cs
#Data hvor kombinatiooner er kombineret. EDGE er inkluderet i DE. OLB og ILB er inklu
data <- data_gruppering %>% select(-Player, -Year, -Team, -AV, -Round, -Pick, -Pfr_ID
data %<>% mutate(data, Pos = fct_recode(Pos, "Tung" = "C")) %>% mutate(data, Pos = fct
nrow(data)
count(data$Pos)
```

Hurtig unsupervised analyse, vi ser 2 clusters og går med det.

```
data.pca <- prcomp(data[,vars_num], center = TRUE,scale. = TRUE)
summary(data.pca)

data[,vars_num] %>%
  scale() %>%
  fviz_nbclust(kmeans, method = "wss")

km <- data[,vars_num] %>%
  scale() %>%
  kmeans( centers = 2, nstart = 2)

km %>%
  fviz_cluster(data = data[,vars_num],
               ggtheme = theme_gray())

data.pca %>%
  fviz_pca_biplot(alpha.ind = "cos2",
                  geom = "point",
                  habillage = data$Pos %>% factor(),
                  addEllipses = TRUE,
                  ggtheme = theme_gray())
```

Her ses en god opdeling som viser de karakteristika som er typiske for spillerne. Her ses at vægt, højde, og lav m hurtighed, broadjump og vertical er for de lettere spillere. Som forventet er mellem spillerne mellem disse.

supervised analyse

```
data %<>% drop_na %>% select(Pos, everything())

index <- createDataPartition(y = data$Pos, p = 0.90, list = FALSE) # 75% to 25% split

training <- data[index,]
test <- data[-index,]

nrow(training)
nrow(test)

reci <- recipe(Pos ~ ., data = training) %>%
  step_center(all_numeric(), -all_outcomes()) %>% # Centers all numeric variables to r
  step_scale(all_numeric(), -all_outcomes()) %>% # scales all numeric variables to sd
  step_zv(all_predictors())  # Removed predictors with zero variance


# knn inputation of missing values
reci %<>%
  step_knnimpute(all_predictors())

```

```
# Recipe in the end has to be prepared on the training data
reci %<>%
  prep(data = training)

reci

x_train <- bake(reci, new_data = training) %>% select(-Pos)
y_train <- pull(training, Pos) %>% as.factor()

# test: split in y and x
x_test <- bake(reci, new_data = test) %>% select(-Pos)
y_test  <- pull(test, Pos) %>% as.factor()


ctrl <- trainControl(method = "cv", # repeatedcv, boot, cv, LOOCV, timeslice OR adapt
                     number = 2, # Number of CV's
                     classProbs = TRUE, # Include probability of class prediction
                     savePredictions = TRUE, # Save the prediction results
                     summaryFunction = multiClassSummary, # Which type of summary sta
                     verboseIter = FALSE,
                     # add adaptive resampling for hyperparamether search
                     adaptive = list(min = 3,
                                     alpha = 0.05,
                                     method = "gls",
                                     complete = TRUE),
                     search = "random" )

metric <- "Accuracy" # Which metric should be optimized (more on that later)
n_tune = 2 # number of tuning rounds

#x_train$Type1 <- as.character(x_train$Type1)

x_train <- cbind(Number = rownames(x_train), x_train)
x_train$Number <- as.numeric(x_train$Number)


x_test <- cbind(Number = rownames(x_test), x_test)
x_test$Number <- as.numeric(x_test$Number)

x_train %>% head()
y_train %>% head()
x_test %>% head()
y_test %>% head()
```

Vi notere at det er de korrekte 3 levels og fitter vores 3 modeller på kknn, decision tree og random forest.

```
fit_kknn <- train(x = x_train,
                  y = y_train,
                  trControl = ctrl,
                  metric = metric,
                  method = "kknn")

fit_kknn

pred_kknn <- predict(fit_kknn, newdata = x_test)
confusionMatrix(pred_kknn, y_test, positive = "Yes")

fit_dt <- train(x = x_train,
                y = y_train,
                trControl = ctrl,
                metric = metric,
                tuneLength = n_tune,
                method = "rpart")

fit_dt

pred_dt <- predict(fit_dt, newdata = x_test)
confusionMatrix(pred_dt, y_test, positive = "Yes")

fit_rf <- train(x = x_train,
                y = y_train,
                trControl = ctrl,
                metric = metric,
                tuneLength = n_tune,
                method = "ranger",
```