Deploying Several Machine Learning Models and Candidate the Best Model “data1”

Samer Hisham Ismail

Cohort A20

Date

1st of April 2021

Course title

Advanced Statistics and Machine Learning

Pr.Christine Malot

Overview

This document is to report the results of building several Machine Learning models with R, with **"data1",** explaining the **Development** response variable, with several independent variables.

The key to this exercise is to assess the performance of the models and candidate the best one.

Main Modeling settings:

- The problem considered as multivariate supervised machine learning problem. Since we are going to explain continuous variable, the models will be in regression techniques.

- This time we will use a slightly different methodologies in our analysis due to dataset nature, as addition of *Caret* package, we will use some classical ways and functions to get deeper into analysis, as it's sometimes good idea to tweak the parameters manually to get the best results.

- Models' performance will be evaluated with RSME metric.

▪ Project Steps:

1. **Choosing the Packages libraries:**
   As mentioned before, the package Caret will be used to Train, Tune and *Predict()* function for testing the models.

2. **Algorithm Selection:**
   Algorithms used in this project are, Full Linear Regression, Linear Regression with the Best subset of variables, Ridge, Lasso, Principal Component Regression, CART and Random Forest.

3. **Data Exploration:**
   Exploring the dataset and variables type and distribution using plots, assessing, and testing the Gaussianity (Normality) of the residuals using KSsmirnov Test, in addition of correlation analysis between variables using ANOVA (will get into details next).

4. **Data Split:**
   Splitting the data into two parts, Training and Testing using *createDataPartition()* function.

5. **Setting Train Control:**
   This time we will use Cross Validation with 2 folds, might not be the best number of folds to use, but it's the most stable one in terms of resampling without generating some missing values that confuses our analysis.

6. **Building Models:**
   In this step we will be using *train()* function within the 'Caret' Package, to train the models using prespecified *trainControl()*, with Root Mean Squared Error (RSME) as performance metric.

7. **Predictions:**
   Again, with '*Caret'* Package, the function used to predict is *predict(),* using the unseen data 'Testing' dataset, although 'Repeated Cross Validation' method generates decent results, but it is a way to assess models with unseen data to avoid any surprises. The performance metric to be used in prediction is 'RMSE' predicted.

8. **Summarize Training and Prediction models:**
   Using *resamples()* function, and data frame where the results will be stored in, also we will use RSME and RMSE_Predicted to assess the models and candidate the best one.

▪ Importing data1 and needed Packages:

```
### Libraries Import & Installation ###
library(ggplot2)                # Plots and Charts
library(olsrr)                  # Linear Regression utility Package
library(e1071)                  # Utilities Package
library(PerformanceAnalytics)   # Utilities Package for Linear Regression
library(corrplot)               # Plot Correlation Package
library(pls)                    # Package for PCA and PLS
library(caret)                  # Main Package that we will be working on
library(rpart)                  # Package for CART
library(rpartScore)             # CART tools
library(explore)                # Tools Package
library(randomForest)           # Random Forest Package
```

a) Data Import and preparation:

```
### Data Import ###

    setwd("C:/Users/Samer/Desktop/ASML_RAW")
    data1  <- read.csv('data1.csv',header = TRUE, sep= ";")
    str(data1)  #showing dataset structure
    data1
```

```
'data.frame':  12 obs. of  4 variables:
 $ Children    : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Treatment   : Factor w/ 2 levels "placebo","produit actif": 1 2 1 1 2 1 2 1 2 2 ..
 $ Psychologist: int  2 1 1 1 2 1 2 2 1 2 ...
 $ Development : num  36.8 93.7 56.6 37.7 70.3 24.2 42.5 50.9 71.8 72.5 ...
```

- At the first look at the data, we can observe that we have some little work to do in terms of data types, first, we are going to ignore the variable Children, as its and ID variable and it does not have any effect on our analysis.

- The second observation is that there are two categorical variables (Treatment, Psychologist).

- The variable Treatment is automatically detected by R, that it is a factor with two levels, while variable Psychologist is also categorical, but it is defined as an Integer.

- As I do not trust any software when making assumptions, I am going to handle those two variables manually by a process called One-Hot-Encoding, or Creating a Dummy variable, which means that we a re going to replace their values with numeric binary variables of (0,1) and then convert them into factors.

- For variable "Treatment", it was replaced by a new variable called "Treatemnt_placebo", as 1 means placebo and 0 for produit_actif, then turning it into a factor.
- Variable "Psychologist" had been converted into a factor instead of integer.

```
#Creating 2 dummy variables for treatment
data1$Treatemnt_placebo       <- ifelse(data1$Treatment == 'placebo', 1, 0)
data1$Treatemnt_produit_actif <- ifelse(data1$Treatment == 'produit actif', 1, 0)
#Converting the new dummy variables into factors
data1$Treatemnt_placebo       <-factor(data1$Treatemnt_placebo)
data1$Treatemnt_produit_actif <-factor(data1$Treatemnt_produit_actif)
#Converting this categorical variable with two levels into factor
data1$Psychologist            <-factor(data1$Psychologist)

#Eyeballing the data to decide the next step
str(data1)
```

Final data to be used in models building:

```
#selecting only the needed variables from dataset
data1<-data1[,3:5]
data1
str(data1)
```

```
'data.frame':  12 obs. of  3 variables:
 $ Psychologist     : Factor w/ 2 levels "1","2": 2 1 1 1 2 1 2 2 1 2 ...
 $ Development      : num  36.8 93.7 56.6 37.7 70.3 24.2 42.5 50.9 71.8 72.5 ...
 $ Treatemnt_placebo: Factor w/ 2 levels "0","1": 2 1 2 2 1 2 1 2 1 1 ...
```

We can see, our data is now having 2 categorical variables (Predictors) and one numeric Response variable (Development). This problem needs ANOVA, and some trees models might generate good results, we will see as we move deeper into our analysis.

Data Split:

The next step is to split the data 65% into Train (obs=12), Test (obs=4)

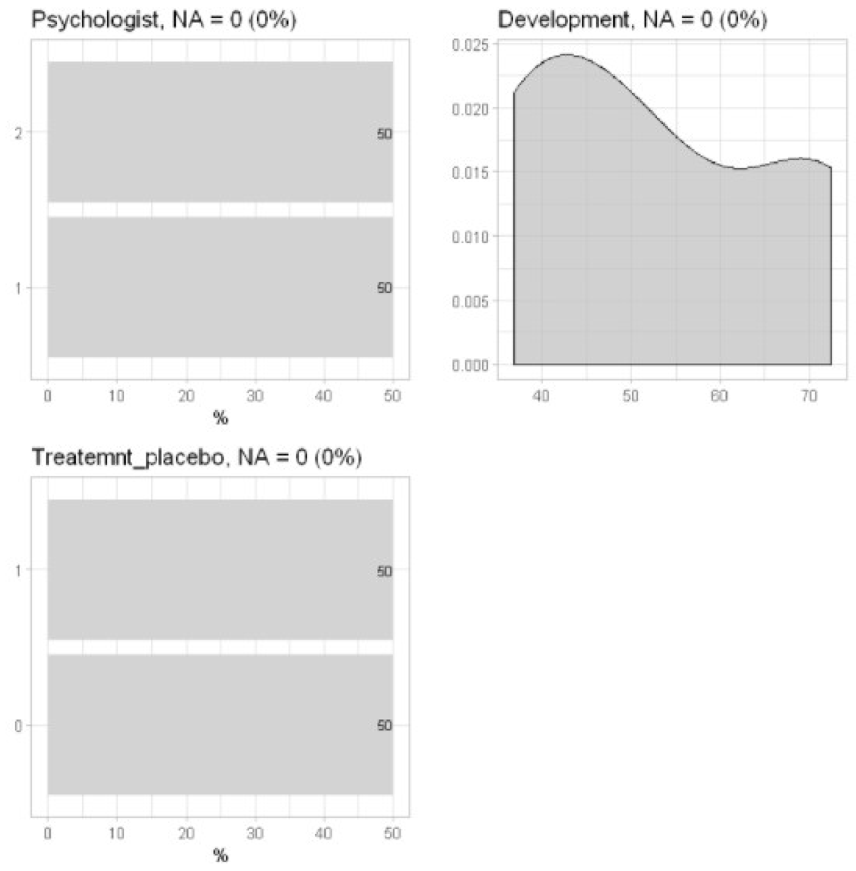
```
### Split the data 65% into Train (obs=12), Test (obs=4) ##
set.seed(1234)
trainingIndex <- createDataPartition(data1$Development, p=0.65, list=FALSE)
training <- data1[trainingIndex,]
testing <- data1[-trainingIndex,]

training
testing
```

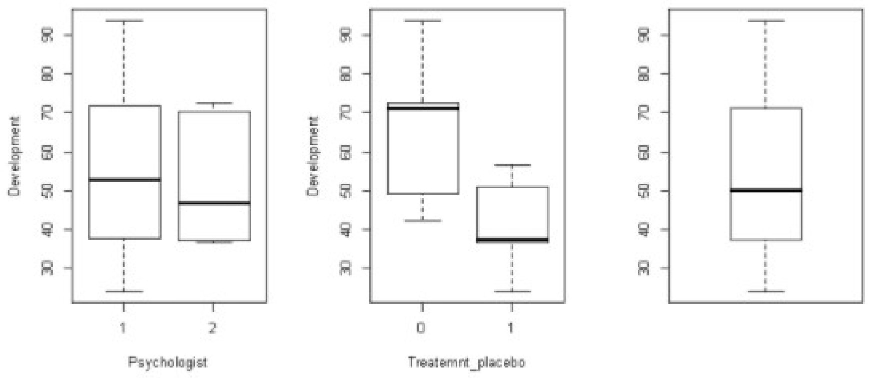
▪ Data Exploration:

At first, we will start with general descriptive statistics analysis to review data and to assess its skewness or whether if data needs to be standardized or normalized.

```
#explore variables
data1 %>%
explore_all()
```

```
# Box Plot
par(mfrow=c(2, 3))
boxplot(Development ~ Psychologist,       data = data1)
boxplot(Development ~ Treatemnt_placebo, data = data1)
boxplot(data1$Development,                 data = data1)
```





- We observe that data is consistent, variables (Treatment_placebo, Psychologist) are equally distributed across the dataset, 6 obs for 1 and 6 obs for 0.

- Also, we found that there is a slight effect on development with psychologist, as children who went to number 1 have slightly more development than psychologist number 2.

- What is more interesting is Treatment_placebo variable, as we can see here, when child have (0 = produit actif) there will be a strong impact on development, this means that when treatment is placebo higher the Development.
  The opposite for placebo, as development tend to decrease according to box blot chart 2.

a. Variables Correlation:

In our case, the best way to find how variables are correlated is by using 2-way ANOVA model additive and interaction effect.

2 Way ANOVA - Additive effect:

```
#two-way ANOVA test
#additive effect
anova_add <- aov(Development ~ Treatemnt_placebo + Psychologist, data = data1)
summary(anova_add)
```

```
                  Df Sum Sq Mean Sq F value Pr(>F)  
Treatemnt_placebo  1 2043.6  2043.6   7.960  0.020 *
Psychologist       1   44.1    44.1   0.172  0.688  
Residuals          9 2310.7   256.7                 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- From the ANOVA table we can conclude that variable "Treatemnt_placebo" is statistically significant, and Psychologist is not significant.

- "Treatemnt_placebo" is the most significant factor variable. These results would lead us to believe that if the child have produit actif or placebo, that will impact Development significantly.
  While Psychologist does not significantly impact mean Development

2 Way ANOVA - Interaction effect:

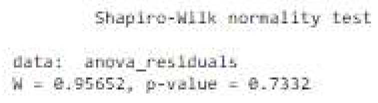
```
#two-way ANOVA test
# Two-way ANOVA with interaction effect
anova_int <- aov(Development ~ Psychologist * Treatemnt_placebo, data = data1)
anova_int <- aov(Development  ~ Psychologist + Treatemnt_placebo +
Psychologist:Treatemnt_placebo, data = data1)
summary(anova_int)
```

```
                               Df Sum Sq Mean Sq F value Pr(>F)
Psychologist                    1   44.1    44.1   0.160 0.6995
Treatemnt_placebo               1 2043.6  2043.6   7.422 0.0261 *
Psychologist:Treatemnt_placebo  1  108.0   108.0   0.392 0.5486
Residuals                       8 2202.7   275.3
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

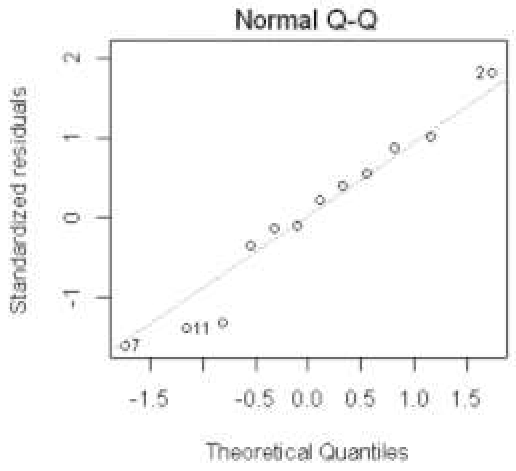
- From the ANOVA Interaction results, we can conclude the following, based on the p-values and a significance level of 0.05:
- The p-value of Treatemnt_placebo is 0.0261 (significant), which indicates that the Treatemnt type is associated with significant development level.
- The p-value of Psychologist is < 0.6995 (not significant), which indicates that the Psychologist has no significant development level.
- The p-value for the interaction between (Psychologist*Treatemnt_placebo) is 0.5486 (not significant), which indicates that the relationships between Psychologist and Treatment_placebo has no effect on Development level.

Normality of the residuals:

```
Normality
par(mfrow=c(2, 2))
plot(anova_add, 2)

# Extract the residuals
anova_residuals <- residuals(object = anova_add)
# Run Shapiro-Wilk test
shapiro.test(x = anova_residuals )
```



```
	Shapiro-Wilk normality test

data:  anova_residuals
W = 0.95652, p-value = 0.7332
```



As all the points fall approximately along this reference line, we can assume normality, by the Shapiro-Wilk test on the ANOVA residuals which finds no indication that normality is violated.

b. *trainControl()* Function Settings:

This function is part of the train function, where we specify the training settings that we want to use in our models, in our models, we will use K-Fold CV with 2 folds, in addition of *savePredictions()* function that allow us to keep the final best model based on RMSE.

Defining training control for cross validation:

```
# Train Control Function
train.control <- trainControl(number = 2, method="cv",savePredictions = "final")
```

▪ Building Models:

As mentioned earlier, we will be using '*Caret*' package to train and *predict()* to test models, in addition we might use another packages like *ols_best_subset*, and also we may have to build the models in a simple classical way, because sometimes I notice that '*Caret*' hides some details underneath its code.

Full Linear Regression Model

```
### Full_Linear Regression Model Cross-Validated ###
set.seed(1234)
lm_full   <- train(Development ~ . , data = training, method = "lm",trControl =
train.control, metric="RMSE")

#Model Results
summary(lm_full$finalModel)
lm_full$results
```

```
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
     X2      X3      X4      X6      X7      X8      X9     X12
 20.561  10.693  -8.207 -21.707 -19.221  16.411  -1.339   2.811

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)           73.14      11.64   6.284   0.0015 **
Psychologist2        -11.42      13.59  -0.840   0.4392
Treatemnt_placebo1   -27.23      13.59  -2.004   0.1015
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.57 on 5 degrees of freedom
Multiple R-squared:  0.4983,   Adjusted R-squared:  0.2977
F-statistic: 2.483 on 2 and 5 DF,  p-value: 0.1783
```

intercept	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
TRUE	36.02484	0.9080448	33.28125	18.01643	0.08366049	15.24699

- We notice some symmetry as first impression, the F- statistics seems to be not significant, and adjusted R- Squared is bad, we will try several models to see if it could be enhanced somehow.

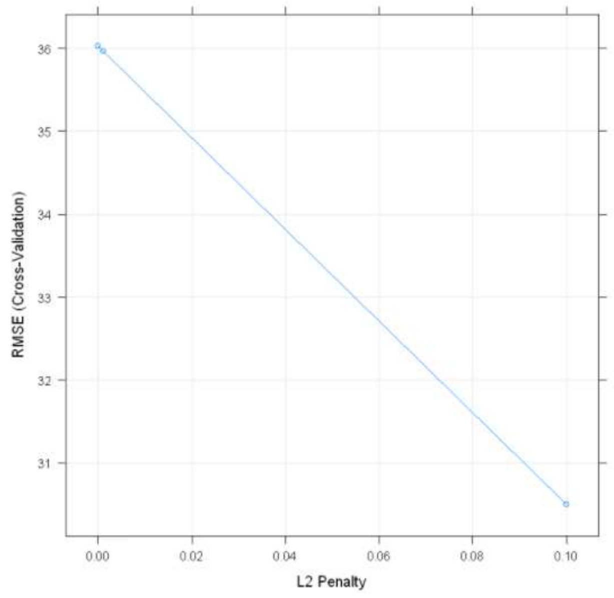
- The p-value of the predictors is not significant as all values are way above 5%

- What is significant is the Variable (Treatement_placebo0) that was included in the intercept.

Regression with Ridge:

```
### Linear Regression with Ridge ###
set.seed(1234)
lm_ridge  <- train(Development ~., data = training, method = "foba",        trControl
= train.control,metric="RMSE")

#Model Results
lm_ridge
print(lm_ridge$results)
# Plot
plot(lm_ridge)
```

```
lambda  RMSE      Rsquared   MAE
  1e-05   36.02420  0.9080447  33.28066
  1e-03   35.96146  0.9080440  33.22267
  1e-01   30.50404  0.9077433  28.12338

Tuning parameter 'k' was held constant at a value of 2
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were k = 2 and lambda = 0.1.
  lambda k     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
1  1e-05 2 36.02420 0.9080447 33.28066 18.015514 0.08366048 15.246142
2  1e-03 2 35.96146 0.9080440 33.22267 17.924904 0.08366019 15.162323
3  1e-01 2 30.50404 0.9077433 28.12338  9.989056 0.08366741  7.757665
```

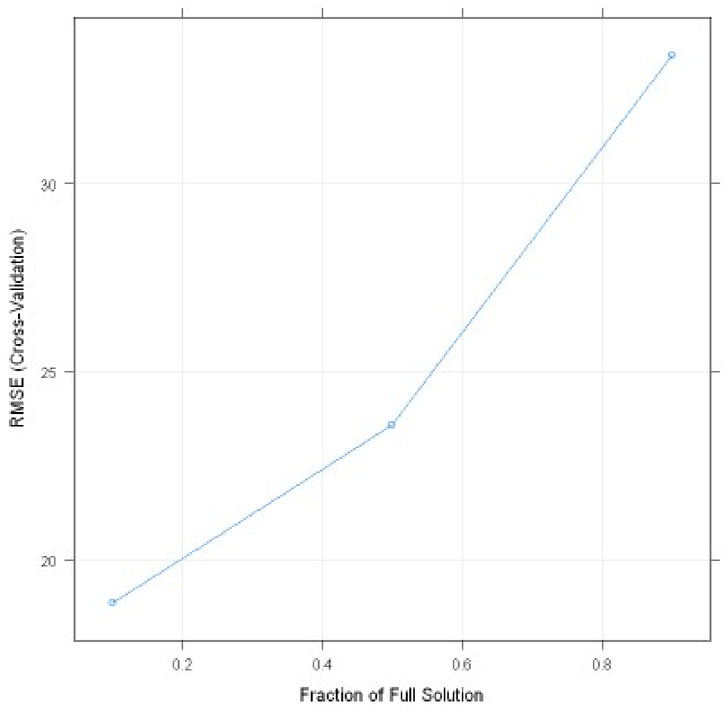


- The best hyperparameters with lowest RMSE, chosen by cross validation are K=2 with Lambda=0.1
- A little bit high RMSE but with good R squared so far.

Regression with Lasso:

```
### Linear Regression with Lasso ###

set.seed(1234)
lm_lasso  <- train(Development ~., data = training, method = "lasso",        trControl
= train.control,metric="RMSE")

#Model Results
lm_lasso
print(lm_lasso$results)
# Plot
plot(lm_lasso)
```

```
fraction  RMSE      Rsquared   MAE
  0.1       18.87387  0.7746757  17.45220
  0.5       23.58066  0.9035587  21.25468
  0.9       33.40878  0.9080028  30.87594

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 0.1.
  fraction     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
1      0.1 18.87387 0.7746757 17.45220 10.362483 0.13574827  9.496019
2      0.5 23.58066 0.9035587 21.25468  1.441882 0.08824471  3.070791
3      0.9 33.40878 0.9080028 30.87594 14.039985 0.08377592 11.583434
```



- The best hyperparameter with lowest RMSE, chosen by cross validation are L1 norm fraction= 0.5
- Very good RMSE, but we sacrificed a little with R squared.

Principal Component Regression:

```
### Principle Component Regression ###

set.seed(1234)
#pca <- pcr(Grade ~ ., data = training, validation = "CV")
pca <- train(Development ~., data = training, method = "pcr",trControl =
train.control,metric="RMSE")

#Model Results
summary(pca)
print(pca$results)
```

```
Data:    X dimension: 8 2
         Y dimension: 8 1
Fit method: svdpc
Number of components considered: 1
TRAINING: % variance explained
          1 comps
X           53.33
.outcome    43.47
  ncomp     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
1     1 19.11733 0.4085012 17.52044 7.171635  0.5624283 7.639648
```

- The selected model is only with one component, at the lowest RMSE.
- Good RMSE, but very low R squared.

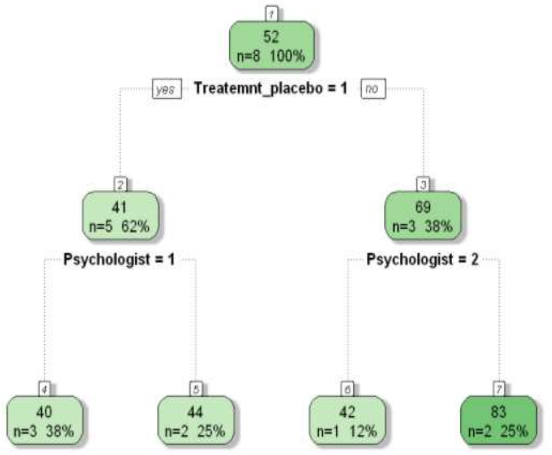
CART Algorithm:

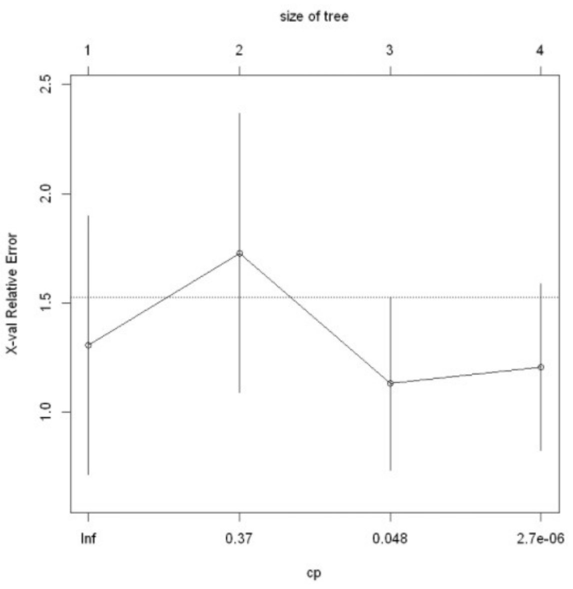
1. CART max:

In this model, we deploy maximum tree with cp=10^-9 and minsplit=2, then we will prune the tree according to 1SE Rule and see the results:

```
## Regression Tree - CART Algorithm - Classical Function
cart_max=rpart(Development ~ ., data=training,cp=10^-9, minsplit=2, method= 'anova')
cart_max
summary(cart_max)
fancyRpartPlot(cart_max)
plotcp(cart_max)
```

```
Call:
rpart(formula = Development ~ ., data = training, method = "anova",
    cp = 10^-9, minsplit = 2)
  n= 8

           CP nsplit rel error   xerror      xstd
1 0.427524941      0 1.0000000 1.306122 0.5923952
2 0.314261706      1 0.5724751 1.728120 0.6397340
3 0.007388357      2 0.2582134 1.130496 0.3971320
4 0.000000001      3 0.2508250 1.204949 0.3815836
```





- Since we cannot use this tree due to over fitting and complexity as it cannot be generalized, so we need to do some pruning, using CV error and 1SE Rule.

- We notice that the CV error is decreasing then increasing and stabilizing above the dotted line which equals to (min of cv error +1*std)

- For the lowest point, its std is very high, which leads to some fluctuation, so we are going try to keep the model with the smallest number of leaves.

- According to 1 SE Rule: we consider all the models with cv error that is smaller than 1 se rule, and we take the smaller model (with smaller leaves), because otherwise we will choose a model which will cause overfitting.

- According to 1SE Rule we choose cp = 0.048, then we will build our model based on this value.

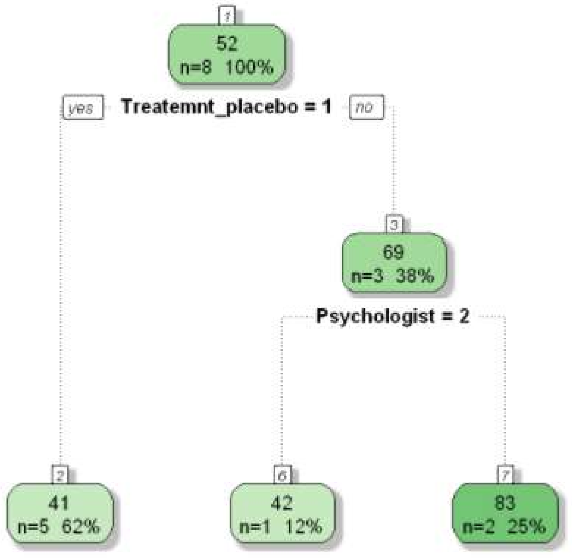
2. CART CP= 0.048:

```
## Regression Tree - CART Algorithm - Classical Function with cp=0.048
cart_cp=rpart(Development ~ ., data=training,cp=0.048, minsplit=2, method= 'anova')
cart_cp
summary(cart_cp)
fancyRpartPlot(cart_cp)
```

```
         CP nsplit rel error   xerror      xstd
1 0.4275249      0 1.0000000 1.306122 0.5923952
2 0.3142617      1 0.5724751 1.728120 0.6397340
3 0.0480000      2 0.2582134 1.044931 0.3943884

Variable importance
Treatemnt_placebo      Psychologist
               58                42
[1] "Train results Cart cp= 0.048"
```

RMSE

10.53219



Variable Importance:

Since there are still some doubts about the code that calculate the Variable Importance, we may not trust it, but we can trust the variable importance generated by Random Forest because it is totally different and because the variable Importance in CART uses surrogate split.

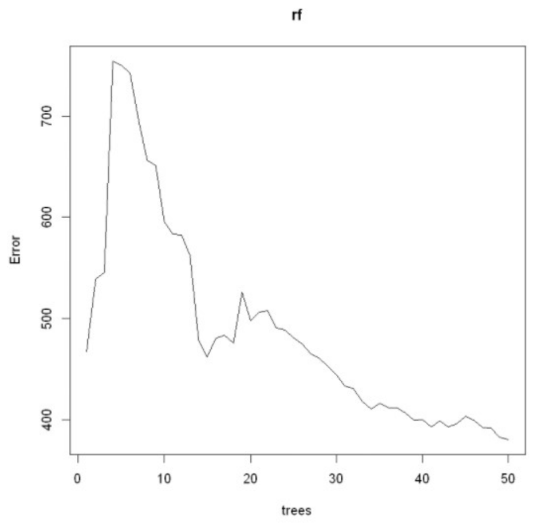
Random Forest:

1. Random Forest – Manually Tuned

To build the best Random forest, there are two parameters which need to be set, mtry and ntree.
We will use the function *bestmtry()* to help us choosing the best mtry for our model, or we can use the rule (floor(p/3)).

```
## Best mtry
bestmtry <- tuneRF(training,training$Development, stepFactor= 1, improve = 0.01, trace=
T, plot= T )

## Random Forest - Manually Tuned
set.seed(1234)
rf_m <- randomForest(Development ~ ., data = training, ntree = 50, mtry = 1,
importance = TRUE, replace = T )
rf_m

# RMSE
print("RMSE")
sqrt(rf_m$mse[length(rf_m$mse)])

plot(rf)
```

```
Call:
 randomForest(formula = Development ~ ., data = training, ntree = 50,      mtry = 1, impo
rtance = TRUE, replace = T)
               Type of random forest: regression
                     Number of trees: 50
No. of variables tried at each split: 1

          Mean of squared residuals: 380.6177
                    % Var explained: 11.4
[1] "RMSE"
19.5094260097989
```



- Noting that we set the value of ntree manually, as we started from 500 and started to increase and decrease according to results, when we select ntree other that 50, % of VAR explained become negative, and this is the best result we could achieve for this model.

2. Random Forest – Cross Validated:

```
### Random Forest - Cross Validated ###

set.seed(1234)
rf_cv <- train(Development ~., data = training, method = "rf",trControl =
train.control, tuneLength = 15,metric="RMSE")

#Model Results
summary(rf_cv)
print(rf_cv$results)
```

```
mtry     RMSE      Rsquared       MAE       RMSESD    RsquaredSD     MAESD
2       19.38933   0.7781806  17.75723    6.624647  0.1016588     7.005996
```

- In the cross validated model, the best mtry was chosen is 2, with almost the same RMSE.

Predict models on Test Dataset:

```
### Predictions
pred_lm      = predict(lm_full,      newdata = testing, metric="RMSE")
pred_pca     = predict(pca,          newdata = testing, metric="RMSE")
pred_ridge   = predict(lm_ridge,     newdata = testing, metric="RMSE")
pred_lasso   = predict(lm_lasso,     newdata = testing, metric="RMSE")
pred_Cart_cp = predict(cart_cp,      newdata = testing, metric="RMSE")
pred_rf_cv   = predict((rf_cv),      newdata = testing, metric="RMSE")
pred_rf_m    = predict((rf_m ),      newdata = testing, metric="RMSE")
```

Summarizing metrics from Training and Testing for Model Comparison:

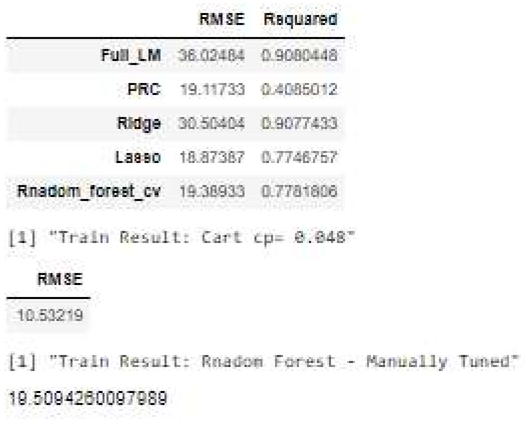
Metrics for Training:

Building the results table to include RMSE and Rsquared:

```
vmetrics <- function(results)
{
  row = c(results$RMSE,
          results$Rsquared)
}
```

Gathering all final best models results:

```
# Results List

Full_LM       = metrics(lm_full$results)
PRC           = metrics(pca$results[1,])
Ridge         = metrics(lm_ridge$results[3,])
Lasso         = metrics(lm_lasso$results[1,])
Rnadom_forest_cv = metrics(rf_cv$results)



model_results = rbind(Full_LM,
                      PRC,
                      Ridge,
                      Lasso,
                      Rnadom_forest_cv)

colnames(model_results) = c("RMSE", "Rsquared")
model_results

## Cart_cp Train Results:
print('Train Result: Cart cp= 0.048')
predictions_train_cart2 = predict(cart_cp, data = training)
eval_results2(training$Development, predictions_train_cart2, training)

## Random Forest - Manually Tuned Train Results:
print("Train Result: Rnadom Forest - Manually Tuned")
sqrt(rf_m$mse[length(rf_m$mse)])
```



	RMSE	Rsquared
Full_LM	36.02484	0.9080448
PRC	19.11733	0.4085012
Ridge	30.50404	0.9077433
Lasso	18.87367	0.7746757
Rnadom_forest_cv	19.38933	0.7781806

```
[1] "Train Result: Cart cp= 0.048"
```

RMSE
10.53219

```
[1] "Train Result: Rnadom Forest - Manually Tuned"
```

19.5094260097989

Metrics for Testing:

```
### Predictions List
pred_RMSE <- list(lm_fullp   =   RMSE(pred_lm,      testing$Development),
                  pcap       =   RMSE(pred_pca,     testing$Development),
                  lm_ridgep  =   RMSE(pred_ridge,   testing$Development),
                  lm_lassop  =   RMSE(pred_lasso,   testing$Development),
                  CART_P     =   RMSE(pred_Cart_cp, testing$Development),
                  RF_CV      =   RMSE(pred_rf_cv,   testing$Development),
                  RF_M       =   RMSE(pred_rf_m,    testing$Development))

pred_RMSE
```

$lm_fullp	13.8150319731214
$pcap	17.777474335518
$lm_ridgep	13.3311728783406
$lm_lassop	14.1387873667838
$CART_P	26.5158165063797
$RF_CV	11.0440302009138
$RF_M	14.0552286203454

Conclusion:

Model	RMSE	RMSE_Predicted (RMSEP)
Full_LM	36.02484	13.81503197
PRC	19.11733	17.77747434
Ridge	30.50404	13.33117288
Lasso	18.87387	14.13878737
Random Forest - CV	19.38933	11.0440302
Random Forest - Manually Tuned	19.50942601	14.05522862
CART- CP = 0.048	10.53219	26.51581651

- The best Training model with lowest RMSE is Linear Regression with Lasso (RMSE 18.87387) with (RMSEP = 14.13878737).

- Our winner model and best candidate is **Random Forest – CV** (RMSE 19.38933) and (RMSEP 11.0440302).

- The reason why I selected this model to be our hero model, is that because it has the second-best Training RMSE with slight difference between it and Lasso. In addition, it generated the lowest RMSEP, so I sacrificed some fractions form training RMSE to let it win :D

- Obviously, we totally ignored CART_CP model because its strongly overfitting, which is not surprising because when we implemented the model, its result was not much different than maximum tree which for sure will cause overfitting and cannot be generalized.

- I am sure if we had larger dataset to generate more robust, as working with small datasets in my opinion is hard, in terms of cross validation for instance, we had to set a small number of folds to make sure that everything run smoothly, but with more data we can for sure reduce the error rate significantly.