Alunos da equipe:

Equipe 03

- Gustavo Costa de Souza
- Marcos Vinicius de Melo
- Marcus Eneas Silveira Galvao do Rio Apa II
- Patrícia Verdugo Pascoal
- Rodrigo de Araujo
- William de Souza Alencar

Seed utilizado: 2038
(Ano atual com 4 dígitos + 2 algarismos do dígito verificador do CPF de um dos integrantes)

Especificações:
O trabalho pode ser feito por uma equipe de 1 a 6 integrantes.
Para cada problema, preencher as colunas dos quadros com o que pede.
Além disso, fazer as solicitações pedidas antes dos quadros.

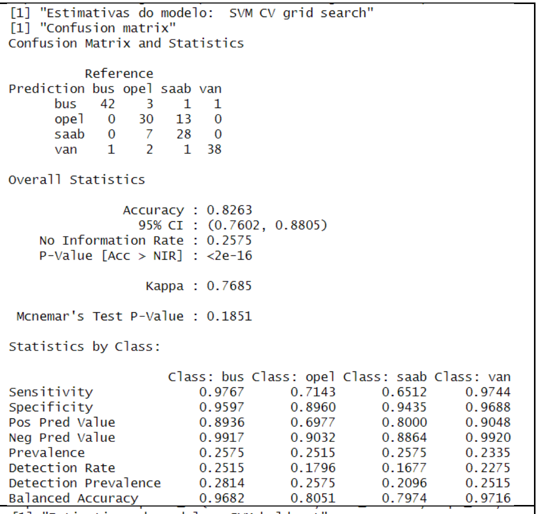
CLASSIFICAÇÃO

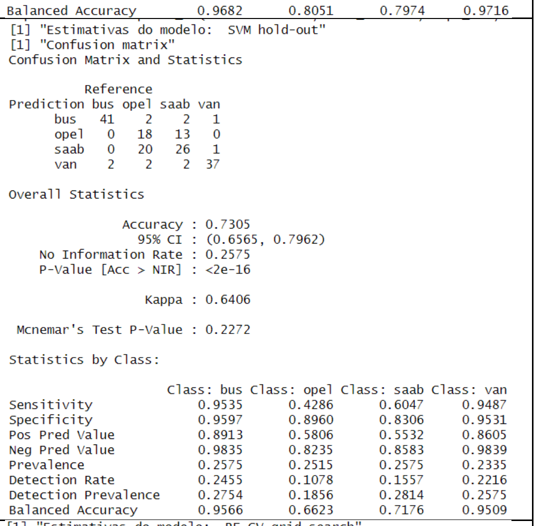
Para o experimento de Classificação:

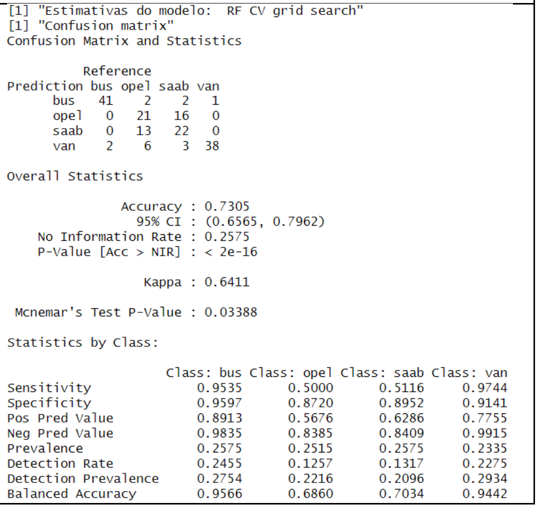
- Ordenar pela Acurácia (descendente), ou seja, a técnica de melhor acurácia ficará em primeiro na tabela.
- Após o quadro colocar:
    - Um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior Acurácia (criar um arquivo com novos casos à sua escolha)
    - A lista de comandos emitidos no RStudio para conseguir os resultados obtidos

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão

SVM – CV	C=100
	Sigma=0.015
		0.82	

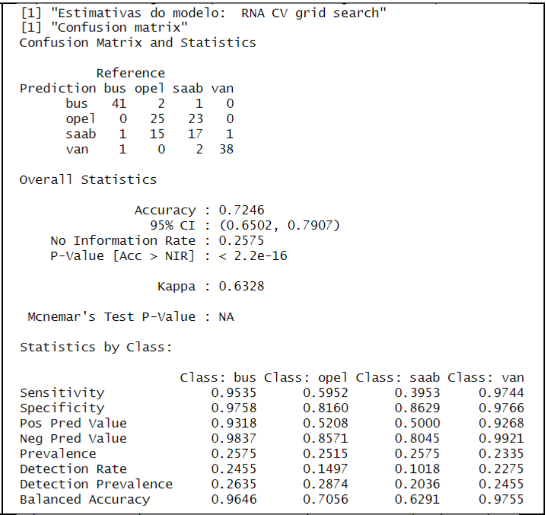
SVM – Hold-out	C=1
	Sigma=0.07156
		0.73	

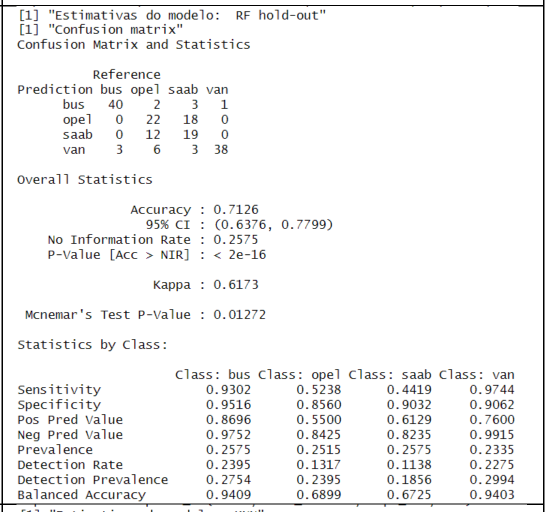
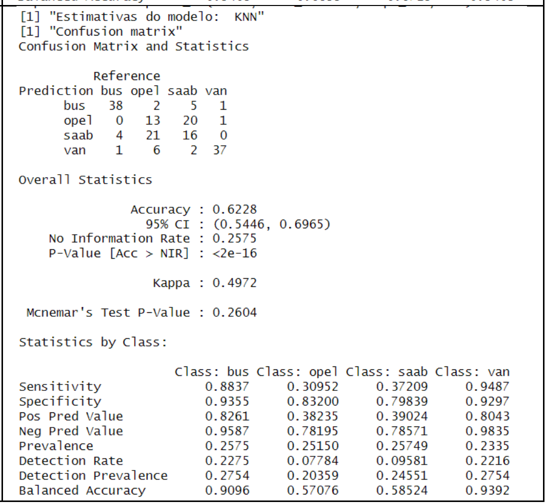
RF – CV	mtry=7	0.73	

```
[1] "Estimativas do modelo:  SVM CV grid search"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   42    3    1   1
      opel   0   30   13   0
      saab   0    7   28   0
      van    1    2    1  38

Overall Statistics

               Accuracy : 0.8263
                 95% CI : (0.7602, 0.8805)
    No Information Rate : 0.2575
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7685

 Mcnemar's Test P-Value : 0.1851

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity              0.9767      0.7143      0.6512     0.9744
Specificity              0.9597      0.8960      0.9435     0.9688
Pos Pred Value           0.8936      0.6977      0.8000     0.9048
Neg Pred Value           0.9917      0.9032      0.8864     0.9920
Prevalence               0.2575      0.2515      0.2575     0.2335
Detection Rate           0.2515      0.1796      0.1677     0.2275
Detection Prevalence     0.2814      0.2575      0.2096     0.2515
Balanced Accuracy        0.9682      0.8051      0.7974     0.9716
```

```
[1] "Estimativas do modelo:  SVM hold-out"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   41    2    2   1
      opel   0   18   13   0
      saab   0   20   26   1
      van    2    2    2  37

Overall Statistics

               Accuracy : 0.7305
                 95% CI : (0.6565, 0.7962)
    No Information Rate : 0.2575
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6406

 Mcnemar's Test P-Value : 0.2272

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity              0.9535      0.4286      0.6047     0.9487
Specificity              0.9597      0.8960      0.8306     0.9531
Pos Pred Value           0.8913      0.5806      0.5532     0.8605
Neg Pred Value           0.9835      0.8235      0.8583     0.9839
Prevalence               0.2575      0.2515      0.2575     0.2335
Detection Rate           0.2455      0.1078      0.1557     0.2216
Detection Prevalence     0.2754      0.1856      0.2814     0.2575
Balanced Accuracy        0.9566      0.6623      0.7176     0.9509
```

```
[1] "Estimativas do modelo:  RF CV grid search"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   41    2    2   1
      opel   0   21   16   0
      saab   0   13   22   0
      van    2    6    3  38

Overall Statistics

               Accuracy : 0.7305
                 95% CI : (0.6565, 0.7962)
    No Information Rate : 0.2575
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6411

 Mcnemar's Test P-Value : 0.03388

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity              0.9535      0.5000      0.5116     0.9744
Specificity              0.9597      0.8720      0.8952     0.9141
Pos Pred Value           0.8913      0.5676      0.6286     0.7755
Neg Pred Value           0.9835      0.8385      0.8409     0.9915
Prevalence               0.2575      0.2515      0.2575     0.2335
Detection Rate           0.2455      0.1257      0.1317     0.2275
Detection Prevalence     0.2754      0.2216      0.2096     0.2934
Balanced Accuracy        0.9566      0.6860      0.7034     0.9442
```

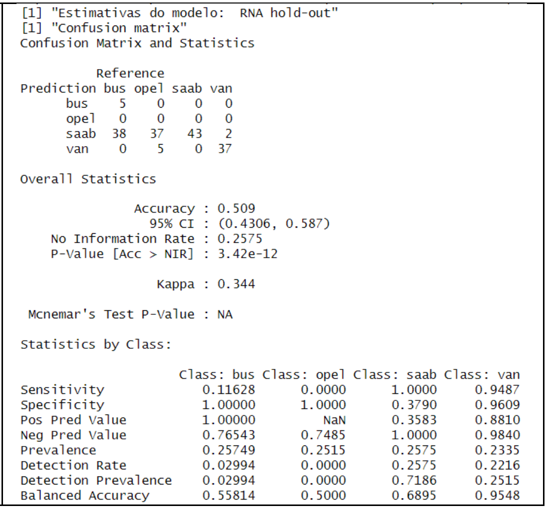
RNA – CV	size=21
	decay=0.7
		0.72	

RF – Hold-out	mtry=10	0.71	
KNN	k=1	0.62	

```
[1] "Estimativas do modelo:  RNA CV grid search"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   41    2    1   0
      opel   0   25   23   0
      saab   1   15   17   1
      van    1    0    2  38

Overall Statistics

               Accuracy : 0.7246
                 95% CI : (0.6502, 0.7907)
    No Information Rate : 0.2575
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6328

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity              0.9535      0.5952      0.3953     0.9744
Specificity              0.9758      0.8160      0.8629     0.9766
Pos Pred Value           0.9318      0.5208      0.5000     0.9268
Neg Pred Value           0.9837      0.8571      0.8045     0.9921
Prevalence               0.2575      0.2515      0.2575     0.2335
Detection Rate           0.2455      0.1497      0.1018     0.2275
Detection Prevalence     0.2635      0.2874      0.2036     0.2455
Balanced Accuracy        0.9646      0.7056      0.6291     0.9755
```

```
[1] "Estimativas do modelo:  RF hold-out"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   40    2    3   1
      opel   0   22   18   0
      saab   0   12   19   0
      van    3    6    3  38

Overall Statistics

               Accuracy : 0.7126
                 95% CI : (0.6376, 0.7799)
    No Information Rate : 0.2575
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6173

 Mcnemar's Test P-Value : 0.01272

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity              0.9302      0.5238      0.4419     0.9744
Specificity              0.9516      0.8560      0.9032     0.9062
Pos Pred Value           0.8696      0.5500      0.6129     0.7600
Neg Pred Value           0.9752      0.8425      0.8235     0.9915
Prevalence               0.2575      0.2515      0.2575     0.2335
Detection Rate           0.2395      0.1317      0.1138     0.2275
Detection Prevalence     0.2754      0.2395      0.1856     0.2994
Balanced Accuracy        0.9409      0.6899      0.6725     0.9403
```

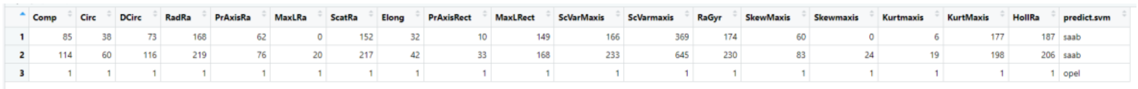
```
[1] "Estimativas do modelo:  KNN"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   38    2    5   1
      opel   0   13   20   1
      saab   4   21   16   0
      van    1    6    2  37

Overall Statistics

               Accuracy : 0.6228
                 95% CI : (0.5446, 0.6965)
    No Information Rate : 0.2575
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.4972

 Mcnemar's Test P-Value : 0.2604

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity              0.8837     0.30952     0.37209     0.9487
Specificity              0.9355     0.83200     0.79839     0.9297
Pos Pred Value           0.8261     0.38235     0.39024     0.8043
Neg Pred Value           0.9587     0.78195     0.78571     0.9835
Prevalence               0.2575     0.25150     0.25749     0.2335
Detection Rate           0.2275     0.07784     0.09581     0.2216
Detection Prevalence     0.2754     0.20359     0.24551     0.2754
Balanced Accuracy        0.9096     0.57076     0.58524     0.9392
```

RNA – Hold-out	size=5
	decay=0.1
		0.50	

```
[1] "Estimativas do modelo:  RNA hold-out"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus    5    0    0   0
      opel   0    0    0   0
      saab  38   37   43   2
      van    0    5    0  37

Overall Statistics

               Accuracy : 0.509
                 95% CI : (0.4306, 0.587)
    No Information Rate : 0.2575
    P-Value [Acc > NIR] : 3.42e-12

                  Kappa : 0.344

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity             0.11628      0.0000      1.0000     0.9487
Specificity             1.00000      1.0000      0.3790     0.9609
Pos Pred Value          1.00000         NaN      0.3583     0.8810
Neg Pred Value          0.76543      0.7485      1.0000     0.9840
Prevalence              0.25749      0.2515      0.2575     0.2335
Detection Rate          0.02994      0.0000      0.2575     0.2216
Detection Prevalence    0.02994      0.0000      0.7186     0.2515
Balanced Accuracy       0.55814      0.5000      0.6895     0.9548
```

O modelo que obteve o melhor desempenho foi o SVM com cross-validation, alcançando uma acurácia de 0,82 com os parâmetros custo (C) = 100 e sigma = 0,015. Esse resultado indica que a técnica de SVM, aliada à validação cruzada e ao ajuste adequado de parâmetros, proporcionou maior capacidade de generalização em comparação aos demais modelos avaliados.

Com base no modelo selecionado, a imagem abaixo apresenta a predição de três novos casos realizada pelo algoritmo SVM.



	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	predict.svm
1	85	38	73	168	62	0	152	32	10	149	166	369	174	60	0	6	177	187	saab
2	114	60	116	219	76	20	217	42	33	168	233	645	230	83	24	19	198	206	saab
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	opel

Abaixo são os comandos no R Studio para chegar no resultado.

```
pacotes <- c("caret","e1071", "mlbench", "mice")

# Instalando e carregando os pacotes necessarios
if(sum(as.numeric(!pacotes %in% installed.packages()))!=0){
  instalador <- pacotes[!pacotes %in% installed.packages()]
  for(i in 1:length(instalador)) {
    install.packages(instalador, dependencies = T)
    break()}
  sapply(pacotes, require, character = T)
} else {
  sapply(pacotes, require, character = T)
}

# Configurando o seed
SEED <- 2038
set.seed(SEED)
```

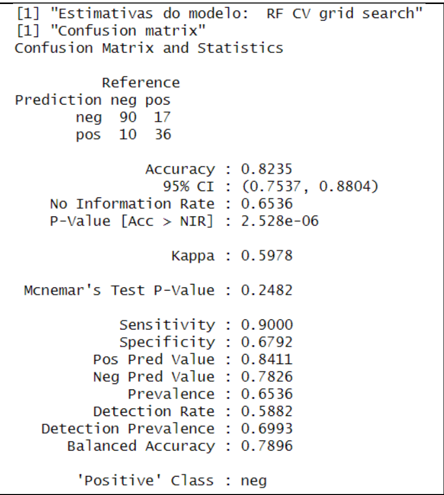
```
# função para dividir as bases
split_train_test <- function(df, target_var) {
  set.seed(SEED)
  indexes <- createDataPartition(df[[target_var]], p=0.80,
list=FALSE)
  train <- df[indexes,]
  test <- df[-indexes, ]
  return(list(train=train, test=test))
}

print_cf <- function(name, df_test, cf_factor, model) {
  print(paste("Estimativas do modelo: ", name))
  predict.model <- predict(model, df_test)

  print("Confusion matrix")
  print(confusionMatrix(predict.model, cf_factor))

  return(predict.model)
}



setwd("C:/Users/rodri/machine-learning/UFPR-IAAP/IAA008 -
Aprendizado de máquina")

# Veículos

df_veiculos <- read.csv("base/06 - Veículos/6 - Veiculos -
Dados.csv")
#View(df_veiculos)
temp_df <- df_veiculos
temp_df$a <- NULL
imp <- mice(temp_df)
df_veiculos <- complete(imp, 1)
df_veiculos[["tipo"]] <- as.factor(df_veiculos[["tipo"]])
View(df_veiculos)

# Divisão da base de dados
split_df <- split_train_test(df_veiculos, "tipo")
train_veiculo = split_df$train
test_veiculo = split_df$test

tipo_fct <- as.factor(test_veiculo$tipo)

ctrl <- trainControl(method="cv", number=10)

# KNN
tuneGrid_knn <- expand.grid(k=c(1,3,5,7,9))
set.seed(SEED)
knn <- train(tipo~., data=train_veiculo, method = "knn", tuneGrid =
tuneGrid_knn)
```

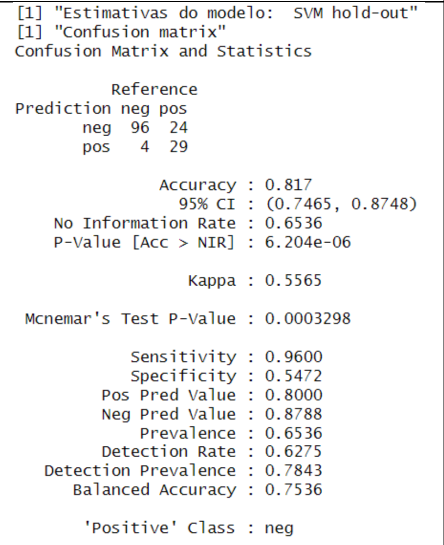
```
knn
predict.knn <- print_cf("KNN", test_veiculo, tipo_fct, knn)

# RNA Hold-out
set.seed(SEED)
rna <- train(tipo~., data=train_veiculo, method = "nnet", trace =
FALSE)
rna
predict.rna <- print_cf("RNA hold-out", test_veiculo, tipo_fct, rna)

# RNA CV
set.seed(SEED)
rna_cv <- train(tipo~., data=train_veiculo, method = "nnet", trace =
FALSE, trControl = ctrl)
rna_cv
predict.rna_cv <- print_cf("RNA CV", test_veiculo, tipo_fct, rna_cv)

# RNA grid search
grid_rna <- expand.grid(size = seq(from=1,to=45, by=10),
decay=seq(from=0.1,to=0.9,by=0.3))
set.seed(SEED)
rna_grid <- train(tipo~., data=train_veiculo, method = "nnet", trace
= FALSE, trControl = ctrl, tuneGrid = grid_rna)
rna_grid
predict.rna_cv_grid <- print_cf("RNA CV grid search", test_veiculo,
tipo_fct, rna_grid)

# SVM
set.seed(SEED)
svm <- train(tipo~., data = train_veiculo, method = "svmRadial")
svm
predict.svm <- print_cf("SVM hold-out", test_veiculo, tipo_fct, svm)

# SVM CV
set.seed(SEED)
svm_cv <- train(tipo~., data = train_veiculo, method = "svmRadial",
trControl = ctrl)
svm_cv
predict.svm_cv <- print_cf("SVM CV", test_veiculo, tipo_fct, svm_cv)

# SVM CV Grid
grid_cv <- expand.grid(C=c(1,2,10,50,100), sigma=c(.01,.015,.2))
set.seed(SEED)
svm_cv_grid <- train(tipo~., data = train_veiculo, method =
"svmRadial", trControl = ctrl, tuneGrid=grid_cv)
svm_cv_grid
predict.svm_cv_grid <- print_cf("SVM CV grid search", test_veiculo,
tipo_fct, svm_cv_grid)

# Randon forest
set.seed(SEED)
rf <- train(tipo~., data = train_veiculo, method="rf")
```

```
rf
predict.rf <- print_cf("RF hold-out", test_veiculo, tipo_fct, rf)

# RF CV
set.seed(SEED)
rf_cv <- train(tipo~., data = train_veiculo, method="rf", trControl
= ctrl)
rf_cv
predict.rf_cv <- print_cf("RF CV", test_veiculo, tipo_fct, rf_cv)

# RF CV grid search
grid_rf = expand.grid(mtry=c(2,5,7,9))
set.seed(SEED)
rf_cv_grid <- train(tipo~., data = train_veiculo, method="rf",
trControl = ctrl, tuneGrid = grid_rf)
rf_cv_grid
predict.rf_cv_grid <- print_cf("RF CV grid search", test_veiculo,
tipo_fct, rf_cv_grid)

### Predições
new_data <- read.csv("base/06 - Veículos/6 - Veiculos - Dados -
Novos Casos.csv")
View(new_data)

predict.svm <- predict(svm_cv_grid, new_data)
new_data$tipo <- NULL
result <- cbind(new_data, predict.svm)
View(result)
```

Diabetes

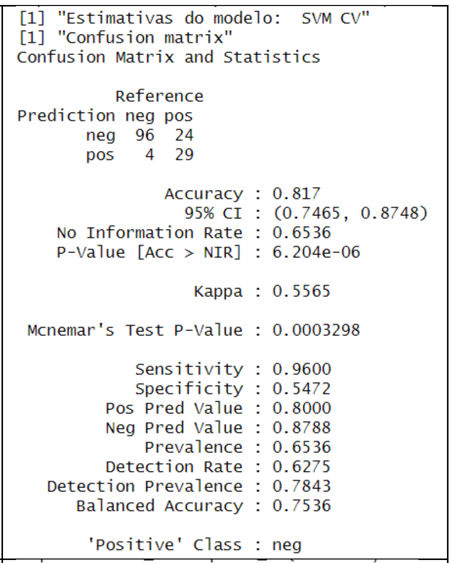
Técnica	Parâmetro	Acurácia	Matriz de Confusão

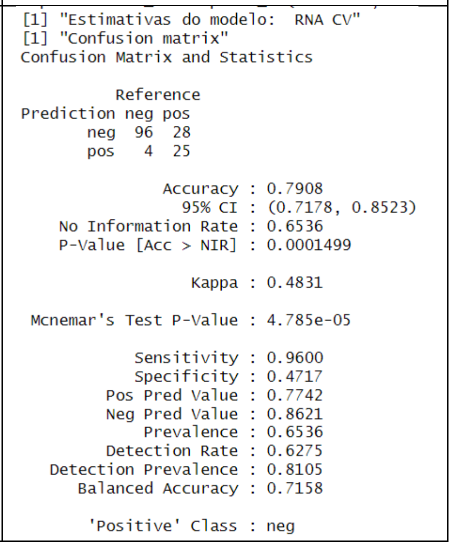
RF – CV	mtry=5	0.82	

SVM – Hold-out	C=0.25
	Sigma=0.128
		0.81	

```
[1] "Estimativas do modelo:  RF CV grid search"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  90  17
       pos  10  36

               Accuracy : 0.8235
                 95% CI : (0.7537, 0.8804)
    No Information Rate : 0.6536
    P-Value [Acc > NIR] : 2.528e-06

                  Kappa : 0.5978

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9000
            Specificity : 0.6792
         Pos Pred Value : 0.8411
         Neg Pred Value : 0.7826
             Prevalence : 0.6536
         Detection Rate : 0.5882
   Detection Prevalence : 0.6993
      Balanced Accuracy : 0.7896

       'Positive' Class : neg
```

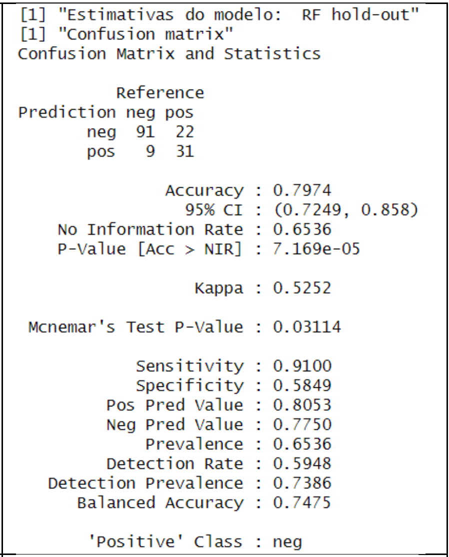
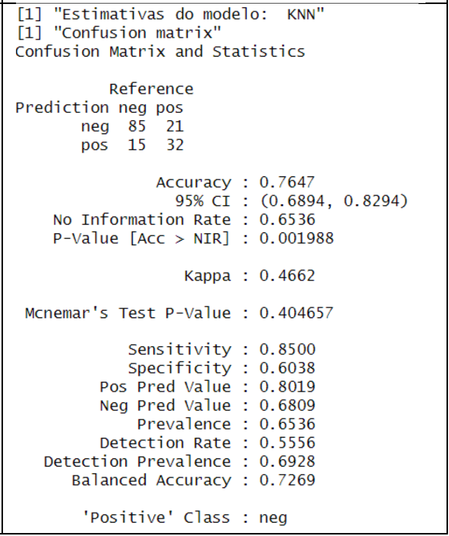
```
[1] "Estimativas do modelo:  SVM hold-out"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  96  24
       pos   4  29

               Accuracy : 0.817
                 95% CI : (0.7465, 0.8748)
    No Information Rate : 0.6536
    P-Value [Acc > NIR] : 6.204e-06

                  Kappa : 0.5565

 Mcnemar's Test P-Value : 0.0003298

            Sensitivity : 0.9600
            Specificity : 0.5472
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.8788
             Prevalence : 0.6536
         Detection Rate : 0.6275
   Detection Prevalence : 0.7843
      Balanced Accuracy : 0.7536

       'Positive' Class : neg
```

SVM – CV	C=0.25
	Sigma=0.128
		0.81	

RNA – CV	size=3 decay=0.1	0.79	

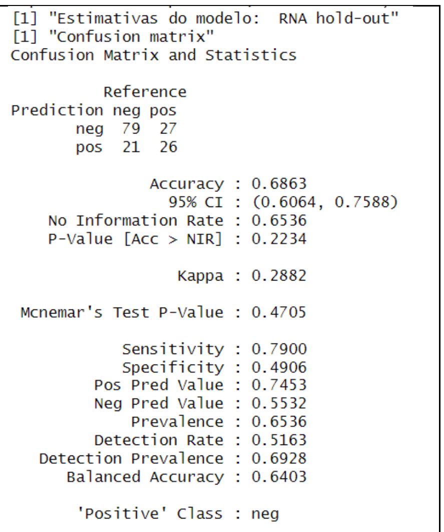
```
[1] "Estimativas do modelo:  SVM CV"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  96  24
       pos   4  29

               Accuracy : 0.817
                 95% CI : (0.7465, 0.8748)
    No Information Rate : 0.6536
    P-Value [Acc > NIR] : 6.204e-06

                  Kappa : 0.5565

 Mcnemar's Test P-Value : 0.0003298

            Sensitivity : 0.9600
            Specificity : 0.5472
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.8788
             Prevalence : 0.6536
         Detection Rate : 0.6275
   Detection Prevalence : 0.7843
      Balanced Accuracy : 0.7536

       'Positive' Class : neg
```

```
[1] "Estimativas do modelo:  RNA CV"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  96  28
       pos   4  25

               Accuracy : 0.7908
                 95% CI : (0.7178, 0.8523)
    No Information Rate : 0.6536
    P-Value [Acc > NIR] : 0.0001499

                  Kappa : 0.4831

 Mcnemar's Test P-Value : 4.785e-05

            Sensitivity : 0.9600
            Specificity : 0.4717
         Pos Pred Value : 0.7742
         Neg Pred Value : 0.8621
             Prevalence : 0.6536
         Detection Rate : 0.6275
   Detection Prevalence : 0.8105
      Balanced Accuracy : 0.7158

       'Positive' Class : neg
```

RF – Hold-out	mtry=2	0.79	
KNN	k=9	0.76	

```
[1] "Estimativas do modelo:  RF hold-out"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  91  22
       pos   9  31

               Accuracy : 0.7974
                 95% CI : (0.7249, 0.858)
    No Information Rate : 0.6536
    P-Value [Acc > NIR] : 7.169e-05

                  Kappa : 0.5252

 Mcnemar's Test P-Value : 0.03114

            Sensitivity : 0.9100
            Specificity : 0.5849
         Pos Pred Value : 0.8053
         Neg Pred Value : 0.7750
             Prevalence : 0.6536
         Detection Rate : 0.5948
   Detection Prevalence : 0.7386
      Balanced Accuracy : 0.7475

       'Positive' Class : neg
```

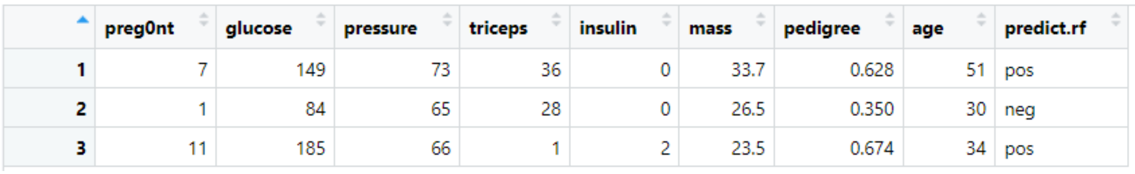
```
[1] "Estimativas do modelo:  KNN"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  85  21
       pos  15  32

               Accuracy : 0.7647
                 95% CI : (0.6894, 0.8294)
    No Information Rate : 0.6536
    P-Value [Acc > NIR] : 0.001988

                  Kappa : 0.4662

 Mcnemar's Test P-Value : 0.404657

            Sensitivity : 0.8500
            Specificity : 0.6038
         Pos Pred Value : 0.8019
         Neg Pred Value : 0.6809
             Prevalence : 0.6536
         Detection Rate : 0.5556
   Detection Prevalence : 0.6928
      Balanced Accuracy : 0.7269

       'Positive' Class : neg
```

RNA – Hold-out	size=3 decay=0.1	0.68	

```
[1] "Estimativas do modelo:  RNA hold-out"
[1] "Confusion matrix"
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  79  27
       pos  21  26

               Accuracy : 0.6863
                 95% CI : (0.6064, 0.7588)
    No Information Rate : 0.6536
    P-Value [Acc > NIR] : 0.2234

                  Kappa : 0.2882

 Mcnemar's Test P-Value : 0.4705

            Sensitivity : 0.7900
            Specificity : 0.4906
         Pos Pred Value : 0.7453
         Neg Pred Value : 0.5532
             Prevalence : 0.6536
         Detection Rate : 0.5163
   Detection Prevalence : 0.6928
      Balanced Accuracy : 0.6403

       'Positive' Class : neg
```

O modelo que obteve o melhor desempenho foi o Random Forest com cross-validation, alcançando uma acurácia de 0,82 com o parâmetro mtry = 5. Esse resultado demonstra que a combinação da técnica Random Forest com validação cruzada e ajuste adequado do parâmetro (mtry) via grid search resultou em maior capacidade de generalização em relação aos demais modelos testados.

Com base no modelo selecionado, a imagem abaixo apresenta a predição de três novos casos realizada pelo algoritmo RF.



	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	predict.rf
1	7	149	73	36	0	33.7	0.628	51	pos
2	1	84	65	28	0	26.5	0.350	30	neg
3	11	185	66	1	2	23.5	0.674	34	pos

Abaixo são os comandos no R Studio para chegar no resultado.

```
pacotes <- c("caret","e1071", "mlbench", "mice")

# Instalando e carregando os pacotes necessarios
if(sum(as.numeric(!pacotes %in% installed.packages()))!=0){
  instalador <- pacotes[!pacotes %in% installed.packages()]
  for(i in 1:length(instalador)) {
    install.packages(instalador, dependencies = T)
    break()}
  sapply(pacotes, require, character = T)
} else {
  sapply(pacotes, require, character = T)
}
```

```
# Configurando o seed
SEED <- 2038
set.seed(SEED)

# função para dividir as bases
split_train_test <- function(df, target_var) {
  set.seed(SEED)
  indexes <- createDataPartition(df[[target_var]], p=0.80,
list=FALSE)
  train <- df[indexes,]
  test <- df[-indexes, ]
  return(list(train=train, test=test))
}

print_cf <- function(name, df_test, cf_factor, model) {
  print(paste("Estimativas do modelo: ", name))
  predict.model <- predict(model, df_test)

  print("Confusion matrix")
  print(confusionMatrix(predict.model, cf_factor))

  return(predict.model)
}



setwd("C:/Users/rodri/machine-learning/UFPR-IAAP/IAA008 -
Aprendizado de máquina")

# Diabetes

df_diabetes <- read.csv("base/10 - Diabetes/10 - Diabetes -
Dados.csv")
#View(df_diabetes)
temp_df <- df_diabetes
temp_df$num <- NULL
imp <- mice(temp_df)
df_diabetes <- complete(imp, 1)
df_diabetes[["diabetes"]] <- as.factor(df_diabetes[["diabetes"]])
View(df_diabetes)

# Divisão da base de dados
split_df <- split_train_test(df_diabetes, "diabetes")
train_diabetes = split_df$train
test_diabetes = split_df$test

diabetes_fct <- as.factor(test_diabetes$diabetes)
ctrl <- trainControl(method="cv", number=10)

# KNN
tuneGrid_knn <- expand.grid(k=c(1,3,5,7,9))
```

```
set.seed(SEED)
knn <- train(diabetes~., data=train_diabetes, method = "knn",
tuneGrid = tuneGrid_knn)
knn
predict.knn <- print_cf("KNN", test_diabetes, diabetes_fct, knn)

# RNA Hold-out
set.seed(SEED)
rna <- train(diabetes~., data=train_diabetes, method = "nnet", trace
= FALSE)
rna
predict.rna <- print_cf("RNA hold-out", test_diabetes, diabetes_fct,
rna)

# RNA CV
set.seed(SEED)
rna_cv <- train(diabetes~., data=train_diabetes, method = "nnet",
trace = FALSE, trControl = ctrl)
rna_cv
predict.rna_cv <- print_cf("RNA CV", test_diabetes, diabetes_fct,
rna_cv)

# RNA grid search
grid_rna <- expand.grid(size = seq(from=1,to=45, by=10),
decay=seq(from=0.1,to=0.9,by=0.3))
set.seed(SEED)
rna_grid <- train(diabetes~., data=train_diabetes, method = "nnet",
trace = FALSE, trControl = ctrl, tuneGrid = grid_rna)
rna_grid
predict.rna_cv_grid <- print_cf("RNA CV grid search", test_diabetes,
diabetes_fct, rna_grid)

# SVM
set.seed(SEED)
svm <- train(diabetes~., data = train_diabetes, method =
"svmRadial")
svm
predict.svm <- print_cf("SVM hold-out", test_diabetes, diabetes_fct,
svm)

# SVM CV
set.seed(SEED)
svm_cv <- train(diabetes~., data = train_diabetes, method =
"svmRadial", trControl = ctrl)
svm_cv
predict.svm_cv <- print_cf("SVM CV", test_diabetes, diabetes_fct,
svm_cv)

# SVM CV Grid
grid_cv <- expand.grid(C=c(1,2,10,50,100), sigma=c(.01,.015,.2))
set.seed(SEED)
```

```
svm_cv_grid <- train(diabetes~., data = train_diabetes, method =
"svmRadial", trControl = ctrl, tuneGrid=grid_cv)
svm_cv_grid
predict.svm_cv_grid <- print_cf("SVM CV grid search", test_diabetes,
diabetes_fct, svm_cv_grid)

# Randon forest
set.seed(SEED)
rf <- train(diabetes~., data = train_diabetes, method="rf")
rf
predict.rf <- print_cf("RF hold-out", test_diabetes, diabetes_fct,
rf)

# RF CV
set.seed(SEED)
rf_cv <- train(diabetes~., data = train_diabetes, method="rf",
trControl = ctrl)
rf_cv
predict.rf_cv <- print_cf("RF CV", test_diabetes, diabetes_fct,
rf_cv)

# RF CV grid search
grid_rf = expand.grid(mtry=c(2,5,7,9))
set.seed(SEED)
rf_cv_grid <- train(diabetes~., data = train_diabetes, method="rf",
trControl = ctrl, tuneGrid = grid_rf)
rf_cv_grid
predict.rf_cv_grid <- print_cf("RF CV grid search", test_diabetes,
diabetes_fct, rf_cv_grid)

### Predições
new_data <- read.csv("base/10 - Diabetes/10 - Diabetes - Dados -
Novos Casos.csv")
View(new_data)

predict.rf <- predict(rf_cv_grid, new_data)
new_data$diabetes <- NULL
result <- cbind(new_data, predict.rf)
View(result)
```

REGRESSÃO

Para o experimento de Regressão:

- Ordenar por R2 descendente, ou seja, a técnica de melhor R2 ficará em primeiro na tabela.

- Após o quadro, colocar:
  - Um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior R2 (criar um arquivo com novos casos à sua escolha)
  - O Gráfico de Resíduos para a técnica/parâmetro de maior R2
  - A lista de comandos emitidos no RStudio para conseguir os resultados obtidos

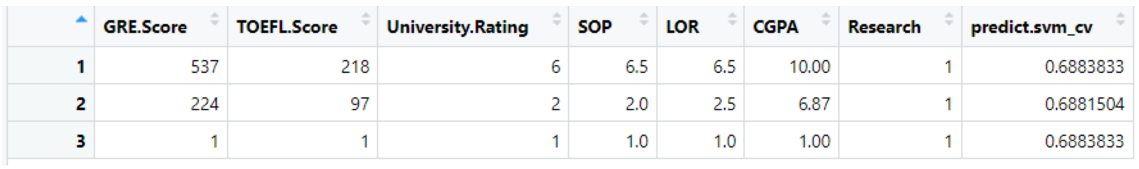
Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM – CV	C=1 Sigma=0.1721	0.83	0.05	0.91	0.05	0.03
SVM – Hold-out	C=0.5 Sigma=0.1721	0.82	0.05	0.90	0.05	0.03
RF – Hold-out	mtry=2	0.82	0.05	0.90	0.05	0.03
RF – CV*	mtry=2	0.82	0.05	0.90	0.05	0.03
RNA – CV*	size=41 decay=0.1	0.80	0.06	0.90	0.05	0.04
RNA – Hold-out	size=5 decay=0.1	0.78	0.06	0.88	0.06	0.04
KNN	K=9	0.77	0.06	0.88	0.06	0.04

* Os modelos utilizaram cross-validation e ajuste adequado de parâmetros via grid Search.

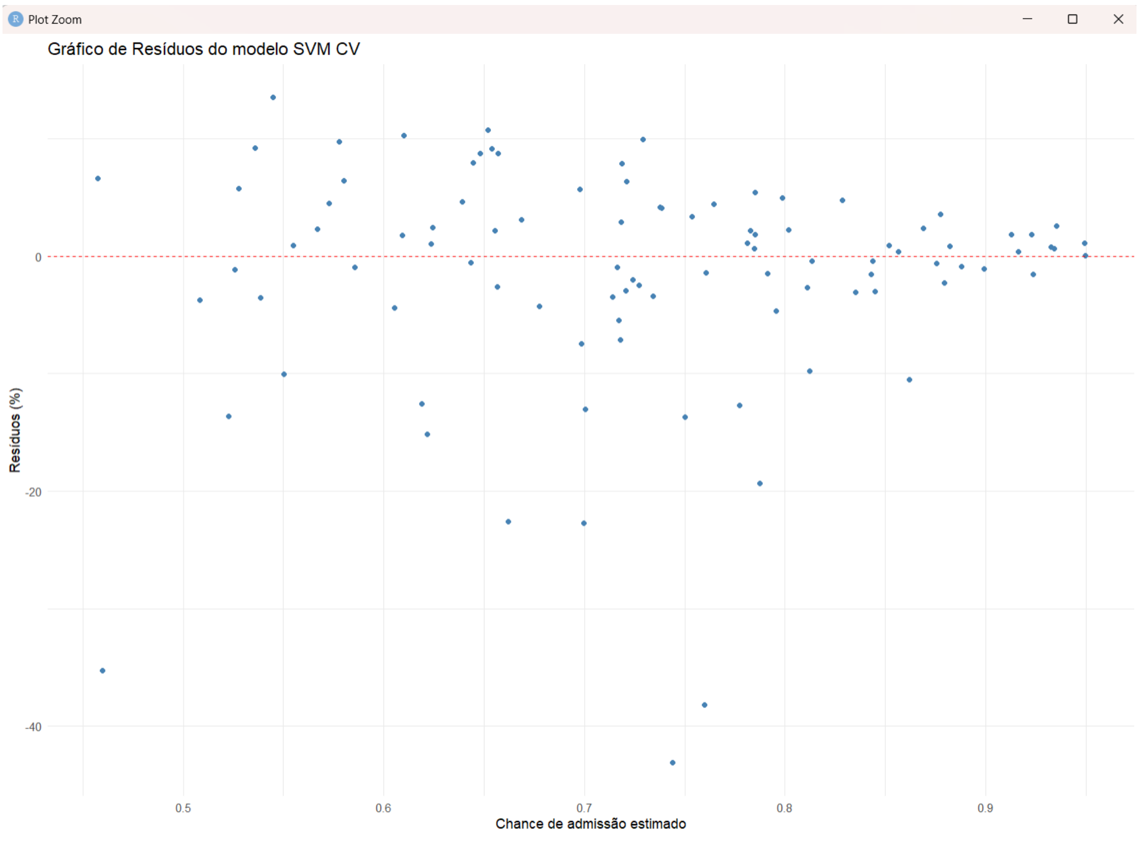
O modelo que obteve o melhor desempenho foi o SVM com cross-validation, alcançando um R^2 de 0.83 com os parâmetros custo(C) = 1 e Sigma=0.1721. Esse resultado demonstra que a combinação da técnica SVM com validação cruzada resultou em maior capacidade de generalização em relação aos demais modelos testados.

Com base no modelo selecionado, a imagem abaixo apresenta a predição de três novos casos realizada pelo algoritmo SVM.



	GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	predict.svm_cv
1	537	218	6	6.5	6.5	10.00	1	0.6883833
2	224	97	2	2.0	2.5	6.87	1	0.6881504
3	1	1	1	1.0	1.0	1.00	1	0.6883833

A figura abaixo apresenta os resíduos percentuais das previsões realizadas pelo modelo SVM com validação cruzada, aplicadas ao conjunto de teste. O resíduo percentual foi calculado como: ((observado - predito)/observado * 100). Esse gráfico permite avaliar a distribuição dos erros e identificar possíveis padrões ou desvios sistemáticos nas estimativas do modelo.



```
pacotes <- c("caret", "ggplot2")

# Instalando e carregando os pacotes necessarios
if(sum(as.numeric(!pacotes %in% installed.packages()))!=0){
  instalador <- pacotes[!pacotes %in% installed.packages()]
  for(i in 1:length(instalador)) {
    install.packages(instalador, dependencies = T)
    break()}
  sapply(pacotes, require, character = T)
} else {
  sapply(pacotes, require, character = T)
}

r2 <- function(predicted, real) {
  return ( 1 - (sum((predicted - real)^2) / sum((real -
mean(real))^2)))
}

syx <- function(predicted, real, p){
  n <- length(predicted)
  return (sqrt(sum((real - predicted) ^ 2) / (n - p)))
}
```

```
pearson <- function(predicted, real) {
  x_mean <- mean(real)
  y_mean <- mean(predicted)
  # cor(predicted, real, method = "pearson")
  return(sum((real - x_mean) * (predicted - y_mean)) /
(sqrt(sum((real - x_mean) ^ 2)) * sqrt(sum((predicted - y_mean) ^
2))))
}

print_model_stats <- function(name, df_test, target_var, model,
number_features, df_stats) {
  predicted <- predict(model, df_test)
  observed <- df_test[[target_var]]


  r2 <- r2(predicted, observed)
  syx <- syx(predicted, observed, number_features)
  pearson <- pearson(predicted, observed)
  rmse <- RMSE(predicted, observed)
  mae <- MAE(predicted, observed)

  cat("Estatísticas do modelo:", name, "\n")
  cat("R2      -->", r2, "\n")
  cat("Syx     -->", syx, "\n")
  cat("Pearson -->", pearson, "\n")
  cat("RMSE    -->", rmse, "\n")
  cat("MAE     -->", mae, "\n\n")

  new_row <- data.frame(
    Modelo = name,
    R2 = r2,
    Syx = syx,
    Pearson = pearson,
    RMSE = rmse,
    MAE = mae,
    stringsAsFactors = FALSE
  )

  return(rbind(df_stats, new_row))
}

# Configurando o seed
SEED <- 2038
set.seed(SEED)

setwd("C:/Users/rodri/machine-learning/UFPR-IAAP/IAA008 -
Aprendizado de máquina")

# Admissão
```

```
df_admissao <- read.csv("base/09 - Admissão/9 - Admissao -
Dados.csv")
df_admissao$num <- NULL
View(df_admissao)

target_var <- "ChanceOfAdmit"
number_features <- ncol(df_admissao) - 1

df_stats <- data.frame(
  Modelo = character(),
  R2 = numeric(),
  Syx = numeric(),
  Pearson = numeric(),
  RMSE = numeric(),
  MAE = numeric(),
  stringsAsFactors = FALSE
)

# Divisão da base de dados
set.seed(SEED)
indexes <- createDataPartition(df_admissao[[target_var]], p=0.80,
list=FALSE)
train <- df_admissao[indexes,]
test <- df_admissao[-indexes, ]

ctrl <- trainControl(method="cv", number=10)

# KNN
tuneGrid_knn <- expand.grid(k=c(1,3,5,7,9))
set.seed(SEED)
knn <- train(ChanceOfAdmit~., data=train, method = "knn", tuneGrid =
tuneGrid_knn)
knn
df_stats <- print_model_stats("KNN", test, target_var, knn,
number_features, df_stats)

# RNA Hold-out
set.seed(SEED)
rna <- train(ChanceOfAdmit~., data=train, method = "nnet", linout=T,
trace = FALSE)
rna
df_stats <- print_model_stats("RNA hold-out", test, target_var, rna,
number_features, df_stats)

# RNA grid search
grid_rna <- expand.grid(size = seq(from=1,to=45, by=10),
decay=seq(from=0.1,to=0.9,by=0.3))
set.seed(SEED)
rna_grid <- train(ChanceOfAdmit~., data=train, method = "nnet",
linout=T, trace = FALSE, trControl = ctrl, tuneGrid = grid_rna,
MaxNWts=10000, maxit=2000)
rna_grid
```

```
df_stats <- print_model_stats("RNA CV grid search", test,
target_var, rna_grid, number_features, df_stats)

# SVM
set.seed(SEED)
svm <- train(ChanceOfAdmit~., data = train, method = "svmRadial")
svm
df_stats <- print_model_stats("SVM hold-out", test, target_var, svm,
number_features, df_stats)

# SVM CV
set.seed(SEED)
svm_cv <- train(ChanceOfAdmit~., data = train, method = "svmRadial",
trControl = ctrl)
svm_cv
df_stats <- print_model_stats("SVM CV", test, target_var, svm_cv,
number_features, df_stats)

# SVM CV Grid
grid_cv <- expand.grid(C=c(1,2,10,50,100), sigma=c(.01,.015,.2))
set.seed(SEED)
svm_cv_grid <- train(ChanceOfAdmit~., data = train, method =
"svmRadial", trControl = ctrl, tuneGrid=grid_cv)
svm_cv_grid
df_stats <- print_model_stats("SVM CV grid search", test,
target_var, svm_cv_grid, number_features, df_stats)

# Randon forest
set.seed(SEED)
rf <- train(ChanceOfAdmit~., data = train, method="rf")
rf
df_stats <- print_model_stats("RF hold-out", test, target_var, rf,
number_features, df_stats)

# RF CV
set.seed(SEED)
rf_cv <- train(ChanceOfAdmit~., data = train, method="rf", trControl
= ctrl)
rf_cv
df_stats <- print_model_stats("RF CV", test, target_var, rf_cv,
number_features, df_stats)

# RF CV grid search
grid_rf = expand.grid(mtry=c(2,5,7,9))
set.seed(SEED)
rf_cv_grid <- train(ChanceOfAdmit~., data = train, method="rf",
trControl = ctrl, tuneGrid = grid_rf)
rf_cv_grid
df_stats <- print_model_stats("RF CV grid search", test, target_var,
rf_cv_grid, number_features, df_stats)

df_stats <- df_stats[order(-df_stats$R2), ]
```

```
View(df_stats)

### Novas predições
new_data <- read.csv("base/09 - Admissão/9 - Admissao - Dados -
Novos Casos.csv")
View(new_data)

predict.svm_cv <- predict(svm_cv, new_data)
new_data$ChanceOfAdmit <- NULL
result <- cbind(new_data, predict.svm_cv)
View(result)


##### Gráfico de resíduos do melhor modelo
svm.pred <- predict(svm_cv, test)
obs <- test$ChanceOfAdmit

df_residual <- data.frame(
  Predito = svm.pred,
  Observado = obs,
  Resíduo = (((obs - svm.pred) / obs) * 100)
)

ggplot(df_residual, aes(x = Predito, y = Resíduo)) +
  geom_point(color = "steelblue") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Gráfico de Resíduos do modelo SVM CV",
       x = "Chance de admissão estimado",
       y = "Resíduos (%)") +
  theme_minimal()
```

Biomassa

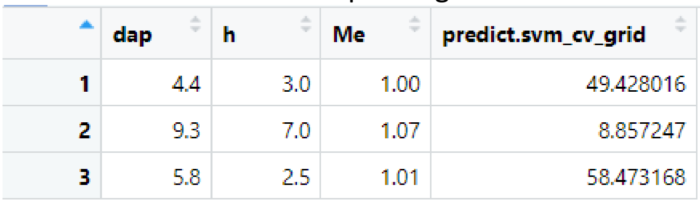
Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM – CV *	C=50 Sigma=0.01	0.85	964.18	0.98	939.76	185.02
RNA – Hold-out	size=5 decay=0.1	0.75	1228.99	0.98	1197.87	265.80
KNN	K=3	0.66	1446.95	0.95	1410.31	239.84
RF – Hold-out	mtry=2	0.66	1453.65	0.96	1416.85	246.46
RF – CV	mtry=2	0.66	1458.59	0.96	1421.66	249.81
RNA – CV *	size=11 decay=0.4	0.51	1741.69	0.93	1697.59	309.09

SVM – Hold-out	C=1 Sigma=1.14	0.11	2361.98	0.44	2302.17	418.74

* Os modelos utilizaram cross-validation e ajuste adequado de parâmetros via grid Search.

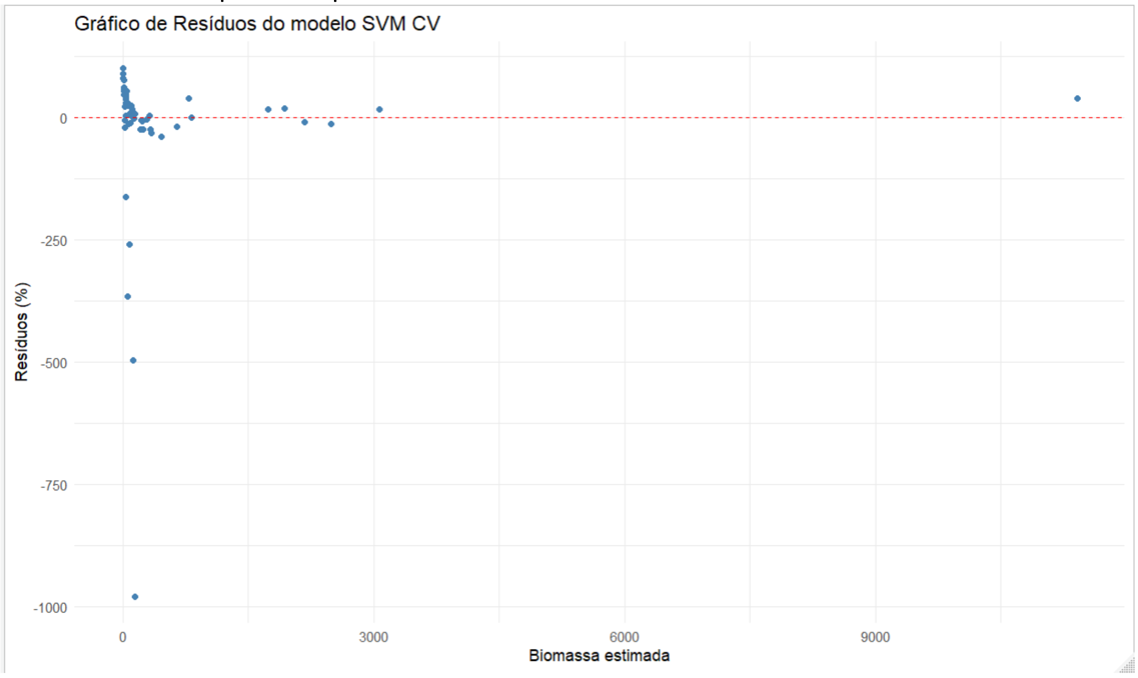
O modelo que obteve o melhor desempenho foi o SVM com cross-validation e busca dos melhores parâmetros com grid search, alcançando um R^2 de 0.85 com os parâmetros custo(C) = 50 e Sigma=0.01. Esse resultado demonstra que a combinação da técnica SVM com validação cruzada e grid Search resultou em maior capacidade de generalização em relação aos demais modelos testados.

Com base no modelo selecionado, a imagem abaixo apresenta a predição de três novos casos realizada pelo algoritmo SVM - CV.



	dap	h	Me	predict.svm_cv_grid
1	4.4	3.0	1.00	49.428016
2	9.3	7.0	1.07	8.857247
3	5.8	2.5	1.01	58.473168

A figura abaixo apresenta os resíduos percentuais das previsões realizadas pelo modelo SVM com validação cruzada, aplicadas ao conjunto de teste. O resíduo percentual foi calculado como: ((observado - predito)/observado * 100). Esse gráfico permite avaliar a distribuição dos erros e identificar possíveis padrões ou desvios sistemáticos nas estimativas do modelo.



```
pacotes <- c("caret", "ggplot2")

# Instalando e carregando os pacotes necessarios
if(sum(as.numeric(!pacotes %in% installed.packages()))!=0){
  instalador <- pacotes[!pacotes %in% installed.packages()]
  for(i in 1:length(instalador)) {
    install.packages(instalador, dependencies = T)
    break()}
  sapply(pacotes, require, character = T)
} else {
  sapply(pacotes, require, character = T)
}

r2 <- function(predicted, real) {
  return ( 1 - (sum((predicted - real)^2) / sum((real -
mean(real))^2)))
}

syx <- function(predicted, real, p){
  n <- length(predicted)
  return (sqrt(sum((real - predicted) ^ 2) / (n - p)))
}

pearson <- function(predicted, real) {
  x_mean <- mean(real)
  y_mean <- mean(predicted)
  # cor(predicted, real, method = "pearson")
  return(sum((real - x_mean) * (predicted - y_mean)) /
(sqrt(sum((real - x_mean) ^ 2)) * sqrt(sum((predicted - y_mean) ^
2))))
}

print_model_stats <- function(name, df_test, target_var, model,
number_features, df_stats) {
  predicted <- predict(model, df_test)
  observed <- df_test[[target_var]]


  r2 <- r2(predicted, observed)
  syx <- syx(predicted, observed, number_features)
  pearson <- pearson(predicted, observed)
  rmse <- RMSE(predicted, observed)
  mae <- MAE(predicted, observed)

  cat("Estatísticas do modelo:", name, "\n")
  cat("R2      -->", r2, "\n")
  cat("Syx     -->", syx, "\n")
  cat("Pearson -->", pearson, "\n")
  cat("RMSE    -->", rmse, "\n")
  cat("MAE     -->", mae, "\n\n")

  new_row <- data.frame(
    Modelo = name,
    R2 = r2,
    Syx = syx,
    Pearson = pearson,
    RMSE = rmse,
    MAE = mae,
    stringsAsFactors = FALSE
  )

  return(rbind(df_stats, new_row))
}

# Configurando o seed
SEED <- 2038
set.seed(SEED)

setwd("C:/Users/rodri/machine-learning/UFPR-IAAP/IAA008 -
Aprendizado de máquina")

# Biomassa

df <- read.csv("base/05 - Biomassa/5 - Biomassa - Dados.csv")
View(df)
df$num <- NULL
View(df)

target_var <- "biomassa"
number_features <- ncol(df) - 1

df_stats <- data.frame(
  Modelo = character(),
  R2 = numeric(),
  Syx = numeric(),
  Pearson = numeric(),
  RMSE = numeric(),
  MAE = numeric(),
  stringsAsFactors = FALSE
)

# Divisão da base de dados
set.seed(SEED)
indexes <- createDataPartition(df[[target_var]], p=0.80, list=FALSE)
df_train <- df[indexes,]
df_test <- df[-indexes, ]

ctrl <- trainControl(method="cv", number=10)

# KNN
tuneGrid_knn <- expand.grid(k=c(1,3,5,7,9))
set.seed(SEED)
knn <- train(biomassa~., data=df_train, method = "knn", tuneGrid =
tuneGrid_knn)
```

```
knn
df_stats <- print_model_stats("KNN", df_test, target_var, knn,
number_features, df_stats)

# RNA Hold-out
set.seed(SEED)
rna <- train(biomassa~., data=df_train, method = "nnet", linout=T,
trace = FALSE)
rna
df_stats <- print_model_stats("RNA hold-out", df_test, target_var,
rna, number_features, df_stats)

# RNA grid search
grid_rna <- expand.grid(size = seq(from=1,to=45, by=10),
decay=seq(from=0.1,to=0.9,by=0.3))
set.seed(SEED)
rna_grid <- train(biomassa~., data=df_train, method = "nnet",
linout=T, trace = FALSE, trControl = ctrl, tuneGrid = grid_rna,
MaxNWts=10000, maxit=2000)
rna_grid
df_stats <- print_model_stats("RNA CV grid search", df_test,
target_var, rna_grid, number_features, df_stats)

# SVM
set.seed(SEED)
svm <- train(biomassa~., data = df_train, method = "svmRadial")
svm
df_stats <- print_model_stats("SVM hold-out", df_test, target_var,
svm, number_features, df_stats)

# SVM CV
set.seed(SEED)
svm_cv <- train(biomassa~., data = df_train, method = "svmRadial",
trControl = ctrl)
svm_cv
df_stats <- print_model_stats("SVM CV", df_test, target_var, svm_cv,
number_features, df_stats)

# SVM CV Grid
grid_cv <- expand.grid(C=c(1,2,10,50,100), sigma=c(.01,.015,.2))
set.seed(SEED)
svm_cv_grid <- train(biomassa~., data = df_train, method =
"svmRadial", trControl = ctrl, tuneGrid=grid_cv)
svm_cv_grid
df_stats <- print_model_stats("SVM CV grid search", df_test,
target_var, svm_cv_grid, number_features, df_stats)

# Randon forest
set.seed(SEED)
rf <- train(biomassa~., data = df_train, method="rf")
rf
```

```
df_stats <- print_model_stats("RF hold-out", df_test, target_var,
rf, number_features, df_stats)

# RF CV
set.seed(SEED)
rf_cv <- train(biomassa~., data = df_train, method="rf", trControl =
ctrl)
rf_cv
df_stats <- print_model_stats("RF CV", df_test, target_var, rf_cv,
number_features, df_stats)

# RF CV grid search
grid_rf = expand.grid(mtry=c(2,5,7,9))
set.seed(SEED)
rf_cv_grid <- train(biomassa~., data = df_train, method="rf",
trControl = ctrl, tuneGrid = grid_rf)
rf_cv_grid
df_stats <- print_model_stats("RF CV grid search", df_test,
target_var, rf_cv_grid, number_features, df_stats)

df_stats <- df_stats[order(-df_stats$R2), ]

View(df_stats)

### Novas predições
new_data <- read.csv("base/05 - Biomassa/5 - Biomassa - Dados -
Novos Casos.csv")
View(new_data)

predict.svm_cv_grid <- predict(svm_cv_grid, new_data)
new_data$biomassa <- NULL
result <- cbind(new_data, predict.svm_cv_grid)
View(result)


##### Gráfico de resíduos do melhor modelo
svm.pred <- predict(svm_cv_grid, df_test)
obs <- df_test$biomassa

df_residual <- data.frame(
  Predito = svm.pred,
  Observado = obs,
  Resíduo = (((obs - svm.pred) / obs) * 100)
)

ggplot(df_residual, aes(x = Predito, y = Resíduo)) +
  geom_point(color = "steelblue") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Gráfico de Resíduos do modelo SVM CV",
       x = "Biomassa estimada",
       y = "Resíduos (%)") +
  theme_minimal()
```

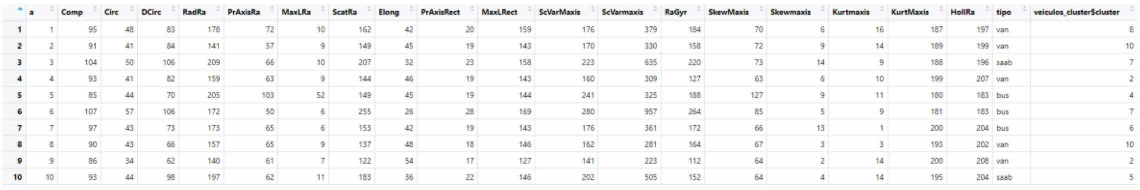
AGRUPAMENTO

Veículo

Lista de Clusters gerados:

10 primeiras linhas do arquivo com o cluster correspondente.
Usa 10 clusters no experimento.
Colocar a lista de comandos emitidos no RStudio para conseguir os resultados obtidos

A figura abaixo apresenta as dez primeiras linhas do resultado da associação. Para tal, foi utilizado o algoritmo *k*-means com o parâmetro de 10 clusters. Antes da aplicação do algoritmo, os dados foram previamente normalizados.



	a	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo	veiculos_cluster$cluster
1	1	95	48	83	178	72	10	162	42	20	159	176	379	184	70	6	16	187	197	van	8
2	2	91	41	84	141	57	9	149	45	19	143	170	330	158	72	9	14	189	199	van	10
3	3	104	50	106	209	66	10	207	32	23	158	223	635	220	73	14	9	188	196	saab	7
4	4	93	41	82	159	63	9	144	46	19	143	160	309	127	63	6	10	199	207	van	2
5	5	85	44	70	205	103	52	149	45	19	144	241	325	188	127	9	11	180	183	bus	4
6	6	107	57	106	172	50	6	255	26	28	169	280	957	264	85	5	9	181	183	bus	7
7	7	97	43	73	173	65	6	153	42	19	143	176	361	172	66	13	1	200	204	bus	6
8	8	90	43	66	157	65	9	137	48	18	146	162	281	164	67	3	3	193	202	van	10
9	9	86	34	62	140	61	7	122	54	17	127	141	223	112	64	2	14	200	208	van	2
10	10	93	44	98	197	62	11	183	36	22	146	202	505	152	64	4	14	195	204	saab	5

```
# Configurando o seed
SEED <- 2038
set.seed(SEED)

setwd("C:/Users/rodri/machine-learning/UFPR-IAAP/IAA008 -
Aprendizado de máquina")

# Veículos

df_veiculos <- read.csv("base/06 - Veículos/6 - Veiculos -
Dados.csv")
View(df_veiculos)

# Ignora a coluna de id e coluna de classes
df_veiculos_clean <- df_veiculos[, -c(1, ncol(df_veiculos))]
df_veiculos_clean = scale(df_veiculos_clean)
View(df_veiculos_clean)

veiculos_cluster <- kmeans(df_veiculos_clean, 10)
veiculos_cluster

table(veiculos_cluster$cluster, df_veiculos$tipo)

resultado <- cbind(df_veiculos, veiculos_cluster$cluster)
```

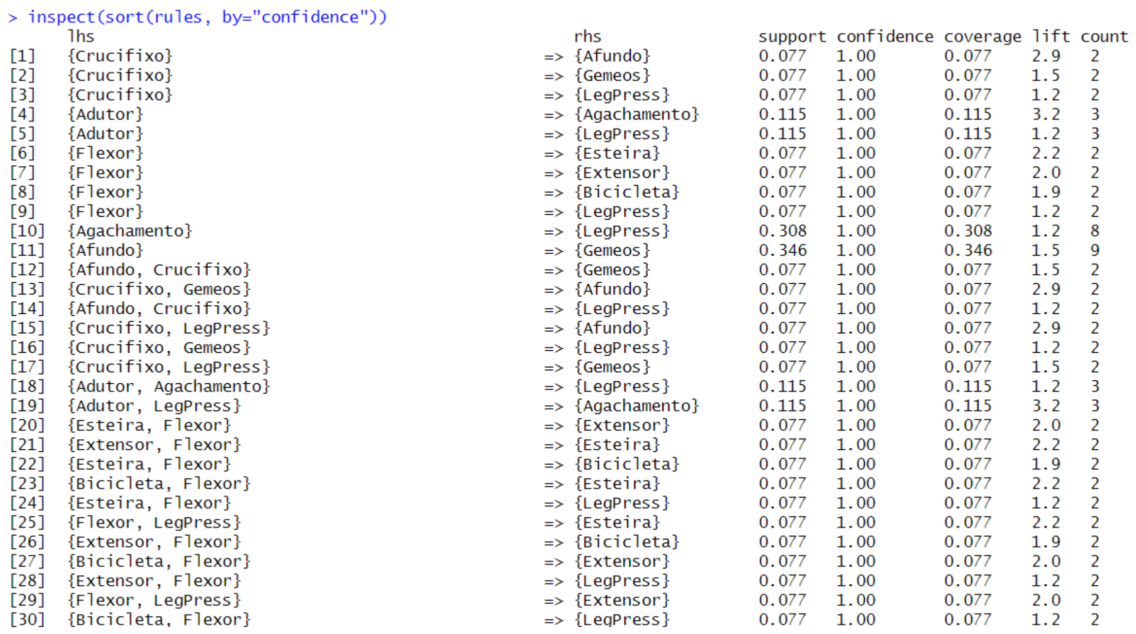
```
View(resultado)
```

REGRAS DE ASSOCIAÇÃO

Musculação

Regras geradas com uma configuração de Suporte e Confiança.
Colocar a lista de comandos emitidos no RStudio para conseguir os resultados obtidos

A figura abaixo apresenta as 30 primeiras regras geradas pela execução do algoritmo Apriori na base de dados de musculação. O algoritmo foi executado com suporte mínimo de 0,001 e confiança mínima de 0,7.



```
> inspect(sort(rules, by="confidence"))
     lhs                                          rhs             support confidence coverage lift count
[1]  {Crucifixo}                              => {Afundo}         0.077   1.00       0.077    2.9  2
[2]  {Crucifixo}                              => {Gemeos}         0.077   1.00       0.077    1.5  2
[3]  {Crucifixo}                              => {LegPress}       0.077   1.00       0.077    1.2  2
[4]  {Adutor}                                 => {Agachamento}    0.115   1.00       0.115    3.2  3
[5]  {Adutor}                                 => {LegPress}       0.115   1.00       0.115    1.2  3
[6]  {Flexor}                                 => {Esteira}        0.077   1.00       0.077    2.2  2
[7]  {Flexor}                                 => {Extensor}       0.077   1.00       0.077    2.0  2
[8]  {Flexor}                                 => {Bicicleta}      0.077   1.00       0.077    1.9  2
[9]  {Flexor}                                 => {LegPress}       0.077   1.00       0.077    1.2  2
[10] {Agachamento}                            => {LegPress}       0.308   1.00       0.308    1.2  8
[11] {Afundo}                                 => {Gemeos}         0.346   1.00       0.346    1.5  9
[12] {Afundo, Crucifixo}                      => {Gemeos}         0.077   1.00       0.077    1.5  2
[13] {Crucifixo, Gemeos}                      => {Afundo}         0.077   1.00       0.077    2.9  2
[14] {Afundo, Crucifixo}                      => {LegPress}       0.077   1.00       0.077    1.2  2
[15] {Crucifixo, LegPress}                    => {Afundo}         0.077   1.00       0.077    2.9  2
[16] {Crucifixo, Gemeos}                      => {LegPress}       0.077   1.00       0.077    1.2  2
[17] {Crucifixo, LegPress}                    => {Gemeos}         0.077   1.00       0.077    1.5  2
[18] {Adutor, Agachamento}                    => {LegPress}       0.115   1.00       0.115    1.2  3
[19] {Adutor, LegPress}                       => {Agachamento}    0.115   1.00       0.115    3.2  3
[20] {Esteira, Flexor}                        => {Extensor}       0.077   1.00       0.077    2.0  2
[21] {Extensor, Flexor}                       => {Esteira}        0.077   1.00       0.077    2.2  2
[22] {Esteira, Flexor}                        => {Bicicleta}      0.077   1.00       0.077    1.9  2
[23] {Bicicleta, Flexor}                      => {Esteira}        0.077   1.00       0.077    2.2  2
[24] {Esteira, Flexor}                        => {LegPress}       0.077   1.00       0.077    1.2  2
[25] {Flexor, LegPress}                       => {Esteira}        0.077   1.00       0.077    2.2  2
[26] {Extensor, Flexor}                       => {Bicicleta}      0.077   1.00       0.077    1.9  2
[27] {Bicicleta, Flexor}                      => {Extensor}       0.077   1.00       0.077    2.0  2
[28] {Extensor, Flexor}                       => {LegPress}       0.077   1.00       0.077    1.2  2
[29] {Flexor, LegPress}                       => {Extensor}       0.077   1.00       0.077    2.0  2
[30] {Bicicleta, Flexor}                      => {LegPress}       0.077   1.00       0.077    1.2  2
```

```
#install.packages('arules', dep=T)
library(arules)

# Configurando o seed
SEED <- 2038
set.seed(SEED)

setwd("C:/Users/rodri/machine-learning/UFPR-IAAP/IAA008 -
Aprendizado de máquina")
```

```
atividades <- read.transactions(
  file = "base/12 - Regras de Associacao - Praticas/12 - Regras de
Associacao - Praticas – 2 - Musculacao/2 - Musculacao - Dados.csv",
  format = "basket",
  sep=";"
  )

inspect(head(atividades, 3))
itemFrequencyPlot(atividades, topN = 10, type='absolute')

summary(atividades)

set.seed(SEED)
rules <- apriori(atividades, parameter = list(supp = 0.001, conf =
0.7, minlen=2))
summary(rules)

options(digits=2)
inspect(sort(rules, by="confidence"))
```