**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:

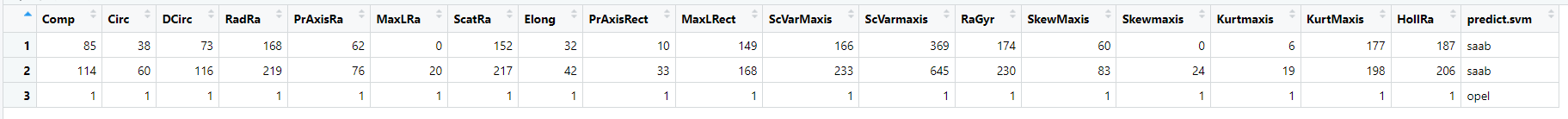
* Ordenar pela Acurácia (descendente), ou seja, a técnica de melhor acurácia ficará em primeiro na tabela.
* Após o quadro colocar:
  + Um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior Acurácia (criar um arquivo com novos casos à sua escolha)
  + A lista de comandos 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 |  |
| RNA – CV | size=21 decay=0.7 | 0.72 |  |
| RF – Hold-out | mtry=10 | 0.71 |  |
| KNN | k=1 | 0.62 |  |
| RNA – Hold-out | size=5 decay=0.1 | 0.50 |  |

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.

****

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

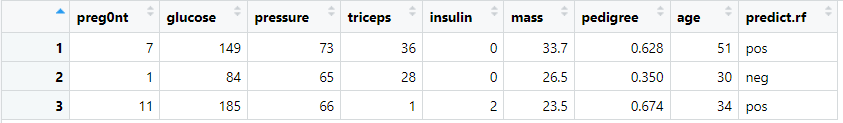
|  |
| --- |
| pacotes <- c("caret","e1071", "mlbench", "mice")  # Instalando e carregando os pacotes necessarios  if(sum(as.numeric(!pacotes %in% installed.packages()))!=0){  instalador <- pacotes[!pacotes %in% installed.packages()]  for(i in 1:length(instalador)) {  install.packages(instalador, dependencies = 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 |  |
| SVM – CV | C=0.25 Sigma=0.128 | 0.81 |  |
| RNA – CV | size=3 decay=0.1 | 0.79 |  |
| RF – Hold-out | mtry=2 | 0.79 |  |
| KNN | k=9 | 0.76 |  |
| RNA – Hold-out | size=3 decay=0.1 | 0.68 |  |

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.

****

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

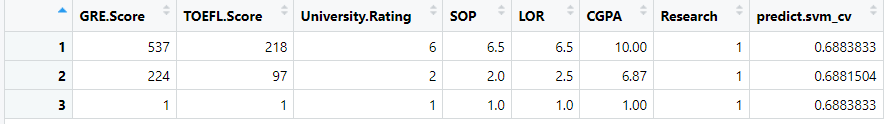
**Admissão**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 |

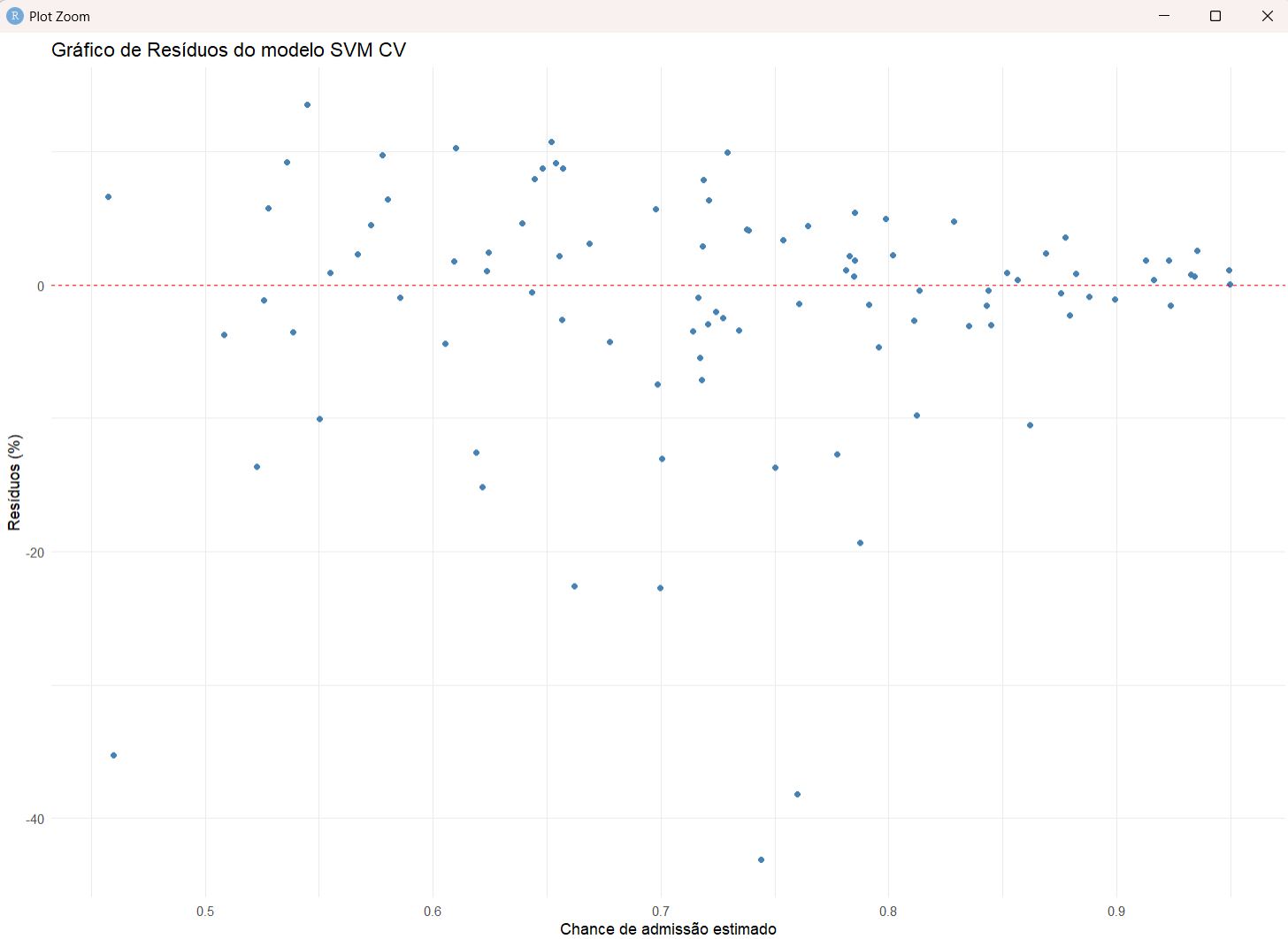
\* Os modelos utilizaram cross-validation e ajuste adequado de parâmetros via grid Search.

O modelo que obteve o melhor desempenho foi o SVM com cross-validation, alcançando um R2 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.



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 |

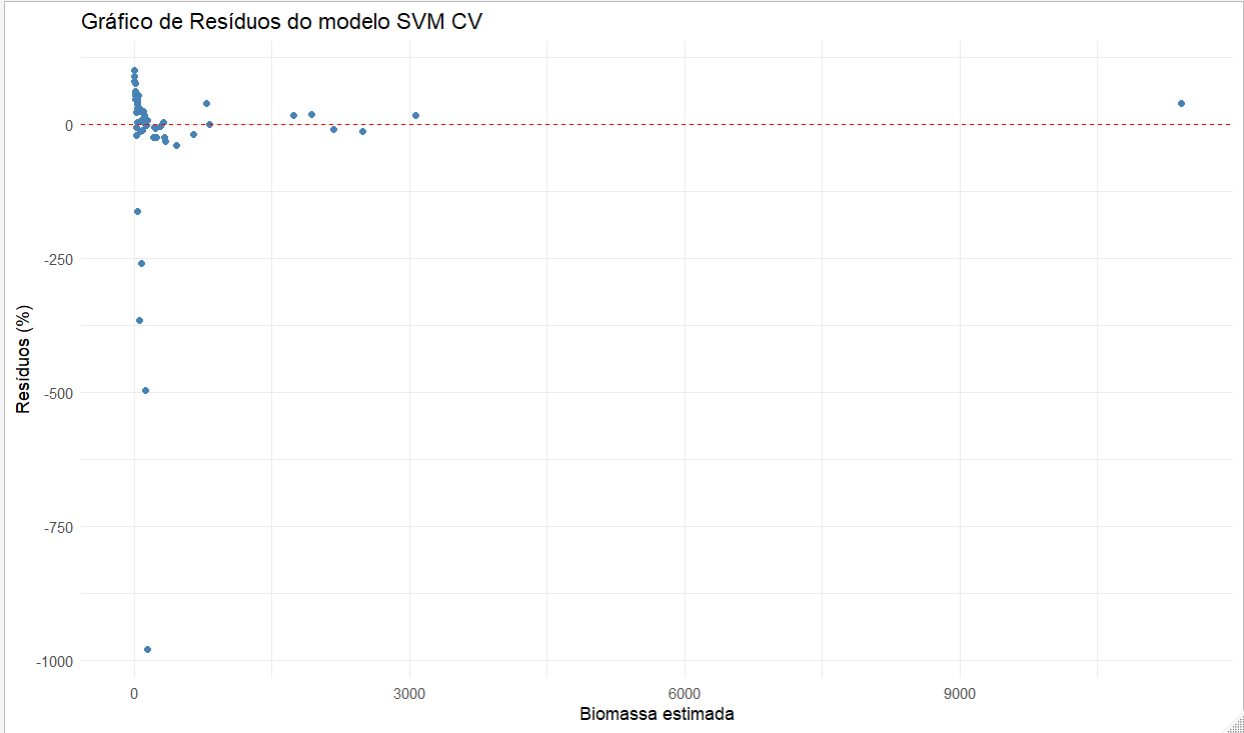
\* Os modelos utilizaram cross-validation e ajuste adequado de parâmetros via grid Search.

O modelo que obteve o melhor desempenho foi o SVM com cross-validation e busca dos melhores parâmetros com grid search, alcançando um R2 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.

****

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**

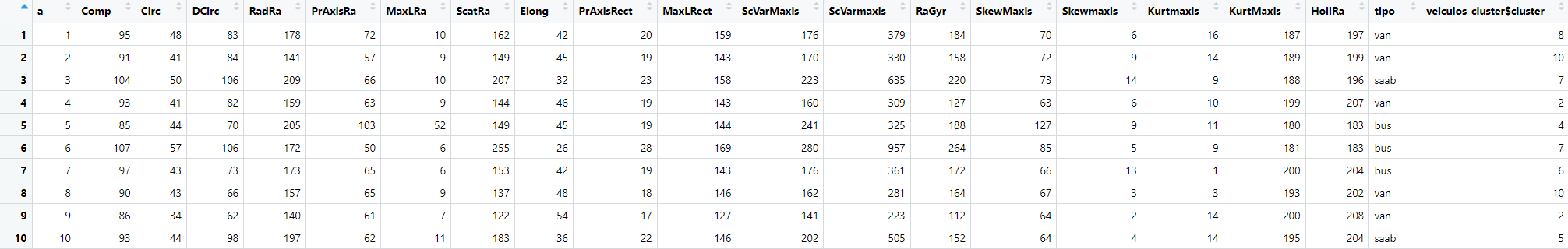
Lista de Clusters gerados:

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.



|  |
| --- |
| # 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) |

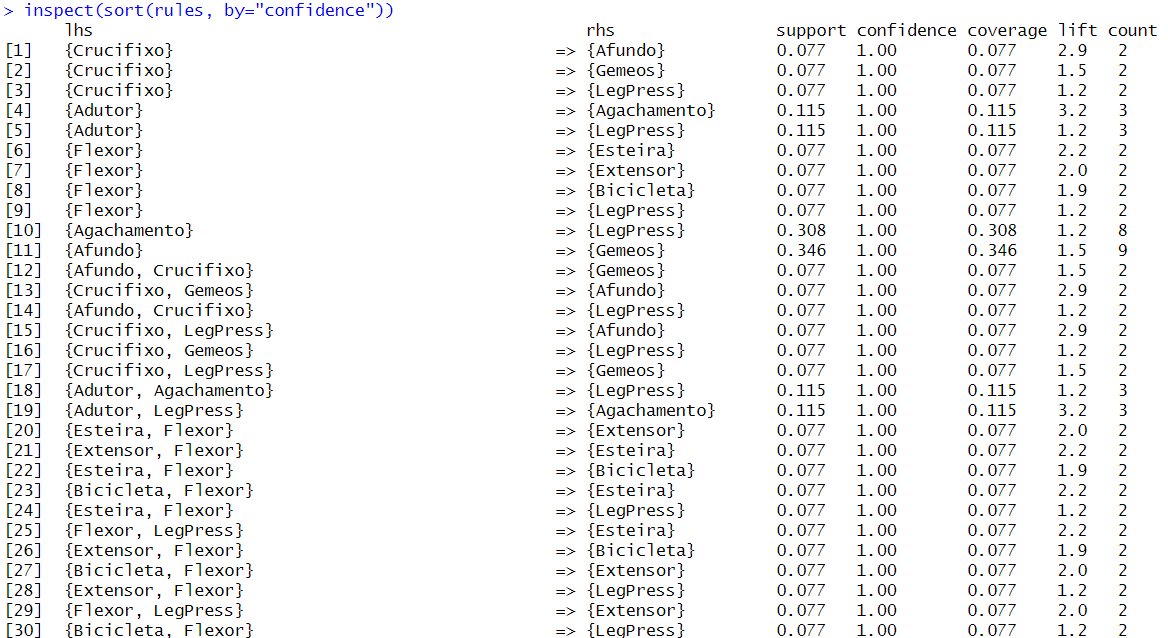
**REGRAS DE ASSOCIAÇÃO**

**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.



|  |
| --- |
| #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")) |