

# Script do Projeto Aprendizagem de Maquina

bibliotecas nescessarias

Hide

```
library(tidyverse)
library(ggplot2)
library(e1071)
library(stringr)
library(rpart)
library(rpart.plot)
library(randomForest)
library(caret)
library(nnet)
library(partykit)
library(plyr)
library(corrplot)
```

## Carregando o banco de dados tratamentos dos dados

Hide

```
#Chamando banco de dados
mush <- read.table(file.choose(), header = T, sep = ",")
# Arquivo mushrooms em csv

#Renomeando cabeçalho
colnames(mush) <- c("classificacao", "formato_chapeu", "superficie_chapeu", "cor_chapeu", "mancha", "odor",
"anexo_branquial", "espacamento_branquial", "tamanho_branquial",
"cor_branquial", "formato_caule", "raiz_caule", "superficie_caule_acima_anel", "superficie_caule_abaixo_anel",
"cor_caule_acima_anel", "cor_caule_abaixo_anel", "tipo_veu", "cor_veu", "numero_aneis", "tipo_aneis", "cor esporos", "populacao", "habitat")

View(mush)
```

O conjunto de dados contem variaveis categóricas, vamos organizar o banco de dados em variaveis factor e vamos renomar os leves(as classes) dos fatores. Note que como a variavel 17 ("tipo\_veu") so tem uma classe vamos tirar ela do conjunto de dados pois não é significativa para o estudo.

Hide

```

#Transformando todas variáveis em fator
mush <- mush %>% map_df(function(.x) as.factor(.x))

#Renomeando os levels
levels(mush$classificacao) <- c("comestivel", "venenoso")
levels(mush$formato_chapeu) <- c("sino", "conico", "plano", "botão", "baixo_relevo", "convexo")
levels(mush$superficie_chapeu) <- c("fibroso", "rachurado", "escamoso", "liso")
levels(mush$cor_chapeu) <- c("pele", "canela", "vermelho", "cinza", "marrom", "rosa", "verde",
"roxo", "branco", "amarelo")
levels(mush$mancha) <- c("nao", "sim")
levels(mush$odor) <- c("amendoa", "creosotose", "falta", "anis", "mofado", "nenhum", "pungente",
"picante", "duvidoso")
levels(mush$anexo_branquial) <- c("anexado", "livre")
levels(mush$espacamento_branquial) <- c("fechado", "coroa")
levels(mush$tamanho_branquial) <- c("ampla", "limitado")
levels(mush$cor_branquial) <- c("pele", "vermelho", "cinza", "chocolate", "preto", "marrom", "laranja", "rosa", "verde", "roxo", "branco", "amarelo")
levels(mush$formato_caule) <- c("ampliado", "afunilado")
levels(mush$raiz_caule) <- c("faltante", "bulbosa", "clube", "igual", "enraizado")
levels(mush$superficie_caule_acima_anel) <- c("fibroso", "sedosa", "liso", "escamosa")
levels(mush$superficie_caule_abaixo_anel) <- c("fibrosa", "sedosa", "lisa", "escamosa")
levels(mush$cor_caule_acima_anel) <- c("pele", "canela", "vermelho", "cinza", "marrom", "rosa", "verde", "roxo", "branco", "amarelo")
levels(mush$cor_caule_abaixo_anel) <- c("pele", "canela", "vermelho", "cinza", "marrom", "rosa", "verde", "roxo", "branco", "amarelo")

levels(mush$tipo_veu) <- c("parcial") ###
levels(mush$cor_veu) <- c("marrom", "laranja", "branco", "amarelo")
levels(mush$numero_aneis) <- c("nenhum", "um", "dois")
levels(mush$tipo_aneis) <- c("evanescente", "deslumbrante", "largo", "nenhum", "penso")
levels(mush$cor_esporos) <- c("pele", "chocolate", "preto", "marrom", "laranja", "verde", "roxo", "branco", "amarelo")
levels(mush$populacao) <- c("abundante", "agrupado", "numeroso", "espalhada", "varias", "solitaria")
levels(mush$habitat) <- c("madeira", "grama", "folhas", "campo", "caminhos", "urbano", "desperdicio")

#Retirando a variável "tipo_veu"
mush <- mush[,-17]

```

Hide

```

#Vusualização dos dados
glimpse(mush)

```

Rows: 8,124

Columns: 22

```
$ classificacao      [3m][38;5;246m<fct>[39m[23m venenoso, comestivel, comest...
$ formato_chapeu    [3m][38;5;246m<fct>[39m[23m convexo, convexo, sino, conv...
$ superficie_chapeu [3m][38;5;246m<fct>[39m[23m escamoso, escamoso, escamoso...
$ cor_chapeu        [3m][38;5;246m<fct>[39m[23m marrom, amarelo, branco, bra...
$ mancha            [3m][38;5;246m<fct>[39m[23m sim, sim, sim, sim, nao, sim...
$ odor              [3m][38;5;246m<fct>[39m[23m pungente, amendoa, anis, pun...
$ anexo_branquial   [3m][38;5;246m<fct>[39m[23m livre, livre, livre, livre, ...
$ espacamento_branquial [3m][38;5;246m<fct>[39m[23m fechado, fechado, fechado, f...
$ tamanho_branquial [3m][38;5;246m<fct>[39m[23m limitado, ampla, ampla, limi...
$ cor_branquial     [3m][38;5;246m<fct>[39m[23m preto, preto, marrom, marrom...
$ formato_caule     [3m][38;5;246m<fct>[39m[23m ampliado, ampliado, ampliado...
$ raiz_caule        [3m][38;5;246m<fct>[39m[23m igual, clube, clube, igual, ...
$ superficie_caule_acima_anel [3m][38;5;246m<fct>[39m[23m liso, liso, liso, liso, liso...
$ superficie_caule_abaixo_anel [3m][38;5;246m<fct>[39m[23m lisa, lisa, lisa, lisa, lisa...
$ cor_caule_acima_anel [3m][38;5;246m<fct>[39m[23m roxo, roxo, roxo, roxo, roxo...
$ cor_caule_abaixo_anel [3m][38;5;246m<fct>[39m[23m roxo, roxo, roxo, roxo, roxo...
$ cor_veu           [3m][38;5;246m<fct>[39m[23m branco, branco, branco, bran...
$ numero_aneis      [3m][38;5;246m<fct>[39m[23m um, um, um, um, um, um, um, ...
$ tipo_aneis        [3m][38;5;246m<fct>[39m[23m penso, penso, penso, penso, ...
$ cor_esporos       [3m][38;5;246m<fct>[39m[23m preto, marrom, marrom, preto...
$ populacao         [3m][38;5;246m<fct>[39m[23m espalhada, numeroso, numeros...
$ habitat           [3m][38;5;246m<fct>[39m[23m urbano, grama, campo, urbano...
```

Hide

```
# head(mush)
```

Com isso vemos que as variaveis do banco esta como factor e todas as classes estão renomeadas. Assim o banco esta pronto para as analise seguintes.

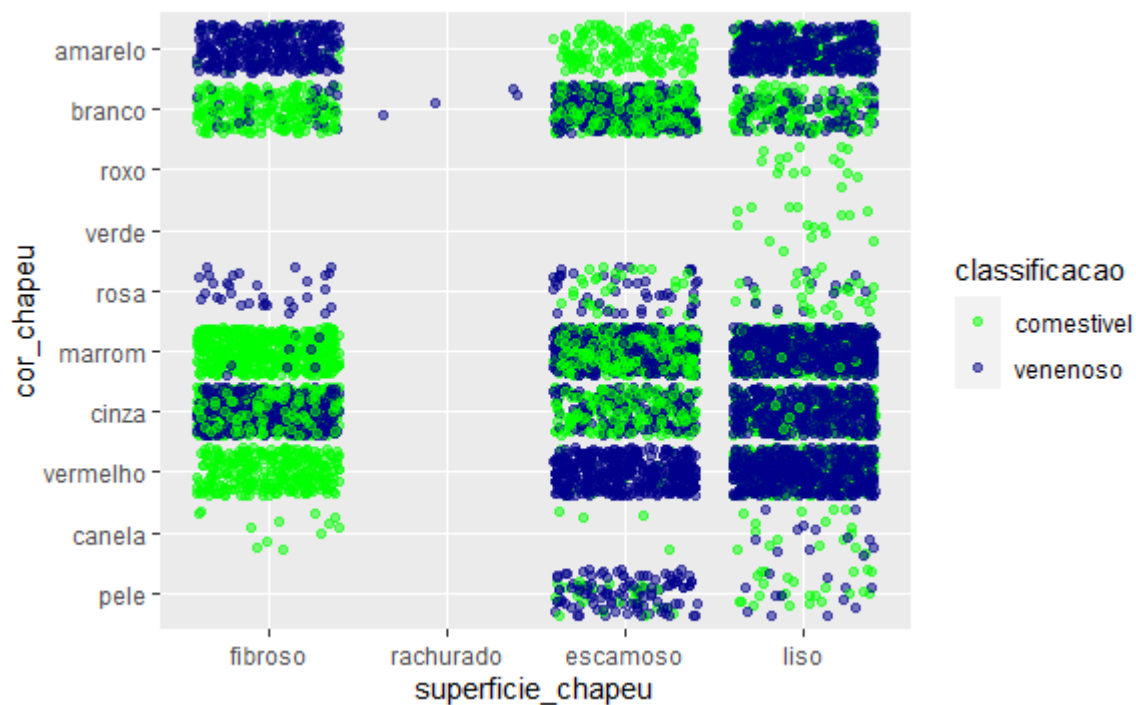
Hide

```
attach(mush)
```

## Analise descirtiva

Hide

```
#Relação entre superficie e cor chapeu para discriminar a classificacao
ggplot(mush, aes(x = superficie_chapeu , y = cor_chapeu, col = classificacao)) + geom_jitter(alp
ha = 0.5) + scale_color_manual(breaks = c("comestivel", "venenoso"), values = c("green", "darkbl
ue"))
```



A superfície fibrosa nos mostrou mais observacoes classificadas como comestivel, enquanto que a lisa mostra o oposto, salvo o caso de quando a cor do chapéu e verde ou roxa. A superfície escamosa com a cor do chapéu amarela também nos mostra total segurança para consumo, enquanto que com o chapéu vermelho totalmente venenoso.

Hide

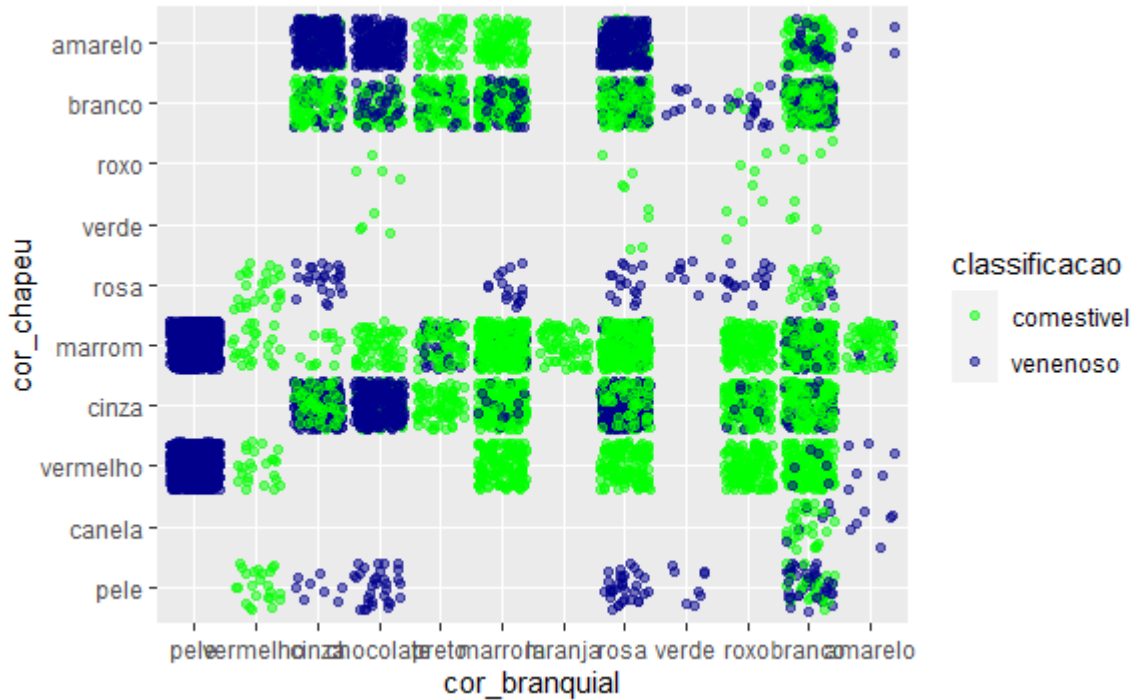
```
ggplot(mush, aes(x = formato_chapeu, y = cor_chapeu, col = classificacao)) + geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("comestivel", "venenoso"),
    values = c("green", "darkblue"))
```



O formato de sino se mostra mais seguro para consumo em relacao aos demais formatos, com excessão do no formato de baixo relevo sinaliza da mesma forma, contudo temos poucas observacoes do mesmo.

Hide

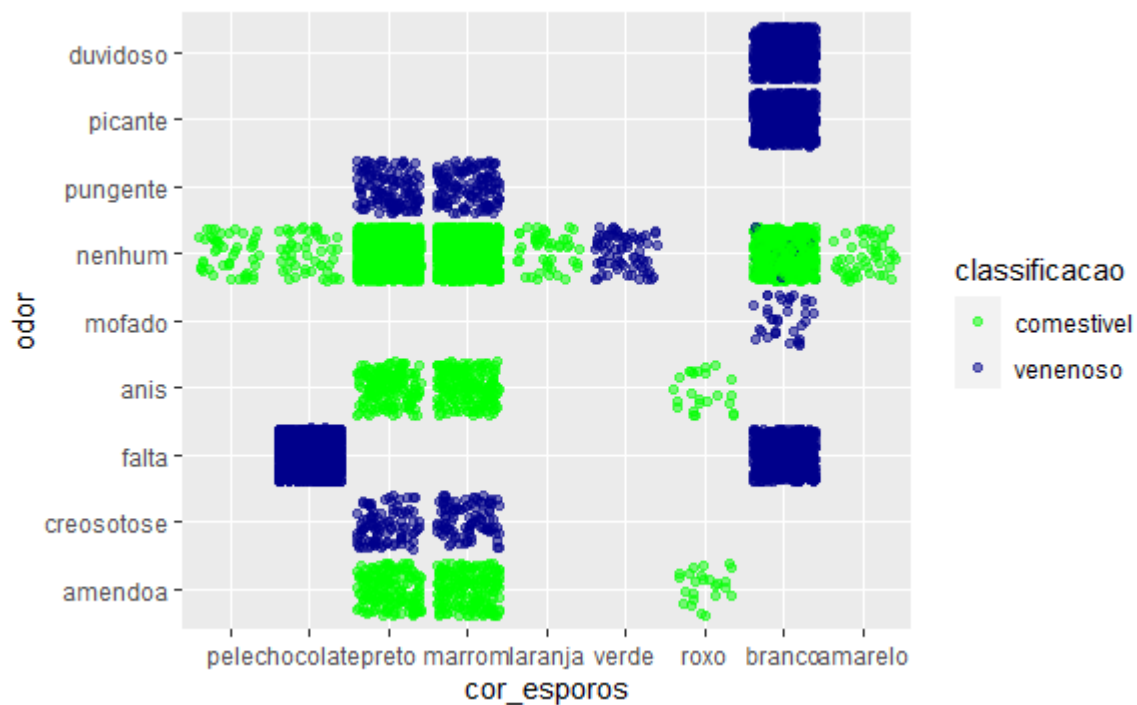
```
ggplot(mush, aes(x = cor_branquial, y = cor_chapeu, col = classificacao)) +  
  geom_jitter(alpha = 0.5) +  
  scale_color_manual(breaks = c("comestivel", "venenoso"),  
                    values = c("green", "darkblue"))
```



Procurar se ajusta angulo do xlab Vemos que para um cogumelo com cor do chapéu vermelho ele é comestível quando a cor do branquio é vermelho, marrom, rosa e roxo. Ou quando a cor do branquio é vermelho essa conclusão se dá para quando a cor do chapéu é cor de pele, vermelha, marrom, e rosa.. Quando a cor branquial for cor de pele temos que o cogumelo é venenoso.

Hide

```
ggplot(mush, aes(x = cor_esporos, y = odor, col = classificacao)) +  
  geom_jitter(alpha = 0.5) +  
  scale_color_manual(breaks = c("comestivel", "venenoso"),  
                    values = c("green", "darkblue"))
```



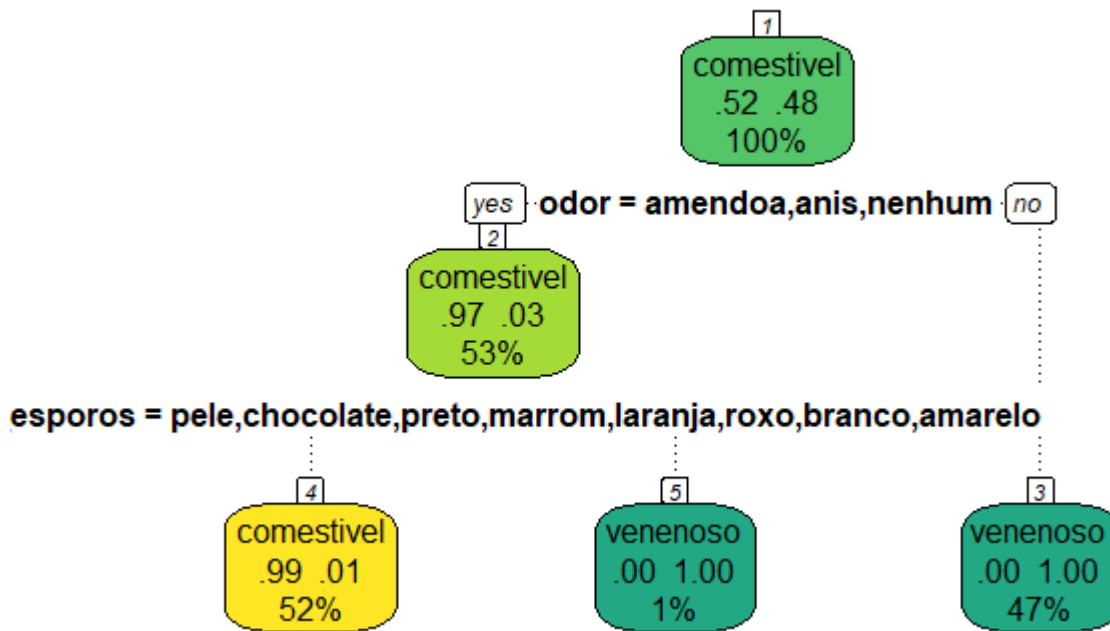
O odor é definitivamente um preditor informativo. Basicamente, se cheira é duvidoso, picante ou pungente, fique longe. Se cheira a anis ou amêndoa, você pode ir em frente. Se não cheira nada, você tem mais chances de ser comestível do que não. E quando a cor do esporos é preto, marrom ou branco e o odor é duvidoso, picante ou pungente terá chances de ser venenoso, e para a cor dos esporos cor de pele, chocolate, preto, marrom ou laranja e o odor é nenhum ou anis o cogumelo será comestível.

## Analise modelo, Redução de dimensionalidade e Correlação

Hide

```
set.seed(2020)
#view(mush)
arvore_mod<-rpart(classificacao ~ ., data = mush)
#_____

#Faz o plot da arvore de decisão
rpart.plot(arvore_mod , extra = 104, box.palette = "YlGnBl",
           branch.lty = 3, nn = TRUE)
```



A proposta de construção desta árvore advém de um particionamento do banco de dados. Isto é, e utilizado 80% dos dados como um conjunto de treino, e 20% dos dados como forma de teste, afim de verificar a assertividade do modelo.

Hide

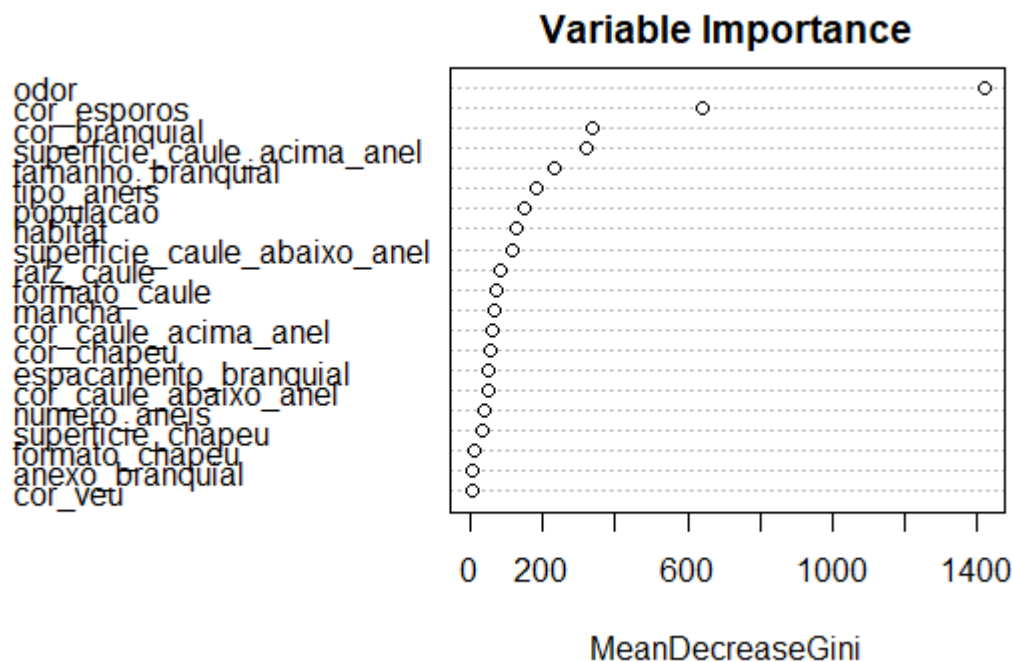
```

# Outra árvore de decisão
# a <- ctree(classificacao ~ ., mush)
# plot(a)

rf = randomForest(classificacao ~ .,
                  ntree = 100,
                  data = mush)

#plot(rf)
varImpPlot(rf,
           sort = T,
           main = "Variable Importance")

```



Hide

```
rf$importance[rf$importance>=145,]
```

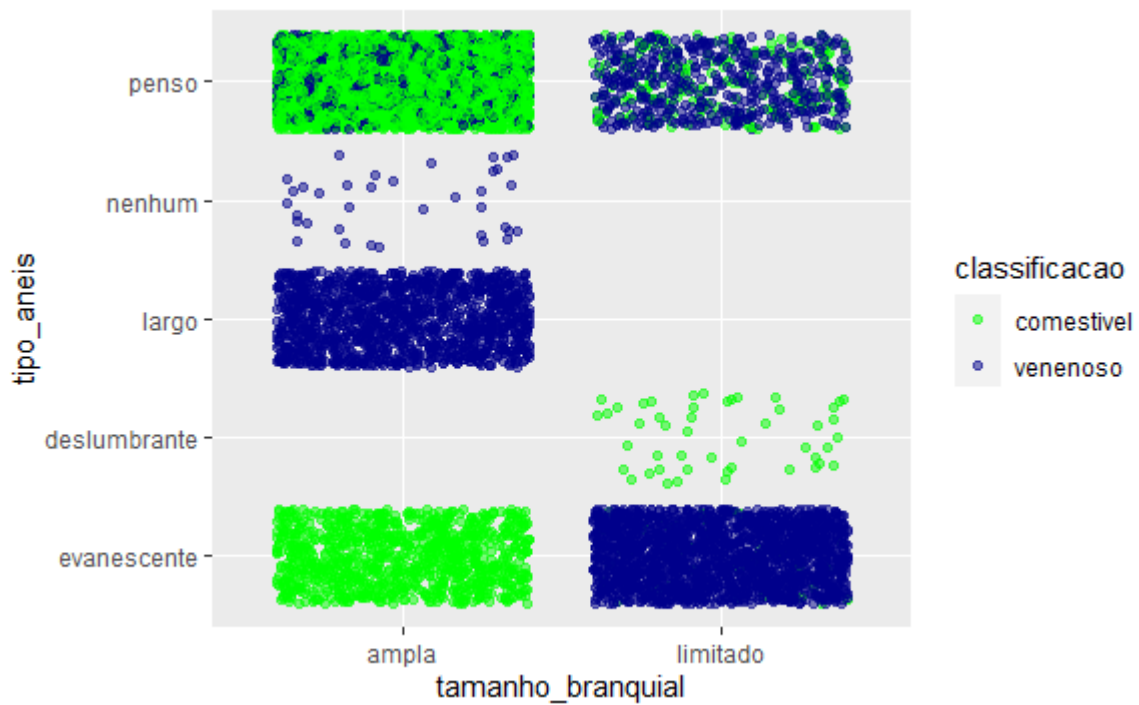
odor	tamanho_branquial
1421.3041	232.6567
cor_branquial	superficie_caule_acima_anel
336.6634	320.0237
tipo_aneis	cor_esporos
181.9546	639.3001
populacao	
147.9658	

Obtemos as importancias das variaveis utilizando o random floreste e escolhemos o corte de 145, onde as variaveis com importancia maior que 15 serão consideradas nos estudos dos modelos adiante, assim conseguindo fazer uma redução de dimensionalidade onde o banco com 21 variáveis exceto a variavel resposta, vemos que 7 variaveis tem grau de importancia alto. Note que no plote das variáveis “odor” e “cor\_esporos” vemos que a separação de cogumelos comestível e venenoso fica nitida e vemos que essas duas variaveis são as que apresenta maior grau de importância, note também que essas duas variáveis são as que ficaram na arvore de decissão. Note também que para o plote com a variável “formato\_chapeu” tem grau de importancia muito e notamos que não fica nitido a separação de cogumelos comestível e venenoso.

Hide

```
ggplot(mush, aes(x = tamanho_branquial, y = tipo_aneis, col = classificacao)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("comestivel", "venenoso"),
    values = c("green", "darkblue"))
```





Note que após apresentar o grau de importância das variáveis, fizemos o plot do “tipo\_anel” com o “tamanho\_branquial” e coloramos com base nos cogumelos comestíveis e venenosos. E vemos que essas duas variáveis são bem nítidas a separação de comestível e venenoso, exceto para a classe “penso” no tipo\_anel.

Para analisarmos as correlações das variáveis vamos criar um subbanco com as variáveis alvo e as variáveis com grau de importância alto, depois como o banco está no formato de factor vamos transformar o banco em numérico para poder calcular as correlações

Hide

```
# vamos pegar as variaveis com grau de importancia alto
df<-subset(mush, select = c(1,6,9,10,13,19,20,21))

# Transformando o banco em numerico
df$classificacao<-as.integer(mapvalues(df$classificacao,c("comestivel", "venenoso"),1:2))
df$odor<-as.integer(mapvalues(df$odor, c("amendoa", "creosotose", "falta", "anis", "mofado", "ne
nhum", "pungente", "picante", "duvidoso"),1:9))
df$tamanho_branquial<-as.integer(mapvalues(df$tamanho_branquial, c("ampla", "limitado"),1:2))
df$cor_branquial<-as.integer(mapvalues(df$cor_branquial, c("pele", "vermelho", "cinza", "chocola
te", "preto", "marrom", "laranja", "rosa", "verde", "roxo", "branco", "amarelo"),1:12))
df$superficie_caule_acima_anel<-as.integer(mapvalues(df$superficie_caule_acima_anel, c("fibroso"
, "sedosa", "liso", "escamosa"),1:4))
df$tipo_aneis<-as.integer(mapvalues(df$tipo_aneis, c("evanescente", "deslumbrante", "largo", "ne
nhum", "penso"),1:5))
df$cor_esporos<-as.integer(mapvalues(df$cor_esporos, c("pele", "chocolate", "preto", "marrom",
"laranja", "verde", "roxo", "branco", "amarelo"),1:9))
df$populacao<-as.integer(mapvalues(df$populacao, c("abundante", "agrupado", "numeroso", "espalha
da", "varias", "solitaria"),1:6))

# Vamos renomar as variaveis para ficar mais agradavel a saida
#tamanho_branquial = , superficie_caule_acima_anel = superf_c_a_anel
colnames(df)<-c("classificacao", "odor","tam_branquial",
               "cor_branquial", "superf_c_a_anel","tipo_aneis","cor_esporos", "populacao")

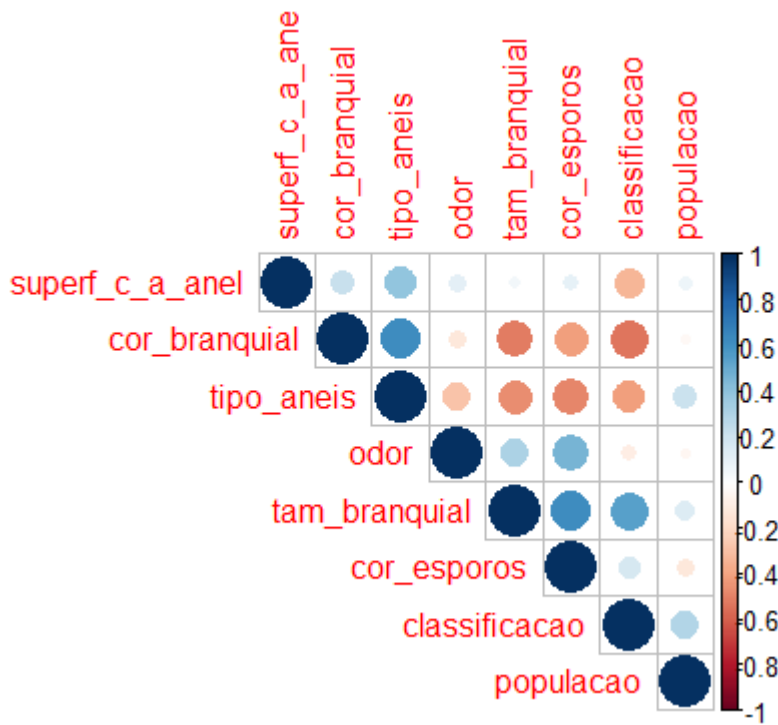
glimpse(df)
```

```
Rows: 8,124
Columns: 8
$ classificacao  [3m][38;5;246m<int>[39m[23m 2, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2,...
$ odor          [3m][38;5;246m<int>[39m[23m 7, 1, 4, 7, 6, 1, 1, 4, 7, 1, 4, 1, 1, 7,...
$ tam_branquial [3m][38;5;246m<int>[39m[23m 2, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2,...
$ cor_branquial [3m][38;5;246m<int>[39m[23m 5, 5, 6, 6, 5, 6, 3, 6, 8, 3, 3, 6, 11, 5...
$ superf_c_a_anel [3m][38;5;246m<int>[39m[23m 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,...
$ tipo_aneis    [3m][38;5;246m<int>[39m[23m 5, 5, 5, 5, 1, 5, 5, 5, 5, 5, 5, 5, 5,...
$ cor_esporos   [3m][38;5;246m<int>[39m[23m 3, 4, 4, 3, 4, 3, 3, 4, 3, 3, 4, 3, 4, 4,...
$ populacao     [3m][38;5;246m<int>[39m[23m 4, 3, 3, 4, 1, 3, 3, 4, 5, 4, 3, 4, 4, 5,...
```

Hide

```
# Calculando as correlações
M<-cor(df)

corrplot(M,type = "upper", order = "hclust", sig.level = 0.01, insig = "blank")
```



## Validação Cruzada

A seguir vamos fazer a análise com o banco de dados, fazendo validação cruzada usando k-fold igual a 10 e cv igual a 10, note que a variável resposta é categórica e os bancos de treinamento e de teste que obtemos na validação cruzada precisam ter a mesma proporção, onde no banco original a variável “classificação” apresenta 0.52 comestível e 0.48 venenoso e os bancos de teste e treinamento a variável classificação precisam ter essas mesmas proporções.

Como estamos lidando com um problema de classificação, na validação cruzada calculamos a matriz de confusão e obtemos os seguintes indicadores Sensitivity, Specificity, Prevalence, PPV(Pos Pred Value), NPV( Neg Pred Value), Detection Rate, Detection Prevalence, Balanced Accuracy, Precision, Recall e F1.

Diante de outros métodos de modelagem, a redução de dimensionalidade não foi satisfatória, como no caso do knn, onde este precisou de 19 das 22 variáveis, para sua construção. #####

Hide

```

### Vamos fazer a validação cruzada para as variáveis com grau de importancia alto

### Organizando o banco
mush.pronto<-subset(mush, select = c(1,3,4,5,6,8,9,10,11,12,13,14,15,16,18,19,20,21,22))

#view(mush.pronto)
# Vamos colocar 90% para o banco de treinamento e 10% para o banco de teste
K=10
cv = 10

R.modelo.arvore.k<-matrix(NA, nrow = K,ncol = 11)
R.modelo.random.k<-matrix(NA, nrow = K,ncol = 11)
R.modelo.Knn.k<-matrix(NA, nrow = K,ncol = 11)
R.modelo.SVM.k<-matrix(NA, nrow = K,ncol = 11)
R.modelo.pred.logit_mod.k<-matrix(NA, nrow = K,ncol = 11)
R.modelo.percep.k<-matrix(NA, nrow = K,ncol = 11)

set.seed(2020)
##### Interacoes K-Fold #####
for(t in 1:K)
{

  R.modelo.arvore.cv<-matrix(NA, nrow = cv,ncol = 11)
  R.modelo.random.cv<-matrix(NA, nrow = cv,ncol = 11)
  R.modelo.Knn.cv<-matrix(NA, nrow = cv,ncol = 11)
  R.modelo.SVM.cv<-matrix(NA, nrow = cv,ncol = 11)
  R.modelo.pred.logit_mod.cv<-matrix(NA, nrow = cv,ncol = 11)
  R.modelo.percep.cv<-matrix(NA, nrow = cv,ncol = 11)

  #group <- sample(group)
  amostra<-createDataPartition(y = mush.pronto$classificacao, times = 10, p = 0.9, list = FALSE)
  # Esta função é muito boa pois quando formos para a tarefa de classificação
  # ele mantém a proporção das classes que é comestível 0.52 , venenoso 0.48

  ##### Interacoes CrossValidation #####

  for(i in 1:cv){

    #####
    round(prop.table(table(mush.pronto$classificacao)), 2)
    #Variaveis temporarias Treinamento
    mush.Trein<-mush.pronto[amostra[,i],]
    print(round(prop.table(table(mush.Trein$classificacao)), 2))

    #Variaveis temporarias Treinamento
    mush.Test<-mush.pronto[-amostra[,i],]
    print(round(prop.table(table(mush.Trein$classificacao)), 2))

    ##### MODELOS LOGISTICO Multinomial #####
    fit.logit_mod <- multinom(classificacao ~ ., data = mush.Trein)
    pred.logit_mod<-predict(fit.logit_mod, newdata = mush.Test[, -1], type = "class")
  }
}

```

```
R.modelo.pred.logit_mod.cv[i,]<-confusionMatrix(data=pred.logit_mod, reference = mush.Test$classificacao, positive="comestivel")$byClass
```

```
##### MODELOS Arvore #####
```

```
fit.arvore<-rpart(classificacao ~ . , data = mush.Trein)
```

```
pred.arvore<-predict(fit.arvore, newdata = mush.Test[,-1], type = "class")
```

```
#Vamos fazer a matriz de confusão
```

```
R.modelo.arvore.cv[i,]<-confusionMatrix(data=pred.arvore, reference = mush.Test$classificacao, positive="comestivel")$byClass
```

```
##### MODELOS Random florest #####
```

```
fit.random<-randomForest(classificacao ~ . , data = mush.Trein)
```

```
pred.random<-predict(fit.random, newdata = mush.Test[,-1], type = "class")
```

```
#Vamos fazer a matriz de confusão
```

```
R.modelo.random.cv[i,]<-confusionMatrix(data=pred.random, reference = mush.Test$classificacao, positive="comestivel")$byClass
```

```
##### MODELOS Knn #####
```

```
fit.knn<-knn3(classificacao ~ . , data = mush.Trein,k=6)
```

```
pred.knn<-predict(fit.knn, newdata = mush.Test[,-1], type = "class")
```

```
#Vamos fazer a matriz de confusão
```

```
R.modelo.Knn.cv[i,]<-confusionMatrix(data=pred.knn, reference = mush.Test$classificacao, positive="comestivel")$byClass
```

```
##### MODELOS SVM #####
```

```
fit.svm<-svm(classificacao ~ . , data = mush.Trein, type='C-classification', kernel='radial')
```

```
pred.svm<-predict(fit.svm, newdata = mush.Test[,-1], type = "class")
```

```
#Vamos fazer a matriz de confusão
```

```
R.modelo.SVM.cv[i,]<-confusionMatrix(data=pred.svm, reference = mush.Test$classificacao, positive="comestivel")$byClass
```

```
##### MODELO Perceptron #####
```

```
fit.percep<-nnet(classificacao ~ . , data = mush.Trein,size = 1)
```

```
pred.percep<-predict(fit.percep, newdata = mush.Test[,-1], type = "class")
```

```
pred.percep<-as.factor(pred.percep)
```

```
#Vamos fazer a matriz de confusão
```

```
R.modelo.percep.cv[i,]<-confusionMatrix(data=pred.percep, reference = mush.Test$classificacao, positive="comestivel")$byClass
```

```
}
```

```
# Salvando os resultados do for anterior
```

```
R.modelo.arvore.k[t,]<-colMeans(R.modelo.arvore.cv)
```

```
R.modelo.random.k[t,]<-colMeans(R.modelo.random.cv)
```

```
R.modelo.Knn.k[t,]<-colMeans(R.modelo.Knn.cv)
```

```
R.modelo.SVM.k[t,]<-colMeans(R.modelo.SVM.cv)
```

```
R.modelo.pred.logit_mod.k[t,]<-colMeans(R.modelo.pred.logit_mod.cv)
```

```
R.modelo.percep.k[t,]<-colMeans(R.modelo.percep.cv)
```

```

}

R.Arvore<-colMeans(R.modelo.arvore.k)
R.Random<-colMeans(R.modelo.random.k)
R.Knn<-colMeans(R.modelo.Knn.k)
R.SVM<-colMeans(R.modelo.SVM.k)
R.Logis.Mod<-colMeans(R.modelo.pred.logit_mod.k)
R.perceptron<-colMeans(R.modelo.percep.k)

Result.Matriz<-matrix(NA,nrow = 11,ncol = 6)
Result.Matriz[,1]<-R.Arvore
Result.Matriz[,2]<-R.Random
Result.Matriz[,3]<-R.SVM
Result.Matriz[,4]<-R.Knn
Result.Matriz[,5]<-R.Logis.Mod
Result.Matriz[,6]<-R.perceptron
row.names(Result.Matriz)<-c("Sensitivity","Specificity","Pos Pred Value","Neg Pred Value",
                           "Precision","Recall","F1","Prevalence","Detection Rate",
                           "Detection Prevalence","Balanced Accuracy")
colnames(Result.Matriz)<-c("Arvore", "Random", "SVM", "Knn", "Logist.Mult", "Perceptron")

```

Hide

Result.Matriz

	Arvore	Random	SVM	Knn
Sensitivity	1.0000000	1.0000000	1.0000000	1.0000000
Specificity	0.9878005	1.0000000	0.9959591	0.9997442
Pos Pred Value	0.9887892	1.0000000	0.9962592	0.9997631
Neg Pred Value	1.0000000	1.0000000	1.0000000	1.0000000
Precision	0.9887892	1.0000000	0.9962592	0.9997631
Recall	1.0000000	1.0000000	1.0000000	1.0000000
F1	0.9943583	1.0000000	0.9981243	0.9998813
Prevalence	0.5178792	0.5178792	0.5178792	0.5178792
Detection Rate	0.5178792	0.5178792	0.5178792	0.5178792
Detection Prevalence	0.5237608	0.5178792	0.5198274	0.5180025
Balanced Accuracy	0.9939003	1.0000000	0.9979795	0.9998721
	Logist.Mult	Perceptron		
Sensitivity	1.0000000	0.9949524		
Specificity	1.0000000	0.9495396		
Pos Pred Value	1.0000000	0.9590368		
Neg Pred Value	1.0000000	0.9947277		
Precision	1.0000000	0.9590368		
Recall	1.0000000	0.9949524		
F1	1.0000000	0.9755881		
Prevalence	0.5178792	0.5178792		
Detection Rate	0.5178792	0.5152651		
Detection Prevalence	0.5178792	0.5395931		
Balanced Accuracy	1.0000000	0.9722460		

Apois rodar a validação cruzada para os modelos “Arvore de decisão”, “Random florest”, “SVM”, “Knn”, “Logistica Multinomial” e “Perceptron” vemos a saída com os indicadores falados anteriormente e vemos que o modelo “Random florest” e “Logistica Multinomial” apresenta a accuracy igual a 1. No geral todos os modelos apresentam resultados muito bons, mais vemos que os dois melhores modelos foram “Random florest” e “Logistica Multinomial”.

Fazendo a validação cruzada para as 8 variáveis que continuaram após a redução de dimensionalidade. Como inicialmente o modelo Knn não rodou para esses dados com a redução nessa validação retiramos o modelo knn.

Hide

```

### Vamos fazer a validação cruzada para as variáveis com grau de importancia alto

### Organizando o banco
mush.pronto<-subset(mush, select = c(1,6,9,10,13,19,20,21))

# Vamos colocar 90% para o banco de treinamento e 10% para o banco de teste
K=10
cv = 10

R.modelo.arvore.k<-matrix(NA, nrow = K,ncol = 11)
R.modelo.random.k<-matrix(NA, nrow = K,ncol = 11)
#R.modelo.Knn.k<-matrix(NA, nrow = K,ncol = 11)
R.modelo.SVM.k<-matrix(NA, nrow = K,ncol = 11)
R.modelo.pred.logit_mod.k<-matrix(NA, nrow = K,ncol = 11)
R.modelo.percep.k<-matrix(NA, nrow = K,ncol = 11)

set.seed(2020)
##### Interacoes K-Fold #####
for(t in 1:K)
{

  R.modelo.arvore.cv<-matrix(NA, nrow = cv,ncol = 11)
  R.modelo.random.cv<-matrix(NA, nrow = cv,ncol = 11)
  #R.modelo.Knn.cv<-matrix(NA, nrow = cv,ncol = 11)
  R.modelo.SVM.cv<-matrix(NA, nrow = cv,ncol = 11)
  R.modelo.pred.logit_mod.cv<-matrix(NA, nrow = cv,ncol = 11)
  R.modelo.percep.cv<-matrix(NA, nrow = cv,ncol = 11)

  #group <- sample(group)
  amostra<-createDataPartition(y = mush.pronto$classificacao, times = 10, p = 0.9, list = FALSE)
  # Esta função é muito boa pois quando formos para a tarefa de classificação
  # ele mantém a proporção das classes que é comestível 0.52 , venenoso 0.48

  ##### Interacoes CrossValidation #####

  for(i in 1:cv){

    #####
    round(prop.table(table(mush.pronto$classificacao)), 2)
    #Variaveis temporarias Treinamento
    mush.Trein<-mush.pronto[amostra[,i],]
    print(round(prop.table(table(mush.Trein$classificacao)), 2))

    #Variaveis temporarias Treinamento
    mush.Test<-mush.pronto[-amostra[,i],]
    print(round(prop.table(table(mush.Trein$classificacao)), 2))

    ##### MODELOS LOGISTICO Multinomial #####
    fit.logit_mod <- multinom(classificacao ~ ., data = mush.Trein)
    pred.logit_mod<-predict(fit.logit_mod, newdata = mush.Test[, -1], type = "class")
  }
}

```



```

R.modelo.pred.logit_mod.cv[i,]<-confusionMatrix(data=pred.logit_mod, reference = mush.Test$classificacao, positive="comestivel")$byClass

##### MODELOS Arvore #####
fit.arvore<-rpart(classificacao ~ . , data = mush.Trein)
pred.arvore<-predict(fit.arvore, newdata = mush.Test[, -1], type = "class")
#Vamos fazer a matriz de confusão
R.modelo.arvore.cv[i,]<-confusionMatrix(data=pred.arvore, reference = mush.Test$classificacao, positive="comestivel")$byClass

##### MODELOS Random florest #####

fit.random<-randomForest(classificacao ~ . , data = mush.Trein)
pred.random<-predict(fit.random, newdata = mush.Test[, -1], type = "class")
#Vamos fazer a matriz de confusão
R.modelo.random.cv[i,]<-confusionMatrix(data=pred.random, reference = mush.Test$classificacao, positive="comestivel")$byClass

##### MODELOS Knn #####
#fit.knn<-knn3(classificacao ~ . , data = mush.Trein,k=6)
#pred.knn<-predict(fit.knn, newdata = mush.Test[, -1], type = "class")
#Vamos fazer a matriz de confusão
#R.modelo.Knn.cv[i,]<-confusionMatrix(data=pred.knn, reference = mush.Test$classificacao, positive="comestivel")$byClass

##### MODELOS SVM #####
fit.svm<-svm(classificacao ~ . , data = mush.Trein, type='C-classification', kernel='radial')
pred.svm<-predict(fit.svm, newdata = mush.Test[, -1], type = "class")
#Vamos fazer a matriz de confusão
R.modelo.SVM.cv[i,]<-confusionMatrix(data=pred.svm, reference = mush.Test$classificacao, positive="comestivel")$byClass

##### MODELO Perceptron #####
fit.percep<-nnet(classificacao ~ . , data = mush.Trein,size = 1)
pred.percep<-predict(fit.percep, newdata = mush.Test[, -1], type = "class")
pred.percep<-as.factor(pred.percep)
#Vamos fazer a matriz de confusão
R.modelo.percep.cv[i,]<-confusionMatrix(data=pred.percep, reference = mush.Test$classificacao, positive="comestivel")$byClass

}

# Salvando os resultados do for anterior
R.modelo.arvore.k[t,]<-colMeans(R.modelo.arvore.cv)
R.modelo.random.k[t,]<-colMeans(R.modelo.random.cv)
#R.modelo.Knn.k[t,]<-colMeans(R.modelo.Knn.cv)
R.modelo.SVM.k[t,]<-colMeans(R.modelo.SVM.cv)
R.modelo.pred.logit_mod.k[t,]<-colMeans(R.modelo.pred.logit_mod.cv)
R.modelo.percep.k[t,]<-colMeans(R.modelo.percep.cv)

}

```

```

R.Arvore<-colMeans(R.modelo.arvore.k)
R.Random<-colMeans(R.modelo.random.k)
#R.Knn<-colMeans(R.modelo.Knn.k)
R.SVM<-colMeans(R.modelo.SVM.k)
R.Logis.Mod<-colMeans(R.modelo.pred.logit_mod.k)
R.perceptron<-colMeans(R.modelo.percep.k)

Result.Matriz<-matrix(NA,nrow = 11,ncol = 6)
Result.Matriz[,1]<-R.Arvore
Result.Matriz[,2]<-R.Random
Result.Matriz[,3]<-R.SVM
#Result.Matriz[,4]<-R.Knn
Result.Matriz[,5]<-R.Logis.Mod
Result.Matriz[,6]<-R.perceptron
row.names(Result.Matriz)<-c("Sensitivity","Specificity","Pos Pred Value","Neg Pred Value",
                           "Precision","Recall","F1","Prevalence","Detection Rate",
                           "Detection Prevalence","Balanced Accuracy")
colnames(Result.Matriz)<-c("Arvore", "Random", "SVM", "Knn", "Logist.Mult", "Perceptron")

```

Hide

Result.Matriz

	Arvore	Random	SVM	Knn	Logist.Mult
Sensitivity	1.0000000	1.0000000	1.0000000	NA	1.0000000
Specificity	0.9879028	1.0000000	0.9879028	NA	1.0000000
Pos Pred Value	0.9888868	1.0000000	0.9888868	NA	1.0000000
Neg Pred Value	1.0000000	1.0000000	1.0000000	NA	1.0000000
Precision	0.9888868	1.0000000	0.9888868	NA	1.0000000
Recall	1.0000000	1.0000000	1.0000000	NA	1.0000000
F1	0.9944065	1.0000000	0.9944065	NA	1.0000000
Prevalence	0.5178792	0.5178792	0.5178792	NA	0.5178792
Detection Rate	0.5178792	0.5178792	0.5178792	NA	0.5178792
Detection Prevalence	0.5237115	0.5178792	0.5237115	NA	0.5178792
Balanced Accuracy	0.9939514	1.0000000	0.9939514	NA	1.0000000
	Perceptron				
Sensitivity	0.9971190				
Specificity	0.9582097				
Pos Pred Value	0.9660379				
Neg Pred Value	0.9966505				
Precision	0.9660379				
Recall	0.9971190				
F1	0.9804889				
Prevalence	0.5178792				
Detection Rate	0.5163872				
Detection Prevalence	0.5365351				
Balanced Accuracy	0.9776644				

Rodando os modelos para todos os dados

Vamos separar o banco de dados em 80% em dados de treinamento e 20% em dados de teste e vamos verificar a matriz de confusão para esses modelos estudados.

Hide

```
# Vamos fazer uma amostra com 80% para o banco de treinamento e 20% para banco de Test
set.seed(2020)

amostra<-createDataPartition(y = mush$classificacao, times = 1, p = 0.8, list = FALSE)

#####
#Variaveis temporarias Treinamento
Mush.Trein<-mush[amostra[,1],]

#Variaveis temporarias Treinamento
Mush.Test<-mush[-amostra[,1],]

##### MODELOS LOGISTICO Multinomial #####
fit.logit_mod <- multinom(classificacao ~ ., data = Mush.Trein)
```

```
# weights:  99 (98 variable)
initial  value 4505.456674
iter   10 value 154.017715
iter   20 value  7.602980
iter   30 value  0.863155
iter   40 value  0.376992
iter   50 value  0.228882
iter   60 value  0.127006
iter   70 value  0.067852
iter   80 value  0.038407
iter   90 value  0.021542
iter  100 value  0.008023
final   value  0.008023
stopped after 100 iterations
```

Hide

```
pred.logit_mod<-predict(fit.logit_mod, newdata = Mush.Test[,-1], type = "class")

confusionMatrix(data=pred.logit_mod, reference = Mush.Test$classificacao, positive="comestivel")
```

## Confusion Matrix and Statistics

	Reference	
Prediction	comestivel	venenoso
comestivel	841	0
venenoso	0	783

Accuracy : 1

95% CI : (0.9977, 1)

No Information Rate : 0.5179

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 1

Mcnemar's Test P-Value : NA

Sensitivity : 1.0000

Specificity : 1.0000

Pos Pred Value : 1.0000

Neg Pred Value : 1.0000

Prevalence : 0.5179

Detection Rate : 0.5179

Detection Prevalence : 0.5179

Balanced Accuracy : 1.0000

'Positive' Class : comestivel

Hide

```
##### MODELOS Arvore #####
fit.arvore<-rpart(classificacao ~ . , data = Mush.Trein)
pred.arvore<-predict(fit.arvore, newdata = Mush.Test[,-1], type = "class")
#Vamos fazer a matriz de confusão
confusionMatrix(data=pred.arvore, reference = Mush.Test$classificacao, positive="comestivel"
)
```

## Confusion Matrix and Statistics

	Reference	
Prediction	comestivel	venenoso
comestivel	841	6
venenoso	0	777

Accuracy : 0.9963

95% CI : (0.992, 0.9986)

No Information Rate : 0.5179

P-Value [Acc > NIR] : < 2e-16

Kappa : 0.9926

Mcnemar's Test P-Value : 0.04123

Sensitivity : 1.0000

Specificity : 0.9923

Pos Pred Value : 0.9929

Neg Pred Value : 1.0000

Prevalence : 0.5179

Detection Rate : 0.5179

Detection Prevalence : 0.5216

Balanced Accuracy : 0.9962

'Positive' Class : comestivel

Hide

```
##### MODELOS Random florest #####
```

```
fit.random<-randomForest(classificacao ~ . , data = Mush.Trein)
```

```
pred.random<-predict(fit.random, newdata = Mush.Test[,-1], type = "class")
```

```
#Vamos fazer a matriz de confusão
```

```
confusionMatrix(data=pred.random, reference = Mush.Test$classificacao, positive="comestivel"
```

```
)
```

## Confusion Matrix and Statistics

	Reference	
Prediction	comestivel	venenoso
comestivel	841	0
venenoso	0	783

Accuracy : 1

95% CI : (0.9977, 1)

No Information Rate : 0.5179

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 1

Mcnemar's Test P-Value : NA

Sensitivity : 1.0000

Specificity : 1.0000

Pos Pred Value : 1.0000

Neg Pred Value : 1.0000

Prevalence : 0.5179

Detection Rate : 0.5179

Detection Prevalence : 0.5179

Balanced Accuracy : 1.0000

'Positive' Class : comestivel

Hide

```
##### MODELOS Knn #####
```

```
fit.knn<-knn3(classificacao ~ . , data = Mush.Trein,k=6)
```

```
pred.knn<-predict(fit.knn, newdata = Mush.Test[,-1], type = "class")
```

```
#Vamos fazer a matriz de confusão
```

```
confusionMatrix(data=pred.knn, reference = Mush.Test$classificacao, positive="comestivel")
```

## Confusion Matrix and Statistics

	Reference	
Prediction	comestivel	venenoso
comestivel	841	0
venenoso	0	783

Accuracy : 1

95% CI : (0.9977, 1)

No Information Rate : 0.5179

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 1

Mcnemar's Test P-Value : NA

Sensitivity : 1.0000

Specificity : 1.0000

Pos Pred Value : 1.0000

Neg Pred Value : 1.0000

Prevalence : 0.5179

Detection Rate : 0.5179

Detection Prevalence : 0.5179

Balanced Accuracy : 1.0000

'Positive' Class : comestivel

Hide

```
##### MODELOS SVM #####
fit.svm<-svm(classificacao ~ . , data = Mush.Trein, type='C-classification', kernel='radial'
)
pred.svm<-predict(fit.svm, newdata = Mush.Test[,-1], type = "class")
#Vamos fazer a matriz de confusão
confusionMatrix(data=pred.svm, reference = Mush.Test$classificacao, positive="comestivel")
```

## Confusion Matrix and Statistics

	Reference	
Prediction	comestivel	venenoso
comestivel	841	2
venenoso	0	781

Accuracy : 0.9988

95% CI : (0.9956, 0.9999)

No Information Rate : 0.5179

P-Value [Acc > NIR] : <2e-16

Kappa : 0.9975

Mcnemar's Test P-Value : 0.4795

Sensitivity : 1.0000

Specificity : 0.9974

Pos Pred Value : 0.9976

Neg Pred Value : 1.0000

Prevalence : 0.5179

Detection Rate : 0.5179

Detection Prevalence : 0.5191

Balanced Accuracy : 0.9987

'Positive' Class : comestivel

Hide

##### MODELO Perceptron #####

```
fit.percep<-nnet(classificacao ~ . , data = Mush.Trein,size = 1)
```

```
# weights: 100
initial value 5188.558286
iter 10 value 572.551624
iter 20 value 314.035689
iter 30 value 267.849971
iter 40 value 265.122832
iter 50 value 265.079377
iter 60 value 265.073959
iter 70 value 265.073639
iter 80 value 265.072144
final value 265.071969
converged
```

Hide



```

pred.percep<-predict(fit.percep, newdata = Mush.Test[,-1], type = "class")
pred.percep<-as.factor(pred.percep)
#Vamos fazer a matriz de confusão
confusionMatrix(data=pred.percep, reference = Mush.Test$classificacao, positive="comestivel"
)

```

#### Confusion Matrix and Statistics

	Reference	
Prediction	comestivel	venenoso
comestivel	841	21
venenoso	0	762

Accuracy : 0.9871

95% CI : (0.9803, 0.992)

No Information Rate : 0.5179

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9741

Mcnemar's Test P-Value : 1.275e-05

Sensitivity : 1.0000

Specificity : 0.9732

Pos Pred Value : 0.9756

Neg Pred Value : 1.0000

Prevalence : 0.5179

Detection Rate : 0.5179

Detection Prevalence : 0.5308

Balanced Accuracy : 0.9866

'Positive' Class : comestivel

Hide