

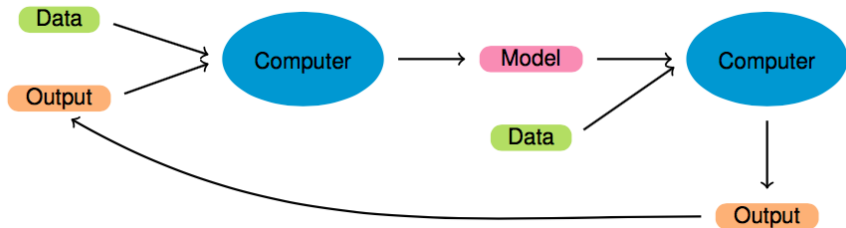
# MAD-CB



## Machine Learning – 2

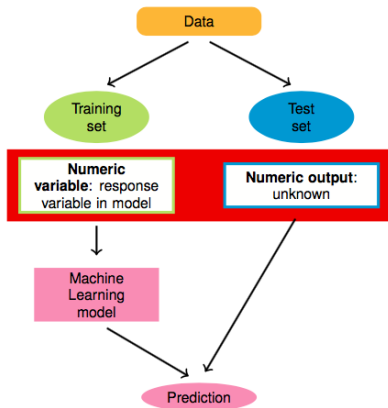
- Inteligência artificial (“AI”)
- Modelo orientado a dados
- Algoritmos **aprendem** por treinamento com dados observados
- E **prever casos desconhecidos**
- Computadores de hoje capazes de tratar essas bases de dados
  - ▶ Mesmo laptops

# Machine Learning – 2

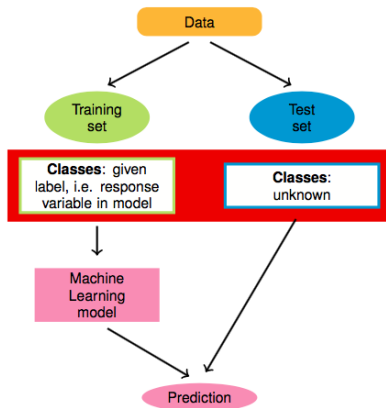


# ## Supervisionada – Classificação vs. Regressão

**Regression**  
e.g. weight loss



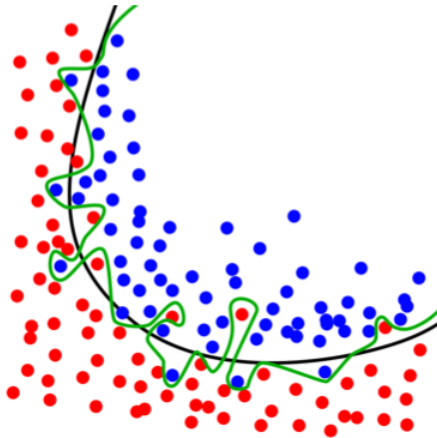
**Classification**  
e.g. healthy vs disease



# Features – Covariáveis

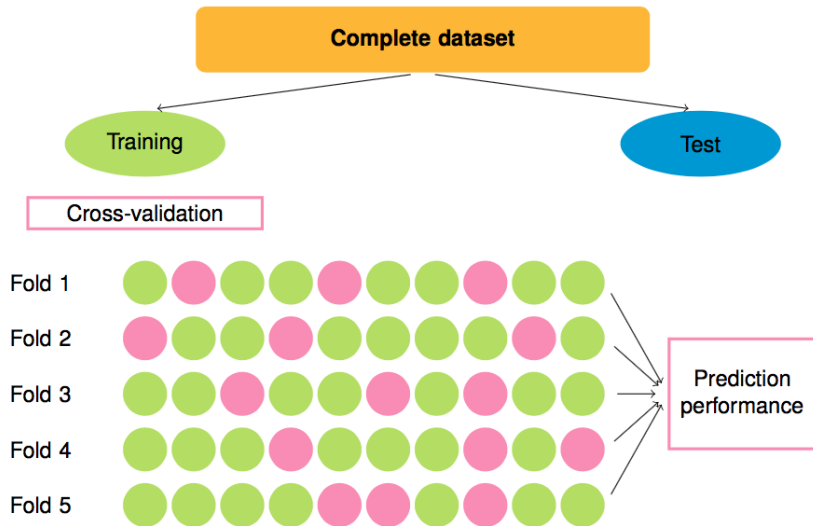
- Variáveis para treinar o modelo
- Selecionar as variáveis certas – **crucial**
- Mais features não necessariamente bom
  - ▶ Perigo de “overfitting”

# Overfitting



*Image Source: Wikipedia*

# Treinamento, Testes & Cross Validation





**NUNCA, JAMAIS, USE OS MESMOS DADOS PARA TESTES  
QUE VOCÊ USOU PARA TREINAMENTO**

# Cross-Validation (*k-fold*)

- Uma de uma serie de técnicas usadas para fortalecer a capacidade do modelo para prever resultados
  - ▶ Bootstrap - reamostragem
- Com os dados de treinamento só
- Divide os dados em  $k$  grupos (“folds”) aleatórios de tamanho igual
- Construir o seu modelo usando todos fora de um grupo
- Testar o modelo nos dados no grupo que você reservou
  - ▶ Calcular o erro entre as previsões com o modelo e os valores observados
- Repetir e fazer a média dos erros
- O modelo (entre os  $k$  que você construiu) com a média menor é o modelo melhor

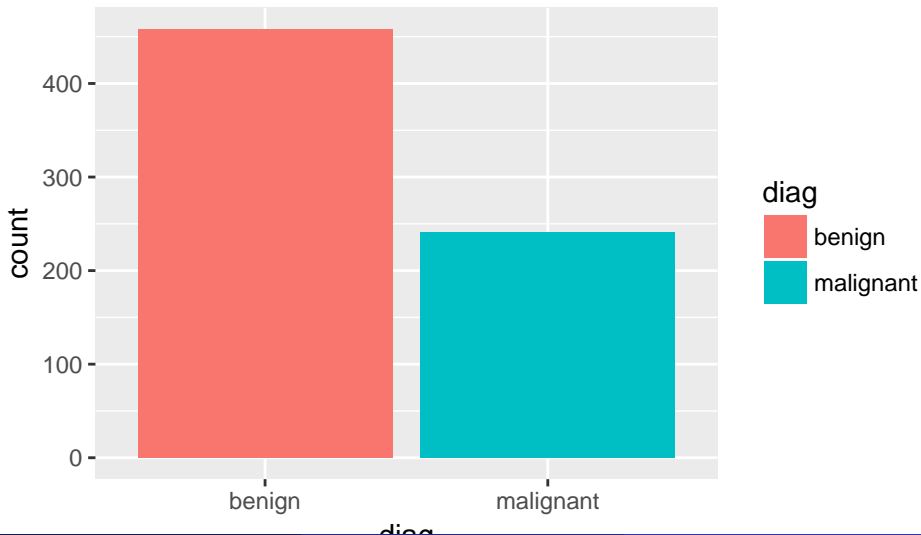
- Vêm de Wisconsin dados sobre câncer de mama
- Características dos tumores de mama
- Variável dependente: diagnose (diag)

```
glimpse(bc_data)
```

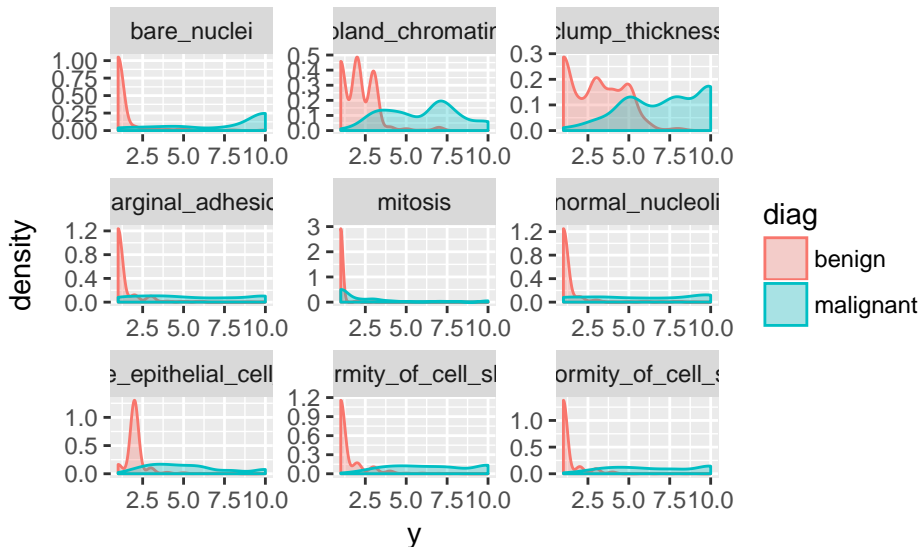
```
## Observations: 699
## Variables: 10
## $ diag                <fctr> benign, benign, benign, benign, b...
## $ clump_thickness      <dbl> 5, 5, 3, 6, 4, 8, 1, 2, 2, 4, 1, 2...
## $ uniformity_of_cell_size <dbl> 1, 4, 1, 8, 1, 10, 1, 1, 1, 2, 1, ...
## $ uniformity_of_cell_shape <dbl> 1, 4, 1, 8, 1, 10, 1, 2, 1, 1, 1, ...
## $ marginal_adhesion    <dbl> 1, 5, 1, 1, 3, 8, 1, 1, 1, 1, 1, 1...
## $ single_epithelial_cell_size <dbl> 2, 7, 2, 3, 2, 7, 2, 2, 2, 2, 1, 2...
## $ bare_nuclei          <dbl> 1, 10, 2, 4, 1, 10, 10, 1, 1, 1, 1...
## $ bland_chromatin       <dbl> 3, 3, 3, 3, 3, 9, 3, 3, 1, 2, 3, 2...
## $ normal_nucleoli       <dbl> 1, 2, 1, 7, 1, 7, 1, 1, 1, 1, 1, 1...
## $ mitosis              <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 5, 1, 1, 1...
```

# Gráfico das Diagnoses

```
brgr1 <- ggplot(bc_data, aes(x = diag, fill = diag)) + geom_bar()  
brgr1
```



# Gráfico das Covariáveis com a Diagnose



- Funções para apoiar machine learning
- Pode conduzir toda a análise dentro de caret
- No grupos dos pacotes iniciais

# Separar Treinamento e Testes

- Utilizar função `caret::createDataPartition()` para criar bases separadas
  - ▶ 1 para treinamento do modelo
  - ▶ 1 para testes
- Especificar (p) porcentagem de dados colocado na base de treinamento
  - ▶ Entre 0.5 (50%) e 0.7 (70%)
- `createDataPartition()` estratifica os dados baseada nas proporções da variável y



# Criar as Bases Treinamento e Testes

```
set.seed(42)
indice <- createDataPartition(bc_data$diag, p = 0.7, list = FALSE)
train_data <- bc_data[indice, ] # use os índices para o treinamento
test_data <- bc_data[-indice, ] # use os outros para testes
```

- Antes de iniciar o passo de treinar o modelos, precisamos decidir qual tipo de validação queremos usar
  - ▶ bootstrap, k-fold cross validation
- Especificar através da função `caret::trainControl()`
- Queremos usar *10-fold cross validation*
- Se pudermos repetir o processo de cross validation, faz a seleção do modelo ainda mais forte
  - ▶ Repetiremos 10 vezes


# trainControl()

```
set.seed(42)
control <- trainControl(method = "repeatedcv",
                        number = 10,
                        repeats = 10,
                        savePredictions = TRUE,
                        verboseIter = FALSE)
```

# Treinamento do Modelo – Regressão Logística

```
model_glm <- caret::train(diag ~ .,  
                           data = train_data,  
                           method = "glm",  
                           preProcess = c("scale", "center"),  
                           trControl = control)
```

# Objeto de Modelo

 <code>model_glm</code>	<code>Large train (23 elements, 1.1 Mb)</code>
--	--

- R preserva todas as iterações do modelo
- Objeto grande (1MB)

```
model_glm
```

```
## Generalized Linear Model
##
## 490 samples
##   9 predictor
##   2 classes: 'benign', 'malignant'
##
## Pre-processing: scaled (9), centered (9)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 442, 441, 441, 441, 441, 441, ...
## Resampling results:
##
##   Accuracy   Kappa
## 0.9594182  0.9097996
##
##
```

# Resumo dos Resultados do Modelo

```
summary(model_glm)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -3.2774  -0.1329  -0.0728   0.0264   2.4573
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.1513     0.3483  -3.306 0.000946 ***
## clump_thickness  1.3536     0.4249   3.186 0.001443 **
## uniformity_of_cell_size 0.1301     0.6604   0.197 0.843845
## uniformity_of_cell_shape 0.9694     0.7195   1.347 0.177891
## marginal_adhesion  0.9932     0.3693   2.689 0.007161 **
## single_epithelial_cell_size 0.2168     0.3728   0.581 0.560960
## bare_nuclei       1.2173     0.3612   3.370 0.000751 ***
## bland_chromatin    1.1771     0.4725   2.492 0.012720 *
## normal_nucleoli    0.3352     0.3666   0.914 0.360500
## mitosis            0.8211     0.5770   1.423 0.154765
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 631.346  on 489  degrees of freedom
## Residual deviance:  85.712  on 480  degrees of freedom
## AIC: 105.71
##
## Number of Fisher Scoring iterations: 8
```

# O Modelo Pode Predizer os Resultados de Treinamento e de Teste?

- Função `predict()`
  - ▶ com modelo e valores para ser usados para previsão
- Aplicado a base de `train` como exemplo
- Mais interessante – base de `test`
  - ▶ Modelo nunca viu esses dados antes
- **Teste ácido**



# Previsões

```
predtr <- predict(model_glm, train_data)
predtestglm <- predict(model_glm, test_data)
prop.table(table(predtestglm))
```

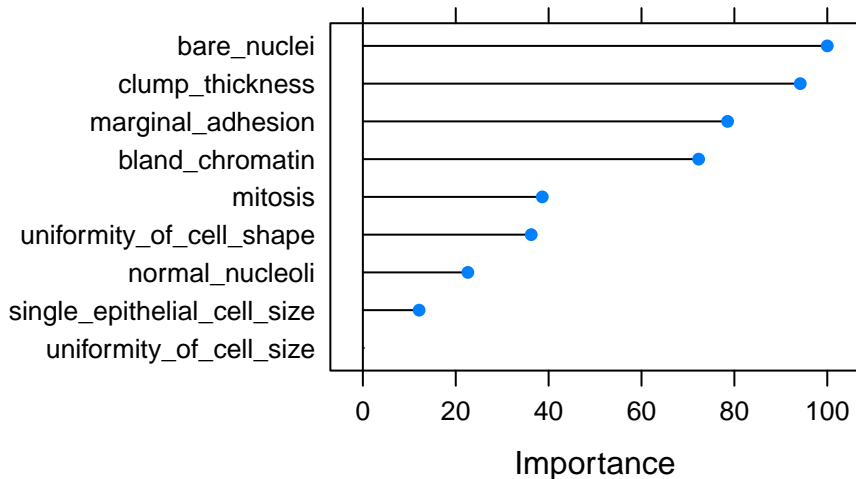
```
## predtestglm
##    benign malignant
## 0.6507177 0.3492823
```

```
prop.table(table(predtr))
```

```
## predtr
##    benign malignant
## 0.6510204 0.3489796
```

# Quais Variáveis Têm Importância para o Modelo

```
plot(caret::varImp(model_glm))
```



# Previsões com os Dados de Teste – Matriz de Confusão

```
confusionMatrix(predtestglm, test_data$diag)
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction  benign malignant
##   benign      133         3
##   malignant    4         69
##
##               Accuracy : 0.9665
##               95% CI : (0.9322, 0.9864)
##   No Information Rate : 0.6555
##   P-Value [Acc > NIR] : <2e-16
##
##               Kappa : 0.9261
##   Mcnemar's Test P-Value : 1
##
##               Sensitivity : 0.9708
##               Specificity : 0.9583
##               Pos Pred Value : 0.9779
##               Neg Pred Value : 0.9452
##               Prevalence : 0.6555
##               Detection Rate : 0.6364
##   Detection Prevalence : 0.6507
##               Balanced Accuracy : 0.9646
##
##   'Positive' Class : benign
##
```

# Previsões com os Dados de Treinamento – Matriz de Confusão

```
confusionMatrix(predtr, train_data$diag)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction  benign malignant
##   benign      313         6
##   malignant    8        163
##
##              Accuracy : 0.9714
##              95% CI : (0.9525, 0.9843)
##   No Information Rate : 0.6551
##   P-Value [Acc > NIR] : <2e-16
##
##              Kappa : 0.9369
##   Mcnemar's Test P-Value : 0.7893
##
##              Sensitivity : 0.9751
##              Specificity : 0.9645
##   Pos Pred Value : 0.9812
##   Neg Pred Value : 0.9532
##   Prevalence : 0.6551
##   Detection Rate : 0.6388
##   Detection Prevalence : 0.6510
##   Balanced Accuracy : 0.9698
##
##   'Positive' Class : benign
##
```

# “Receiver Operating Characteristic” (ROC) Validação do Modelo

- Desenvolvido ao início da WWII para determinar o que foi o sinal recebido pela nova tecnologia, *radar*
  - ▶ Avião ou pássaro
- Mede *sensibilidade* vs. *especificidade* de um modelo
- *Sensibilidade* = % do resultado positivo correto
  - ▶ Teste mede % dos resultados positivos das pessoas com uma doença
  - ▶ Taxa de previsões positivas certas (“True positive rate”, TPR)
- *Especificidade* = % do resultado negativo correto
  - ▶ Teste mede % dos resultados negativos das pessoas sem uma doença
  - ▶ Taxa de previsões positivas erradas (“False positive rate”, FPR)
  - ▶ Visualização da troca entre alta sensibilidade do modelo vs. alta especificidade
  - ▶ Não pode ter os 2 juntos

# AUC (Área abaixo da Curva)

- AUC mede quanto porcentagem da área do gráfico a curva do modelo cobre
- 100% quer dizer que o modelo é perfeitamente sensível e específico
- 50% quer dizer que o resultado é puramente aleatório
- Modelos com AUC maiores prevêm melhor que eles com AUC menores
- Pergunta:
  - ▶ Como calcular área abaixo de uma curva qualquer em matemática?

- 2 Pacotes
  - ▶ pROC
  - ▶ ROCR
- Iguais (basicamente)
- Começamos com pROC
  - ▶ Comando principal – roc

- Compara as previsões contra as observações
- Previsões precisam ser numéricas (não factor)
- Use as opções seguintes:
  - ▶ `plot = TRUE, percent = TRUE, ci = TRUE, grid = TRUE`
- Produz um gráfico e dados sobre o AUC

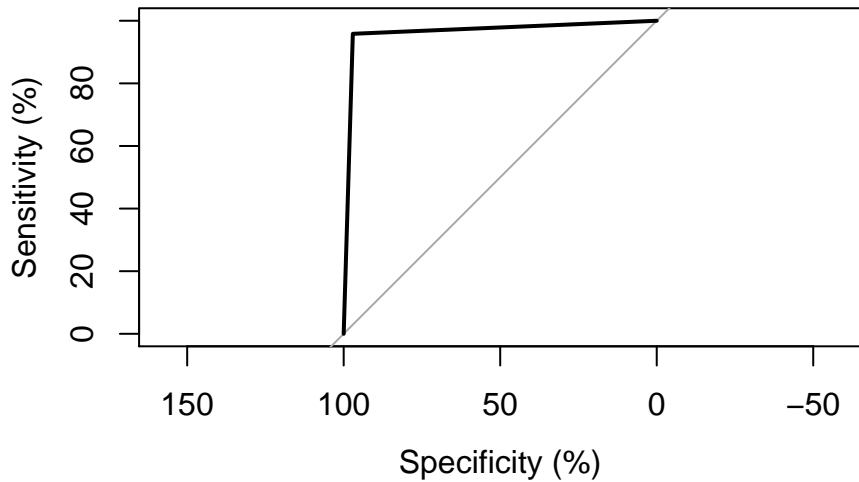


# Chamada e Estatísticas

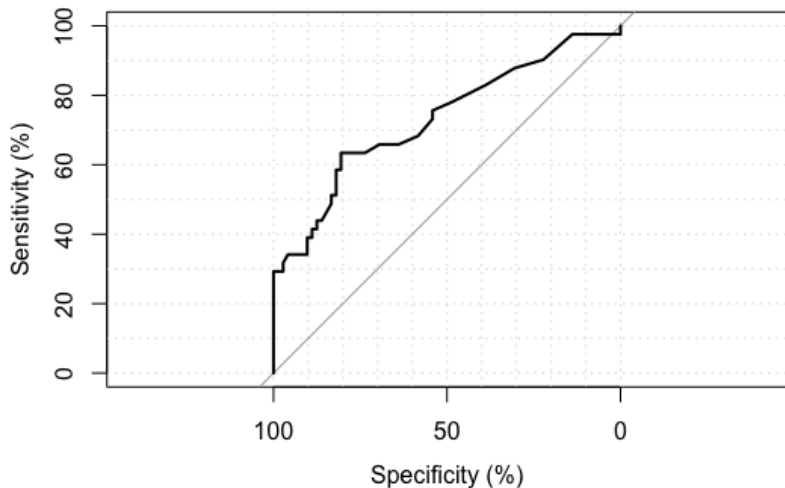
```
## colocar pretest na faixa de 0:1 (atualmente 1:2)
pretestroc <- as.numeric(pretestglm) -1
rocteste <- roc(response = test_data$diag,
               predictor = pretestroc,
               levels = c("benign", "malignant"),
               plot = FALSE, percent = TRUE,
               ci = TRUE, grid = TRUE)

rocteste
```

```
##
## Call:
## roc.default(response = test_data$diag, predictor = pretestroc, levels = c("
##
## Data: pretestroc in 137 controls (test_data$diag benign) < 72 cases (test_data$
## Area under the curve: 96.46%
## 95% CI: 93.74%-99.18% (DeLong)
```



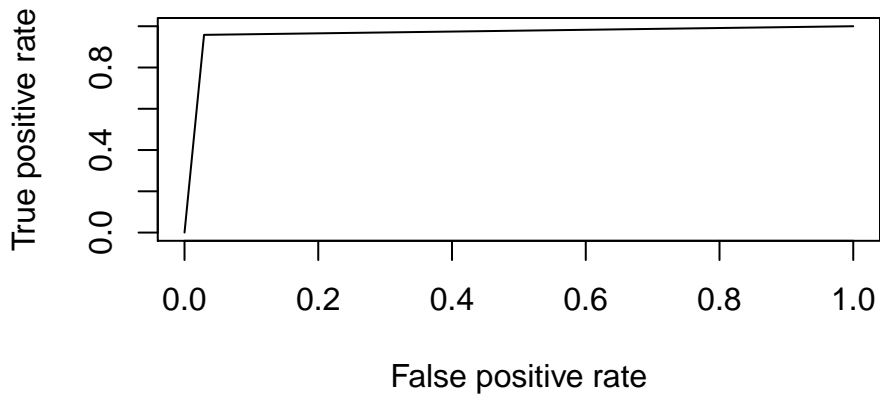
## Outra Curva ROC com Dados Mais Variáveis



# Procedimento com ROCR

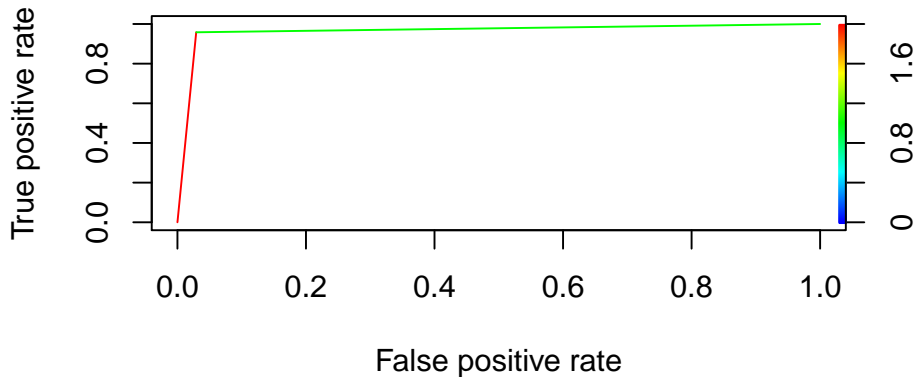
- ROCR quer os dados num formato específico
- Precisa refazer a previsão utilizando a função deste pacote
- Função usará uma versão numérica das previsões `predtest`
- Depois calcular os valores da curva e fazer o gráfico
- ROCR utiliza a terminologia “tpr” e “fpr” para gráfico ROC
- Pode imprimir sensibilidade e especificidade com `sens`, `spec`

```
## Fazer previsão do modelo com ROCR
ROCRpred <- prediction(as.numeric(predtestroc), test_data$diag)
ROCRperf <- performance(ROCRpred, "tpr", "fpr")
```



# Gráfico com Cores

```
plot(ROCperf, colorize = TRUE)
```



- Onde no gráfico fica a troca ótima?
  - ▶ No ponto mais para cima e para esquerda
- `pROC::coords()` pode calcular este ponto
- Precisa dar as seguintes informações a função:
  - ▶ nome de objeto de ROC
  - ▶ Palavra “best”
  - ▶ Coordenados para retornar a você (“threshold”)

# Limites de Nosso Modelo

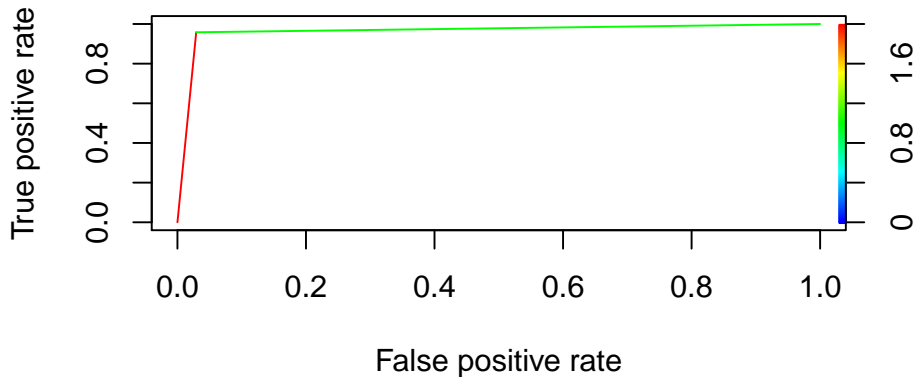
```
coords(rocteste, "best", ret = "threshold")
```

```
## [1] 0.5
```



# Gráfico com Cores

```
plot(ROCRperf, colorize = TRUE)
```



# Novo Modelo – Modelos de Arvore – rpart

- Modelos que constroem arvores de decisão
- Excelentes para problemas de classificação
- Pacote rpart
- Gráficos mostra como escolha das classes está sendo feita
  - ▶ Gráfico vem do pacote `rpart.plot`

# Como Funciona uma Arvore

- Cf. Kuhn & Johnson, *Applied Predictive Modeling* (2013)
- Feita de *nodos* e *ramos*
- Ramos conectam nodos até que chegar num nodo terminal
- Algoritmo cria uma serie de partilhas (divisões) baseado em testes lógicos aninhados
- Os testes lógicos definem a previsão que o modelo faria com novos dados

# Exemplo de uma Regra de uma Arvore

```
if Predictor A >= 1.7 then
|   if Predictor B >= 202.1 then Outcome = 1.3
|   else Outcome = 5.6
else Outcome 2.5
```

# Árvores São uma Técnica de Machine Learning Popular

- Interpretação fácil
- Podem lidar com muitas convariáveis de vários tipos
- Não precisa descrever exatamente a relação entre
  - ▶ Variável dependente
  - ▶ Variáveis independentes
- NA's não criam problemas
- Mas, tem desvantagens também
  - ▶ São instáveis (pequena mudança numa variável pode cause grande mudança no resultado)
  - ▶ Exatidão de previsões não tão boa que outros tipos de modelos

# Funcionamento do Modelo de Arvore

- Algoritmo divide os dados em grupos menores que são mais homogêneos com a dependente
- 3 Critérios para divisão
  - ▶ Qual variável de previsão para usar para o “split”
  - ▶ Profundidade da arvore
  - ▶ A equação de previsão nos nodos terminais
- Metodologia de rpart vem de Breiman et. al (1984)
  - ▶ Classification and regression tree (CART)

# Parâmetros Chaves para rpart

- `method`

- ▶ Para classificação: "class"
- ▶ Para regressão: "anova"

- `control`

- ▶ Vai chamar `rpart.control` explicito
- ▶ `xval`: número de cross-validations
- ▶ `minbucket`: número mínimo de observações em um nodo terminal

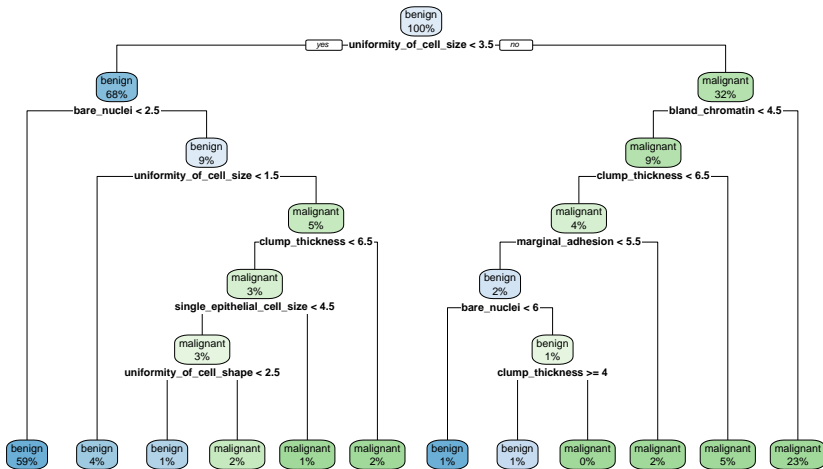
- `parms` – parâmetros para dividindo os casos

- ▶ Só usado para classificação
- ▶ `information`

# Vamos Construir Um Modelo de Câncer de Mama

```
set.seed(42)
fitree1 <- rpart(diag ~ .,
  data = train_data,
  method = "class",
  control = rpart.control(xval = 10,
    minbucket = 2,
    cp = 0),
  parms = list(split = "information"))
```





# Resumo do Modelo de rpart

```
summary(fitree1, cp = 1)
```

```
## Call:
## rpart(formula = diag ~ ., data = train_data, method = "class",
##       parms = list(split = "information"), control = rpart.control(xval = 10,
##       minbucket = 2, cp = 0))
## n= 490
##
##           CP nsplit  rel error    xerror      xstd
## 1 0.822485207      0 1.00000000 1.0000000 0.06226029
## 2 0.038461538      1 0.17751479 0.1775148 0.03140182
## 3 0.009861933      3 0.10059172 0.1360947 0.02770369
## 4 0.005917160      6 0.07100592 0.1597633 0.02988737
## 5 0.000000000     11 0.04142012 0.1360947 0.02770369
##
## Variable importance
##      uniformity_of_cell_size  uniformity_of_cell_shape
##                20                17
##      bare_nuclei             bland_chromatin
##                17                14
##      normal_nucleoli single_epithelial_cell_size
##                14                13
##      clump_thickness             mitosis
##                2                1
##      marginal_adhesion
##                1
##
## Node number 1: 490 observations
## predicted class=benign expected loss=0.344898 P(node) =1
## class counts: 321 169
## probabilities: 0.655 0.345
```

# Splits com cp = 0.1

Node number 1: 490 observations, complexity param=0.8224852  
predicted class=benign expected loss=0.344898 P(node) =1  
class counts: 321 169  
probabilities: 0.655 0.345  
left son=2 (333 obs) right son=3 (157 obs)

## Primary splits:

uniformity_of_cell_size	< 3.5 to the left,	improve=202.8455, (0 missing)
uniformity_of_cell_shape	< 2.5 to the left,	improve=189.0331, (0 missing)
bare_nuclei	< 2.5 to the left,	improve=164.8588, (0 missing)
bland_chromatin	< 3.5 to the left,	improve=156.3825, (0 missing)
single_epithelial_cell_size	< 2.5 to the left,	improve=152.2275, (0 missing)

## Surrogate splits:

uniformity_of_cell_shape	< 3.5 to the left,	agree=0.939, adj=0.809, (0 split)
single_epithelial_cell_size	< 3.5 to the left,	agree=0.894, adj=0.669, (0 split)
normal_nucleoli	< 2.5 to the left,	agree=0.890, adj=0.656, (0 split)
bland_chromatin	< 3.5 to the left,	agree=0.888, adj=0.650, (0 split)
bare_nuclei	< 3.5 to the left,	agree=0.880, adj=0.624, (0 split)

Node number 2: 333 observations  
predicted class=benign expected loss=0.06306306 P(node) =0.6795918  
class counts: 312 21  
probabilities: 0.937 0.063

Node number 3: 157 observations  
predicted class=malignant expected loss=0.05732484 P(node) =0.3204082  
class counts: 9 148  
probabilities: 0.057 0.943

# Previsões com a Arvore

```
predtesttr <- predict(fitree1, newdata = test_data, type = "class")  
prop.table(table(predtesttr))
```

```
## predtesttr  
##      benign malignant  
## 0.6698565 0.3301435
```

# Confusion Matrix – Arvore

```
confusionMatrix(predtesttr, test_data$diag)
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction  benign malignant
##   benign      133         7
##   malignant     4        65
##
##               Accuracy : 0.9474
##               95% CI : (0.9078, 0.9734)
##   No Information Rate : 0.6555
##   P-Value [Acc > NIR] : <2e-16
##
##               Kappa : 0.8823
##   Mcnemar's Test P-Value : 0.5465
##
##               Sensitivity : 0.9708
##               Specificity : 0.9028
##               Pos Pred Value : 0.9500
##               Neg Pred Value : 0.9420
##               Prevalence : 0.6555
##               Detection Rate : 0.6364
##   Detection Prevalence : 0.6699
##               Balanced Accuracy : 0.9368
##
##   'Positive' Class : benign
##
```

# ROC Dados

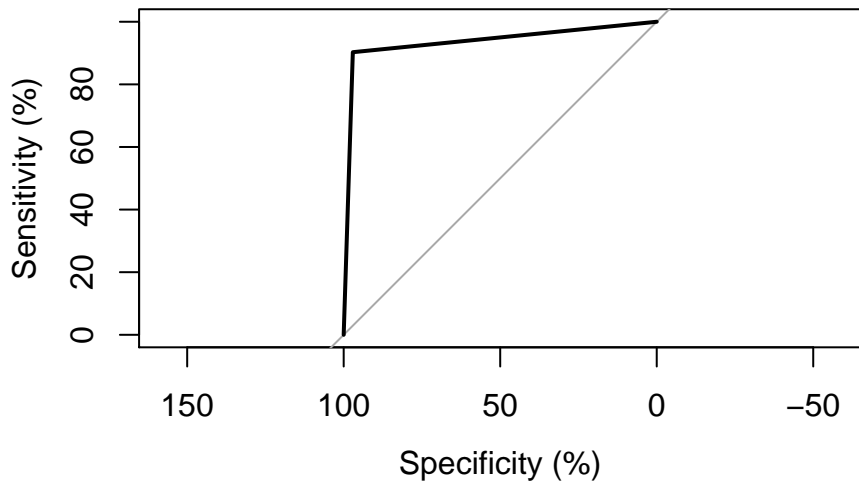
```
## colocar predtest na faixa de 0:1 (atualmente 1:2)
predtesttrroc <- as.numeric(predtesttr) -1
rocteste <- roc(response = test_data$diag,
               predictor = predtesttrroc,
               levels = c("benign", "malignant"),
               plot = FALSE, percent = TRUE,
               ci = TRUE, grid = TRUE)

rocteste
```

```
##
## Call:
## roc.default(response = test_data$diag, predictor = predtesttrroc,      levels = c
##
## Data: predtesttrroc in 137 controls (test_data$diag benign) < 72 cases (test_dat
## Area under the curve: 93.68%
## 95% CI: 89.95%-97.4% (DeLong)
```

```
suppressMessages(coords(rocteste, "best", ret = "threshold"))
```

```
## [1] 0.5
```



# Arvores Mais Robustas – Random Forests

- Random Forests elaborado como algoritmo por Breiman em 2000
- Ideia básica: Combinando resultados de muitas arvores vai produzir uma arvore final melhor

*Grow many deep regression trees to randomized versions of the training data, and average them. Efron & Hastie, 2016*

- “Randomized versions” – pode ser bootstrapping ou outras técnicas de re-amostragem



# Algoritmo de Random Forests

```
1 Select the number of models to build,  $m$ 
2 for  $i = 1$  to  $m$  do
3   Generate a bootstrap sample of the original data
4   Train a tree model on this sample
5   for each split do
6     Randomly select  $k$  ( $< P$ ) of the original predictors
7     Select the best predictor among the  $k$  predictors and
       partition the data
8   end
9   Use typical tree model stopping criteria to determine when a
     tree is complete (but do not prune)
10 end
```

**Algorithm 8.2:** Basic Random Forests

Kuhn & Johnson (2013)

# Random Forests em R

- Pacote randomForest
- Formato:

```
randomForest(y ~ xvars, data = dados, ntrees = 1000,  
             importance = TRUE)
```

- y deve ser expressa como factor para classificação
- Argumentos chaves:
  - ▶ ntrees: número de arvores para a calcular; deve ser muito maior que o número das covariáveis
  - ▶ importance = TRUE: para calcular os valores para importância dos variáveis

# Random Forests Aplicado ao Câncer de Mama

```
arvores = 100
rffit <- randomForest(as.factor(diag) ~ ., data = train_data,
                      ntree = arvores, importance = TRUE, proximity = TRUE)
rffit
```

```
##
## Call:
## randomForest(formula = as.factor(diag) ~ ., data = train_data,      ntree = ar
##               Type of random forest: classification
##               Number of trees: 100
## No. of variables tried at each split: 3
##
##               OOB estimate of  error rate: 3.67%
## Confusion matrix:
##               benign malignant class.error
## benign         311         10 0.03115265
## malignant       8         161 0.04733728
```

Confusion Matrix aqui é dos dados de *treinamento*

# OOB Error????

- “Out of Bag”
  - ▶ Para todas as árvores, os erros associados com os valores não utilizados no treinamento do modelo
  - ▶ Como fizemos com cross-validation

# Previsões com a Random Forest

```
predtestrf <- predict(rffit, newdata = test_data, type = "class")  
prop.table(table(predtestrf))
```

```
## predtestrf  
##      benign malignant  
## 0.6507177 0.3492823
```

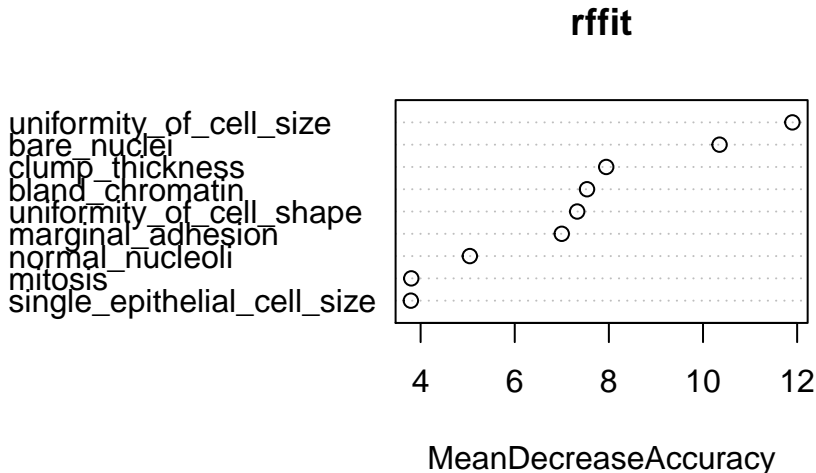
# Desempenho de Random Forest

```
confusionMatrix(predtestrf, test_data$diag)
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction  benign malignant
##   benign      133         3
##   malignant    4         69
##
##               Accuracy : 0.9665
##               95% CI : (0.9322, 0.9864)
##   No Information Rate : 0.6555
##   P-Value [Acc > NIR] : <2e-16
##
##               Kappa : 0.9261
##   Mcnemar's Test P-Value : 1
##
##               Sensitivity : 0.9708
##               Specificity : 0.9583
##               Pos Pred Value : 0.9779
##               Neg Pred Value : 0.9452
##               Prevalence : 0.6555
##               Detection Rate : 0.6364
##   Detection Prevalence : 0.6507
##               Balanced Accuracy : 0.9646
##
##   'Positive' Class : benign
##
```

# Importância das Variáveis

```
randomForest::varImpPlot(rffit, type = 1) ## NB, função dentro de randomForest
```



# O Que Quer Dizer “Mean Decrease Accuracy”

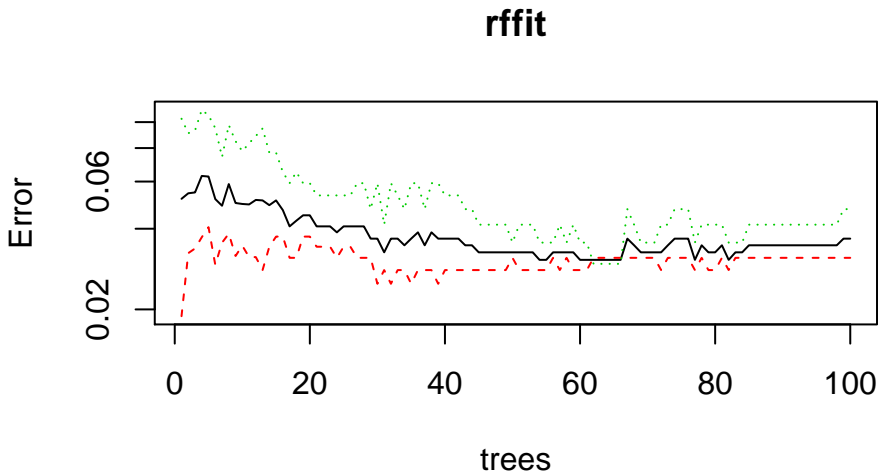
- Através de todos as arvores
  - ▶ A variável causa uma perda de precisão no modelo
- Variáveis que podem causar perda de precisão são mais importantes
- Exemplos:
  - ▶ “bare nuclei” é a mais importante porque pode causar mais perda
  - ▶ “mitosis” é o menos importante, porque qualquer valor que assuma não vai afetar o resultado do modelo, diag



# Controle de Erros

- Gráfico de redução de MSE com o número de arvores calculadas

```
plot(rffit, log = 'y')
```



# Curva ROC e AUC para Random Forests

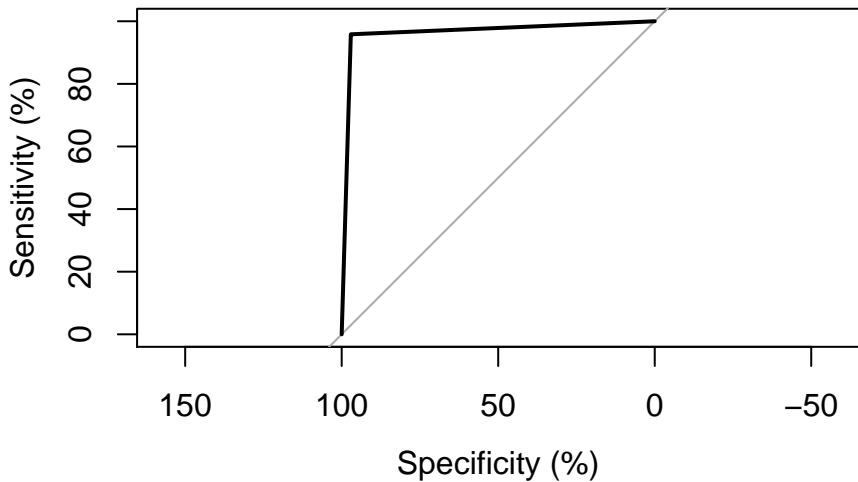
```
## colocar predtest na faixa de 0:1 (atualmente 1:2)
predtestrfroc <- as.numeric(predtestrf) -1
rocteste <- roc(response = test_data$diag,
               predictor = predtestrfroc,
               levels = c("benign", "malignant"),
               plot = FALSE, percent = TRUE,
               ci = TRUE, grid = TRUE)

rocteste
```

```
##
## Call:
## roc.default(response = test_data$diag, predictor = predtestrfroc,      levels = c
##
## Data: predtestrfroc in 137 controls (test_data$diag benign) < 72 cases (test_data$diag malignant)
## Area under the curve: 96.46%
## 95% CI: 93.74%-99.18% (DeLong)
```

```
suppressMessages(coords(rocteste, "best", ret = "threshold"))
```

```
## [1] 0.5
```



# Fazer Random Forests com caret

- Só precisa mudar o a especificação de train
- `method = "rf"`
- caret chama `randomForest` para fazer os calculos
  - ▶ *wrapper* função
- Aqui vamos fazer `set.seed(42)` para ser consistente com os outros métodos

# Calcular os Random Forests

```
set.seed(42)
model_rf <- caret::train(diag ~ .,
                        data = train_data,
                        method = "rf",
                        preProcess = c("scale", "center"),
                        trControl = control)
```

# Resultados Básicos – RF – caret

```
model_rf
```

```
## Random Forest
##
## 490 samples
##   9 predictor
##   2 classes: 'benign', 'malignant'
##
## Pre-processing: scaled (9), centered (9)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 442, 441, 441, 441, 441, 441, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.9702276  0.9344996
##   5     0.9651248  0.9230624
##   9     0.9632881  0.9186159
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

# Calcular as Variáveis Importantes

```
imp <- model_rf$finalModel$importance # Guarda em unidades originais  
importance <- varImp(model_rf, scale = TRUE) # Scale coloca em escala de 100 -> 0
```

# Variáveis Importantes – Escala Original

- % das arvores em que a variável aparece (eu acho??)

```
imp[order(imp, decreasing = TRUE), ]
```

```
##      uniformity_of_cell_size      uniformity_of_cell_shape
##              52.726865              43.797886
##              bare_nuclei              bland_chromatin
##              33.250447              28.200817
##      normal_nucleoli single_epithelial_cell_size
##              16.924956              15.957973
##      clump_thickness      marginal_adhesion
##              14.192735              12.234159
##              mitosis
##              2.885178
```



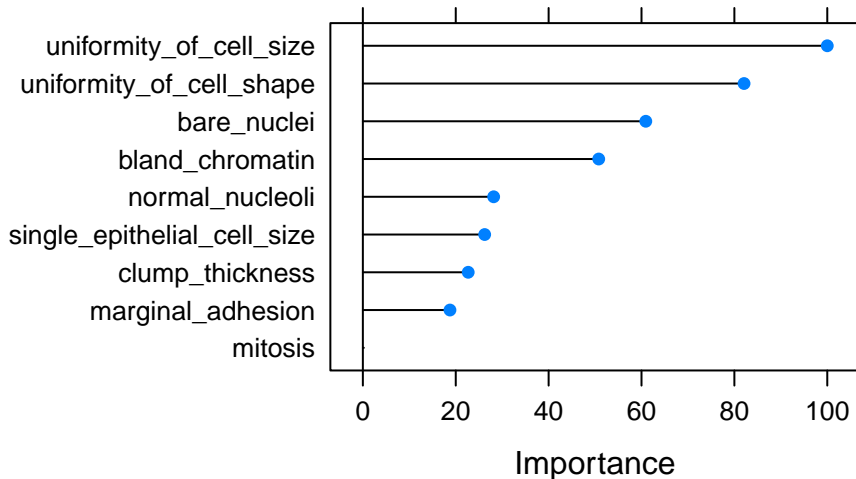
# Variáveis Importantes - Escala 100 -> 0

```
importance
```

```
## rf variable importance
##
## Overall
## uniformity_of_cell_size      100.00
## uniformity_of_cell_shape     82.09
## bare_nuclei                  60.92
## bland_chromatin              50.79
## normal_nucleoli              28.17
## single_epithelial_cell_size  26.23
## clump_thickness              22.69
## marginal_adhesion            18.76
## mitosis                      0.00
```

# Variáveis Importantes – Gráfico

```
plot(importance)
```



# Previsões do Modelo de RF de caret

```
predrfx <- predict(model_rf, test_data)  
prop.table(table(predrfx))
```

```
## predrfx  
##      benign malignant  
## 0.645933 0.354067
```

# Matriz de Confusão – Random Forest – caret

```
confusionMatrix(predrfx, test_data$diag)
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction  benign malignant
##   benign      133         2
##   malignant    4         70
##
##               Accuracy : 0.9713
##               95% CI : (0.9386, 0.9894)
##   No Information Rate : 0.6555
##   P-Value [Acc > NIR] : <2e-16
##
##               Kappa : 0.9369
##   Mcnemar's Test P-Value : 0.6831
##
##               Sensitivity : 0.9708
##               Specificity : 0.9722
##               Pos Pred Value : 0.9852
##               Neg Pred Value : 0.9459
##               Prevalence : 0.6555
##               Detection Rate : 0.6364
##   Detection Prevalence : 0.6459
##               Balanced Accuracy : 0.9715
##
##   'Positive' Class : benign
##
```

# Previsões no Formato de Probabilidades

- `type = "prob"` de `'predict()'` põe os valores em probabilidades
- Deixa você decidir qual seria o limite para diferenciar entre “benign” e “malignant”
  - ▶ Até agora, sempre foi 0.5

```
results <- data.frame(actual = test_data$diag, predict(model_rf, test_data, type = "prob"))
results$prediction <- ifelse(results$benign > 0.5, "benign",
                             ifelse(results$malignant > 0.5, "malignant", NA))
kable(head(results, 8))
```

	actual	benign	malignant	prediction
3	benign	1.000	0.000	benign
8	benign	1.000	0.000	benign
9	benign	0.976	0.024	benign
12	benign	1.000	0.000	benign
13	malignant	0.554	0.446	benign
17	benign	1.000	0.000	benign
19	malignant	0.054	0.946	malignant
21	malignant	0.214	0.786	malignant

# Acertamos com o Novo Modelo?

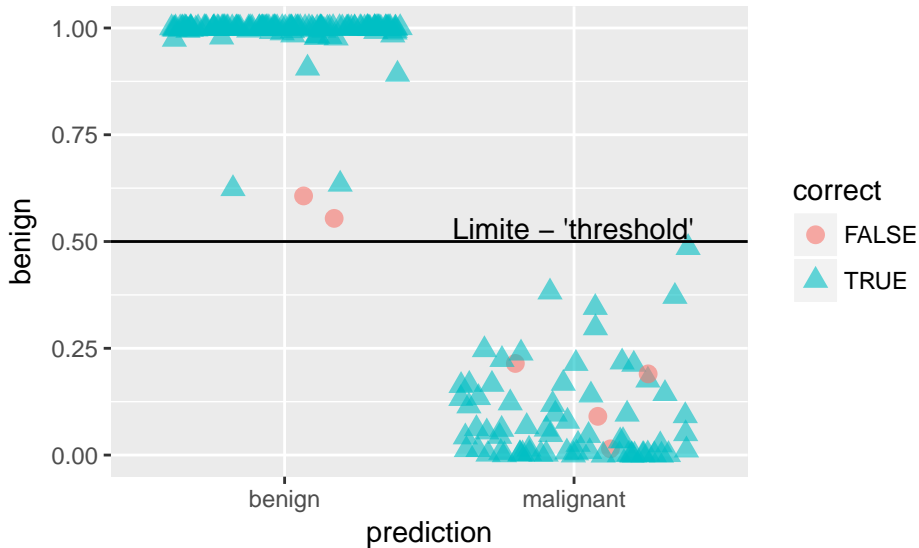
```
results$correct <- ifelse(results$actual == results$prediction, TRUE, FALSE)  
prop.table(table(results$correct))
```

```
##
```

```
##      FALSE      TRUE
```

```
## 0.02870813 0.97129187
```

# Gráfico dos Resultados



# Este Gráfico Mostra

- Erros de “benign”
  - ▶ Os 2 são perto a 0.50
- Erros de “malignant”
  - ▶ Mais espalhadas
  - ▶ Alguns com probabilidades bem perto a verdadeiro “malignant” (0.0)
- Mais confiança nas previsões de “benign”
- Parece que 0.5 é um bom “threshold” entre determinação de “benign” ou “malignant”
  - ▶ Discrimina bem



# Curva ROC e AUC para RF com caret

```
## colocar pretest na faixa de 0:1 (atualmente 1:2)
pretestroc <- as.numeric(predrfx) -1
rocteste <- roc(response = test_data$diag,
               predictor = pretestroc,
               levels = c("benign", "malignant"),
               plot = FALSE, percent = TRUE,
               ci = TRUE, grid = TRUE)

rocteste
```

```
##
## Call:
## roc.default(response = test_data$diag, predictor = pretestroc, levels = c("
##
## Data: pretestroc in 137 controls (test_data$diag benign) < 72 cases (test_data$
## Area under the curve: 97.15%
## 95% CI: 94.77%-99.53% (DeLong)
```

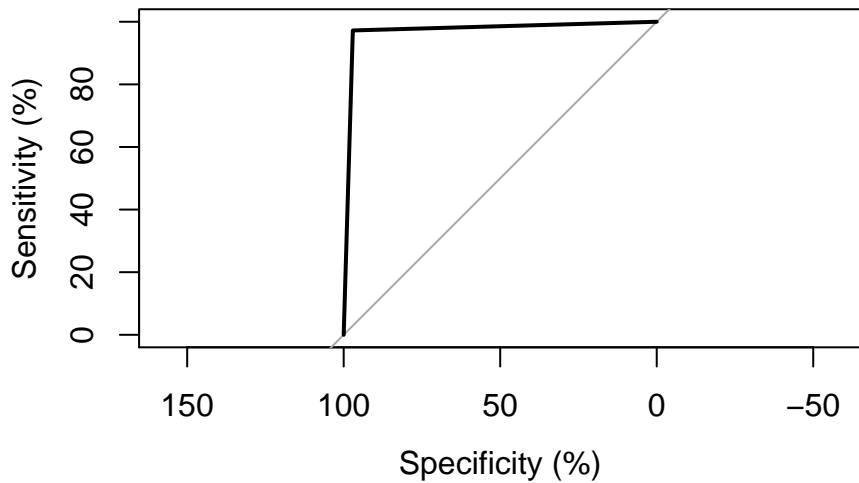


Table 2: Comparação de Precisão de 3 Modelos

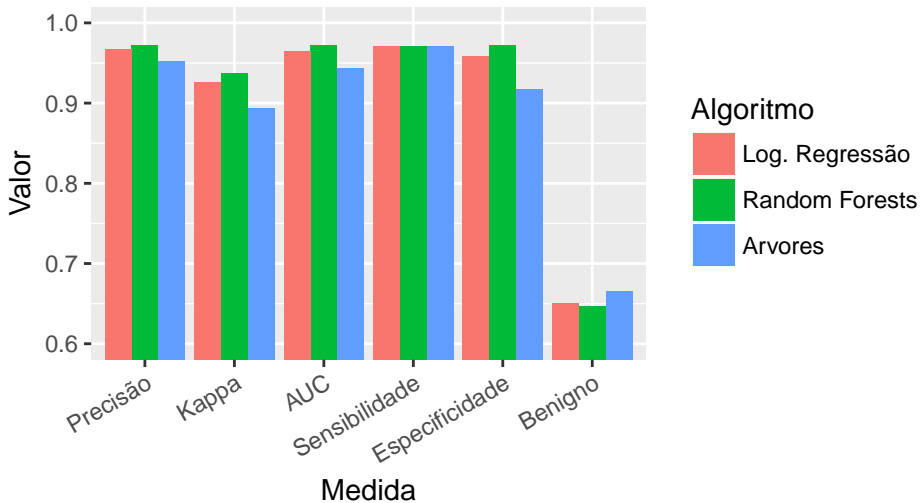
nomes	acc	kappa	sens	spec	auc	beg
modelGLM	0.9665	0.9261	0.9708	0.9583	0.9646	0.6507
rpart	0.9522	0.8934	0.9708	0.9167	0.9437	0.6651
rf	0.9713	0.9369	0.9708	0.9722	0.9715	0.6459

# Gráfico de Comparação

```
ordem <- c("acc", "kappa", "auc", "sens", "spec", "beg")
labs <- c("Precisão", "Kappa", "AUC", "Sensibilidade", "Especificidade", "Benigno")
grcomp <- ggplot(metlong, aes(x = tipo, y = valor, fill = nomes))
grcomp <- grcomp + geom_bar(stat = "identity", position = "dodge")
grcomp <- grcomp + coord_cartesian(ylim = c(.6, 1))
grcomp <- grcomp + scale_x_discrete(limits = ordem,
                                   labels = labs)

grcomp <- grcomp + theme(axis.text.x = element_text(angle=30, hjust=1, vjust=1))
grcomp <- grcomp + labs(title = "Comparação dos Algoritmos de Machine Learning",
                       x = "Medida", y = "Valor", fill = "Algoritmo")
grcomp <- grcomp + scale_fill_discrete(labels = c("Log. Regressão",
                                                  "Random Forests",
                                                  "Arvores"))
```

# Comparação dos Algoritmos de Machine Learning



- Machine Learning Não-Supervisionado
  - ▶ Cluster Analysis