# APS – Modelagem Preditiva

Otávio Farhat Fernandes

## Introdução:

Este trabalho tem o objetivo de aplicar o conteúdo estudado nas aulas de Modelagem Preditiva para realizar a análise de duas bases de dados: a) *churn*: informações sobre os clientes de uma instituição bancária, tentar determinar se o cliente cancelará o serviço ou não; b) *used_cars*: informações sobre os preços de veículos usados da Mercedes, tentar prever o preço. Para isso, utilizo os métodos de Regressão Logística, Árvore de Classificação, *Random Forest* e *Boosting*. As bases serão separadas em um conjunto de treino, onde o modelo será obtido, e um conjunto de testes, onde o modelo será aplicado para verificar sua capacidade de previsão.

Depois da realização dos métodos para cada base de dados, será realizada uma discussão quanto à acurácia no conjunto de testes. Para isso, serão comparadas as curvas ROC de cada método para a base *churn* e para a base *used_cars* utilizo a raiz quadrada do erro quadrático médio de teste para observar qual modelo foi "melhor".

O trabalho está separado em duas partes. Na primeira, a Parte Teórica, onde realizo uma breve explicação histórica de cada método e como funcionam seus mecanismos. Na segunda, a Parte Aplicada, separo em duas seções onde analiso as bases de dados, apresento os códigos do R utilizados e realizo uma discussão dos resultados.

## 1. Parte Teórica:

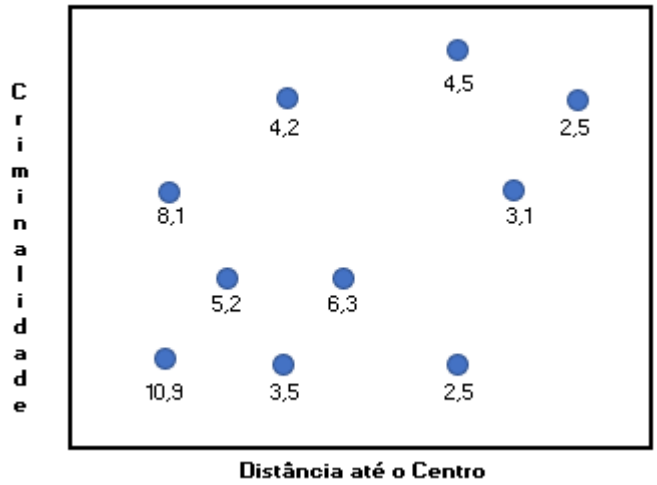
Para prover uma explicação de como funciona uma *Random Forest*, é necessário primeiro explicar as árvores de regressão individuais e o mecanismo de *bagging*.

### a. Árvores de Regressão:

As Árvores de Regressão fazem parte do conjunto de instrumentos de aprendizagem supervisionada. Ou seja, possuem tanto a variável explicada (y) quanto as explicativas (x). Para as Árvores de Regressão, y deve ser um número real, não um valor discreto, como 0 ou 1, por exemplo, como é o caso das Árvores de Classificação. Ambos os métodos utilizam o

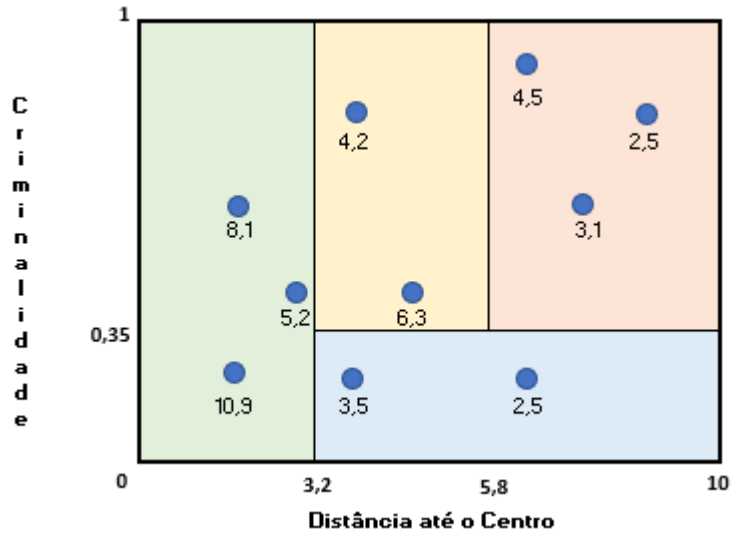
CART (*Classification And Regression Trees*) como algoritmo para criar suas separações. Esse mecanismo foi introduzido por Breiman et al. (1984).

O CART vai primeiro definir a raiz da árvore, e a partir de um parâmetro de decisão que será discutido a seguir, vai criando as divisões da árvore a partir de nós, criando níveis diferentes. Pegando um conjunto de dados, o CART vai dividir esses dados (vai criar um split) em conjuntos menores, como pode ser observado na imagem abaixo para o caso de duas variáveis.

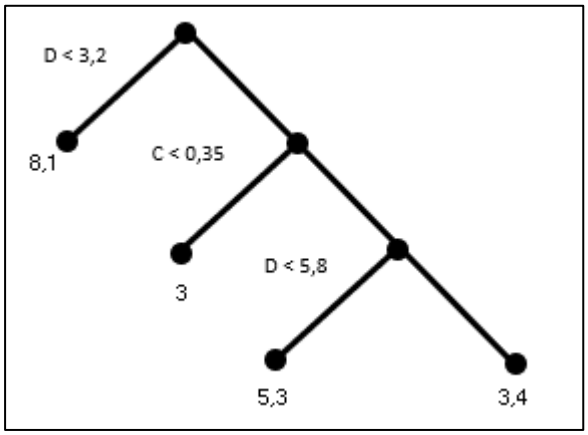



Conjunto de Dados

No exemplo, temos dados do preço de uma casa, representado pelo valor abaixo dos pontos, temos a taxa de criminalidade do local em que está a casa e a distância até o centro da cidade. Admitindo que estes são os dados de treinamento, ou seja, apenas uma parte dos conjunto total de dados, o CART vai separá-los em splits, de acordo com uma medida. No caso, utilizamos o EQM (Erro Quadrático Médio). O CART vai encontrar o método de separação dos dados que minimize o EQM, formando os retângulos verificados na figura.




Conjunto após a separação




Árvore resultante da separação

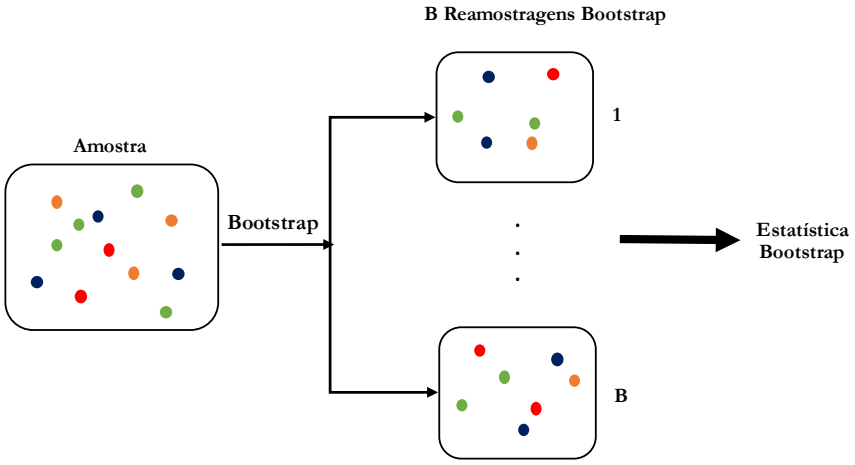
A imagem acima representa a árvore formada pelo CART. Nos nós, temos a média dos valores dentro de cada retângulo. Começando na parte de cima, caso a distância do centro seja menor que 3.2, a previsão do valor da casa será de $8,1. Caso contrário, se a taxa de criminalidade for menor do que 35%, o valor previsto é de 3 etc.

Para verificar a eficiência do modelo, deve-se utilizar o conjunto de testes, ou seja, os dados que não foram utilizados após a escolha dos dados de treinamento. Então, rodamos um modelo de Regressão Linear, um OLS (*Ordinary Least Squares*), e verificamos qual possui a melhor previsão.

### *b. Bagging*

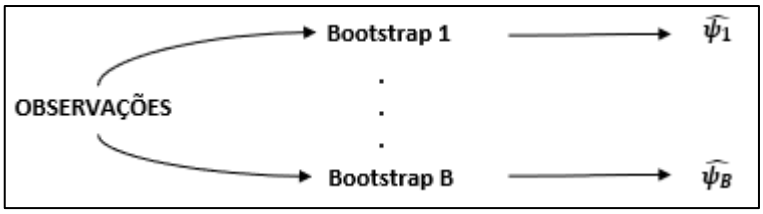
O *Bootstrap Aggregation* (*Bagging*) foi introduzido por Leo Breinman em 1996. O objetivo desse mecanismo é de reduzir a variância de algum método de aprendizagem. Para isso, utiliza da técnica de *Bootstrap*, proposta em 1979, que consiste em formar um conjunto de dados a partir do sorteio de observações dentro de uma amostra, com reposição.

O *Bootstrap* se vale do Teorema de Glivenko-Cantelli, onde a Função de Distribuição Empírica vai se aproximar da Função Geradora dos Dados na medida em que a quantidade de observações coletadas aumenta. Nisso, o objetivo do *bootstrap* é obter uma estimativa do erro quadrático médio de um estimador através do erro quadrático médio de amostras da função de distribuição empírica.




O *Bagging* basicamente permite utilizar o *Bootstrap* para agregar diversas árvores de decisão de modo que a performance da árvore melhore, dado que árvores muito altas podem ter uma variância e um viés grandes, tornando-as sensíveis a pequenas mudanças nos dados. O *Bagging* permite que se contorne o problema de não ter diversas bases de dados para treinar o método de aprendizagem. Para uma árvore de regressão, são geradas B amostras *bootstrap* que servirão como dados de treinamento para cada regressor treinado, e o regressor agregado será igual à média deles. A variância do método será derrubada quando se tira essa média.

Abaixo temos o esquema de funcionamento do *Bagging* e qual será a previsão para um problema de regressão.

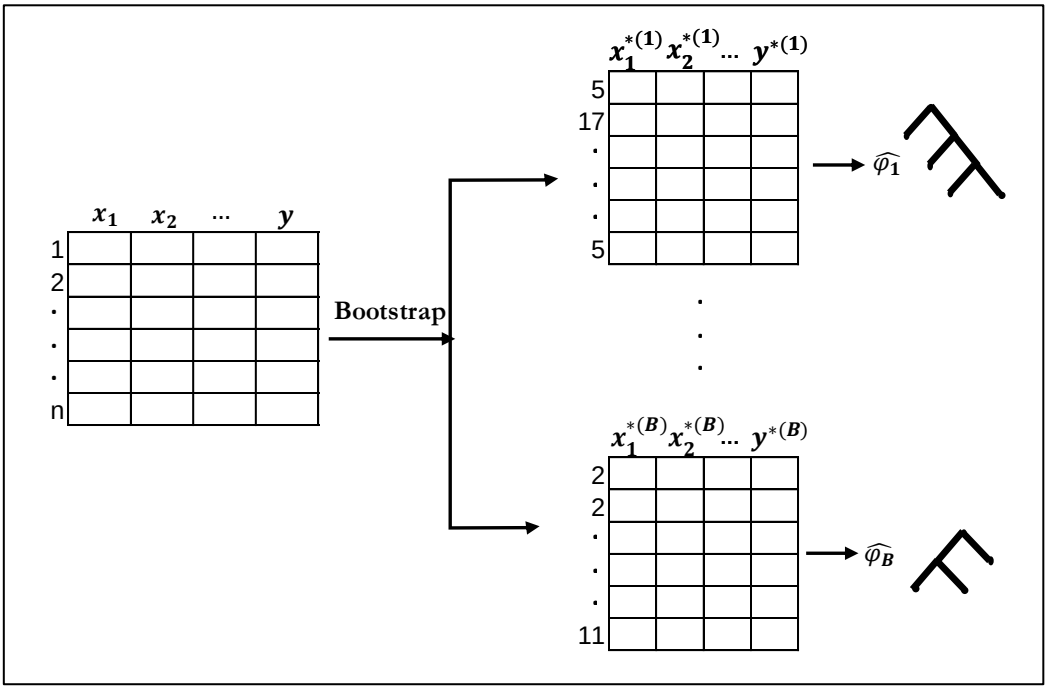



$$\widehat{\psi_B} = \frac{1}{B}\sum \widehat{\psi_i}$$

**c. Random Forests:**

*Random Forest* é um método de aprendizagem supervisionada de classificação e de regressão de alta performance. É um método que utiliza árvores mais altas, via CART e é muito bem-sucedido principalmente para dados tabulares. A *Random Forest* foi inventada por Leo Breiman no começo dos anos 2000. Dado um conjunto de dados, a *Random Forest* usa a ideia de *Bagging* de árvores de classificação ou regressão e acrescenta um novo elemento.

O *Bootstrap* vai pegar uma tabela de dados e amostrar cada linha com repetição, onde o fundamento matemático disso é a função de distribuição empírica e o Teorema de Glivenko-Cantelli. A partir disso, formam-se B novas tabelas com dados originados da amostragem, e é como se tivessem vindo da mesma distribuição. De cada tabela, teremos uma árvore de classificação ou de regressão, dependendo do problema que está sendo avaliado. As B árvores formadas serão diferentes entre si, pois as amostras são diferentes.



Para formar os preditores pelo *Bagging*, varia se o problema é de classificação ou de regressão. Para classificação, pode utilizar o voto da maioria. o estimador será a classe majoritária dentro do conjunto de previsões dos B preditores. Para regressão, a previsão *Bagging* de uma nova observação será a média das previsões de cada uma das B árvores.

Algumas observações importantes são: 1) as árvores não são podadas ao fazer o *Bagging*, elas podem ser altas, onde o viés será baixo e a variância alta; 2) o *Bagging* foi criado para reduzir a variância, mas pode não funcionar se a covariância entre as previsões for alta; para diminuir as covariâncias, Breiman introduziu uma nova ideia que deu origem às *Random Forests*, que consiste em reduzir a correlação entre as árvores. Para isso, utilizou a ideia de subespaço aleatório. No primeiro *split* da árvores, utiliza apenas um subconjunto aleatório m de variáveis preditoras para decisão. Em cada *split* de cada árvores, utiliza essa ideia, mas sem reposição. 3) Erro *Out-of-bag*, onde aproximadamente 37% das observações não entram na amostra gerada pelo *Bootstrap*.

Para encontrar quais são as variáveis mais importantes se verifica as que estão em splits mais acima e que foram utilizadas mais vezes nas decisões. As duas medidas de importância dada para uma variável na *Random Forest* são: 1) quanto que a precisão diminui quando a variável é excluída; 2) quanto a medida de impureza Gini diminui quando uma variável é escolhida para dividir um nó. Para medir a primeira, se utiliza a amostra *out-of-bag*, que não foi utilizada no processo de construção das árvores. Se calcula a acurácia da predição nessa amostra, depois, os valores das variáveis são embaralhados, e por último, se calcula a diminuição da acurácia da predição na amostra *out-of-bag* embaralhada[1]. Wager et al. (2014) mostram métodos para estimar o erro-padrão de preditores de *Bagging* e de *Random Forests*, que podem ser utilizados para se calcular o intervalo de confiança de uma predição de uma *Random Forest*. Para isso, utilizam o método de *jackknife-after-bootstrap* e *infinitesimal jackknife* se baseando nas predições da amostra *out-of-bag*.

## 2. Parte Aplicada:

Importação das Bibliotecas e da base de dados:

```
library(Rcpp)
library(pROC)
library(ISLR)
library(skimr)
library(tree)
```

```
library(randomForest)
library(fastAdaboost)
library(MASS)
library(gbm)
library(caret)

setwd("C:/Users/ferna/Desktop/Modelagem Preditiva/APS")

churn <- read.csv("churn.csv")
used <- read.csv("used_cars.csv")
```

## a. CHURN

**Objetivo:** prever a variável Exited, que indica se o cliente cancelará o serviço ou não.

**Separação em dados de teste e de treinamento:**

```
#fazendo de Yes/No para 1/0
require(dplyr)
churn <- churn %>%
  mutate(Exited = factor(ifelse(Exited == "No",0,1)))

set.seed(1234)

idx <- sample(1:nrow(churn), size = round(0.5*nrow(churn)),
replace = FALSE)

training <- churn[idx, ]
test <- churn[-idx, ]
```

**Regressão Logística:**

```
model_RegLog <- glm(Exited ~., data = training, family =
binomial)

summary(model_RegLog)

prob <- predict(model_RegLog, newdata = test, type =
"response")

y_hat <- ifelse(prob > 0.5, 1, 0)

table(Predicted = y_hat, Observed = test$Exited)
```

**Árvore de Classificação:**

```
ctree <- tree(Exited ~ ., data = training)

y_hat_tree <- predict(ctree, newdata = test, type =
"class")

table(Predicted = y_hat_tree, Observed = test$Exited)

pr_tree <- predict(ctree, newdata = test, type =
"vector")[, 2]
```

**Random Forest:**

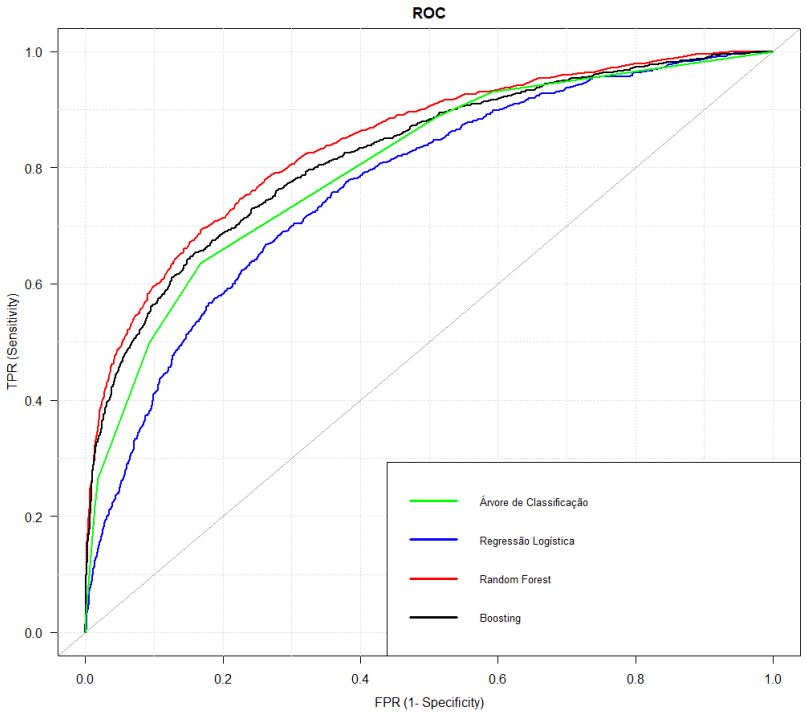
```
rf <- randomForest(Exited ~ .,data = training)

y_hat_rf <- predict(rf, newdata = test)

table(Predicted = y_hat_rf, Observed = test$Exited)

pr_rf <- predict(rf, newdata = test, type = "prob")[, 2]
```

**Boosting:**

```
boost <- adaboost(Exited ~ ., data = training, nIter = 50)

pred_boost <- predict(boost, newdata = test)

y_hat_boost <- pred_boost$class

table(Predicted = y_hat_boost, Observed = test$Exited)

pr_boost <- pred_boost$prob[, 2]
```

**Curvas ROC:**

```
roc_rlog <- roc(test$Exited, prob)
roc_tree <- roc(test$Exited, pr_tree)
roc_rf <- roc(test$Exited, pr_rf)
roc_boost <- roc(test$Exited, pr_boost)
```




**AUC:**

```
AUC <- data.frame(Modelo = c("Regressão Logística",
"Árvores de Classificação","Random Forest","Boosting"),
                          AUC = c(auc(roc_rlog),
                                  auc(roc_tree),
                                  auc(roc_rf),
                                  auc(roc_boost)))
AUC
```

**Acurácia:**

```
rl_acur <- mean(y_hat == test$Exited)
tree_acur <- mean(y_hat_tree == test$Exited)
rf_acur <- mean(y_hat_rf == test$Exited)
boost_acur <- mean(y_hat_boost == test$Exited)

acuracia <- data.frame(Modelo = c("Regressão Logística",
"Árvores de Classificação",
                          "Random Forest","Boosting"),
                   Acurácia = c(rl_acur, tree_acur, rf_acur,
                   boost_acur))

acuracia
```

| | Modelo | AUC |
|---|---|---|
| 1 | Regressão Logística | 0.7669193 |
| 2 | Arvores de Classificação | 0.8029938 |
| 3 | Random Forest | 0.8432743 |
| 4 | Boosting | 0.8229200 |

| | Modelo | Acurácia |
|---|---|---|
| 1 | Regressão Logística | 0.8134 |
| 2 | Árvores de Classificação | 0.8388 |
| 3 | Random Forest | 0.8606 |
| 4 | Boosting | 0.8500 |

**Análise dos resultados:**

Analisando os resultados, a Random Forest foi a que obteve a maior Area Under The Curve, seguida pelo Boosting. O resultado é dentro do esperado, dado que os dois modelos obtiveram performance melhor do que a regressão linear a as árvores de classificação. Isso se reflete na curva ROC, onde a Random Forest ficou mais próxima do canto superior esquerdo, onde a taxa de falsos positivos é zero e a de verdadeiros positivos é 1. Pela acurácia, a random forest perfomou melhor, acertando quase 86% dos dados do conjunto de teste. O boosting obteve acurácia de 85%. De qualquer modo, podemos notar que a "distância" de performance entre os modelos não é tão grande.

## b. USED_CARS

**Objetivo:** prever a variável price.

**Separação em dados de teste e de treinamento:**

```
price = used$price
used = model.matrix(price ~ ., data = used)
used = data.frame(used)
used$price = price
str(used)

set.seed(1234)

idx_used <- sample(1:nrow(used), size = round(0.7 *
nrow(used)), replace = FALSE)

training_used <- used[idx_used, ]
test_used <- used[-idx_used, ]
```

**Regressão Linear Múltipla:**

```
ols <- lm(price ~ ., data = training_used)

y_hat_ols <- predict(ols, newdata = test_used)

(RMSE_ols <- sqrt(mean((y_hat_ols - test_used$price)^2)))
```

**Árvore de Regressão:**

```
regtree <- tree(price ~ ., data = training_used)

summary(regtree)

plot(regtree, type = "uniform")
text(regtree, cex = 0.75)

y_hat_regtree <- predict(regtree, newdata = test_used)

(RMSE_regtree <- sqrt(mean((y_hat_regtree -
test_used$price)^2)))
```

**Random Forest:**

```
rf_used <- randomForest(price ~ ., data = training_used)

y_hat_rf_used <- predict(rf_used, newdata = test_used)
(RMSE_rf_used <- sqrt(mean((y_hat_rf_used -
test_used$price)^2)))
```

**Boosting:**

```
training_used$isOneOwner =
as.numeric(training_used$isOneOwner)
test_used$isOneOwner = as.numeric(test_used$isOneOwner)

boost_used <- gbm(price ~ ., data = training_used,
             distribution = "gaussian",
             n.trees = 1000,
             interaction.depth = 4,
             shrinkage = 0.001)

y_hat_boost_used <- predict(boost_used, newdata =
test_used, n.trees = 5000)

(RMSE_boost_used <- sqrt(mean((y_hat_boost_used -
test_used$price)^2)))
```

**Raiz Quadrada do Erro Quadrático Médio:**

```
RMSE <- data.frame(Modelo = c("OLS", "Árvore de Regressão"
                              ,"Random Forest","Boosting"),
                   RMSE = c(RMSE_ols,
                            RMSE_regtree,
                            RMSE_rf_used,
                            RMSE_boost_used))
RMSE
```

| | Modelo | RMSE |
|---|---|---|
| 1 | OLS | 6.764242 |
| 2 | Arvore de Regressão | 6.724560 |
| 3 | Random Forest | 4.208799 |
| 4 | Boosting | 5.677370 |

**Análise dos Resultados:**

Pelos resultados, é possível observar que a Random Forest foi a que obteve em média o menor RMSE, ou seja, é o modelo mais eficiente. A RMSE de aproximadamente 4,208 indica que está errando a previsão de preço do carro em aproximadamente 4208 dólares. Em seguida, o Boosting foi o que obteve em média a segunda melhor previsão, com uma RMSE de 5,677. A performance dos modelos segue o mesmo padrão observado no problema de churn.

**Referências:**


https://www.displayr.com/how-is-variable-importance-calculated-for-a-random-forest/

Stefan Wager, Trevor Hastie, and Bradley Efron. 2014. **Confidence Intervals for Random Forests: The Jackknife and The Infinitesimal Jackknife**. J. Mach. Learn. Res. 15, 1 (January 2014), 1625–1651.