Modelos

October 29, 2018

1 1 - Random Forest

Floresta Aleatória (Random Forest) é um algoritmo de aprendizagem de máquina supervisionado flexível criado em cima de *Árvores de Decisão* e técnicas de amostragem como *[Bootstrapping](https://en.wikipedia.org/wiki/Bootstrapping_(statistics)* que produz excelentes resultados na maioria das vezes, mesmo sem ajuste de hiperparâmetros e tratamento de dados faltantes. É também um dos algoritmos mais utilizados, devido à sua simplicidade e o fato de que pode ser utilizado tanto para tarefas de classificação como também de regressão.

Para mais informações sobre como funciona este algoritmo, seguem algumas referências:

- Wikipedia - Random forest
- An Introduction to Statistical Learning paginas 319-321
- StatQuest: Random Forests Parte 1 - Construindo, Usando e Avaliando

Iremos usar este algoritmo para gerar nosso primeiro modelo de classificação. Serão três modelos, um para prever se uma loja será, ou permanecerá, inadimplente no próximo mês, um outro modelo para prever a mesma situação dois meses à frente e um terceiro para prever inadimplência três meses à frente.

O modelo consome um conjunto de informações inciais e retorna 0 ou 1, que representa respectivamente adimplente e inadimplente.

```
In [1]: library(randomForest)
        library(caret)
        library(sqldf)
```

```
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.
Loading required package: lattice
Loading required package: ggplot2

Attaching package: ggplot2

The following object is masked from package:randomForest:

    margin

Loading required package: gsubfn
Loading required package: proto
```

```
Loading required package: RSQLite
```

```
In [2]: df = read.csv('data/Archive/ContasReceber&Faturamento_24-10.csv', sep=',', fileEncoding
```

```
In [3]: head(df, 3)
```

fant_shop_data	NrMes	date_type	Shopping	NmFantasia
REI DO MATE_BAN_201610	201610	2016-10-01	BAN	REI DO MATE
FICCUS PLUS SIZE_BAN_201610	201610	2016-10-01	BAN	FICCUS PLUS SIZE
ESTAÇÃO PARAÍSO JEANS_BAN_201610	201610	2016-10-01	BAN	ESTAÇÃO PARAÍSO JEAN

```
In [4]: dim(df)
```

1. 93857 2. 163

```
In [5]: # removendo valores na
        df = na.omit(df)
```

```
In [6]: dim(df)
```

1. 81756 2. 163

1.0.1 1.1 Preparando data sets que serão usados

```
In [7]: # removendo colunas que não serão usadas, que são:
        # 'fant_shop_data', 'NrMes', 'date_type', 'Shopping', 'NmFantasia',
        #  'GrupoAtividade', 'RamoAtividade', 'TipoAtividade'

        df2 = df[, -c(1:8)]
```

```
In [8]: dim(df2)
```

1. 81756 2. 155

```
In [9]: inad1 = df2[, -c(153,154)]
        head(inad1)
```

VlrFaturadoMes	VlrRecebidoMes	VlrRecebidoAntecipadoMes	VlrRecebidoAnteriorMes	VlrRecebid
8790.03	8790.03	0	0	0
7700.00	7700.00	0	0	0
15604.44	9761.31	0	0	2550
78196.86	78196.86	0	0	0
42395.66	42395.66	0	0	0
9593.67	9593.67	0	0	0

```
In [10]: # data set com idicação de
         # inadimplência para o próximo mês
         inad_1 = df2[, -c(153,154)]
         inad_1 = subset(inad_1, inad1 != -1)
```

```
# data set com idicação de
# inadimplência 2 mês à frente
inad_2 = df2[, -c(154, 155)]
inad_2 = subset(inad_2, inad2 != -1)

# data set com idicação de
# inadimplência 3 mês à frente
inad_3 = df2[, -c(153, 155)]
inad_3 = subset(inad_3, inad3 != -1)
```

1.0.2 1.2 Modelando para próximo mês

In [11]:
```
dim(inad_1)
```

1. 72793 2. 153

In [12]:
```
head(inad_1)
```

	VlrFaturadoMes	VlrRecebidoMes	VlrRecebidoAntecipadoMes	VlrRecebidoAnteriorMes	VlrI
3366	8756.21	8756.21	0	0	0
3367	25411.30	12000.00	0	0	0
3368	18418.58	18418.58	0	0	255(
3369	99201.24	99201.24	0	0	0
3370	41285.34	41285.34	0	0	0
3371	12175.12	12175.12	0	0	0

In [13]:
```
# gerando índices do conjunto de treino
train = sample (1: nrow(inad_1), nrow(inad_1) / 2)
```

In [14]:
```
length(train)
```

36396

In [15]:
```
inad_1['inad1'] = as.factor(inad_1$inad1)
```

In [16]:
```
inad_1['inad'] = as.factor(inad_1$inad)
```

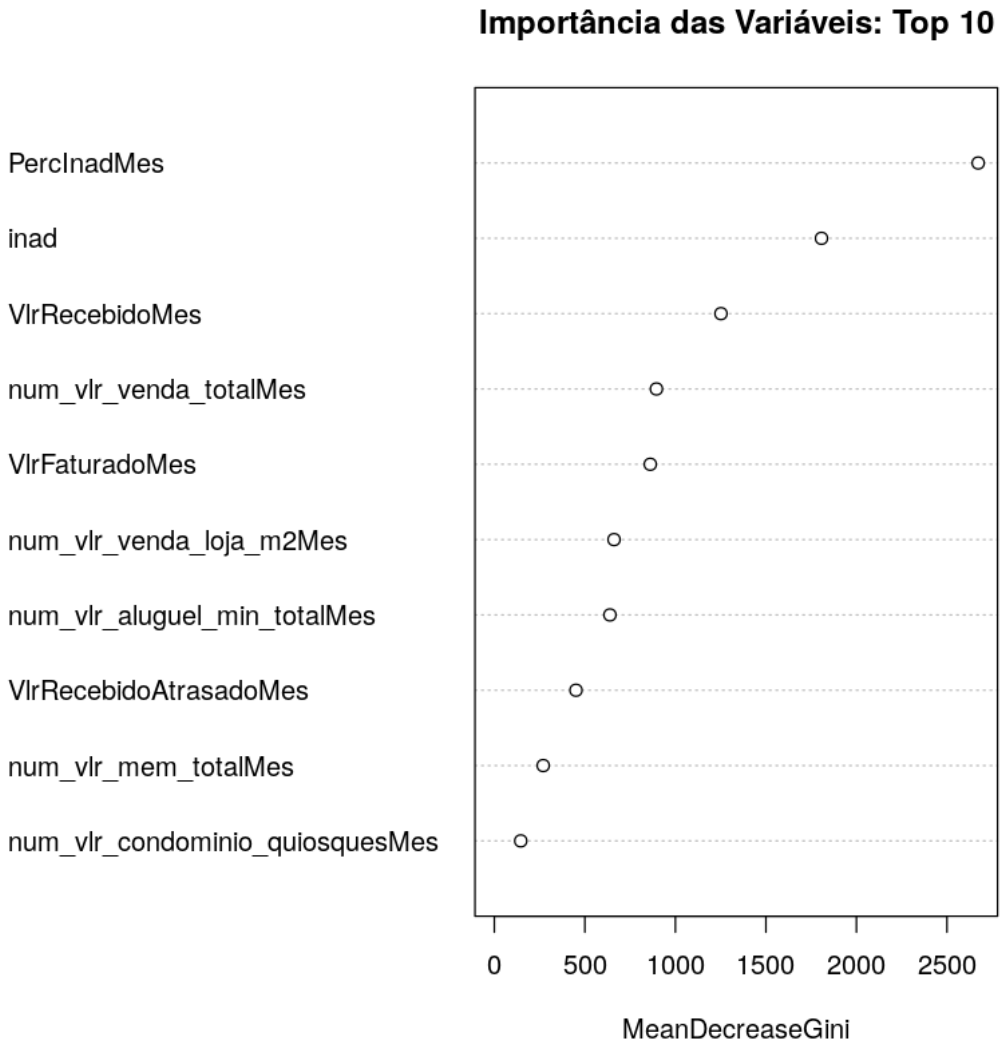
In [17]:
```
inad_1Modelo = randomForest(formula =  inad1 ~ ., data = inad_1, subset = train,
                                    mtry = 50, ntree = 100)
```

In [18]:
```
inad_1Modelo
```

```
Call:
 randomForest(formula = inad1 ~ ., data = inad_1, mtry = 50, ntree = 100,      subset = train)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 50
```

```
        OOB estimate of  error rate: 12.51%
Confusion matrix:
      0    1 class.error
0 26725 1627  0.05738572
1  2925 5119  0.36362506
```

In [19]:
```
varImpPlot(inad_1Modelo,
                    sort = T,
                    n.var=10,
                    main="Importância das Variáveis: Top 10")
```



In [21]:
```
teste_inad_1 = inad_1[-train, ]
```

In [26]:
```
# Fazendo previsão num data set teste
teste_inad_1[ 'previsao'] = predict(inad_1Modelo , newdata = teste_inad_1)
```

In [34]:
```
head(teste_inad_1, 3)
```

	VlrFaturadoMes	VlrRecebidoMes	VlrRecebidoAntecipadoMes	VlrRecebidoAnteriorMes	Vlr
3366	8756.21	8756.21	0	0	0
3368	18418.58	18418.58	0	0	255
3373	18698.34	18698.34	0	0	0

In [40]:
```
# Matriz de Confusão

confusionMatrix(data = teste_inad_1$previsao,
                reference = teste_inad_1$inad1,
                positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 26717  2914
         1  1651  5115

               Accuracy : 0.8746
                 95% CI : (0.8711, 0.878)
    No Information Rate : 0.7794
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6135
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.6371
            Specificity : 0.9418
         Pos Pred Value : 0.7560
         Neg Pred Value : 0.9017
             Prevalence : 0.2206
         Detection Rate : 0.1405
   Detection Prevalence : 0.1859
      Balanced Accuracy : 0.7894

       'Positive' Class : 1
```

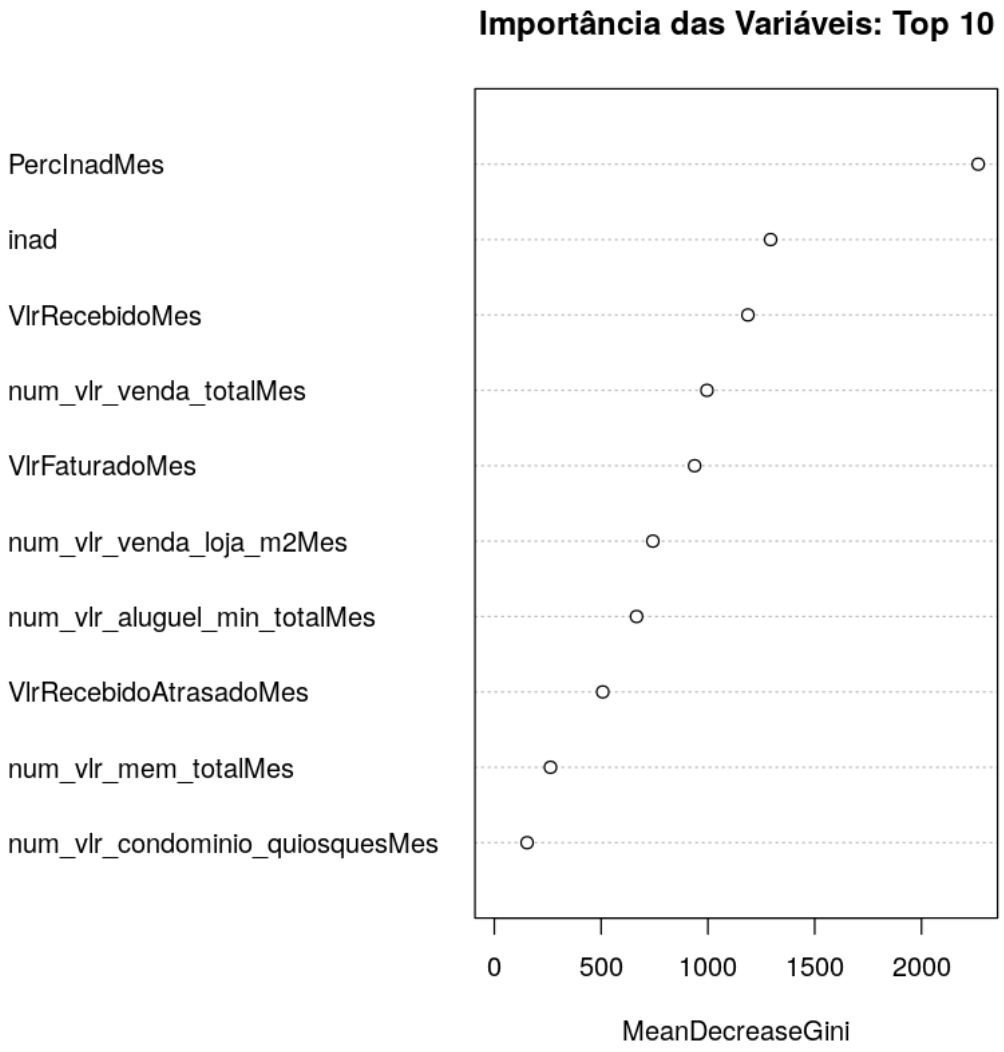
1.0.3 1.3 - Modelando para previsão com 2 meses de antecedência

In [22]:
```
# gerando índices do conjunto de treino
train2 = sample (1: nrow(inad_2), nrow(inad_2) / 2)
```

```
In [23]: length(train2)
```

34850

```
In [24]: inad_2['inad2'] = as.factor(inad_2$inad2)

In [25]: inad_2['inad'] = as.factor(inad_2$inad)

In [26]: inad_2Modelo = randomForest(formula =  inad2 ~ ., data = inad_2, subset = train2,
                                     mtry = 50, ntree = 100)

In [27]: inad_2Modelo


Call:
 randomForest(formula = inad2 ~ ., data = inad_2, mtry = 50, ntree = 100,      subset = train2)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 50

        OOB estimate of  error rate: 14.15%
Confusion matrix:
      0    1 class.error
0 25552 1677  0.06158875
1  3253 4368  0.42684687


In [50]: varImpPlot(inad_2Modelo,
                    sort = T,
                    n.var=10,
                    main="Importância das Variáveis: Top 10")
```




```
In [51]: teste_inad_2 = inad_2[-train2, ]
```

```
In [52]: # Fazendo previsão num data set teste
         teste_inad_2[ 'previsao'] = predict(inad_2Modelo , newdata = teste_inad_2)
```

```
In [53]: # Matriz de Confusão

         confusionMatrix(data = teste_inad_2$previsao,
                         reference = teste_inad_2$inad2,
                         positive = '1')
```

```
Confusion Matrix and Statistics
```

```
          Reference
Prediction     0     1
         0 25416  3335
         1  1628  4472

               Accuracy : 0.8576
                 95% CI : (0.8539, 0.8612)
    No Information Rate : 0.776
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5558
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.5728
            Specificity : 0.9398
         Pos Pred Value : 0.7331
         Neg Pred Value : 0.8840
             Prevalence : 0.2240
         Detection Rate : 0.1283
   Detection Prevalence : 0.1750
      Balanced Accuracy : 0.7563

       'Positive' Class : 1
```

1.0.4 1.4 Modelando para previsão com 3 meses de antecedência

In [28]:
```
# gerando índices do conjunto de treino
train3 = sample (1: nrow(inad_3), nrow(inad_3) / 2)
```

In [29]:
```
length(train3)
```

32442

In [30]:
```
inad_3['inad3'] = as.factor(inad_3$inad3)
inad_3['inad'] = as.factor(inad_3$inad)
```

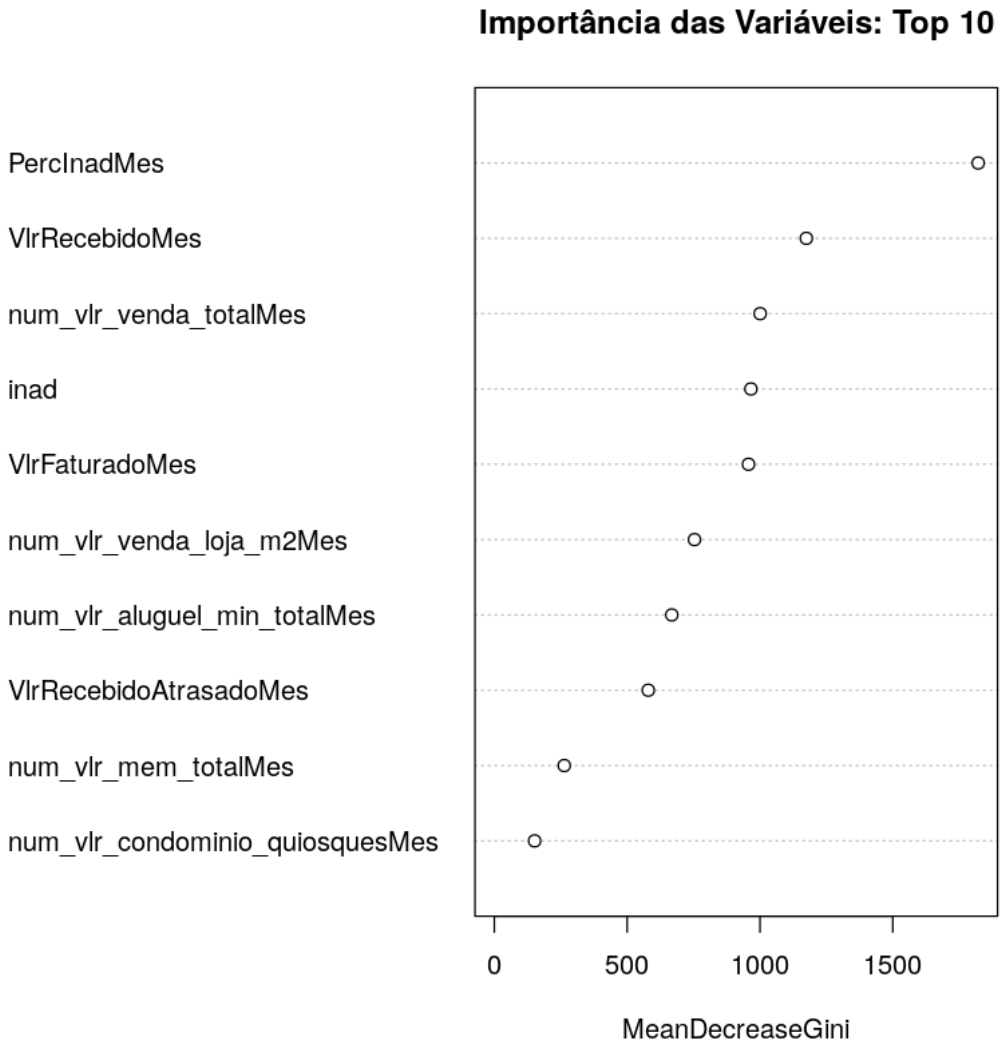
In [31]:
```
inad_3Modelo = randomForest(formula =  inad3 ~ ., data = inad_3, subset = train3,
                                    mtry = 50, ntree = 100)
```

In [58]:
```
inad_3Modelo
```

```
Call:
 randomForest(formula = inad3 ~ ., data = inad_3, mtry = 50, ntree = 100,      subset = train3)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 50
```

```
        OOB estimate of  error rate: 14.95%
Confusion matrix:
      0    1 class.error
0 23599 1639  0.06494175
1  3212 3992  0.44586341
```

```
In [59]: varImpPlot(inad_3Modelo,
                    sort = T,
                    n.var=10,
                    main="Importância das Variáveis: Top 10")
```

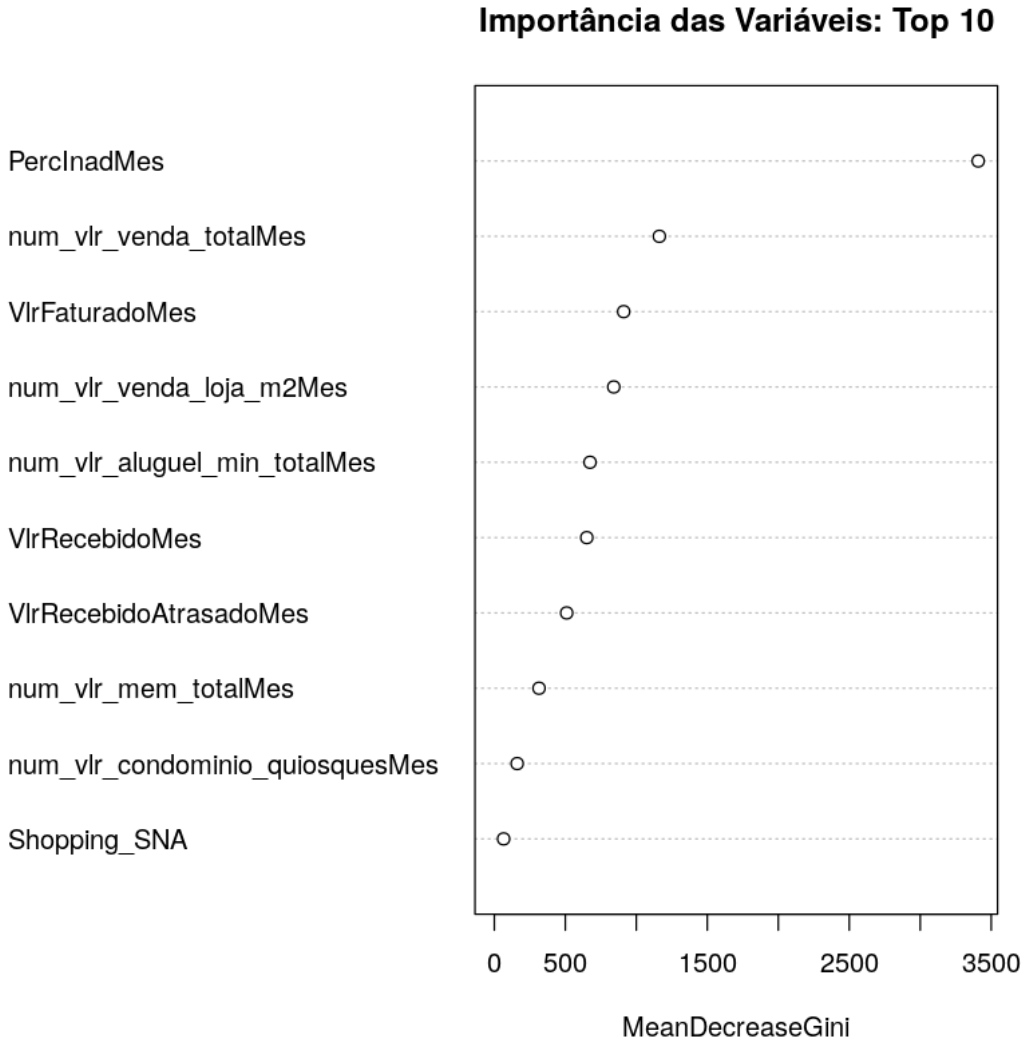



```
In [32]: teste_inad_3 = inad_3[-train3, ]
```

```
In [33]: # Fazendo previsão num data set teste
         teste_inad_3[ 'previsao'] = predict(inad_3Modelo , newdata = teste_inad_3)

In [34]: confusionMatrix(data = teste_inad_3$previsao,
                         reference = teste_inad_3$inad3,
                         positive = '1')

Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 23643  3112
         1  1710  3977

               Accuracy : 0.8514
                 95% CI : (0.8474, 0.8552)
    No Information Rate : 0.7815
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5314
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.5610
            Specificity : 0.9326
         Pos Pred Value : 0.6993
         Neg Pred Value : 0.8837
             Prevalence : 0.2185
         Detection Rate : 0.1226
   Detection Prevalence : 0.1753
      Balanced Accuracy : 0.7468

       'Positive' Class : 1



In [68]: # modelo de 3 meses usando todas as variáveis
         inad_3Modelo_td = randomForest(formula =  inad3 ~ ., data = inad_3, subset = train3,
                                        mtry = 152, ntree = 500)

In [69]: inad_3Modelo_td


Call:
 randomForest(formula = inad3 ~ ., data = inad_3, mtry = 152,      ntree = 500, subset = train3
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 152

        OOB estimate of  error rate: 15.1%
```

```
Confusion matrix:
      0    1 class.error
0 23517 1721  0.06819082
1  3178 4026  0.44114381
```

```
In [74]: varImpPlot(inad_3Modelo_td,
                    sort = T,
                    n.var=10,
                    main="Importância das Variáveis: Top 10")
```




```
In [71]: # Fazendo previsão num data set teste
         teste_inad_3[ 'previsao_td'] = predict(inad_3Modelo_td , newdata = teste_inad_3)
```

```
In [72]: confusionMatrix(data = teste_inad_3$previsao_td,
                         reference = teste_inad_3$inad3,
                         positive = '1')

Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 23572  3254
         1  1688  3928

               Accuracy : 0.8477
                 95% CI : (0.8437, 0.8516)
    No Information Rate : 0.7786
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5207
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.5469
            Specificity : 0.9332
         Pos Pred Value : 0.6994
         Neg Pred Value : 0.8787
             Prevalence : 0.2214
         Detection Rate : 0.1211
   Detection Prevalence : 0.1731
      Balanced Accuracy : 0.7400

       'Positive' Class : 1
```

1.0.5 1.5 - Prevendo Inadimplência para Julho, Agosto e Setembro de 2018

```
In [41]: df_201806 = subset(df, df$NrMes == 201806)
```

```
In [42]: head(df_201806)
```

	fant_shop_data	NrMes	date_type	Shopping	NmFantasia
76737	REI DO MATE_BAN_201806	201806	2018-06-01	BAN	REI DO MATE
76738	ESTAÇÃO PARAÍSO JEANS_BAN_201806	201806	2018-06-01	BAN	ESTAÇÃO PARAÍS
76739	BURGER KING_BAN_201806	201806	2018-06-01	BAN	BURGER KING
76740	RADICAL VEST_BAN_201806	201806	2018-06-01	BAN	RADICAL VEST
76741	PROJETO SPORT_BAN_201806	201806	2018-06-01	BAN	PROJETO SPORT
76742	BRILHO BRASIL_BAN_201806	201806	2018-06-01	BAN	BRILHO BRASIL

```
In [43]: dim(df_201806)
```

1. 3589 2. 163

```
In [44]: # removendo as colunas
         # inad1, inad2 e inad3
         df_201806 = df_201809[,-c(161, 162, 163)]
```

```
In [45]: head(df_201806, 3)
```

	fant_shop_data	NrMes	date_type	Shopping	NmFantasia
76737	REI DO MATE_BAN_201806	201806	2018-06-01	BAN	REI DO MATE
76738	ESTAÇÃO PARAÍSO JEANS_BAN_201806	201806	2018-06-01	BAN	ESTAÇÃO PARAÍS
76739	BURGER KING_BAN_201806	201806	2018-06-01	BAN	BURGER KING

```
In [46]: df_201806['inad'] = as.factor(df_201806$inad)
```

```
In [47]: # Fazendo previsão usando os modelos
         # Random Forest já gerados. As colunas
         # 1 à 8 não são consideradas como input.
         df_201806['pred_julho'] = predict(inad_1Modelo , newdata = df_201806[ , -c(1:8)])
         df_201806['pred_agosto'] = predict(inad_2Modelo, newdata = df_201806[ , -c(1:8)])
         df_201806['pred_setembro'] = predict(inad_3Modelo, newdata = df_201806[ , -c(1:8)])
```

```
In [50]: head(df_201806[, c('NrMes', 'Shopping', 'NmFantasia','pred_julho', 'pred_agosto', 'pre
```

	NrMes	Shopping	NmFantasia	pred_julho	pred_agosto	pred_setembro
76737	201806	BAN	REI DO MATE	0	0	0
76738	201806	BAN	ESTAÇÃO PARAÍSO JEANS	0	1	0
76739	201806	BAN	BURGER KING	0	0	0
76740	201806	BAN	RADICAL VEST	0	1	1
76741	201806	BAN	PROJETO SPORT	0	0	0
76742	201806	BAN	BRILHO BRASIL	0	0	0

```
In [52]: write.csv(df_201806[, c('NrMes', 'Shopping', 'NmFantasia','pred_julho', 'pred_agosto'
                  ,'data/df_2018-06-modelo-random-forest.csv', sep=',', fileEncoding="utf-8",
```

```
Warning message in write.csv(df_201806[, c("NrMes", "Shopping", "NmFantasia", "pred_julho", :
attempt to set 'sep' ignored
```

2 2 - Regressão Logística

A regressão logística é uma técnica estatística de aprendizagem supervisionada que tem como objetivo produzir, a partir de um conjunto de observações, um modelo que permita a predição de valores tomados por uma variável categórica, frequentemente binária, a partir de uma série de variáveis explicativas contínuas e/ou categóricas.

Para mais informações este tipo de classificador, seguem algumas referências:

- Logistic regression - Wikipedia
- An Introduction to Statistical Learning paginas 131-137
- StatQuest: Logistic Regression

2.0.1 2.2 Modelando para próximo mês

```
In [84]: glm_inad_1 = glm ( inad1 ~., data = inad_1 ,
                            family = binomial, subset = train)
```

```
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
In [87]: glm_probs_inad_1 = predict(glm_inad_1, teste_inad_1 ,type = "response")
```

```
Warning message in predict.lm(object, newdata, se.fit, scale = 1, type = ifelse(type == :
prediction from a rank-deficient fit may be misleading
```

```
In [88]: glm_probs_inad_1[1:10]
```

3366 0.11609447677638 **3368** 0.164629906932179 **3373** 0.100377345632062 **3374** 0.139059552249848 **3375** 0.022408881272981 **3376** 0.121350816976011 **3378** 0.615965752472994 **3380** 0.074504034377811 **3382** 0.0921249004946192 **3384** 0.0120453403559639

```
In [81]: contrasts(inad_1$inad1)
```

	1
0	0
1	1

```
In [89]: dim(teste_inad_1)
```

1. 36397 2. 154

```
In [90]: glm_pred_inad1 = rep("0" , 36397)
         glm_pred_inad1[glm_probs_inad_1 > 0.5] = '1'
```

```
In [94]: teste_inad_1['pred1_logit'] = as.factor(glm_pred_inad1)
```

```
In [95]: head(teste_inad_1)
```

	VlrFaturadoMes	VlrRecebidoMes	VlrRecebidoAntecipadoMes	VlrRecebidoAnteriorMes	Vlr
3366	8756.21	8756.21	0	0.00	0
3368	18418.58	18418.58	0	0.00	255
3373	18698.34	18698.34	0	0.00	0
3374	11164.98	11164.98	0	0.00	0
3375	39388.14	39388.14	0	0.00	0
3376	28078.85	0.00	0	28078.85	0

```
In [96]: confusionMatrix(data = teste_inad_1$pred1_logit,
                        reference = teste_inad_1$inad1,
                        positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 26680  2907
         1  1688  5122

               Accuracy : 0.8738
                 95% CI : (0.8703, 0.8771)
    No Information Rate : 0.7794
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6117
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.6379
            Specificity : 0.9405
         Pos Pred Value : 0.7521
         Neg Pred Value : 0.9017
             Prevalence : 0.2206
         Detection Rate : 0.1407
   Detection Prevalence : 0.1871
      Balanced Accuracy : 0.7892

       'Positive' Class : 1
```

2.0.2 2.3 Modelando para previsão com 2 meses de antecedência

```
In [98]: glm_inad_2 = glm ( inad2 ~., data = inad_2 ,
                            family = binomial, subset = train2)
```

```
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
In [99]: dim(teste_inad_2)
```

1. 34851 2. 154

```
In [100]: glm_probs_inad_2 = predict(glm_inad_2, teste_inad_2 ,type = "response")
          glm_pred_inad2 = rep("0" , 34851)
          glm_pred_inad2[glm_probs_inad_2 > 0.5] = '1'
          teste_inad_2['pred2_logit'] = as.factor(glm_pred_inad2)
```

```
Warning message in predict.lm(object, newdata, se.fit, scale = 1, type = ifelse(type == :
prediction from a rank-deficient fit may be misleading
```

```
In [101]: confusionMatrix(data = teste_inad_2$pred2_logit,
                          reference = teste_inad_2$inad2,
                          positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 25390  3400
         1  1654  4407

               Accuracy : 0.855
                 95% CI : (0.8512, 0.8587)
    No Information Rate : 0.776
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5468
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.5645
            Specificity : 0.9388
         Pos Pred Value : 0.7271
         Neg Pred Value : 0.8819
             Prevalence : 0.2240
         Detection Rate : 0.1265
   Detection Prevalence : 0.1739
      Balanced Accuracy : 0.7517

       'Positive' Class : 1
```

2.0.3 2.4 Modelando para previsão com 3 meses de antecedência

```
In [102]: glm_inad_3 = glm ( inad3 ~., data = inad_3 ,
                             family = binomial, subset = train3)
```

```
Warning message:
glm.fit: algorithm did not convergeWarning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
In [103]: dim(teste_inad_3)
```

1. 32442 2. 155

```
In [104]: glm_probs_inad_3 = predict(glm_inad_3, teste_inad_3 ,type = "response")
          glm_pred_inad3 = rep("0" , 32442)
          glm_pred_inad3[glm_probs_inad_3 > 0.5] = '1'
          teste_inad_3['pred3_logit'] = as.factor(glm_pred_inad3)
```

```
Warning message in predict.lm(object, newdata, se.fit, scale = 1, type = ifelse(type == :
prediction from a rank-deficient fit may be misleading
```

```
In [106]: confusionMatrix(data = teste_inad_3$pred3_logit,
                          reference = teste_inad_3$inad3,
                          positive = '1')

Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 23588  3390
         1  1672  3792

               Accuracy : 0.844
                 95% CI : (0.84, 0.8479)
    No Information Rate : 0.7786
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.505
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.5280
            Specificity : 0.9338
         Pos Pred Value : 0.6940
         Neg Pred Value : 0.8743
             Prevalence : 0.2214
         Detection Rate : 0.1169
   Detection Prevalence : 0.1684
      Balanced Accuracy : 0.7309

       'Positive' Class : 1
```

2.0.4 3. 1 - Variáveis que não foram inclusas na modelagem

int_cd_periodo_mes
- int_cd_shopping
- int_cd_tenant_mix
- int_cd_contrato
- num_vlr_aluguel_min_loja_bruto
- num_vlr_aluguel_min_loja_desc
- num_vlr_aluguel_min_loja_caren
- num_vlr_aluguel_perc
- num_vlr_aluguel_laje
- num_vlr_aluguel_outros
- num_vlr_condominio_especif
- num_vlr_condominio_laje
- num_vlr_condominio_ressarcim
- num_vlr_fpp

num_vlr_fpp_outros
num_vlr_fpp_ressarcim
num_vlr_fpp_cota_empreendedor
num_vlr_fpp_total
num_vlr_custo_ocup_com
num_vlr_custo_ocup_com_esp
num_vlr_custo_ocup_com_m2
num_vlr_custo_ocup_com_esp_m2
num_vlr_grocc_com
num_vlr_grocc_com_esp
num_prc_perfil_fat_com
num_prc_perfil_fat_com_esp
num_nr_abl
num_vlr_condominio_outros
num_vlr_fpp_complementar
num_vlr_cota_extra
num_vlr_tx_administracao
int_fl_possui_venda_maa
int_fl_possui_alug_maa
int_fl_alterou_abl
num_nr_abl_vago
num_nr_abl_ocupado

3 4 - Conclusão

Tanto os modelos Random Forest como Logístico demostraram desempenho similares.

Ainda é preciso "afinar" os modelos ajustando os parâmetros de modelagem, no caso do Random Forest encontrar o número de variáveis aleatórias *mtry* e o número de árvores *ntree* e no caso na Regressão Logística encontrar o limiar (usamos limiar igual 0.5) que resultem em modelos com maior "sensibilidade".

Também é preciso fazer **Análise de Componentes Principais** para reduzir a complexidade dos modelos.