

Análise Preditiva Avançada

Adriano Abelaira Paz

13/04/2020

Análise Preditiva Avançada

Case

A multinacional de varejo Waldata está querendo expandir a sua presença na américa latina e por isso decide firmar uma parceria com a FGV para desenvolver um modelo preditivo do valor de vendas. Além disso a companhia decide apostar em um segundo modelo de ‘targetads’ tornando mais efetiva as campanhas de marketing. Assim a rede varejista pretende melhorar suas projeções de fluxo de caixa e otimizar a distribuição de seus produtos por departamentos.

Para desenvolver seu modelo você irá realizar as seguintes tarefas:

1. Importar os datasets RETAIL e MARKETING para o ambiente R.
2. Fazer uma exploração detalhada dos dados. (Distribuições, valores faltantes etc..)
3. Dividir as bases em 70% para treino e 30% para teste do modelo. (Utilize sempre seed(314)).
4. Testar modelos de classificação para as campanhas de marketing: 1. Regressão Logística, Árvores de Decisão, SVM , Redes Neurais e Algoritmo Genético para featureselection.
5. Testar modelos de regressão para o valor de vendas das lojas: 1. Regressão Linear, Árvore de Decisão, e Redes Neurais.
6. Validar a performance dos modelos (R2 & Matriz de Confusão).
7. Fazer o “scoring” dos modelos para os dados nas respectivas bases de teste.

Importando, analisando e tratando o *Data Set* de Marketing.

Importando os dados de Marrketing e ajustando o *Data Set*.

Descrição dos dados:

Variável	Descrição
AGE	Idade
JOB	Profissão
MARITAL_STATUS	Estado civil
EDUCATION	Educação
DEFAULT	Contas Atrasadas ?
HOUSING	Hipoteca ?
LOAN	Empréstimo Pessoal ?
CONTACT	Tipo de Contato
MONTH	Último Mês de Contato
DAY_OF_WEEK	Último Dia da Semana de Contato
DURATION	Duração do Último Contato em segundos

Variável	Descrição
CAMPAIGN	Tipo de Campanha de Marketing
PDAYS	Número de Dias desde Último Contato (-1: Não Houve Contato)
PREVIOUS	Número de Contatos Antes da Campanha
POUTCOME	Resultado da Última Campanha
EMP_VAR_RATE	Taxa de Desemprego da Região
CONS_PRICE_IDX	IGPM
CONS_CONF_IDX	Índice de Confiança do Consumidor
SUBSCRIBED	Aderiu ao Serviço ?

```

market <- fread(file = '../data//Marketing.csv')

market[, ':='(JOB = as.factor(market$JOB),
             MARITAL_STATUS = as.factor(market$MARITAL_STATUS),
             EDUCATION = factor(market$EDUCATION,
                                 ordered = FALSE,
                                 levels = c('unknown', 'illiterate',
                                           'basic_4y', 'basic_6y', 'basic_9y',
                                           'high_school', 'professional_course',
                                           'university_degree')),
             DEFAULT = as.factor(market$DEFAULT),
             HOUSING = as.factor(market$HOUSING),
             LOAN = as.factor(market$LOAN),
             CONTACT = as.factor(market$CONTACT),
             MONTH = as.factor(market$MONTH),
             DAY_OF_WEEK = as.factor(market$DAY_OF_WEEK),
             POUTCOME = as.factor(market$POUTCOME),
             EMP_VAR_RATE = as.numeric(str_replace_all(string = market$EMP_VAR_RATE,
                                                       pattern = '_', replacement = '.')),
             CONS_PRICE_IDX = as.numeric(market$CONS_PRICE_IDX),
             CONS_CONF_IDX =
               as.numeric(str_replace_all(string = market$CONS_CONF_IDX,
                                         pattern = '_', replacement = '.')),
             SUBSCRIBED = as.factor(market$SUBSCRIBED))]

str(market)

## Classes 'data.table' and 'data.frame': 41188 obs. of 19 variables:
## $ AGE : int 56 57 37 40 56 45 59 41 24 25 ...
## $ JOB : Factor w/ 12 levels "admin","blue-collar",...: 4 8 8 1 8 8 1 2 10 8 ...
## $ MARITAL_STATUS: Factor w/ 4 levels "divorced","married",...: 2 2 2 2 2 2 2 3 3 ...
## $ EDUCATION : Factor w/ 8 levels "unknown","illiterate",...: 3 6 6 4 6 5 7 1 7 6 ...
## $ DEFAULT : Factor w/ 3 levels "no","unknown",...: 1 2 1 1 1 2 1 2 1 1 ...
## $ HOUSING : Factor w/ 3 levels "no","unknown",...: 1 1 3 1 1 1 1 3 3 ...
## $ LOAN : Factor w/ 3 levels "no","unknown",...: 1 1 1 1 3 1 1 1 1 1 ...
## $ CONTACT : Factor w/ 2 levels "cellular","telephone": 2 2 2 2 2 2 2 2 2 2 ...
## $ MONTH : Factor w/ 10 levels "apr","aug","dec",...: 7 7 7 7 7 7 7 7 7 7 ...
## $ DAY_OF_WEEK : Factor w/ 5 levels "fri","mon","thu",...: 2 2 2 2 2 2 2 2 2 2 ...
## $ DURATION : int 261 149 226 151 307 198 139 217 380 50 ...
## $ CAMPAIGN : int 1 1 1 1 1 1 1 1 1 ...
## $ PDAYS : int -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
## $ PREVIOUS : int 0 0 0 0 0 0 0 0 0 ...
## $ POUTCOME : Factor w/ 3 levels "failure","nonexistent",...: 2 2 2 2 2 2 2 2 2 2 ...

```

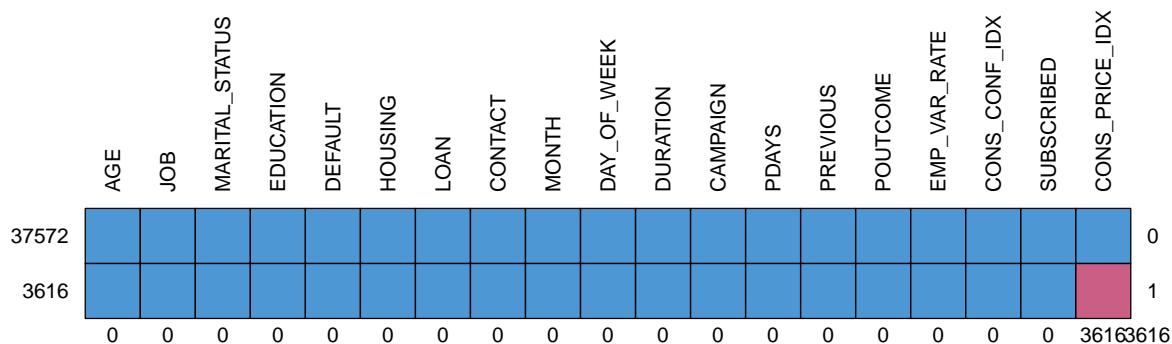
```

## $ EMP_VAR_RATE : num 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
## $ CONS_PRICE_IDX: num 94 94 94 94 94 ...
## $ CONS_CONF_IDX : num -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
## $ SUBSCRIBED    : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
## - attr(*, ".internal.selfref")=<externalptr>

```

Verificando a existência de NA's no *Data Set* de Marketing (**Market**).

```
# ----- Para o Dataset MARKET -----
na_market <- md.pattern(market, rotate.names = TRUE)
```



O *Data Set* possui NA's somente na variável **CONS_PRICE_IDX**.

```

na_market_ord <- setnames(as.data.table(na_market),
                           'V20',
                           'Qtd.NA')[, Qtd.Linhas := rownames(na_market)][,.SD, keyby=Qtd.NA]
na_market_ord[3,
              c('AGE', 'JOB', 'MARITAL_STATUS', 'EDUCATION', 'DEFAULT', 'HOUSING', 'LOAN',
                'CONTACT', 'MONTH', 'DAY_OF_WEEK', 'DURATION', 'CAMPAIGN', 'PDAYS',
                'PREVIOUS', 'POUTCOME', 'EMP_VAR_RATE',
                'CONS_PRICE_IDX', 'CONS_CONF_IDX', 'SUBSCRIBED)] / nrow(market) * 100.0

##      AGE JOB MARITAL_STATUS EDUCATION DEFAULT HOUSING LOAN CONTACT MONTH
## 1: 0 0 0 0 0 0 0 0 0 0
##      DAY_OF_WEEK DURATION CAMPAIGN PDAYS PREVIOUS POUTCOME EMP_VAR_RATE
## 1: 0 0 0 0 0 0 0
##      CONS_PRICE_IDX CONS_CONF_IDX SUBSCRIBED
## 1: 8.779256 0 0

```

Cerca de 8% das linhas estão com NA's na coluna **CONS_PRICE_IDX**.

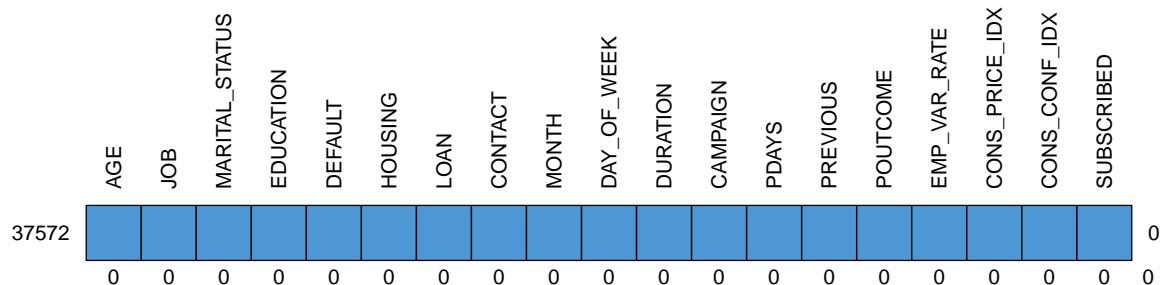
Como percentualmente os valores de NA's são poucos em relação ao total de linhas, optou-se por retirá-los.

```
# Primeiro dataset para Market
market2 <- na.omit(market, cols = 'CONS_PRICE_IDX')
```

Com a eliminação de algumas linhas, gerou um *Data Set* de Marketing sem NA's.

```
md.pattern(market2, rotate.names = TRUE)
```

```
##  /\      /\
## {   ---' }
## {   0   0  }
## ==> V <== No need for mice. This data set is completely observed.
##  \  \|/  /
##    `-----'
```



```
##          AGE  JOB MARITAL_STATUS EDUCATION DEFAULT  HOUSING  LOAN CONTACT MONTH
## 37572    1    1            1        1       1       1       1       1     1
##          0    0            0        0       0       0       0       0     0
##          DAY_OF_WEEK DURATION CAMPAIGN PDAYS PREVIOUS POUTCOME EMP_VAR_RATE
## 37572           1         1        1       1       1       1       1
##          0         0        0       0       0       0       0
##          CONS_PRICE_IDX CONS_CONF_IDX SUBSCRIBED
## 37572           1             1       1 0
##          0             0       0 0
```

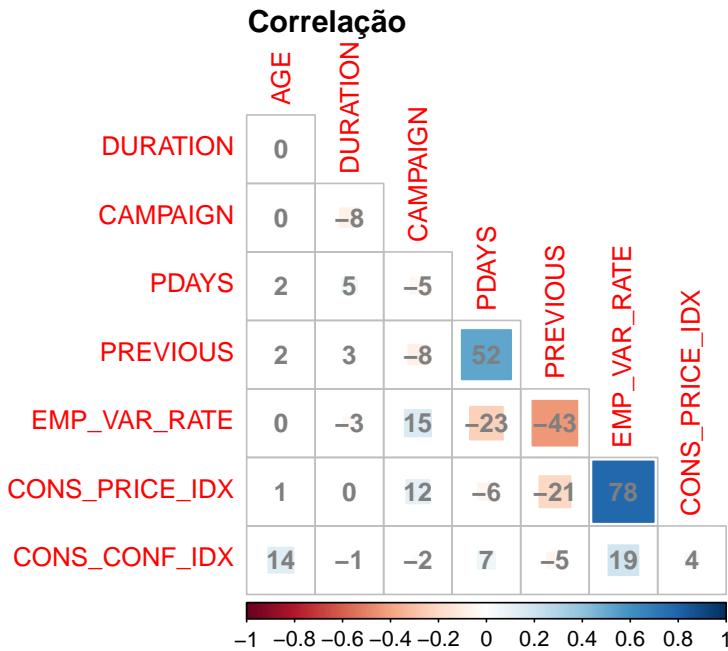
Verificando a correlação no *Data Set* tratado de Marketing.

```
corrplot(cor(market2[,-c('JOB','MARITAL_STATUS','EDUCATION','DEFAULT',
                      'HOUSING','LOAN','CONTACT',
                      'MONTH','DAY_OF_WEEK','POUTCOME','SUBSCRIBED')]),
        method = 'square',
        type = 'lower',
        diag = FALSE,
        title = 'Correlação',
```

```

mar = c(1,1,1,1),
addCoefAsPercent = TRUE,
addCoef.col = 'gray50',
number.digits = 0)

```



O Data Set possui uma alta correlação positiva (>70%) entre **EMP_VAR_RATE** e **CONS_PRICE_IDX**, cerca de 78%. Possui uma correlação mediana positiva (entre 50% e 70%) entre **PDAYS** e **PREVIOUS**, cerca de 52%. Possui uma baixa correlação positiva (entre 15% e 50%) entre **EMP_VAR_RATE** e **CONS_CONF_IDX**, cerca de 19% e **CAMPNIGN** e **EMP_VAR_RATE**, cerca de 15%. Não possui variáveis com alta (>70%) correlação negativa ou mediana (entre 50% e 70%). Possui baixa correlação negativa (entre 15% e 50%) entre **PREVIOUS** e **EMP_VAR_RATE**, cerca de 43%, **PDAYS** e **EMP_VAR_RATE**, cerca de 23% e entre **PREVIOUS** e **CONS_PRICE_IDX**, cerca de 21%.

Dado as correlações envolvendo **CONS_PRICE_IDX**: 1. **CONS_PRICE_IDX** (Índice de preço) e **EMP_VAR_RATE** (Taxa de desemprego) (78%);

2. **CONS_PRICE_IDX** e **PREVIOUS** (Número de Contatos Antes da Campanha) (-21%);

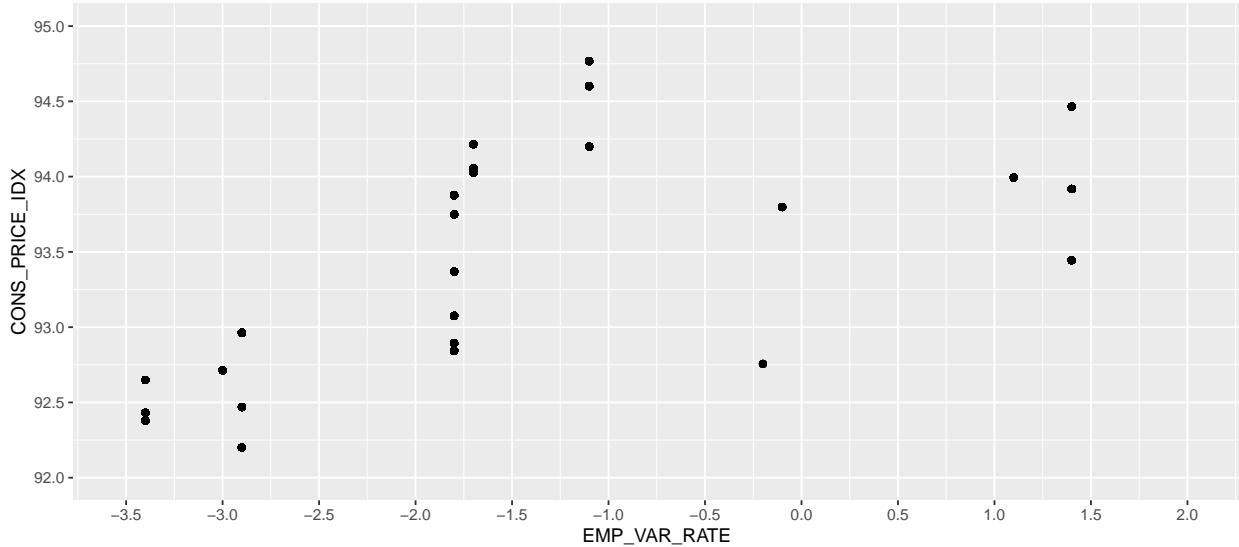
3. **CONS_PRICE_IDX** e **CAMPAIGN** (Tipo de Campanha de Marketing) (12%).

Como **PREVIOUS** e **CAMPAIGN** possuem baixa correlação com **CONS_PRICE_IDX** e não possuem uma relação aparente com a variável **CONS_PRICE_IDX**, serão desconsideradas. Faremos uma análise somente com a variável **EMP_VAR_RATE**, pois **CONS_PRICE_IDX** (IGPM/Inflação) e **EMP_VAR_RATE** (Taxa de Desemprego) são medidas que se relacionam.

```

market2[,c('CONS_PRICE_IDX', 'EMP_VAR_RATE')] %>%
  ggplot(aes(y = CONS_PRICE_IDX, x = EMP_VAR_RATE)) +
  geom_point() +
  scale_y_continuous(limits = c(92,95),
                     breaks = c(92,92.5,93,93.5,94,94.5,95)) +
  scale_x_continuous(limits = c(-3.5,2),
                     breaks = c(-3.5,-3,-2.5,-2,-1.5,-1,-0.5,0,0.5,1,1.5,2))

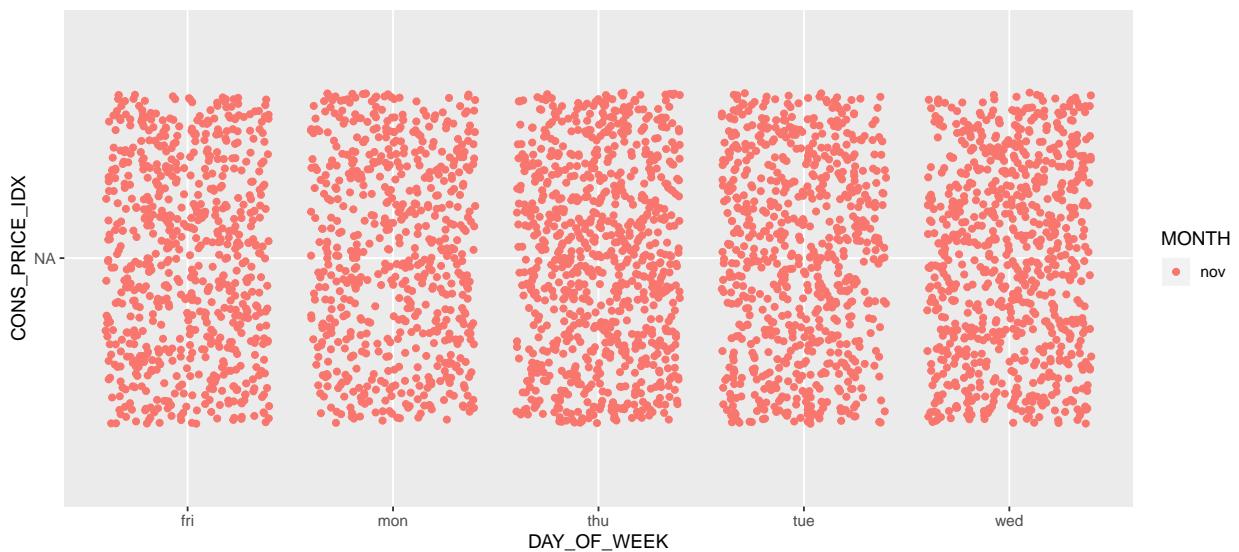
```



Não se percebe qualquer relação linear entre as duas variáveis. Inclusive, para o mesmo valor de **EMP_VAR_RATE** encontramos diferentes valores para **CONS_PRICE_IDX**. Talvez para valores médios de **CONS_PRICE_IDX** tenhamos uma relação linear com a média de **EMP_VAR_RATE**. Portanto, foi desconsiderada a possibilidade de imputação de **CONS_PRICE_IDX** a partir de **EMP_VAR_RATE**.

Considerando que **CONS_PRICE_IDX** seja uma variável que capta flutuações da inflação, logo torna-se dependente do tempo em que foi apurada. Faremos uma análise de **CONS_PRICE_IDX**, **MONTH** e **DAY_OF_WEEK**, pois foi considerado que o valor de **CONS_PRICE_IDX** foi obtido no mês e dia da semana de contato com o consumidor.

```
market[is.na(CONS_PRICE_IDX),c('CONS_PRICE_IDX','MONTH','DAY_OF_WEEK')] %>%
  ggplot(aes(x = DAY_OF_WEEK, y = as.factor(CONS_PRICE_IDX))) +
  geom_jitter(aes(color = MONTH)) +
  labs(y = 'CONS_PRICE_IDX')
```



Percebe-se que os valores da variável **CONS_PRICE_IDX** preenchidos com **NA**'s, estão todos no mês de NOVEMBRO, sendo distribuídos de forma igualitária entre SEGUNDA, TERÇA, QUARTA, QUINTA e

SEXTA.

Fazendo uma análise no *Data Set* direcionado para as variáveis **MONTH** igual a NOVEMBRO, teremos:

```
market[MONTH == 'nov', c('CONS_PRICE_IDX', 'MONTH')] %>%
  ggplot(aes(y = MONTH, x = as.factor(CONS_PRICE_IDX))) +
  geom_jitter() +
  labs(x = 'CONS_PRICE_IDX')
```



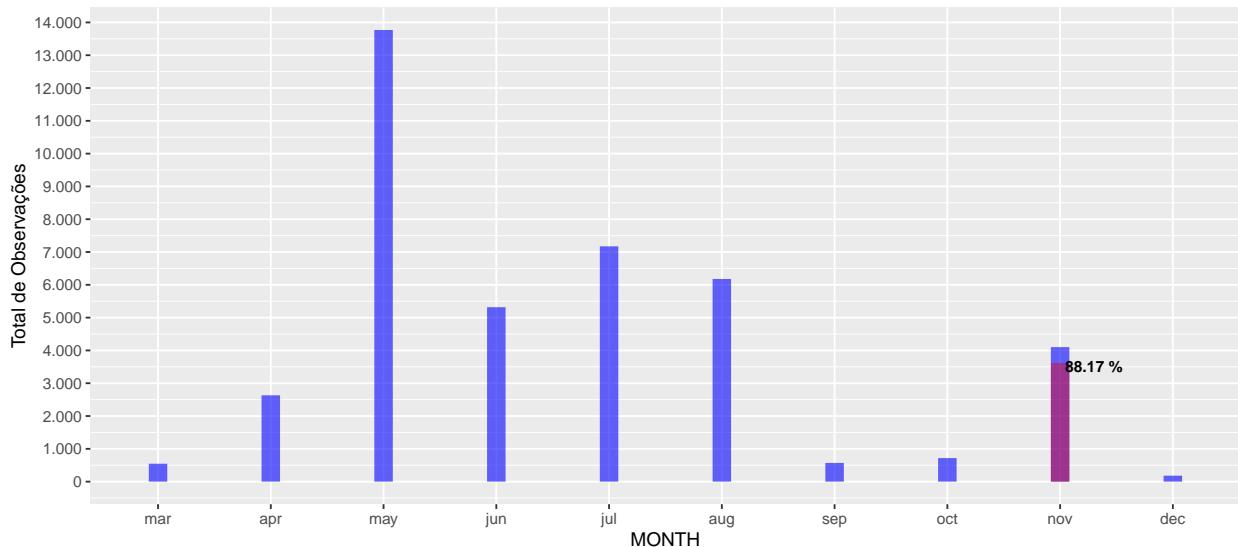
Mostrando que para o mês de NOVEMBRO temos dois valores conhecidos para a variável **CONS_PRICE_IDX**: 92,649 e 94,767. Considerando que **CONS_PRICE_IDX** seja uma representação de medida de inflação mensal, que não varia de acordo com os dias da semana dentro do mesmo mês. Os dois valores constatados no mês de NOVEMBRO nos leva a concluir que eles representam anos diferentes para a variável **CONS_PRICE_IDX**. Como não temos a informação de ano no *Data Set*, não temos como imputar valor em **CONS_PRICE_IDX** através da variável **MONTH**, pois nos falta informação do ano.

```
market_mes <- data.table()
market_mes[, ':='(ID = c(3,4,5,6,7,8,9,10,11,12),
                 MES = factor(c('mar', 'apr', 'may', 'jun', 'jul',
                               'aug', 'sep', 'oct', 'nov', 'dec'))),
            QTD = c(nrow(market[MONTH == 'mar', c('CONS_PRICE_IDX', 'MONTH')]),
                    nrow(market[MONTH == 'apr', c('CONS_PRICE_IDX', 'MONTH')]),
                    nrow(market[MONTH == 'may', c('CONS_PRICE_IDX', 'MONTH')]),
                    nrow(market[MONTH == 'jun', c('CONS_PRICE_IDX', 'MONTH')]),
                    nrow(market[MONTH == 'jul', c('CONS_PRICE_IDX', 'MONTH')]),
                    nrow(market[MONTH == 'aug', c('CONS_PRICE_IDX', 'MONTH')]),
                    nrow(market[MONTH == 'sep', c('CONS_PRICE_IDX', 'MONTH')]),
                    nrow(market[MONTH == 'oct', c('CONS_PRICE_IDX', 'MONTH')]),
                    nrow(market[MONTH == 'nov', c('CONS_PRICE_IDX', 'MONTH')]),
                    nrow(market[MONTH == 'dec', c('CONS_PRICE_IDX', 'MONTH')])))]
as.data.frame(market_mes) %>% ggplot(aes(x = as.factor(ID), y = QTD)) +
  geom_point(colour = 'red', shape = -9, size = 3) +
  geom_segment(aes(x = as.factor(market_mes$ID),
                  xend = as.factor(market_mes$ID), y = 0, yend = QTD),
               colour = 'blue', size = 5, alpha = 0.6) +
  geom_segment(aes(x = '11', xend = '11', y = 0,
```

```

yend = nrow(market[is.na(CONS_PRICE_IDX) &
                  MONTH == 'nov', c('CONS_PRICE_IDX', 'MONTH')]]),
colour = 'red', size = 5, alpha = 0.05) +
scale_x_discrete(labels = market_mes$MES) +
scale_y_continuous(breaks = c(0, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000,
                             10000, 11000, 12000, 13000, 14000),
                   labels = c('0', '1.000', '2.000', '3.000', '4.000', '5.000', '6.000', '7.000', '8.000', '9.000',
                             '10.000', '11.000', '12.000', '13.000', '14.000')) +
annotate("text",
        label = paste(round(nrow(market[is.na(CONS_PRICE_IDX) &
                                         MONTH == 'nov', c('CONS_PRICE_IDX', 'MONTH')]]) /
                      nrow(market[MONTH == 'nov',
                                         c('CONS_PRICE_IDX', 'MONTH')])) * 100, 2), "%"),
        x = 9.3, y = 3500, size = 3, colour = "black", fontface = 'bold') +
labs(x = 'MONTH', y = 'Total de Observações')

```



Uma solução a adotar nesses casos seria a retirada de toda a variável **CONS_PRICE_IDX** ao invés de retirar as linhas que possuem NA's nessa variável. Essa solução possivelmente produzirá menos viés, dado que os valores de NA's, que estão localizados todos no mês de NOVEMBRO, representam 88,17% do conjunto desse mês.

```

# Segundo dataset para Market
market3 <- copy(market)
market3[, CONS_PRICE_IDX := NULL]

```

Dividindo os dois *Data Set* de Marketing tratados em *Data Set* para treinamento e teste. O primeiro *Data Set* obtido retirando as linhas em que a coluna **CONS_PRICE_IDX** apresentava valor ausente, ficou dividido em *Data Set* **market2_train** para treinamento e **market2_test** para teste. O segundo *Data Set* obtido toda a coluna **CONS_PRICE_IDX**, ficou dividido em *Data Set* **market3_train** para treinamento e ***market3_test*** para teste.

```

set.seed(314)
i_market <- createDataPartition(market2$SUBSCRIBED, p = 0.7, list = FALSE)
market2_train <- market2[i_market,]
market2_test <- market2[-i_market,]

```

```

set.seed(314)
i_market <- createDataPartition(market3$SUBSCRIBED, p = 0.7, list = FALSE)
market3_train <- market3[i_market,]
market3_test <- market3[-i_market,]

```

Com o objetivo de avaliar qual o melhor *Data Set* de Marketing para ser utilizado, foi utilizado uma modelagem por *Regressão Logística*, que é um modelo robusto e com boa performance para *Data Sets* grandes como esse.

A métrica que a análise se balizará será o AUC, que quantifica a Área sob a Curva ROC, e o LogLoss, que quantifica a Acuracidade de um classificador penalizando classificações falsas. Para o AUC, quanto maior melhor, já o LogLoss, quanto menor melhor.

```

cv <- trainControl(method = "repeatedcv", number = 10,
                    repeats = 5, savePredictions = TRUE,
                    summaryFunction = twoClassSummary, classProbs = TRUE)

# -----
# Usando o Dataset Market2 (sem as linhas com NA's da CONS_PRICE_IDX)
# ----- Rregressão Logística -----
mod_market2_rl <- train(SUBSCRIBED ~ .,
                         data = market2_train,
                         method = "glm",
                         metric = "ROC",
                         trControl = cv)
pred_market2_rl <- predict(mod_market2_rl, newdata = market2_test, type = 'prob')

# Performance sem as linhas com NAs da variável CONS_PRICE_IDX
print(paste('AUC:',
            AUC(pred_market2_rl$yes, as.integer(market2_test$SUBSCRIBED)-1)))

```

```
## [1] "AUC: 0.936833713293166"
```

```

print(paste('LogLos:',
            LogLoss(pred_market2_rl$yes, as.integer(market2_test$SUBSCRIBED)-1)))

```

```
## [1] "LogLos: 0.212690511373347"
```

```

cutoff <- optimalCutoff(actuals = as.integer(market2_test$SUBSCRIBED)-1,
                         predictedScores = pred_market2_rl$yes)
pred_market2 <- ifelse(pred_market2_rl$yes > cutoff, 'yes', 'no')
pred_market2 <- as.factor(pred_market2)
(mc_market2_rl <- caret::confusionMatrix(data = pred_market2, positive = 'yes',
                                             reference = market2_test$SUBSCRIBED,
                                             mode = 'everything'))

```

```

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    no    yes
##       no     9502   600

```

```

##      yes 434 735
##
##          Accuracy : 0.9083
##             95% CI : (0.9028, 0.9135)
##      No Information Rate : 0.8816
##      P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 0.5357
##
##      Mcnemar's Test P-Value : 2.878e-07
##
##          Sensitivity : 0.55056
##          Specificity : 0.95632
##      Pos Pred Value : 0.62874
##      Neg Pred Value : 0.94061
##          Precision : 0.62874
##          Recall : 0.55056
##          F1 : 0.58706
##          Prevalence : 0.11845
##          Detection Rate : 0.06521
##      Detection Prevalence : 0.10372
##          Balanced Accuracy : 0.75344
##
##      'Positive' Class : yes
##
```

```

# -----
# Usando o Dataset Market3 (sem a variável CONS_PRICE_IDX)
# ----- Regressão Logística -----
mod_market3_rl <- train(SUBSCRIBED ~ .,
                         data = market3_train,
                         method = "glm",
                         metric = "ROC",
                         trControl = cv)
pred_market3_rl <- predict(mod_market3_rl, newdata = market3_test, type = 'prob')

# Performance sem a variável CONS_PRICE_IDX
print(paste('AUC:', 
            AUC(pred_market3_rl$yes, as.integer(market3_test$SUBSCRIBED)-1)))

```

```

## [1] "AUC: 0.928248261289822"

print(paste('LogLoss:', 
            LogLoss(pred_market3_rl$yes, as.integer(market3_test$SUBSCRIBED)-1)))

```

```

## [1] "LogLoss: 0.217032958284501"

cutoff <- optimalCutoff(actuals = as.integer(market3_test$SUBSCRIBED)-1,
                         predictedScores = pred_market3_rl$yes)
pred_market3 <- ifelse(pred_market3_rl$yes > cutoff, 'yes', 'no')
pred_market3 <- as.factor(pred_market3)
(mc_market3_rl <- caret::confusionMatrix(data = pred_market3, positive = 'yes',

```

```

        reference = market3_test$SUBSCRIBED,
        mode = 'everything'))
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    no     yes
##       no    10678    823
##       yes     286    569
##
##                 Accuracy : 0.9102
##                 95% CI : (0.9051, 0.9152)
##       No Information Rate : 0.8873
##       P-Value [Acc > NIR] : < 2.2e-16
##
##                 Kappa : 0.4602
##
## McNemar's Test P-Value : < 2.2e-16
##
##                 Sensitivity : 0.40876
##                 Specificity : 0.97391
##       Pos Pred Value : 0.66550
##       Neg Pred Value : 0.92844
##                 Precision : 0.66550
##                 Recall : 0.40876
##                 F1 : 0.50645
##                 Prevalence : 0.11266
##                 Detection Rate : 0.04605
##       Detection Prevalence : 0.06920
##       Balanced Accuracy : 0.69134
##
##       'Positive' Class : yes
##

```

A retirada das linhas com **CONS_PRICE_IDX** com NA's se mostrou mais eficaz que a retirada de toda a variável. O *AUC* da regressão com o primeiro Dataset (**Market2**) foi de 93,68% e o *LogLoss* foi de 0,2126905, otimizando o *Cutoff* foi possível obter um *F1 Score* de 58,71%. Com o segundo *Data Set* (**Market3**) que foi criado retirando toda a variável **CONS_PRICE_IDX**, o *AUC* foi de 92,82%, um pouco inferior, e um *LogLoss* de 0,217033, superior ao modelo anterior. Otimizando o *Cutoff* foi possível obter *F1 Score* de 50,64%.

Nota-se um *AUC* ligeiramente maior do primeiro modelo e um *LogLoss* menor, evidenciando a capacidade de distinguir clientes que irão converter dos que não irão maior em relação ao segundo modelo. Outro ponto mostrado foi a métrica *F1 Score*, que no primeiro modelo ficou sensivelmente melhor que no segundo modelo. Como essa métrica é uma junção da *Sensibilidade (Recall)* e *Precisão*, ela mostra a capacidade de distinguir clientes que irão converter, minimizando tanto os falsos positivo quanto os falsos negativos. Essa evidência do *F1 Score* se soma ao do *AUC* e *LogLos* que apontam que o melhor *Data Set* para Marketing é o utilizado no primeiro modelo, que é o *Data Set* (**Market2**).

```

# Setando o melhor Dataset para Market
dt_market_train <- market2_train
dt_market_test <- market2_test

```

Importando, analisando e tratando o *Data Set* de Faturamento das Lojas.

Importando os dados de Faturamento das Lojas e ajustando o *Data Set*.

Descrição dos dados:

Variável	Descrição
STORE	ID da Loja
DATE	Datetime
TEMPERATURE	Temperatura em Fahrenheit
FUEL_PRICE	Preço Galão Combustível em USD
MARKDOWN1	Redução de Preço
MARKDOWN2	Redução de Preço
MARKDOWN3	Redução de Preço
MARKDOWN4	Redução de Preço
MARKDOWN5	Redução de Preço
CPI	Consumer Price Index - Inflação
UNEMPLOYMENT	Taxa de Desemprego
ISHOLIDAY	Feriado ?
WEEKLY_SALES	Valor Total de Vendas na Semana em USD

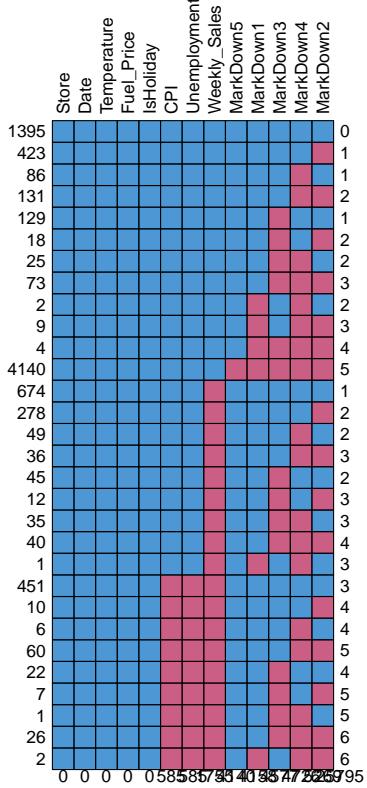
```
# Importando os dados
retail <- fread(file = '../data/Retail.csv')

# Tratando os data sets
retail[,':='](IsHoliday = as.factor(retail$IsHoliday),
              Weekly_Sales = as.double(retail$Weekly_Sales),
              Date = dmy(retail$Date))]
str(retail)

## Classes 'data.table' and 'data.frame':  8190 obs. of  13 variables:
## $ Store      : int  1 1 1 1 1 1 1 1 1 ...
## $ Date       : Date, format: "2010-02-05" "2010-02-12" ...
## $ Temperature : num  42.3 38.5 39.9 46.6 46.5 ...
## $ Fuel_Price  : num  2.57 2.55 2.51 2.56 2.62 ...
## $ MarkDown1   : num  NA NA NA NA NA NA NA NA NA ...
## $ MarkDown2   : num  NA NA NA NA NA NA NA NA NA ...
## $ MarkDown3   : num  NA NA NA NA NA NA NA NA NA ...
## $ MarkDown4   : num  NA NA NA NA NA NA NA NA NA ...
## $ MarkDown5   : num  NA NA NA NA NA NA NA NA NA ...
## $ CPI         : num  211 211 211 211 211 ...
## $ Unemployment: num  8.11 8.11 8.11 8.11 8.11 ...
## $ IsHoliday    : Factor w/ 2 levels "FALSE","TRUE": 1 2 1 1 1 1 1 1 1 ...
## $ Weekly_Sales: num  1643691 1641957 1611968 1409728 1554807 ...
## - attr(*, ".internal.selfref")=<externalptr>
```

Verificando a existência de NA's nas variáveis do *Data Set* de Faturamento das Lojas (**Retail**).

```
# ----- Para o Dataset RETAIL -----
na_retail <- md.pattern(retail, rotate.names = TRUE)
```



O *Data Set* possui NA's nas colunas **CPI**, **Unemployment**, **Weekly_Sales**, **MarkDown1**, **MarkDown2**, **MarkDown3**, **MarkDown4**, **MarkDown5**.

```

na_retail_ord <-
  setnames(as.data.table(na_retail),
          'V14', 'Qtd.NA')[, Qtd.Linhas:=rownames(na_retail)][,.SD, keyby=Qtd.NA]
na_retail_ord[31,
  c('Store', 'Date', 'Temperature', 'Fuel_Price', 'IsHoliday', 'CPI',
    'Unemployment', 'Weekly_Sales',
    'MarkDown1', 'MarkDown2', 'MarkDown3',
    'MarkDown4', 'MarkDown5')] / nrow(retail) * 100.0

##      Store Date Temperature Fuel_Price IsHoliday      CPI Unemployment
## 1:      0     0            0        0        0 7.142857    7.142857
##      Weekly_Sales MarkDown1 MarkDown2 MarkDown3 MarkDown4 MarkDown5
## 1: 21.42857  50.76923  64.33455  55.88523  57.70452  50.54945

```

Cerca de 50% das linhas estão sem os *MarkDown1* até *MarkDown5*. Não dá para descartá-los. Cerca de 7% não possuem *CPI* (Inflação) e *Unemployment* (Taxa de Desemprego). Cerca de 21% não possui *Weekly_Sales* (Valor Total das Vendas na Semana).

Para as variáveis *CPI* e *Unemployment* preencheremos com o último valor disponível na linha anterior do *Data Set* ordenado por loja e data. Assim o último valor disponível da loja será a melhor estimativa para os valores faltantes na datas seguintes da mesma loja.

```

retail2 <- copy(retail)

setorder(retail2, Store, Date)

# Preenchendo os NA's das colunas MarkDown's1-5 (Redução de Preço) com zero
setnafill(retail2, type = c('const'), fill = 0,
          cols = c('MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5'))

setnafill(retail2, type = c('locf'), cols = c('CPI', 'Unemployment'))

retail2 <- na.omit(retail, cols = 'Weekly_Sales')

na_mark <- is.na(retail2$MarkDown1)
retail2[which(na_mark), MarkDown1 := 0]
na_mark <- is.na(retail2$MarkDown2)
retail2[which(na_mark), MarkDown2 := 0]
na_mark <- is.na(retail2$MarkDown3)
retail2[which(na_mark), MarkDown3 := 0]
na_mark <- is.na(retail2$MarkDown4)
retail2[which(na_mark), MarkDown4 := 0]
na_mark <- is.na(retail2$MarkDown5)
retail2[which(na_mark), MarkDown5 := 0]

```

Para as variáveis *MarkDown1* até *MarkDown5*, optou-se por preenchê-las com zero, pois são variáveis que mostram a redução de preço nas lojas e a melhor alternativa, levando-se em consideração a relação de custo e benefício, seria considerá-las como zero, ou seja, sem redução de preço.

Para a variável *Weekly_Sales*, como é a variável *target* para o modelo de previsão de faturamento, qualquer método de imputação pode criar um viés no modelo. Portanto, a melhor alternativa nessa caso será o descarte das linhas com NA's. Como isso perderemos 21% do *Data Set* de faturamento.

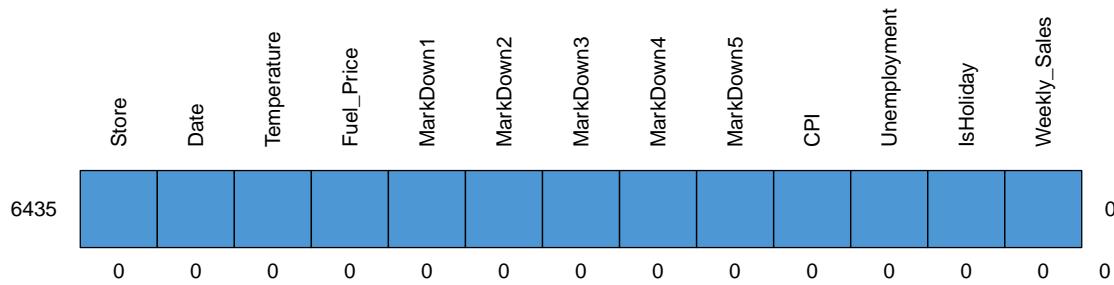
Com a imputação dos NA's e a eliminação de algumas linhas, gerou um *Data Set* de Faturamento das Lojas sem NA's.

```

md.pattern(retail2, rotate.names = TRUE)

##  /\      /\
## { `---' }
## { 0 0  }
## ==> V <== No need for mice. This data set is completely observed.
##  \ \|\ / /
##    `-----'

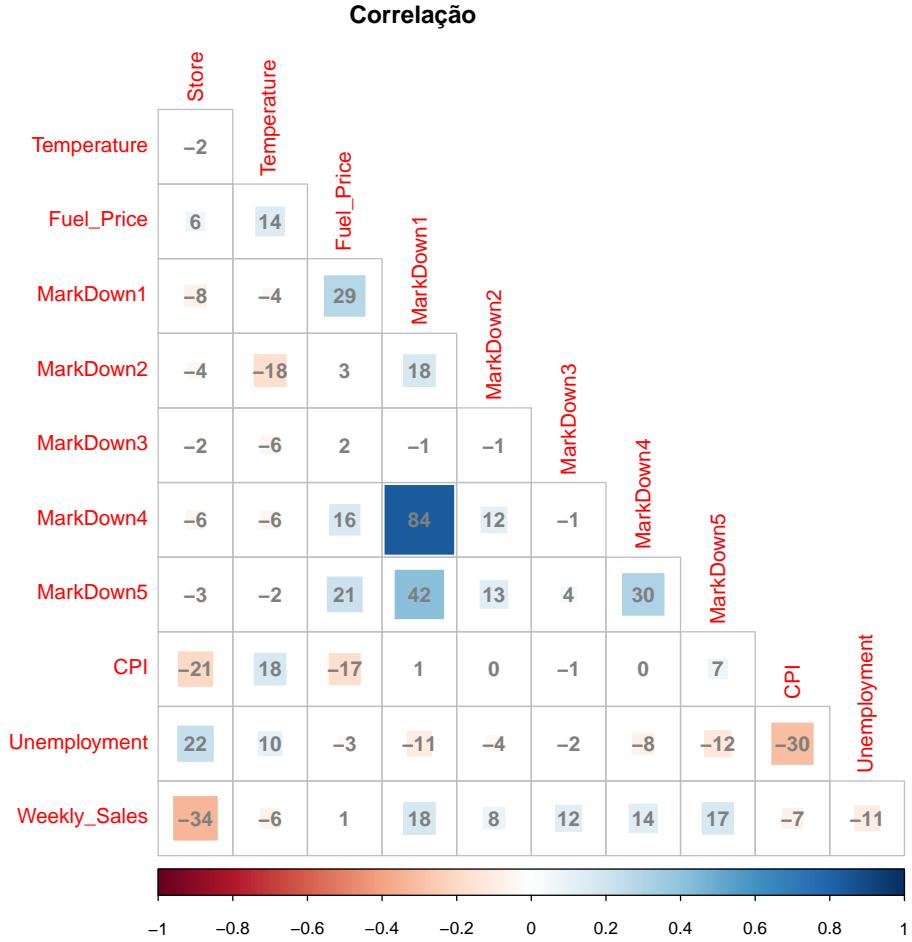
```



```
##      Store Date Temperature Fuel_Price MarkDown1 MarkDown2 MarkDown3 MarkDown4
## 6435     1     1           1         1       1       1       1       1       1
##          0     0           0         0       0       0       0       0       0
##      MarkDown5 CPI Unemployment IsHoliday Weekly_Sales
## 6435     1     1           1         1       1     0
##          0     0           0         0       0     0
```

Verificando a correlação no *Data Set* tratado de Faturamento das Lojas.

```
corrplot(cor( retail2[,-c('Date','IsHoliday')]),  
        method = 'square',  
        type = 'lower',  
        diag = FALSE,  
        title = 'Correlação',  
        mar = c(1,1,1,1),  
        addCoefasPercent = TRUE,  
        addCoef.col = 'gray50',  
        number.digits = 0)
```



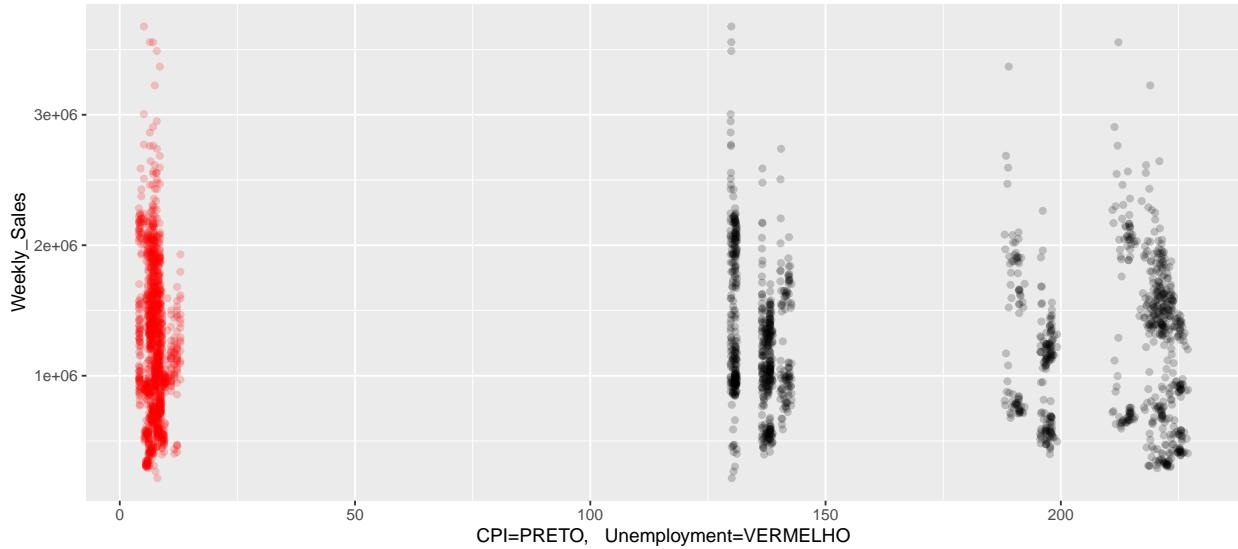
O *Data Set* possui uma alta correlação positiva ($> 70\%$) entre **MarkDown1** e **MarkDown4**, cerca de 83%. Não possui variáveis com uma correlação positiva mediana (entre 50% e 70%). Possui as seguintes variáveis com uma baixa correlação positiva (entre 15% e 50%): . **Fuel_Price** e **MarkDown1**, cerca de 26%; . **Unemployment** e **Store**, cerca de 22%; . **MarkDown1** e **Weekly_Sales**, cerca de 20%; . **MarkDown1** e **MarkDown5**, cerca de 18%; . **Temperature** e **CPI**, cerca de 16%; . **MarkDown1** e **MarkDown2**, cerca de 16%; . **Fuel_Price** e **MarkDown4**, cerca de 15%; . **MarkDown4** e **Weekly_Sales**, cerca de 15%. Não possui nenhum par de variáveis com alta ($>70\%$) correlação negativa e nem mediana (entre 50% e 70%). Possui as seguintes variáveis com baixa correlação negativa (entre 15% e 50%): . **Store** e **Weekly_Sales**, cerca de 33%; . **CPI** e **Unemployment**, cerca de 30%; . **Temperature** e **MorkDown2**, cerca de 22%; . **CPI** e **Store**, cerca de 21%; . **CPI** e **Fuel_Price**, cerca de 19%.

A alta correlação positiva entre **MarkDown1** e **MarkDown4** pode ter sido influenciada pelo método de imputação de NA's nessas variáveis. Como também ele não faz muito sentido, uma correlação entre redução de preços, não foi dado muita atenção.

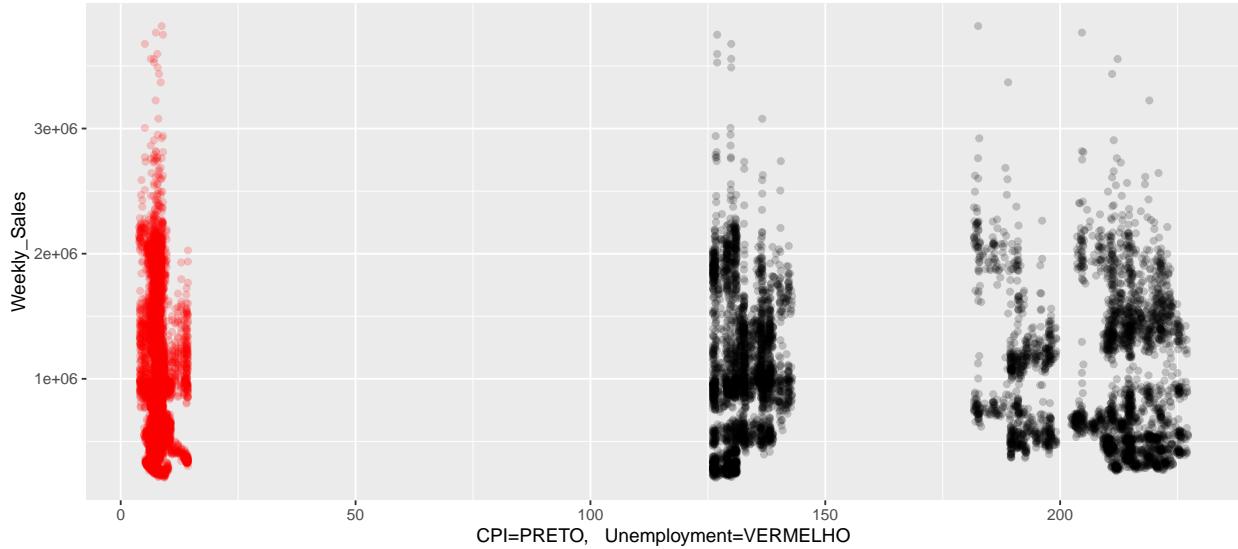
Como a variável *target* é **Weekly_Sales**, as correlações entre ela e as demais são de interesse nessa análise preliminar. Temos uma correlação média com **markDown1** e **MarkDown4** e uma correlação baixa com **Store**.

Avaliando mais detalhadamente o método de imputação para as variáveis **CPI** e **Unemployment**, percebe-se que houve um acréscimo grande nos grupos em que os dados estavam originalmente.

```
ggplot(data = na.omit(retail), aes(y = Weekly_Sales)) +
  geom_point(aes(x = CPI), color = 'black', alpha = 0.2) +
  geom_point(aes(x = Unemployment), color = 'red', alpha = 0.2) +
  labs(x = 'CPI=PRETO, Unemployment=VERMELHO')
```



```
ggplot(data = retail2, aes(y = Weekly_Sales)) +
  geom_point(aes(x = CPI), color = 'black', alpha = 0.2) +
  geom_point(aes(x = Unemployment), color = 'red', alpha = 0.2) +
  labs(x = 'CPI=PRETO, Unemployment=VERMELHO')
```

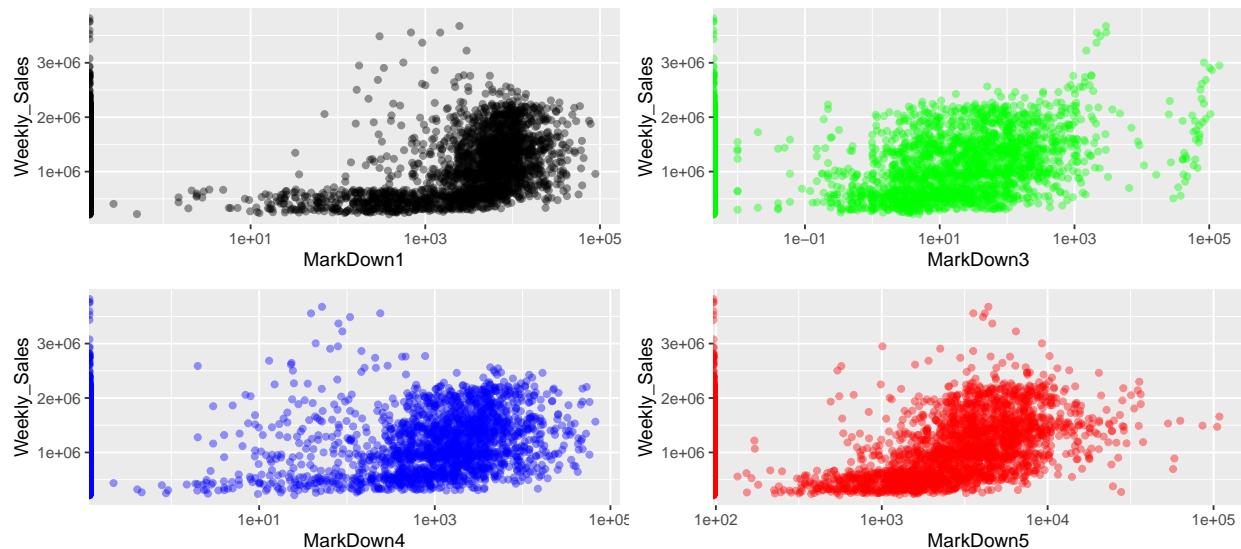


O gráfico a seguir mostra a dependência das variáveis **MarkDown1**, **MarkDown3**, **MarkDown4** e **MarkDown5**, com a variável **Weekly_Sales**, evidenciando o impacto que ocorreu com a imputação dos valores ausentes com valores zero. A concentração dos valores zero podem estar distorcendo demasiadamente a relação com a variável *target* **Weekly_Sales**.

```

g1 <- ggplot(data = retail2, aes(y = Weekly_Sales)) +
  geom_point(aes(x = MarkDown1), color = 'black', alpha = 0.4) +
  scale_x_log10()
g2 <- ggplot(data = retail2, aes(y = Weekly_Sales)) +
  geom_point(aes(x = MarkDown3), color = 'green', alpha = 0.4) +
  scale_x_log10()
g3 <- ggplot(data = retail2, aes(y = Weekly_Sales)) +
  geom_point(aes(x = MarkDown4), color = 'blue', alpha = 0.4) +
  scale_x_log10()
g4 <- ggplot(data = retail2, aes(y = Weekly_Sales)) +
  geom_point(aes(x = MarkDown5), color = 'red', alpha = 0.4) +
  scale_x_log10()
grid.arrange(g1 , g2 ,
             g3 , g4 ,
             ncol=2, nrow=2)

```



Para tentar corrigir essas distorções entre as variáveis **MarkDown1** até **MarkDown5**, **CPI** e **Unemployment**, optou-se por um tratamento de imputação de valores ausentes através do KNN-5.

```

retail3 <- na.omit(retail, cols = 'Weekly_Sales')

target <- retail3$Weekly_Sales

# Ajustando o processamento paralelo em CORES (NÚCLEOS)
cl <- makePSOCKcluster(cores)
registerDoParallel(cl)

retail3 <- knnImputation(retail3[!names(retail3) %in% "Weekly_Sales", with=F],
                         k = 5,
                         meth = 'weighAvg')

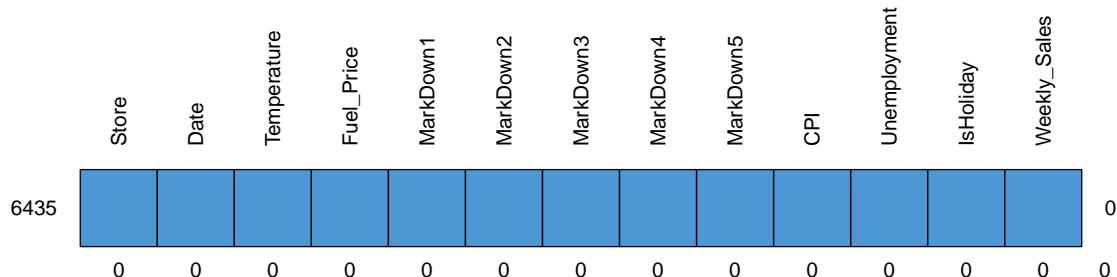
# Parando o processamento paralelo
stopCluster(cl)

retail3[,Weekly_Sales := target]

```

```
# Verificação de existência de NA no Data Set
md.pattern( retail3, rotate.names = TRUE)
```

```
##  /\      \
## { `---' }
## { 0 0  }
## ==> V <== No need for mice. This data set is completely observed.
##  \ \|/ /
##    `-----'
```

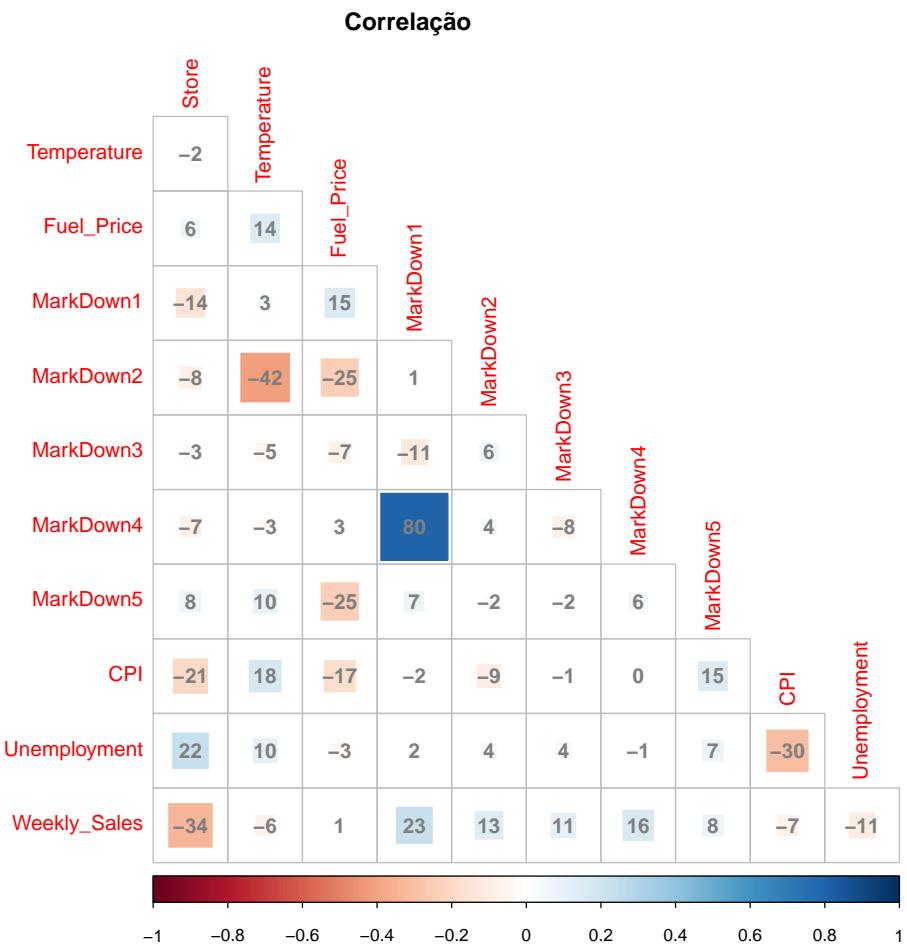


```
##      Store Date Temperature Fuel_Price MarkDown1 MarkDown2 MarkDown3 MarkDown4
## 6435     1    1           1          1        1        1        1        1        1
##      0    0           0          0        0        0        0        0        0
##      MarkDown5 CPI Unemployment IsHoliday Weekly_Sales
## 6435     1    1           1          1        1  0
##      0    0           0          0        0        0  0
```

A imputação pelo KNN-5 dos valores ausentes foi bem sucedida.

Verificando a correlação resultante no *Data Set* tratado.

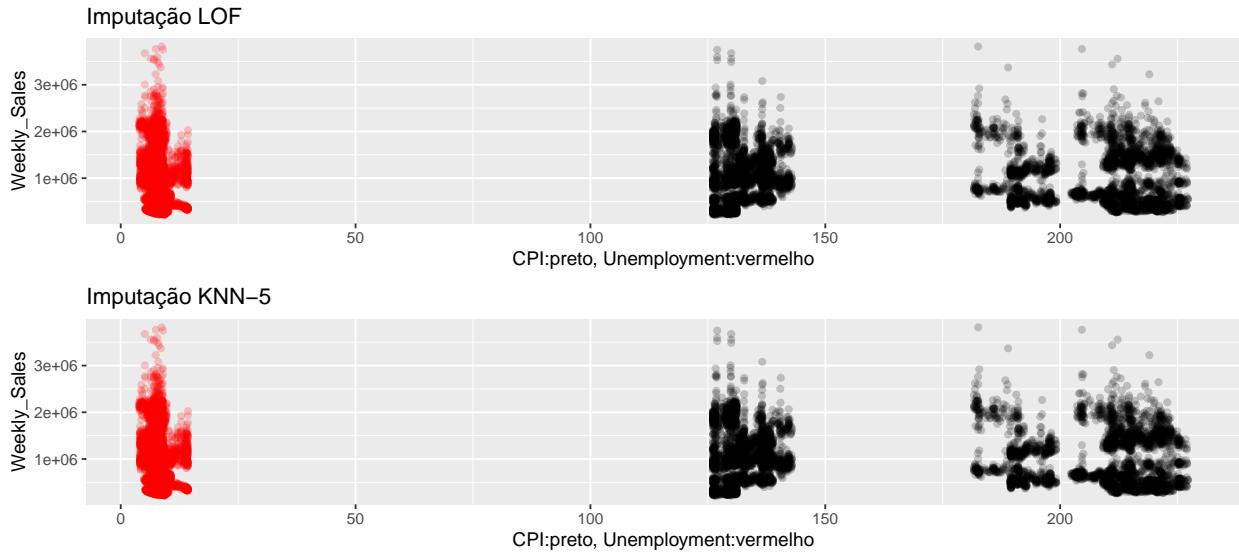
```
corrplot(cor(retail3[,-c('Date','IsHoliday')]),
         method = 'square',
         type = 'lower',
         diag = FALSE,
         title = 'Correlação',
         mar = c(1,1,1,1),
         addCoefAsPercent = TRUE,
         addCoef.col = 'gray50',
         number.digits = 0)
```



Percebe-se que houve uma mudança razoável entre as correlações das variáveis **MarkDown1** até **MarkDown5** com a variável *target* **Weekly_Sales**.

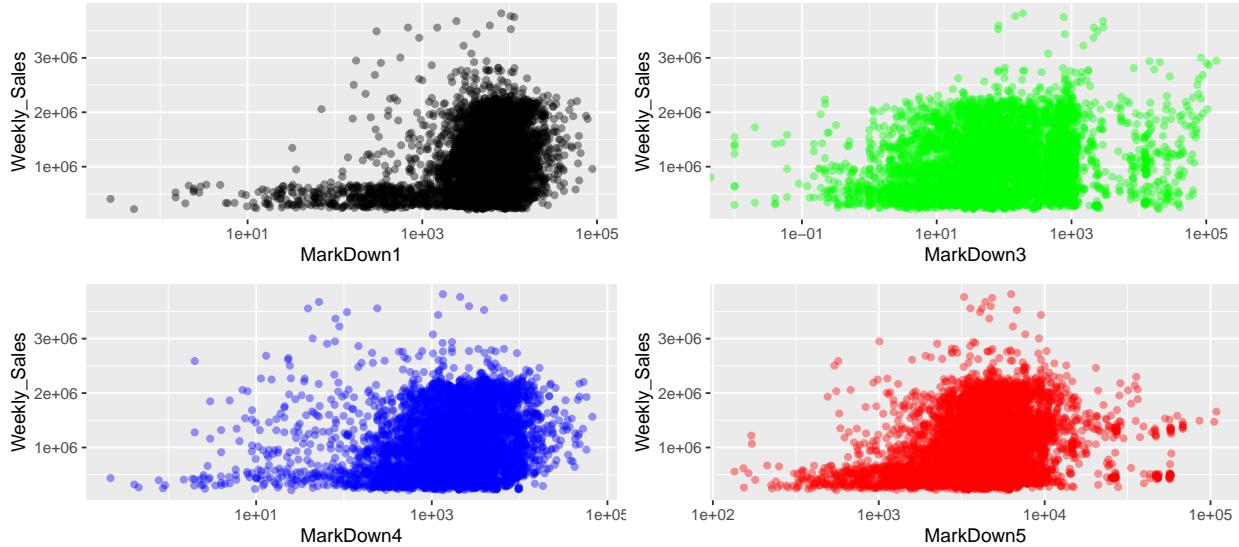
Avaliando mais detalhadamente as variáveis **CPI** e **Unemployment**, percebe-se que não houve mudança significativa entre a imputação pelo método LOF (Last Observation Carried Forward), que preenche o NA com o último valor disponível e o método KNN-5.

```
# Para as variáveis CPI e Unemployment
g1 <- ggplot(data = retail2, aes(y = Weekly_Sales)) +
  geom_point(aes(x = CPI), color = 'black', alpha = 0.2) +
  geom_point(aes(x = Unemployment), color = 'red', alpha = 0.2) +
  labs(x = 'CPI:preto, Unemployment:vermelho', title = 'Imputação LOF')
g2 <- ggplot(data = retail3, aes(y = Weekly_Sales)) +
  geom_point(aes(x = CPI), color = 'black', alpha = 0.2) +
  geom_point(aes(x = Unemployment), color = 'red', alpha = 0.2) +
  labs(x = 'CPI:preto, Unemployment:vermelho', title = 'Imputação KNN-5')
grid.arrange(g1 , g2 ,
             ncol=1, nrow=2)
```



Para as variáveis **MarkDown1**, **MarkDown3**, **MarkDown4** e **MarkDown5**, percebe-se que o viés produzido pela imputação dos NA's com valores zerados desapareceu. Isso ficou evidenciado pelo gráfico das correlações que mostra uma diminuição entre a correlação dessas variáveis com a variável *target* **Weekly_Sales**.

```
# Para as variáveis MarkDowns1-5
g1 <- ggplot(data = retail3, aes(y = Weekly_Sales)) +
  geom_point(aes(x = MarkDown1), color = 'black', alpha = 0.4) +
  scale_x_log10()
g2 <- ggplot(data = retail3, aes(y = Weekly_Sales)) +
  geom_point(aes(x = MarkDown3), color = 'green', alpha = 0.4) +
  scale_x_log10()
g3 <- ggplot(data = retail3, aes(y = Weekly_Sales)) +
  geom_point(aes(x = MarkDown4), color = 'blue', alpha = 0.4) +
  scale_x_log10()
g4 <- ggplot(data = retail3, aes(y = Weekly_Sales)) +
  geom_point(aes(x = MarkDown5), color = 'red', alpha = 0.4) +
  scale_x_log10()
grid.arrange(g1 , g2 ,
             g3 , g4 ,
             ncol=2, nrow=2)
```



Dividindo os dois *Data Set* de Faturamento das Lojas tratados em *Data Set* para treinamento e teste. O primeiro *Data Set* obtido com preenchimento de valores zerados para **MarkDowns1-5** e LOF para **CPI** e **Unemployment**, ficou dividido em *Data Set* **retail2_train** para treinamento e **retail2_test** para teste. O segundo *Data Set* obtido com preenchimento para **MarkDowns1-5**, **CPI** e **Unemployment** com KNN-5, ficou dividido em *Data Set* **retail3_train** para treinamento e **retail3_test** para teste.

```
set.seed(314)
i_retail <- createDataPartition(retail2$Weekly_Sales, p = 0.7, list = FALSE)
retail2_train <- retail2[i_retail,]
retail2_test <- retail2[-i_retail,]

set.seed(314)
i_retail <- createDataPartition(retail3$Weekly_Sales, p = 0.7, list = FALSE)
retail3_train <- retail3[i_retail,]
retail3_test <- retail3[-i_retail,]
```

Com o objetivo de avaliar qual o melhor *Data Set* do Faturamento da Loja que deverá ser utilizado, foi implementado uma modelagem por *Regressão Linear*, que é um modelo com boa performance para *Data Sets* grandes como esse, apesar do gráfico de correlações apontar uma baixa relação entre as variáveis independente e a variável dependente **Weekly_Sales**.

A métrica que a análise se balizará será o *R²* e o *RMSE*.

```
# -----
# Usando o Dataset Retail2
# (com preenchimento de valores zerados para MarkDowns1-5 e LOF para CPI e Unemployment)
# ----- Regressão Linear -----
# Treinando o modelo
set.seed(314)

mod_retail2_rl <- lm(Weekly_Sales ~ .,
                      data = retail2_train[,-c('IsHoliday','Temperature','MarkDown4')])
summary(mod_retail2_rl)

##
```

```

## Call:
## lm(formula = Weekly_Sales ~ ., data = retail2_train[, -c("IsHoliday",
## "Temperature", "MarkDown4")])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1976828 -359097 -34399  340966 2755772
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 8.361e+06 7.015e+05 11.919 < 2e-16 ***
## Store       -1.483e+04 6.080e+02 -24.393 < 2e-16 ***
## Date        -4.575e+02 5.123e+01 -8.929 < 2e-16 ***
## Fuel_Price   1.216e+05 2.873e+04  4.232 2.36e-05 ***
## MarkDown1    1.251e+01 1.510e+00  8.281 < 2e-16 ***
## MarkDown2    6.006e+00 1.580e+00  3.802 0.000145 ***
## MarkDown3    1.224e+01 1.424e+00  8.595 < 2e-16 ***
## MarkDown5    2.779e+01 2.485e+00 11.184 < 2e-16 ***
## CPI         -1.958e+03 2.161e+02 -9.060 < 2e-16 ***
## Unemployment -2.512e+04 4.504e+03 -5.578 2.57e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 505400 on 4497 degrees of freedom
## Multiple R-squared:  0.2092, Adjusted R-squared:  0.2076
## F-statistic: 132.2 on 9 and 4497 DF,  p-value: < 2.2e-16

pred_retail2_rl <- predict(mod_retail2_rl, newdata = retail2_test)

# Performance com valores zerados para MarkDowns1-5 e LOF para CPI e Unemployment:
print(paste('R2:', 
            R2_Score(y_pred = pred_retail2_rl, y_true = retail2_test$Weekly_Sales)))

```

```

## [1] "R2: 0.178142668461905"

print(paste('RMSE:', 
            RMSE(y_pred = pred_retail2_rl, y_true = retail2_test$Weekly_Sales)))

```

```

## [1] "RMSE: 504436.789804949"

```

```

# -----
# Usando o Dataset Retail3
# (com preenchimento para MarkDowns1-5, CPI e Unemployment) com KNN-5
# ----- Regressão Linear -----
set.seed(314)

mod_retail3_rl <- lm(Weekly_Sales ~ .,
                      data = retail3_train[,-c('Fuel_Price','Temperature','MarkDown4')])
summary(mod_retail3_rl)

```

```

##
## Call:
## lm(formula = Weekly_Sales ~ ., data = retail3_train[, -c("Fuel_Price", "Temperature", "MarkDown4")])
## 
```

```

## lm(formula = Weekly_Sales ~ ., data = retail3_train[, -c("Fuel_Price",
##           "Temperature", "MarkDown4")])
##
## Residuals:
##      Min       1Q   Median     3Q    Max
## -1671609 -372446 -20608  340769 2604763
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.210e+05 4.396e+05  1.185 0.23600
## Store       -1.438e+04 6.183e+02 -23.252 < 2e-16 ***
## Date        8.377e+01 2.826e+01   2.964 0.00305 **
## MarkDown1   1.732e+01 1.297e+00  13.361 < 2e-16 ***
## MarkDown2   1.029e+01 1.175e+00   8.754 < 2e-16 ***
## MarkDown3   1.307e+01 1.344e+00   9.726 < 2e-16 ***
## MarkDown5   1.123e+01 1.211e+00   9.271 < 2e-16 ***
## CPI         -2.398e+03 2.083e+02 -11.513 < 2e-16 ***
## Unemployment -2.838e+04 4.441e+03 -6.391 1.82e-10 ***
## IsHolidayTRUE -1.481e+05 3.748e+04 -3.951 7.90e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 502500 on 4497 degrees of freedom
## Multiple R-squared:  0.218, Adjusted R-squared:  0.2164
## F-statistic: 139.3 on 9 and 4497 DF, p-value: < 2.2e-16

pred_retail3_rl <- predict(mod_retail3_rl, newdata = retail3_test)

# Performance com KNN-5 para MarkDowns1-5 e para CPI e Unemployment:
print(paste('R2: ',
            R2_Score(y_pred = pred_retail3_rl, y_true = retail3_test$Weekly_Sales)))

```

```

## [1] "R2: 0.193554094131231"

print(paste('RMSE: ',
            RMSE(y_pred = pred_retail3_rl, y_true = retail3_test$Weekly_Sales)))

```

```

## [1] "RMSE: 499684.821070386"

```

O uso do KNN-5 para imputação dos NA's do *Data Set Retail* se mostrou melhor que q imputação de zeros nas variáveis variáveis **MarkDown1** até **MarkDown5** e preenchimento com o último valor conhecido para as variáveis **CPI** e **Unemployment**. Apesar do *R2* ter sido muito baixo pelo dois métodos, ele ficou melhor pelo segundo método. Essa evidência do *R2*, somada ao *RMSE* apontam que o melhor *Data Set* para o Faturamento das Lojas (**Retail**) é o utilzado pelo segundo modelo, que é o *Data Set* (**Retail3**).

```

# Setando o melhor Dataset para Market
dt_retail_train <- retail3_train
dt_retail_test <- retail3_test

```

Modelagem para o *Data Set* de Marketing.

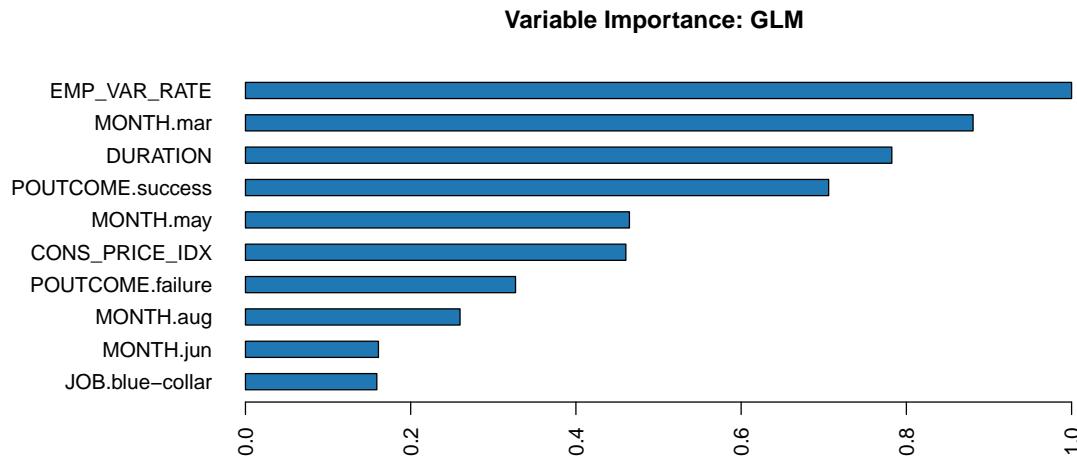
Configurando os *Data Sets* de Marketing para uso com o pacote **H2O**.

```
## |  
## |
```

Regressão Logística

O primeiro modelo que será treinado é o de *Regressão Logística*, utilizando o pacote **H2O**.

```
## |
```



As três variáveis mais importantes do modelo, em ordem de importância, são a **EMP_VAR_RATE** (Taxa de Desemprego da Região), **MONTH.mar**, variável binária para a variável **MONTH** (Último Mês de Contato) para o mês de MARÇO, e **DURATION** (Duração do Último Contato em segundos).

```
## |  
  
## [1] "AUC: 0.937137304215142"  
  
## [1] "LogLoss: 0.21239806750584"  
  
## Confusion Matrix and Statistics  
##  
##           Reference  
## Prediction  no  yes  
##       no  9454  549  
##       yes   482  786  
##  
##           Accuracy : 0.9085  
##             95% CI : (0.9031, 0.9138)  
##    No Information Rate : 0.8816  
##    P-Value [Acc > NIR] : < 2e-16  
##  
##           Kappa : 0.5522  
##
```

```

##  Mcnemar's Test P-Value : 0.03983
##
##          Sensitivity : 0.58876
##          Specificity : 0.95149
##      Pos Pred Value : 0.61987
##      Neg Pred Value : 0.94512
##          Precision : 0.61987
##          Recall : 0.58876
##          F1 : 0.60392
##          Prevalence : 0.11845
##      Detection Rate : 0.06974
##  Detection Prevalence : 0.11250
##      Balanced Accuracy : 0.77013
##
##      'Positive' Class : yes
##

```

Avaliando a performance do modelo, ele obteve um *AUC* de 93,71% e um *LogLoss* de 0,2123981. Otimizando a linha de corte, foi obtido *Acurácia* de 90,85%, uma *Sensibilidade* de 58,88%, uma *Especificidade* de 95,15% e uma *Precisão* de 61,99%. Como métrica balizadora de performance, foi escolhido o *F1 Score* pois é uma junção da *Sensibilidade* (*Recall*) e *Precisão*, mostrando a capacidade de distinguir clientes que irão converter, minimizando tanto os falsos positivo quanto os falsos negativos. O valor do *F1 Score* para esse modelo foi de 60,39%.

Com o objetivo de aprimorar o modelo de *Regressão Logística* e torná-lo mais parcimonioso, foi aplicado os métodos de *feature selection*: *Step Wise* e *Genetic Algorithm (GA)*.

```

# ----- Regressão Logística com Step Wise
# Treinando o modelo
set.seed(314)
arquivo <- '../model/model-Market-RL-StepAIC.rds'
if (file.exists(arquivo)) {
  mod_market_r1SW <- readRDS(file = arquivo)
} else {
  # Ajustando o processamento paralelo em CORES (NÚCLEOS)
  cl <- makePSOCKcluster(cores)
  registerDoParallel(cl)

  cv <- trainControl(method = "cv", number = 10, savePredictions = TRUE,
                      summaryFunction = twoClassSummary, classProbs = TRUE)

  system.time(mod_market_r1SW <- train(SUBSCRIBED ~ .,
                                         data = dt_market_train,
                                         method = "glmStepAIC",
                                         metric = 'ROC',
                                         trControl = cv))

  # Parando o processamento paralelo
  stopCluster(cl)
}
plot(varImp_r1SW)

pred_market_r1SW <- predict(mod_market_r1SW, newdata = dt_market_test, type = 'prob')
cutoff <- optimalCutoff(actuals = as.integer(dt_market_test$SUBSCRIBED)-1,
                         predictedScores = pred_market_r1SW$yes)

```

```

# Performance com Step Wise
print(paste('AUC:', 
            AUC(pred_market_r1SW$yes, as.integer(dt_market_test$SUBSCRIBED)-1)))
print(paste('LogLoss:', 
            LogLoss(pred_market_r1SW$yes, as.integer(dt_market_test$SUBSCRIBED)-1)))

pred_market <- ifelse(pred_market_r1SW$yes > cutoff, 'yes', 'no')
pred_market <- as.factor(pred_market)
(mc_market_r1SW <- caret::confusionMatrix(data = pred_market,
                                              positive = 'yes',
                                              reference = dt_market_test$SUBSCRIBED,
                                              mode = 'everything'))

```

Avaliando a performance do modelo, ele obteve um *AUC* de 93,69% e um *LogLoss* de 0,2127598, similares ao modelo sem *Step Wise*. Otimizando a linha de corte, foi obtido *Acurácia* de 90,86%, uma *Sensibilidade* de 56,33%, uma *Especificidade* de 95,50% e uma *Precisão* de 62,72%. Como métrica balizadora de performance, o valor do *F1 Score* para esse modelo foi de 59,35%. Apesar de ligeiramente menor que o modelo sem *Step Wise*, o modelo com *Step Wise* gerou uma *Precisão* melhor e é um modelo mais parcimonioso que o modelo com todas as variáveis. Portanto, a preferência será pelo modelo com *Step Wise*.

```

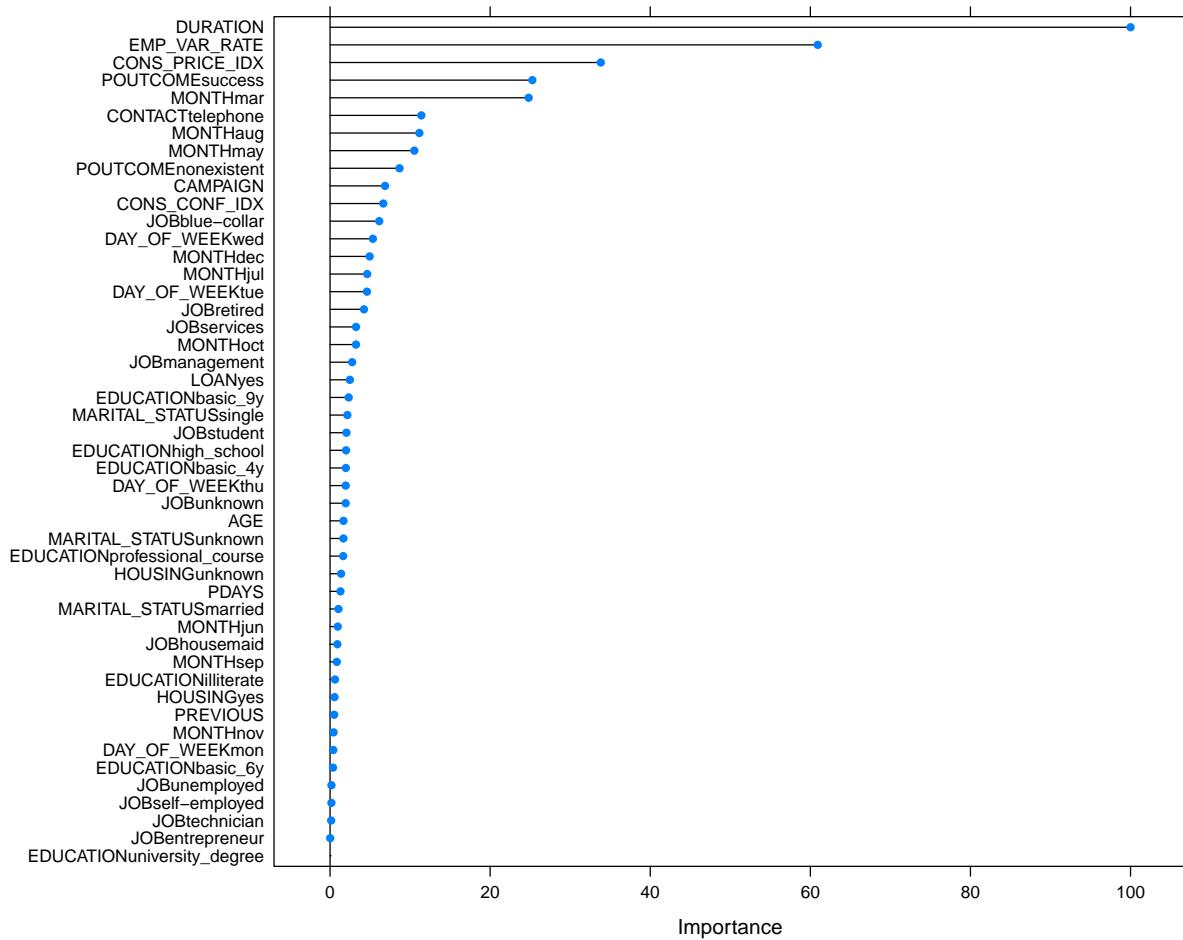
# ----- Regressão Logística com Genetic Algorithm
# Treinando o modelo
set.seed(314)
arquivo <- '../model/model-Market-RL-GA.rds'
if (file.exists(arquivo)) {
  obj_rl_ga <- readRDS(file = arquivo)
} else {
  cl <- makePSOCKcluster(cores)
  registerDoParallel(cl)

  ga_ctrl <- gafsControl(functions = caretGA,
                           genParallel = TRUE,
                           allowParallel = TRUE,
                           number = 5,
                           method = "cv")

  obj_rl_ga <- gafs(x = dt_market_train[,-c('SUBSCRIBED','DEFAULT')],
                     y = dt_market_train$SUBSCRIBED,
                     iters = 3,
                     popSize = 5,
                     gafsControl = ga_ctrl,
                     method = "glm")

  # Parando o processamento paralelo
  stopCluster(cl)
}
mod_market_rlGA <- obj_rl_ga$fit
varImp_rlGA <- varImp(mod_market_rlGA)
plot(varImp_rlGA)

```



```

# Avaliando a performance
pred_market_rlGA <- predict(mod_market_rlGA, newdata = dt_market_test, type = 'prob')
cutoff <- optimalCutoff(actuals = as.integer(dt_market_test$SUBSCRIBED)-1,
                         predictedScores = pred_market_rlGA$yes)

# Performance com Genetic Algorithm
print(paste('AUC: ',
            AUC(pred_market_rlGA$yes, as.integer(dt_market_test$SUBSCRIBED)-1)))

## [1] "AUC: 0.936619533554072"

print(paste('LogLoss: ',
            LogLoss(pred_market_rlGA$yes, as.integer(dt_market_test$SUBSCRIBED)-1)))

## [1] "LogLoss: 0.213113689982769"

pred_market <- ifelse(pred_market_rlGA$yes > cutoff, 'yes', 'no')
pred_market <- as.factor(pred_market)
(mc_market_rlGA <- caret::confusionMatrix(data = pred_market,
                                             positive = 'yes',

```

```

        reference = dt_market_test$SUBSCRIBED,
        mode = 'everything'))
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   no   yes
##       no    9474   568
##      yes     462   767
##
##                  Accuracy : 0.9086
##                  95% CI : (0.9031, 0.9139)
##      No Information Rate : 0.8816
##      P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.5468
##
## Mcnemar's Test P-Value : 0.001069
##
##                  Sensitivity : 0.57453
##                  Specificity : 0.95350
##      Pos Pred Value : 0.62408
##      Neg Pred Value : 0.94344
##                  Precision : 0.62408
##                  Recall : 0.57453
##                  F1 : 0.59828
##                  Prevalence : 0.11845
##      Detection Rate : 0.06805
##      Detection Prevalence : 0.10904
##      Balanced Accuracy : 0.76402
##
##      'Positive' Class : yes
##

```

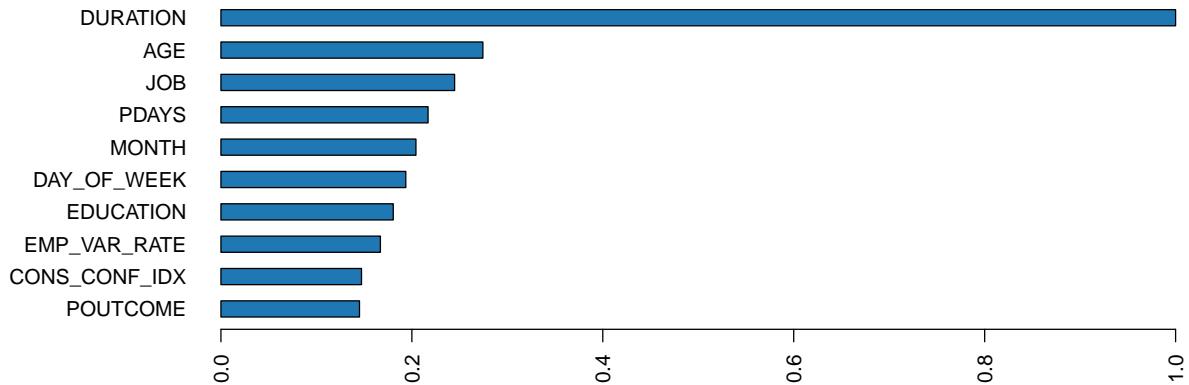
Avaliando a performance do modelo usando *Genetic Algorithm*, ele obteve um *AUC* de 93,66% e um *LogLoss* de 0,2130927, similares aos demais modelos usando *Regressão Logística*. Otimizando a linha de corte, foi obtido *Acurácia* de 90,81%, uma *Sensibilidade* de 57,23%, uma *Especificidade* de 95,32% e uma *Precisão* de 62,16%. Como métrica balizadora de performance, o valor do *F1 Score* para esse modelo foi de 59,59%.

Random Forest

O modelo seguinte a ser treinado será o *Random Forest*, utilizando o pacote **H2O**.

```
## |
```

Variable Importance: DRF



As três variáveis mais importantes do modelo, em ordem de importância, são a **DURATION** (Duração do Último Contato em segundos), a **AGE** (Idade) e o **JOB** (Profissão).

```

## | 
## [1] "AUC: 0.941665121195124"
## [1] "LogLoss: 0.187198841465861"

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   no   yes
##       no    9451   519
##       yes    485   816
## 
##           Accuracy : 0.9109
##             95% CI : (0.9055, 0.9161)
##   No Information Rate : 0.8816
##   P-Value [Acc > NIR] : <2e-16
## 
##           Kappa : 0.5687
## 
##   Mcnemar's Test P-Value : 0.2977
## 
##           Sensitivity : 0.6112
##           Specificity : 0.9512
##   Pos Pred Value : 0.6272
##   Neg Pred Value : 0.9479
##           Precision : 0.6272
##           Recall : 0.6112
##           F1 : 0.6191
##           Prevalence : 0.1184
##   Detection Rate : 0.0724
## Detection Prevalence : 0.1154

```

```

##      Balanced Accuracy : 0.7812
##
##      'Positive' Class : yes
##

```

Avaliando a performance do modelo, foi obtido um *AUC* de 94,17% e um *LogLoss* de 0,1871988. Otimizando a linha de corte, foi obtido *Acurácia* de 91,09%, uma *Sensibilidade* de 61,12%, uma *Especificidade* de 95,12% e uma *Precisão* de 62,72%. Como métrica balizadora de performance, o *F1 Score* para esse modelo foi de 61,91%.

Na busca de um aprimoramento no modelo de *Random Forest*, availou-se o modelo com os métodos de *feature selection*: *Genetic Algorithm (GA)* e *Recursive Feature Elimination (RFE)*.

```

# ----- Random Forest com Genetic Algorithm
# Treinando o modelo
set.seed(314)
arquivo <- '../model/model-Market-RF-GA.rds'
if (file.exists(arquivo)) {
  obj_rf_ga <- readRDS(file = arquivo)
} else {
  # Ajustando o processamento paralelo em CORES (NÚCLEOS)
  cl <- makePSOCKcluster(cores)
  registerDoParallel(cl)

  ga_ctrl <- gafsControl(functions = rfGA,
                          genParallel = TRUE,
                          allowParallel = TRUE,
                          number = 5,
                          method = "cv")

  obj_rf_ga <- gafs(x = dt_market_train[,-c('SUBSCRIBED')],
                     y = dt_market_train$SUBSCRIBED,
                     iters = 3,
                     popSize = 10,
                     gafsControl = ga_ctrl)

  # Parando o processamento paralelo
  stopCluster(cl)
}
mod_market_rfGA <- obj_rf_ga$fit
plot(varImp_rfGA)

# Avaliando a performance
pred_market_rfGA <- predict(mod_market_rfGA, newdata = dt_market_test, type = 'prob')
cutoff <- optimalCutoff(actuals = as.integer(dt_market_test$SUBSCRIBED)-1,
                         predictedScores = (as.data.frame(pred_market_rfGA))$yes)

# Performance com Genetic Algorithm
print(paste('AUC: ',
           AUC((as.data.frame(pred_market_rfGA))$yes,
                as.integer(dt_market_test$SUBSCRIBED)-1)))
print(paste('LogLoss: ',
           LogLoss((as.data.frame(pred_market_rfGA))$yes,
                    as.integer(dt_market_test$SUBSCRIBED)-1)))

```

```

pred_market <- ifelse(as.data.frame(pred_market_rfGA)$yes > cutoff, 'yes', 'no')
pred_market <- as.factor(pred_market)
(mc_market_rfGA <- caret::confusionMatrix(data = pred_market,
                                             positive = 'yes',
                                             reference = dt_market_test$SUBSCRIBED,
                                             mode = 'everything'))

```

Avaliando a performance do modelo usando *Genetic Algorithm*, foi obtido um *AUC* de 94,29% e um *LogLoss* de 0,192436. Otimizando a linha de corte, foi obtido Acurácia de 91,11%, uma Sensibilidade de 60,00%, uma Especificidade de 95,29% e uma Precisão de 63,12%. Como métrica balizadora de performance, o *F1 Score* para esse modelo foi de 61,52%.

```

# ----- Random Forest com Recursive Feature Elimination
# Treinando o modelo
set.seed(314)
arquivo <- '../model/model-Market-RF-RFE.rds'
if (file.exists(arquivo)) {
  obj_rf_rfe <- readRDS(file = arquivo)
} else {
  # Ajustando o processamento paralelo em CORES (NÚCLEOS)
  cl <- makePSOCKcluster(cores)
  registerDoParallel(cl)

  subsets <- c(1:5,10,15,ncol(dt_market_train)-1)
  fStats <- function(...) c(twoClassSummary(...),
                            defaultSummary(...))

  rfFuncs_new <- rfFuncs
  rfFuncs_new$summary <- fStats
  rfe_ctrl <- rfeControl(functions = rfFuncs_new,
                           allowParallel = TRUE,
                           number = 10,
                           method = "cv")

  obj_rf_rfe <- rfe(x = dt_market_train[,-c('SUBSCRIBED')],
                     y = dt_market_train$SUBSCRIBED,
                     sizes = subsets,
                     metric = 'ROC',
                     rfeControl = rfe_ctrl,
                     ntree = 200)

  # Parando o processamento paralelo
  stopCluster(cl)
}

mod_market_rfRFE <- obj_rf_rfe$fit
plot(varImp_rfRFE)

# Avaliando a performance
pred_market_rfRFE <- predict(mod_market_rfRFE, newdata = dt_market_test, type = 'prob')
cutoff <- optimalCutoff(actuals = as.integer(dt_market_test$SUBSCRIBED)-1,
                         predictedScores = (as.data.frame(pred_market_rfRFE))$yes)

# Performance com Recursive Feature Elimination
print(paste('AUC:', 

```

```

AUC((as.data.frame(pred_market_rfRFE))$yes,
     as.integer(dt_market_test$SUBSCRIBED)-1)))
print(paste('LogLoss:',
            LogLoss((as.data.frame(pred_market_rfRFE))$yes,
                     as.integer(dt_market_test$SUBSCRIBED)-1)))

pred_market <- ifelse((as.data.frame(pred_market_rfRFE))$yes > cutoff, 'yes', 'no')
pred_market <- as.factor(pred_market)
(mc_market_rfRFE <- caret::confusionMatrix(data = pred_market,
                                              positive = 'yes',
                                              reference = dt_market_test$SUBSCRIBED,
                                              mode = 'everything'))

```

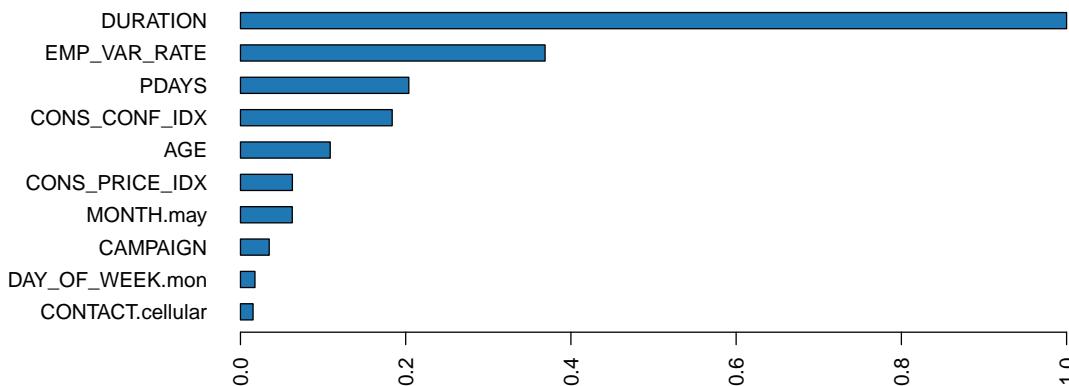
Avaliando a performance do modelo usando *Recursive Feature Elimination*, foi obtido um *AUC* de 94,43% e um *LogLoss* de 0,1976715. Otimizando a linha de corte, foi obtido *Acurácia* de 91,17%, uma *Sensibilidade* de 61,72%, uma *Especificidade* de 9513% e uma *Precisão* de 63,00%. Como métrica balizadora de performance, o *F1 Score* para esse modelo foi de 62,35%.

eXtreme Gradient Boosting

O modelo seguinte a ser treinado será o *eXtreme Gradient Boosting*, utilizando o pacote **H2O**.

```
## |
```

Variable Importance: XGBOOST



As três variáveis mais importantes do modelo, em ordem de impotância, são a **DURATION** (Duração do Último Contato em segundos), similar ao obtido pelo modelo de *Random Forest*, **EMP_VAR_RATE** (Taxa de Desemprego da Região) e **PDAYS** (Número de Dias desde Último Contato).

```
## |
```

```
## [1] "AUC: 0.945356875765197"
```

```
## [1] "LogLoss: 0.185490688390293"
```

```

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   no   yes
##       no    9499   559
##      yes     437   776
##
##                  Accuracy : 0.9116
##                  95% CI : (0.9062, 0.9168)
##      No Information Rate : 0.8816
##      P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.5594
##
## McNemar's Test P-Value : 0.0001261
##
##      Sensitivity : 0.58127
##      Specificity : 0.95602
##      Pos Pred Value : 0.63974
##      Neg Pred Value : 0.94442
##      Precision : 0.63974
##      Recall : 0.58127
##      F1 : 0.60911
##      Prevalence : 0.11845
##      Detection Rate : 0.06885
##      Detection Prevalence : 0.10762
##      Balanced Accuracy : 0.76865
##
##      'Positive' Class : yes
##

```

Avaliando a performance do modelo, foi obtido um *AUC* de 94,50% e um *LogLoss* de 0,1859924. Otimizando a linha de corte, foi obtido *Acurácia* de 91,18%, uma *Sensibilidade* de 61,80%, uma *Especificidade* de 95,13% e uma *Precisão* de 63,03%. Como métrica balizadora de performance, o *F1 Score* para esse modelo foi de 62,41%.

Support Vector Machine (LINEAR)

O modelo seguinte a ser treinado será o *Support Vector Machine (LINEAR)*, utilizando o pacote **CARET**.

```

# ----- Support Vector Machine (LINEAR)
# Treinando o modelo
set.seed(314)
arquivo <- '../model/model-Market-SVM-Linear.rds'
if (file.exists(arquivo)) {
  mod_market_svmLin <- readRDS(file = arquivo)
} else {
  # Ajustando o processamento paralelo em 8 CORES (NÚCLEOS)
  cl <- makePSOCKcluster(cores)
  registerDoParallel(cl)

  cv <- trainControl(method = "cv", number = 10, savePredictions = TRUE,
                      summaryFunction = twoClassSummary, classProbs = TRUE)

```

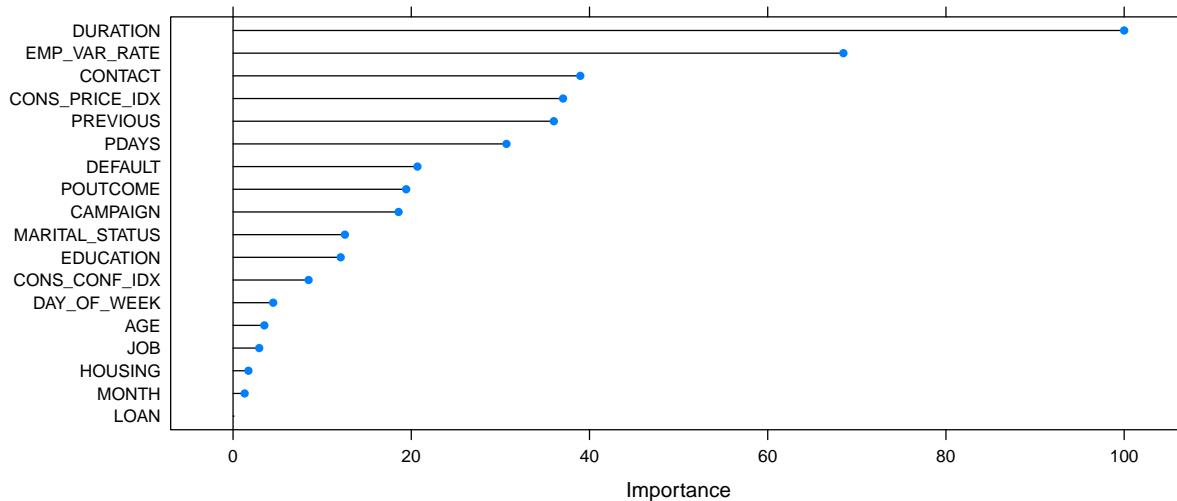
```

mod_market_svmLin <- train(SUBSCRIBED ~ .,
                            data = dt_market_train,
                            method = "svmLinear",
                            metric = 'ROC',
                            trControl = cv,
                            preProcess = c("center", "scale"))

# Parando o processamento paralelo
stopCluster(cl)
}

varImp_svmLin <- varImp(mod_market_svmLin)
plot(varImp_svmLin)

```



As três mais importantes variáveis do modelo, em ordem de importância, são **DURATION** (Duração do Último Contato em segundos), **EMP_VAR RATE** (Taxa de Desemprego da Região) e **CONTACT** (Tipo de Contato).

```

pred_market_svmLin <- predict(mod_market_svmLin, newdata = dt_market_test, type = 'prob')
cutoff <- optimalCutoff(actuals = as.integer(dt_market_test$SUBSCRIBED)-1,
                         predictedScores = pred_market_svmLin$yes)

print(paste('AUC:',
           AUC(pred_market_svmLin$yes, as.integer(dt_market_test$SUBSCRIBED)-1)))

## [1] "AUC: 0.93654844186313"

print(paste('LogLoss:',
           LogLoss(pred_market_svmLin$yes, as.integer(dt_market_test$SUBSCRIBED)-1)))

## [1] "LogLoss: 0.225067204616777"

pred_market <- ifelse(pred_market_svmLin$yes > cutoff, 'yes', 'no')
pred_market <- as.factor(pred_market)

```

```

(mc_market_svmLin <- caret::confusionMatrix(data = pred_market,
                                              positive = 'yes',
                                              reference = dt_market_test$SUBSCRIBED,
                                              mode = 'everything'))

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    no   yes
##       no    9450   572
##      yes     486   763
##
##                  Accuracy : 0.9061
##                         95% CI : (0.9006, 0.9115)
##      No Information Rate : 0.8816
##      P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.5376
##
##      Mcnemar's Test P-Value : 0.008969
##
##                  Sensitivity : 0.5715
##                  Specificity : 0.9511
##      Pos Pred Value : 0.6109
##      Neg Pred Value : 0.9429
##                  Precision : 0.6109
##                  Recall : 0.5715
##                  F1 : 0.5906
##      Prevalence : 0.1184
##      Detection Rate : 0.0677
##      Detection Prevalence : 0.1108
##      Balanced Accuracy : 0.7613
##
##      'Positive' Class : yes
##

```

Avaliando a performance do modelo, foi obtido um *AUC* de 93,65% e um *LogLoss* de 0,2250672. Otimizando a linha de corte, foi obtido *Acurácia* de 90,61%, uma *Sensibilidade* de 57,15%, uma *Especificidade* de 95,11% e uma *Precisão* de 61,09%. Como métrica balizadora de performance, o *F1 Score* para esse modelo foi de 59,06%.

Support Vector Machine (RADIAL)

O modelo seguinte a ser treinado será o *Support Vector Machine (RADIAL)*, utilizando o pacote **CARET**.

```

# ----- Support Vector Machine (RADIAL)
# Treinando o modelo
set.seed(314)
arquivo <- '../model/model-Market-SVM-Radial.rds'
if (file.exists(arquivo)) {
  mod_market_svmRad <- readRDS(file = arquivo)
} else {

```

```

# Ajustando o processamento paralelo em 8 CORES (NÚCLEOS)
cl <- makePSOCKcluster(cores)
registerDoParallel(cl)

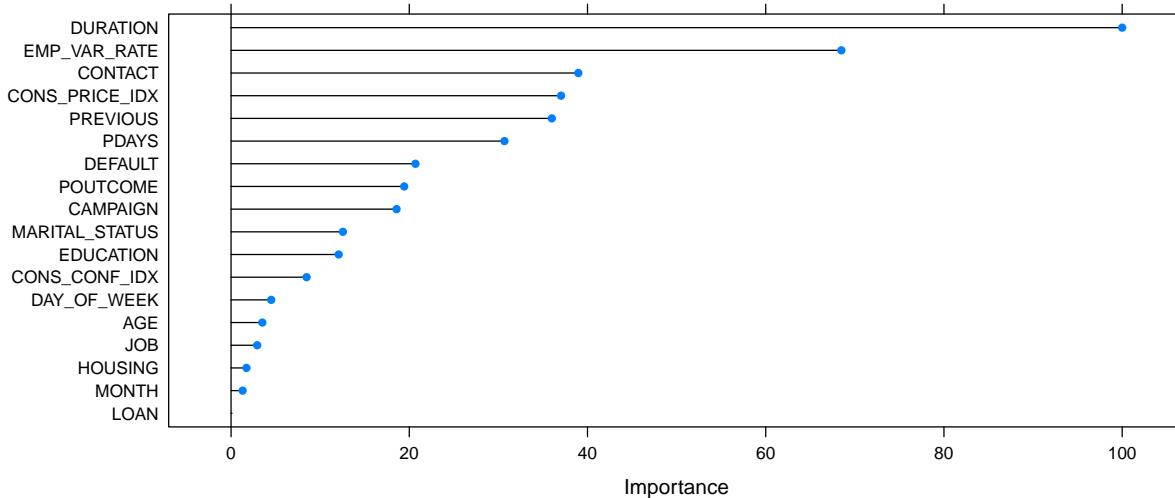
cv <- trainControl(method = "cv", number = 10, savePredictions = TRUE,
                     summaryFunction = twoClassSummary, classProbs = TRUE)

mod_market_svmRad <- train(SUBSCRIBED ~ .,
                            data = dt_market_train,
                            method = "svmRadial",
                            metric = 'ROC',
                            trControl = cv,
                            preProcess = c("center", "scale"))

# Parando o processamento paralelo
stopCluster(cl)
}

varImp_svmRad <- varImp(mod_market_svmRad)
plot(varImp_svmRad)

```



As três mais importantes variáveis do modelo, em ordem de importância, são **DURATION** (Duração do Último Contato em segundos), **EMP_VAR RATE** (Taxa de Desemprego da Região) e **CONTACT** (Tipo de Contato), similar ao modelo *Support Vector Machine (LINEAR)*.

```

pred_market_svmRad <- predict(mod_market_svmRad, newdata = dt_market_test, type = 'prob')
cutoff <- optimalCutoff(actuals = as.integer(dt_market_test$SUBSCRIBED)-1,
                         predictedScores = pred_market_svmRad$yes)

print(paste('AUC:',
           AUC(pred_market_svmRad$yes, as.integer(dt_market_test$SUBSCRIBED)-1)))

## [1] "AUC: 0.934921927301019"

```

```

print(paste('LogLoss: ',
           LogLoss(pred_market_svmRad$yes, as.integer(dt_market_test$SUBSCRIBED)-1)))

## [1] "LogLoss: 0.223607438039648"

pred_market <- ifelse(pred_market_svmRad$yes > cutoff, 'yes', 'no')
pred_market <- as.factor(pred_market)
(mc_market_svmRad <- caret::confusionMatrix(data = pred_market,
                                              positive = 'yes',
                                              reference = dt_market_test$SUBSCRIBED,
                                              mode = 'everything'))

```

```

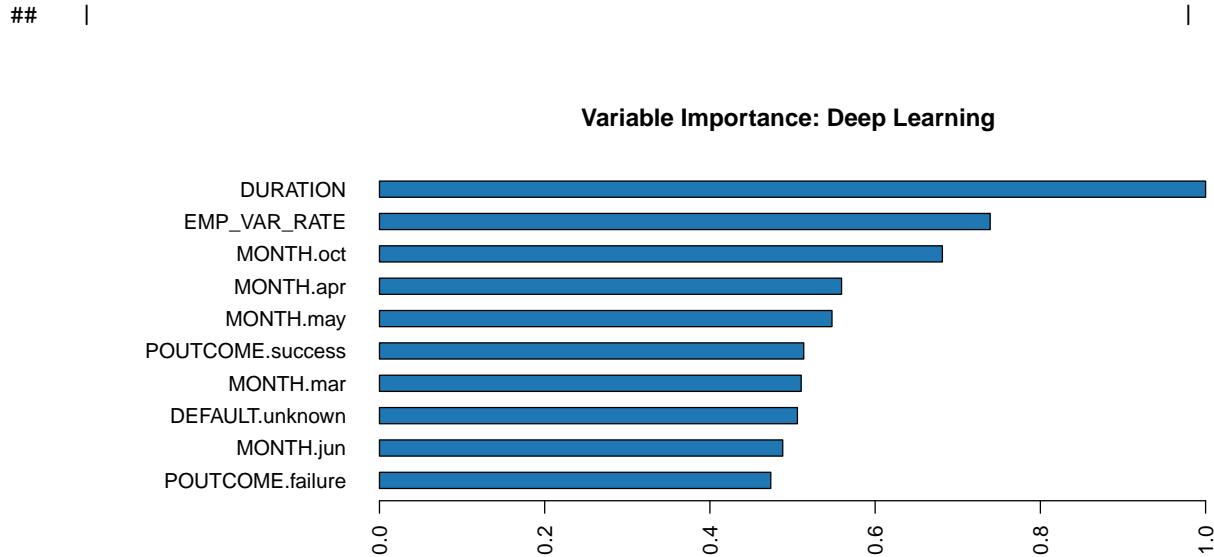
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    no    yes
##       no  9528   653
##       yes   408   682
##
##          Accuracy : 0.9059
##                 95% CI : (0.9003, 0.9112)
##      No Information Rate : 0.8816
##      P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 0.5103
##
##  Mcnemar's Test P-Value : 6.842e-14
##
##          Sensitivity : 0.51086
##          Specificity  : 0.95894
##      Pos Pred Value : 0.62569
##      Neg Pred Value : 0.93586
##          Precision  : 0.62569
##          Recall     : 0.51086
##          F1         : 0.56247
##          Prevalence  : 0.11845
##      Detection Rate : 0.06051
##  Detection Prevalence : 0.09671
##      Balanced Accuracy : 0.73490
##
##      'Positive' Class : yes
##

```

Avaliando a performance do modelo, foi obtido um *AUC* de 93,49% e um *LogLoss* de 0,2236074. Otimizando a linha de corte, foi obtido *Acurácia* de 90,59%, uma *Sensibilidade* de 51,09%, uma *Especificidade* de 95,89% e uma *Precisão* de 62,57%. Como métrica balizadora de performance, o *F1 Score* para esse modelo foi de 56,25%.

Rede Neural

O modelo seguinte a ser treinado será o de *Rede Neural*, utilizando o pacote **CARET**.



As três variáveis mais importantes do modelo, em ordem de importância, são **DURATION** (Duração do Último Contato em segundos), **EMP_VAR_RATE** (Taxa de Desemprego da Região) e **MONTH.oct**, variável binária para a variável **MONTH** (Último Mês de Contato) para o mês de OUTUBRO.

```
## |
```

```
## [1] "AUC: 0.945926513959001"

## [1] "LogLoss: 0.1837950278947"

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    no    yes
##       no 9391   450
##       yes  545  885
## 
##           Accuracy : 0.9117
##                 95% CI : (0.9063, 0.9169)
##      No Information Rate : 0.8816
##      P-Value [Acc > NIR] : < 2.2e-16
## 
##           Kappa : 0.5899
## 
##  Mcnemar's Test P-Value : 0.002882
## 
##           Sensitivity : 0.66292
##           Specificity  : 0.94515
##      Pos Pred Value : 0.61888
##      Neg Pred Value : 0.95427
##          Precision  : 0.61888
##          Recall    : 0.66292
##          F1        : 0.64014
```

```

##          Prevalence : 0.11845
##          Detection Rate : 0.07852
##  Detection Prevalence : 0.12687
##          Balanced Accuracy : 0.80404
##
##          'Positive' Class : yes
##

```

Avaliando a performance do modelo, foi obtido um *AUC* de 94,59% e um *LogLoss* de 0,183795. Otimizando a linha de corte, foi obtido *Acurácia* de 91,17%, uma *Sensibilidade* de 66,29%, uma *Especificidade* de 94,51% e uma *Precisão* foi de 61,89%. Como métrica balizadora de performance, o *F1 Score* para esse modelo foi de 64,01%.

Modelagem para o *Data Set* de Faturamento das Lojas.

Regressão Linear

O primeiro modelo que será treinado é o de *Regressão Linear*.

```

# Treinando o modelo
set.seed(314)

mod_retail_rl <- lm(Weekly_Sales ~ .,
                     data = dt_retail_train[,-c('Store')])
summary(mod_retail_rl)

##
## Call:
## lm(formula = Weekly_Sales ~ ., data = dt_retail_train[, -c("Store")])
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1973007 -405215  -94081   346224  2559665 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 6.495e+05  6.519e+05   0.996 0.319133  
## Date        5.319e+01  4.794e+01   1.110 0.267265  
## Temperature 4.312e+02  4.944e+02   0.872 0.383119  
## Fuel_Price  -1.617e+04 3.120e+04  -0.518 0.604315  
## MarkDown1   2.912e+01  2.411e+00  12.077 < 2e-16 *** 
## MarkDown2   1.354e+01  1.336e+00  10.139 < 2e-16 *** 
## MarkDown3   1.562e+01  1.419e+00  11.007 < 2e-16 *** 
## MarkDown4  -1.100e+01  3.174e+00  -3.466 0.000533 *** 
## MarkDown5   7.419e+00  1.290e+00   5.752 9.40e-09 *** 
## CPI        -1.519e+03 2.322e+02  -6.540 6.84e-11 *** 
## Unemployment -4.821e+04 4.757e+03 -10.135 < 2e-16 *** 
## IsHolidayTRUE -2.008e+05 3.965e+04  -5.063 4.28e-07 *** 
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 531200 on 4495 degrees of freedom
## Multiple R-squared:  0.1266, Adjusted R-squared:  0.1244 
## F-statistic: 59.22 on 11 and 4495 DF,  p-value: < 2.2e-16

```

A princípio, foi obtido um modelo sem a variável ID da Loja. O resultado obtido foi um R^2 muito baixo e um RSE demasiadamente alto. Optou-se em colocar a variável **Store** no treinamento do modelo, com o argumento que seria uma variável da loja que não seria alterada no decorrer da vida útil da mesma e a identificaria univocamente. Consequentemente, a região em que a loja estaria instalada seria relacionada no modelo por conta disso. Isso traria para o modelo a posição geográfica em que elas atuariam, levando para a modelagem informações que não estariam disponíveis no *Data Set* explicitamente. Isso poderia enriquecer o modelo com mais informações, trazendo informação geográfica que estaria relacionado ao nível de renda da população da região.

```
mod_retail_rl <- lm(Weekly_Sales ~ .,
                     data = dt_retail_train)
summary(mod_retail_rl)

##
## Call:
## lm(formula = Weekly_Sales ~ ., data = dt_retail_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1649960 -372535  -18895  342336  2583197
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.495e+05 6.168e+05 1.215 0.224356
## Store       -1.434e+04 6.240e+02 -22.979 < 2e-16 ***
## Date        6.595e+01 4.535e+01 1.454 0.145996
## Temperature -3.539e+02 4.689e+02 -0.755 0.450537
## Fuel_Price   1.445e+04 2.955e+04 0.489 0.624836
## MarkDown1    1.948e+01 2.319e+00 8.398 < 2e-16 ***
## MarkDown2    1.003e+01 1.273e+00 7.881 4.03e-15 ***
## MarkDown3    1.305e+01 1.347e+00 9.689 < 2e-16 ***
## MarkDown4    -3.696e+00 3.019e+00 -1.224 0.220957
## MarkDown5    1.137e+01 1.232e+00 9.227 < 2e-16 ***
## CPI         -2.332e+03 2.225e+02 -10.480 < 2e-16 ***
## Unemployment -2.858e+04 4.581e+03 -6.240 4.77e-10 ***
## IsHolidayTRUE -1.442e+05 3.760e+04 -3.835 0.000127 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 502600 on 4494 degrees of freedom
## Multiple R-squared:  0.2184, Adjusted R-squared:  0.2163
## F-statistic: 104.7 on 12 and 4494 DF,  p-value: < 2.2e-16
```

Buscando uma melhora no modelo, optou-se por retirar os outliers, pois o *Data Set* é relativamente grande, não trazendo maiores prejuízos para o treinamento. Foi considerado outliers os pontos do modelo que tenham a distância de Cook maior que $\frac{4}{\text{nr. de observações}}$.

```
outliers <- cooks.distance(mod_retail_rl) > 4/nrow(dt_retail_train)
mod_retail_rl <- lm(Weekly_Sales ~ .,
                     data = dt_retail_train[!which(outliers),])
summary(mod_retail_rl)

##
```

```

## Call:
## lm(formula = Weekly_Sales ~ ., data = dt_retail_train[!which(outliers),
##           ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1062729 -353134 -13754  319108 1544028
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.403e+05 5.751e+05  0.592  0.5540
## Store       -1.333e+04 5.820e+02 -22.900 < 2e-16 ***
## Date        8.647e+01 4.233e+01  2.043  0.0412 *
## Temperature 6.232e+02 4.441e+02  1.403  0.1606
## Fuel_Price  2.682e+03 2.769e+04  0.097  0.9228
## MarkDown1   2.495e+01 2.331e+00 10.704 < 2e-16 ***
## MarkDown2   1.097e+01 1.248e+00  8.791 < 2e-16 ***
## MarkDown3   1.298e+01 1.652e+00  7.861 4.77e-15 ***
## MarkDown4   -3.351e+00 3.044e+00 -1.101  0.2710
## MarkDown5   2.148e+01 1.517e+00 14.156 < 2e-16 ***
## CPI         -2.413e+03 2.078e+02 -11.611 < 2e-16 ***
## Unemployment -3.263e+04 4.260e+03 -7.659 2.30e-14 ***
## IsHolidayTRUE -1.629e+05 3.762e+04 -4.329 1.53e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 460000 on 4318 degrees of freedom
## Multiple R-squared:  0.264, Adjusted R-squared:  0.262
## F-statistic: 129.1 on 12 and 4318 DF, p-value: < 2.2e-16

```

Houve uma melhora significativa do modelo, o *R²* passou para 0,435 e o *RSE* 337.300.

```

mod_retail_rl <- lm(Weekly_Sales ~ .,
                     data = dt_retail_train[!which(outliers), -c('MarkDown4', 'Fuel_Price')])
summary(mod_retail_rl)

##
## Call:
## lm(formula = Weekly_Sales ~ ., data = dt_retail_train[!which(outliers),
##           -c("MarkDown4", "Fuel_Price")])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1061194 -352911 -13484  318679 1544807
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.685e+05 4.099e+05  0.655 0.512372
## Store       -1.339e+04 5.789e+02 -23.127 < 2e-16 ***
## Date        9.195e+01 2.645e+01  3.476 0.000514 ***
## Temperature 6.644e+02 4.426e+02  1.501 0.133362
## MarkDown1   2.302e+01 1.381e+00 16.663 < 2e-16 ***
## MarkDown2   1.097e+01 1.241e+00  8.837 < 2e-16 ***

```

```

## MarkDown3      1.297e+01  1.651e+00   7.856 4.95e-15 ***
## MarkDown5      2.152e+01  1.501e+00  14.337 < 2e-16 ***
## CPI          -2.431e+03  1.975e+02 -12.311 < 2e-16 ***
## Unemployment -3.226e+04  4.210e+03 -7.663 2.23e-14 ***
## IsHolidayTRUE -1.638e+05  3.760e+04 -4.357 1.35e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 460000 on 4320 degrees of freedom
## Multiple R-squared:  0.2638, Adjusted R-squared:  0.2621
## F-statistic: 154.8 on 10 and 4320 DF,  p-value: < 2.2e-16

dt <- dt_retail_train[!which(outliers), -c('MarkDown4', 'Fuel_Price')]

```

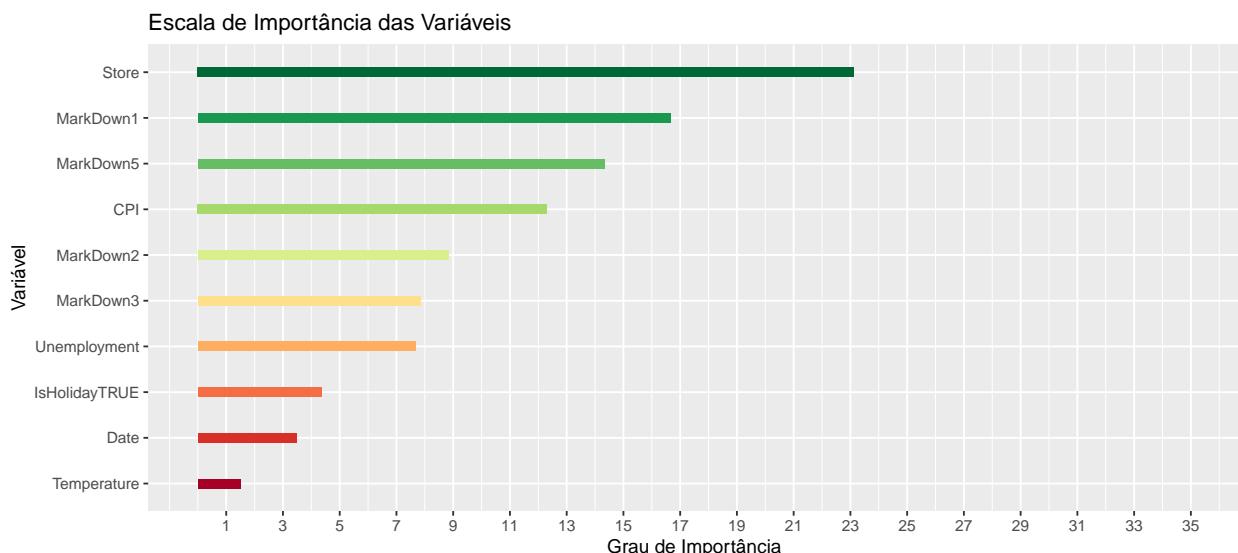
Como resultado final, foi obtido um modelo com todas as variáveis presentes significantes, pois o *p-valor* dos coeficientes delas passaram no teste t. Assim como todo o modelo de acordo com o teste F que obteve um *p-valor* igual a $2.2e-16$, apesar do *R²* baixo de 0,435 e um *RSE* alto de 337.200.

Verificando a contribuição das variáveis no modelo.

```

varImp_rl <- caret::varImp(mod_retail_rl, useModel = TRUE, scale = TRUE)
Imp_rl <- data.table()
Imp_rl[, ' :=' (Variable = rownames(varImp_rl),
             Importance = varImp_rl$Overall)]
setorder(Imp_rl, Importance)
Imp_rl$Variable <- factor(Imp_rl$Variable, levels = Imp_rl$Variable)
as.data.frame(Imp_rl) %>% ggplot(aes(y = Importance, x = Variable)) +
  geom_col(aes(fill = as.factor(Variable)), width = 0.2, show.legend = FALSE) +
  scale_fill_brewer(direction = 1, palette = 8, type = 'div') +
  scale_y_continuous(limits = c(0,35), breaks = seq(1,35,2)) +
  labs(y = 'Grau de Importância', x = "Variável",
       title = "Escala de Importância das Variáveis") +
  coord_flip()

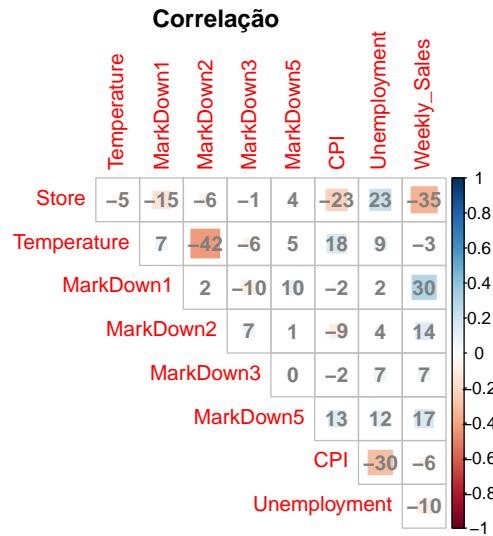
```



As três variáveis mais importantes do modelo, em ordem de importância, são **Store** (ID da Loja), **MarkDown1** e **MarkDown5** (Redução de Preço).

Verificando a existência de Multicolinearidade no modelo.

```
corrplot(cor(dt[,-c('Date','IsHoliday')]),
         method = 'square',
         type = 'upper',
         diag = FALSE,
         title = 'Correlação',
         mar = c(1,1,1,1),
         addCoefAsPercent = TRUE,
         addCoef.col = 'gray50',
         number.digits = 0)
```



```
# VIF (Variance Inflation Factor)
vif(mod_retail_rl)
```

```
##           Store        Date Temperature   MarkDown1   MarkDown2   MarkDown3
## 1.155873 1.212585 1.323240 1.070453 1.648638 1.240790
##   MarkDown5          CPI Unemployment IsHoliday
## 1.132470 1.238192 1.282312 1.585572
```

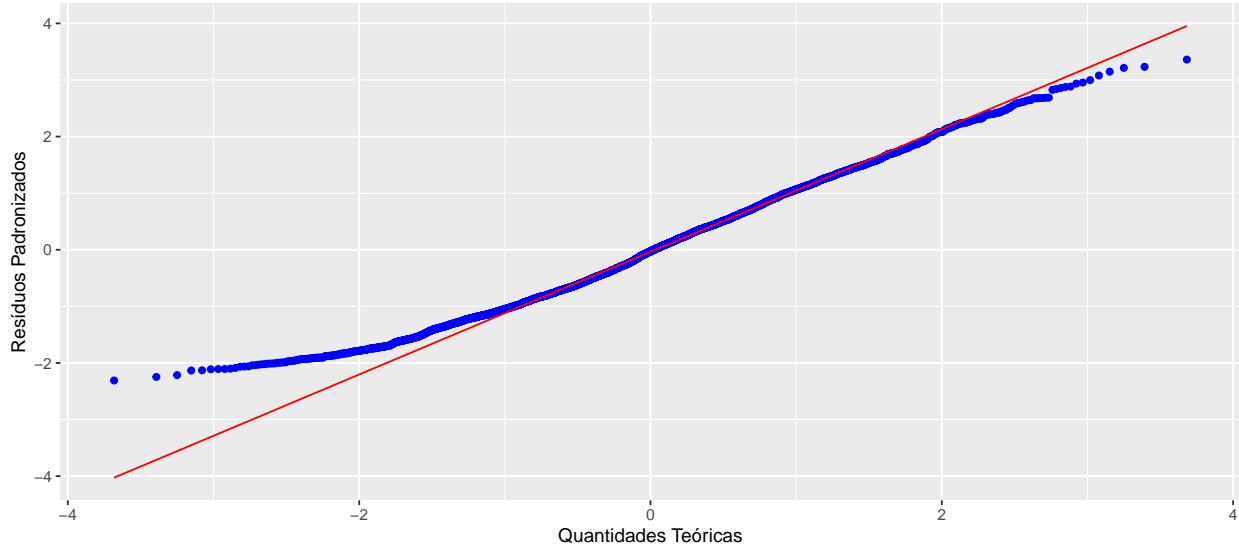
Graficamente não há indícios de alta correlação entre as variáveis dependentes (>70%).

Avaliando através do teste estatístico VIF (Variance Inflation Factor), utilizou-se a regra prática se o valor do VIF for maior que 10, existe forte multicolinearidade entre as variáveis indenpendentes. Como o teste não apontou a existência de multicolinearidade, o modelo atende ao pressuposto de ausência de multicolinearidade entre as variáveis independentes.

Verificando o pressuposto de normalidade dos resíduos.

```
residuo <- rstandard(mod_retail_rl)

# Avaliando graficamente
ggplot() +
  geom_qq(aes(sample = residuo), color = 'blue') +
  geom_qq_line(aes(sample = residuo), color = 'red') +
  labs(x = "Quantidades Teóricas", y = "Resíduos Padronizados")
```



```
# Avaliando a normalidade dos resíduos através do teste formal de Anderson-Darling
ad.test(residuo)
```

```
##
##  Anderson-Darling normality test
##
## data: residuo
## A = 10.29, p-value < 2.2e-16
```

Com exceção dos primeiros e últimos pontos que representam um descolamento da reta, o gráfico sugere que os resíduos são distribuídos normalmente. Utilizando um teste formal para a avaliação, constatou-se que os resíduos não possuem distribuição normal. O teste utilizado foi o de Anderson-Darling, que gerou um *p*-*valor* menor que 5%, portanto rejeitou-se a hipótese nula de existência de normalidade.

Na tentativa de buscar uma normalidade dos resíduos, foi aplicado uma transformação nas variáveis independentes, que no caso foi utilizado a função logarítmica..

```
mod_retail_r12 <- lm(Weekly_Sales ~ log(Store) + log(MarkDown1) +
                      log(MarkDown5) + log(CPI) +
                      log(Temperature) + log(Unemployment) + 1,
                      data = dt)
summary(mod_retail_r12)
```

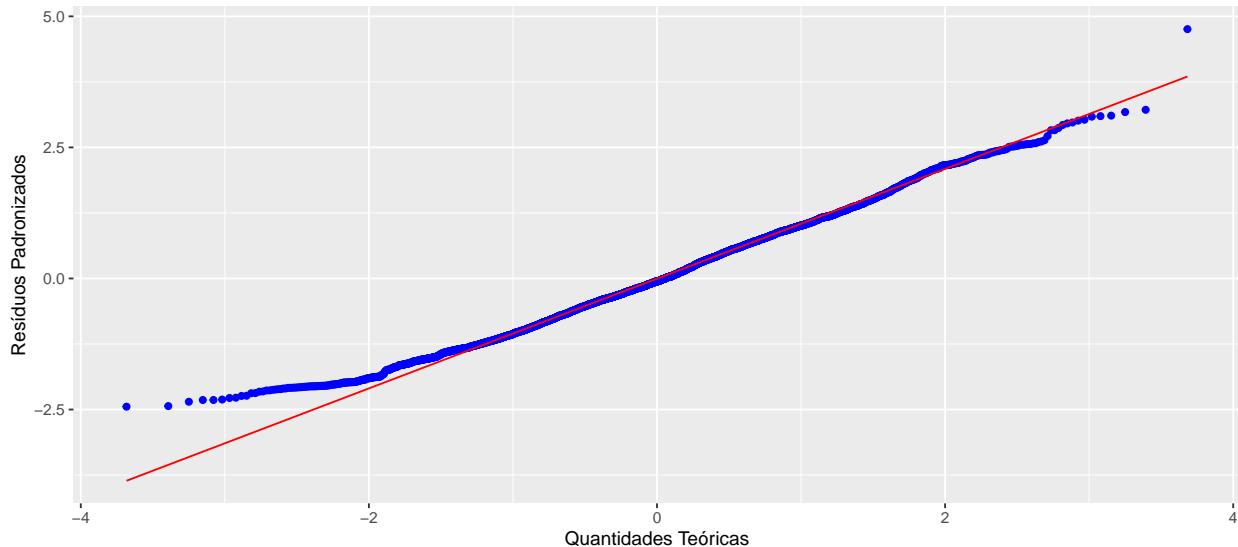
```
##
## Call:
## lm(formula = Weekly_Sales ~ log(Store) + log(MarkDown1) + log(MarkDown5) +
##     log(CPI) + log(Temperature) + log(Unemployment) + 1, data = dt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1140726 -331176 -25937  327714  2216949 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  1079.78    10.67 101.00  <2e-16 ***
## log(Store)   13.77    0.31  44.13  <2e-16 ***
## log(MarkDown1)  0.00    0.00  1.00  0.3177    
## log(MarkDown5)  0.00    0.00  1.00  0.3177    
## log(CPI)     -0.01    0.00 -1.00  0.3177    
## log(Temperature)  0.00    0.00  1.00  0.3177    
## log(Unemployment)  0.00    0.00  1.00  0.3177    
```

```

## (Intercept) 2458814    214849  11.444 <2e-16 ***
## log(Store)   -185445      8972 -20.669 <2e-16 ***
## log(MarkDown1) 79309     7210  11.000 <2e-16 ***
## log(MarkDown5) 176324    11727  15.036 <2e-16 ***
## log(CPI)     -455181    33847 -13.448 <2e-16 ***
## log(Temperature) -17329    20490  -0.846  0.398
## log(Unemployment) -310928   33985  -9.149 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 466900 on 4324 degrees of freedom
## Multiple R-squared:  0.2407, Adjusted R-squared:  0.2397
## F-statistic: 228.5 on 6 and 4324 DF,  p-value: < 2.2e-16

residuo2 <- rstandard(mod_retail_rl2)
ggplot() +
  geom_qq(aes(sample = residuo2), color = 'blue') +
  geom_qq_line(aes(sample = residuo2), color = 'red') +
  labs(x = "Quantidades Teóricas", y = "Resíduos Padronizados")

```



```

ad.test(residuo2)

##
## Anderson-Darling normality test
##
## data:  residuo2
## A = 4.2886, p-value = 1.19e-10

```

O resultado foi um modelo com R^2 mais baixo e RSE mais alto.

Validando graficamente, os resíduos parecem estar distribuídos normalmente com exceção das extremidades, mas formalmente o teste ainda acusa a falta de normalidade. O p -valor é menor que 5%, rejeitando a hipótese de normalidade.

Apesar do pressuposto de normalidade dos resíduos comprometer a qualidade do modelo obtido, prosseguiremos com sua avaliação no demais pressupostos.

Verificando a existência de auto correlação serial.

```
# Teste de Durbin-Watson para modelo sem transformação
dwtest(mod_retail_rl, alternative = 'two.sided')

##
##  Durbin-Watson test
##
## data: mod_retail_rl
## DW = 0.24679, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is not 0

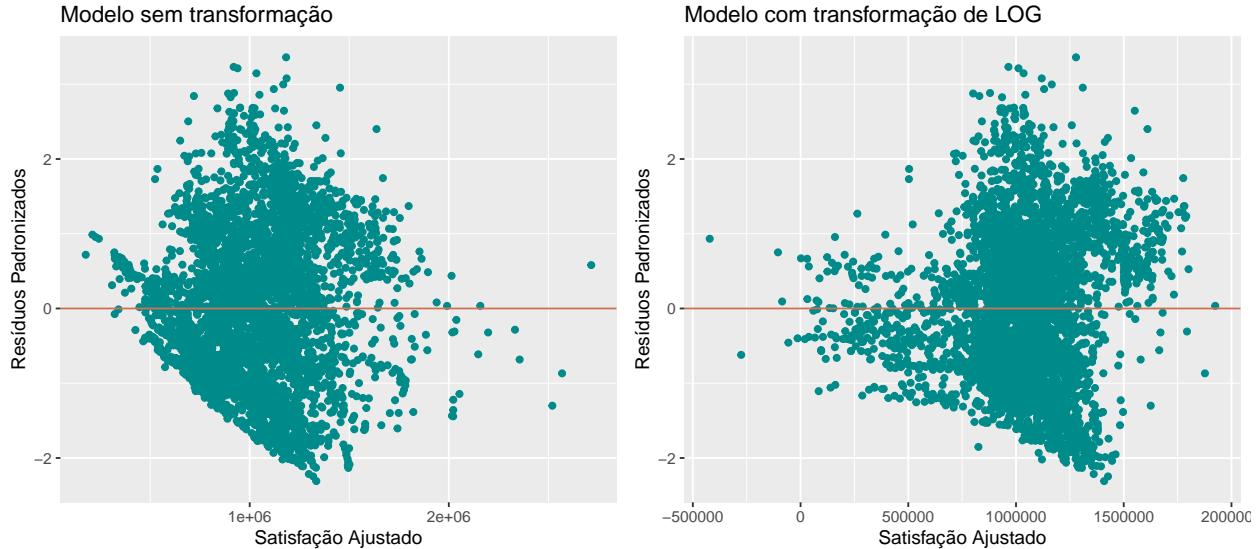
# Teste de Durbin-Watson para modelo com transformação LOG
dwtest(mod_retail_rl2, alternative = 'two.sided')

##
##  Durbin-Watson test
##
## data: mod_retail_rl2
## DW = 0.20856, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is not 0
```

Avaliando através do teste formal de Durbin-Watson, percebe-se um *p-valor* menor que 5%. Com isso rejeita-se a hipótese nula de ausência de correlação serial nos dois modelos. Portanto, há indícios de correlação serial nos dois modelos, quebrando mais um pressuposto da *Regressão Linear*.

Verificando a existência de homocedasticidade.

```
g1 <- ggplot() +
  geom_point(aes(x = fitted.values(mod_retail_rl), y = residuo), color = 'cyan4') +
  geom_hline(yintercept = 0, color = 'salmon3') +
  labs(x = "Satisfação Ajustado", y = 'Resíduos Padronizados',
       title = "Modelo sem transformação")
g2 <- ggplot() +
  geom_point(aes(x = fitted.values(mod_retail_rl2), y = residuo), color = 'cyan4') +
  geom_hline(yintercept = 0, color = 'salmon3') +
  labs(x = "Satisfação Ajustado", y = 'Resíduos Padronizados',
       title = "Modelo com transformação de LOG")
grid.arrange(g1, g2,
             ncol = 2, nrow = 1)
```



```
# Teste de Breusch-Pagan para modelo sem transformação
bptest(mod_retail_rl)
```

```
##
## studentized Breusch-Pagan test
##
## data: mod_retail_rl
## BP = 520.16, df = 10, p-value < 2.2e-16
```

```
# Teste de Breusch-Pagan para modelo sem transformação
bptest(mod_retail_r12)
```

```
##
## studentized Breusch-Pagan test
##
## data: mod_retail_r12
## BP = 191.24, df = 6, p-value < 2.2e-16
```

Graficamente os pontos parecem apresentar um padrão nos dois modelos, como a forma de um losango por exemplo, que sugere uma possível heterocedasticidade.

Verificando formalmente através do teste de Breusch-Pagan, constata-se um *p-valor* menor que 5% nos dois modelos, rejeitando a hipótese nula de que a variância dos resíduos é constante, e consequentemente a variância da variável dependente não é constante. Portanto, existe heterocedasticidade nos dois modelos, quebrando esse pressuposto.

Como o teste de vários pressupostos falharam para os dois modelos, eles não serão utilizado para previsão do Faturamento das Lojas. Acrescenta-se o baixo nível do *R²* obtido, menor que 60% nos dois modelos.

Seguiremos com a avaliação de performance do modelo para fins didáticos. Optou-se por utilizar o modelo sem transformação, pois possui um *R²* melhor.

```
pred_retail_rl <- predict(mod_retail_rl, newdata = dt_retail_test)
R2_Score(y_pred = pred_retail_rl, y_true = dt_retail_test$Weekly_Sales)
```

```

## [1] 0.1728032

RMSE(y_pred = pred_retail_rl, y_true = dt_retail_test$Weekly_Sales)

## [1] 506072.8

```

O modelo gerado obteve um R^2 de 0,1728 e um $RMSE$ de 506.072,8.

Random Forest

O modelo seguinte a ser treinado será o *Random Forest* para regressão, utilizando o pacote **CARET**.

```

# Ajustando o processamento paralelo em CORES (NÚCLEOS)
cl <- makePSOCKcluster(cores)
registerDoParallel(cl)

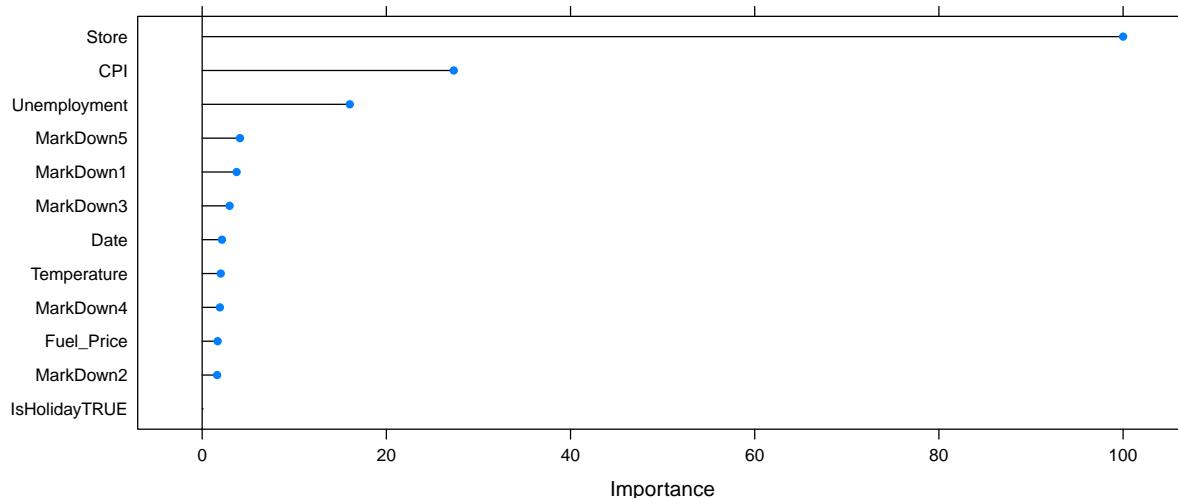
# Treinando o modelo
set.seed(314)
cv <- trainControl(method = "cv", number = 10, savePredictions = TRUE,
                     summaryFunction = defaultSummary)

mod_retail_rf <- train(Weekly_Sales ~ .,
                       data = dt_retail_train,
                       method = "rf",
                       metric = 'RMSE',
                       trControl = cv,
                       tuneLength = 5)

# Parando o processamento paralelo
stopCluster(cl)

varImp_rf <- varImp(mod_retail_rf)
plot(varImp_rf)

```



As três variáveis mais importantes do modelo, em ordem de importância, são **Store** (ID da Loja), **CPI** (Consumer Price Index - Inflação) e **Unemployment** (Taxa de Desemprego).

```
pred_retail_rf <- predict(mod_retail_rf, newdata = dt_retail_test)
R2_Score(y_pred = pred_retail_rf, y_true = dt_retail_test$Weekly_Sales)
```

```
## [1] 0.9417
```

```
RMSE(y_pred = pred_retail_rf, y_true = dt_retail_test$Weekly_Sales)
```

```
## [1] 134351.6
```

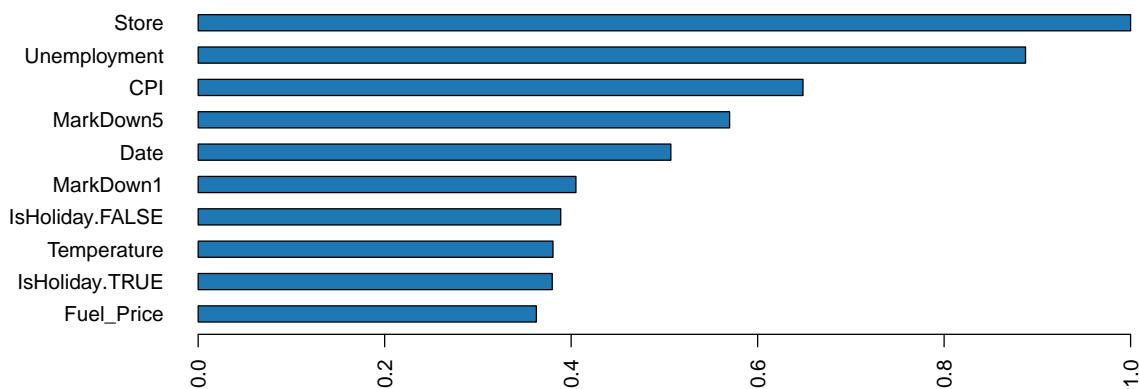
O modelo gerado obteve um *R2* de *0,9417* e um *RMSE* de *135.551,6*, bem superior ao modelo de *Regressão Linear*.

Rede Neural

O modelo seguindo a ser treinado é o de *Rede Neural*, utilizando o pacote **H2O**.

```
## |  
## |  
## |
```

Variable Importance: Deep Learning



As três variáveis mais importantes do modelo, em ordem de importância, são **Store** (ID da Loja), **Unemployment** (Taxa de Desemprego) e **CPI** (Consumer Price Index - Inflação).

```
## |
```

```

## H2OResgressionMetrics: deeplearning
##
## MSE: 49626804887
## RMSE: 222770.7
## MAE: 129138.2
## RMSLE: 0.2233751
## Mean Residual Deviance : 49626804887

## [1] 0.8397127

```

O modelo gerado obteve um *R2* de 0,8397 e um *RMSE* de 222.770,7, bem superior ao modelo de *Regressão Linear*, mas inferior ao modelo de *Random Forest*.

Conclusão

Marketing

Fazendo uma avaliação da performance dos modelos de classificação para o *Data Set* de **Marketing**, verifica-se na tabela abaixo o desempenho de acordo com as métricas: *LogLoss*, *AUC*, *Precisão* e *F1 Score*.

Avaliação da Performance dos modelos de Classificação para Marketing:

Modelo	LogLoss	AUC	Precisão	F1 Score
Regressão Logística	0,2123981	93,71%	61,99%	60,39%
Regressão Logística com StepWise	0,2127598	93,69%	62,72%	59,35%
Regressão Logística com GA	0,2130927	93,66%	62,16%	59,59%
Random Forest	0,1871988	94,17%	62,72%	61,39%
Random Forest com GA	0,192436	94,29%	63,12%	61,52%
Random Forest com RFE	0,1976715	94,43%	63,00%	62,35%
eXtreme Gradient Boosting	0,1859924	94,50%	63,03%	62,41%
SVM Linear	0,2250672	93,65%	61,09%	59,06%
SVM Radial	0,2236074	93,49%	62,57%	56,25%
Rede Neural	0,1837795	94,59%	61,89%	64,01%

Os melhores modelos de acordo com a métrica *F1 Score* foram: *Regressão Logística*, *Random Forest*, *eXtreme Gradient Boosting* e *Rede Neural*.

O modelo de *Regressão Logística*, apesar de um *LogLoss* de 0,212391, que mostra um erro na capacidade de identificar um cliente que irá contratar o plano de um que não, semelhante aos demais medianos (modelos com *LogLoss* maior que 0,2), apresenta capacidade de acertar os clientes que convertem, evidenciado por sua *Precisão* no valor de 61,99%.

Random Forest possui um *LogLoss* muito bom, superior a maioria dos modelos, acrescido de um *F1 Score* maior que 60%. Além disso, utilizando processo de *feature selection* houve uma melhora na previsão do modelo, atingindo *F1 Score* de 62,41% com RFE (Recursive Feature Elimination).

O modelo de *eXtreme Gradient Boosting* possui um *LogLoss* que é o segundo melhor dentre os modelos treinados, perdendo somente para o modelo de *Rede Neural* e também possui um *F1 Score* maior que 60%. Mas notou-se que o tempo gasto no treinamento desse modelo foi maior que no modelo de *Random Forest*.

Rede Neural possui o melhor *LogLoss* dentre os modelos treinados e um *F1 Score* mais alto. Seu ponto de preocupação foi o tempo necessário no treinamento do modelo, que foi bem superior aos demais.

Dentre os melhores modelos, a escolha foi pelo modelo de *Random Forest*, pois possui métricas muito boas em comparação com os melhores modelos e um tempo de treinamento abaixo dos demais, com exceção do

modelo de *Regressão Logística* que possui o menor tempo de treinamento dentre os modelos.

Faturamento das Lojas

Fazendo uma avaliação da performance dos modelos de regressão para o *Data Set* de **Faturamento das Lojas**, verifica-se na tabela abaixo o desempenho de acordo com as métricas: *R2* e *RMSE*.

Avaliação da Performance dos modelos de Regressão para o Faturamento das Lojas:

Modelo	R2	RMSE
Regressão Linear	17,28%	506.072,8
Random Forest	96,17%	135.551,6
Rede Neural	83,97%	222.770,7

Os melhores modelos de acordo com a métrica *F1 Score* foram: *Regressão Logística*, *Random Forest*, *eXtreme Gradient Boosting* e *Rede Neural*.

Os melhores modelos de acordo com a métrica *R2* foram: *Random Forest* e *Rede Neural*.

O modelo de *Random Forest* utilizado para regressão funcionou com boa performance, gerando um alto *R2* com baixo erro medido através do *RMSE*, em comparação com os demais modelos.

Rede Neural também gerou um bom modelo com um valor de *R2* acima dos 80%. Mas em relação ao custo de treinamento, foi superior ao *Random Forest*.

A escolha pelo modelo que será utilizado na previsão do faturamento das lojas recairá novamente no modelo de *Random Forest*. Possui o maior *R2* e menor *RMSE* dentre os modelos testados, aliado ao fato de ser um modelo com mais baixo tempo de treinamento em comparação com o de *Rede Neural*.