Laboratório 4 – Predição e Decisão

ME905

Instruções Gerais

- Baixe os arquivos `produtos_treino.csv`, `produtos_valida.csv` e `produtos_teste.csv` disponíveis em:
  https://drive.google.com/drive/folders/1wLey8vhtYhYPEnI2lqL3FOEhSSJisb_k?usp=sharing
- Utilize qualquer pacote ou função que achar adequado para ajuste dos modelos e validação cruzada, desde que os métodos tenham sido discutidos em aula.
- Além do código, inclua explicações em texto sempre que considerar necessário. Justifique suas escolhas de métodos, parâmetros e comente os resultados obtidos.
- Seu relatório deve ser claro, organizado e bem documentado.

---

1. Leitura dos Dados

Carregue o conjunto de treino (`produtos_treino.csv`), que contém um `data.frame` com:

- **20 colunas (`x1` a `x20`)**: covariáveis descritivas dos produtos (cada linha corresponde a uma unidade);
- **Coluna `y`**: variável resposta, indicando se o produto está com defeito (`y = 1`) ou funcionando (`y = 0`);
- **Coluna `cost`**: custo de compra de cada produto (para os exercícios a partir do 4).

---

2. Ajuste de Modelos

Utilizando os métodos estudados ao longo do semestre, ajuste um modelo para estimar a probabilidade de que um produto esteja com defeito (`y = 1`), com base nas variáveis `x1` a `x20`.

Orientações:

- Utilize apenas os dados de treino (`produtos_treino.csv`) nesta etapa.
- Caso julgue necessário, use validação cruzada para avaliar o desempenho dos modelos.
- Apresente as diferentes abordagens testadas (métodos, escolha de hiperparâmetros, etc).
- Comente as métricas de desempenho obtidas (por exemplo: acurácia), destacando a escolha do modelo final.

```r
dados_treino <- read_csv("produtos_treino.csv")
```

```
## Rows: 1000 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (22): x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
dados_treino$y <- as.factor(dados_treino$y)
dados = dados_treino[-22]
```

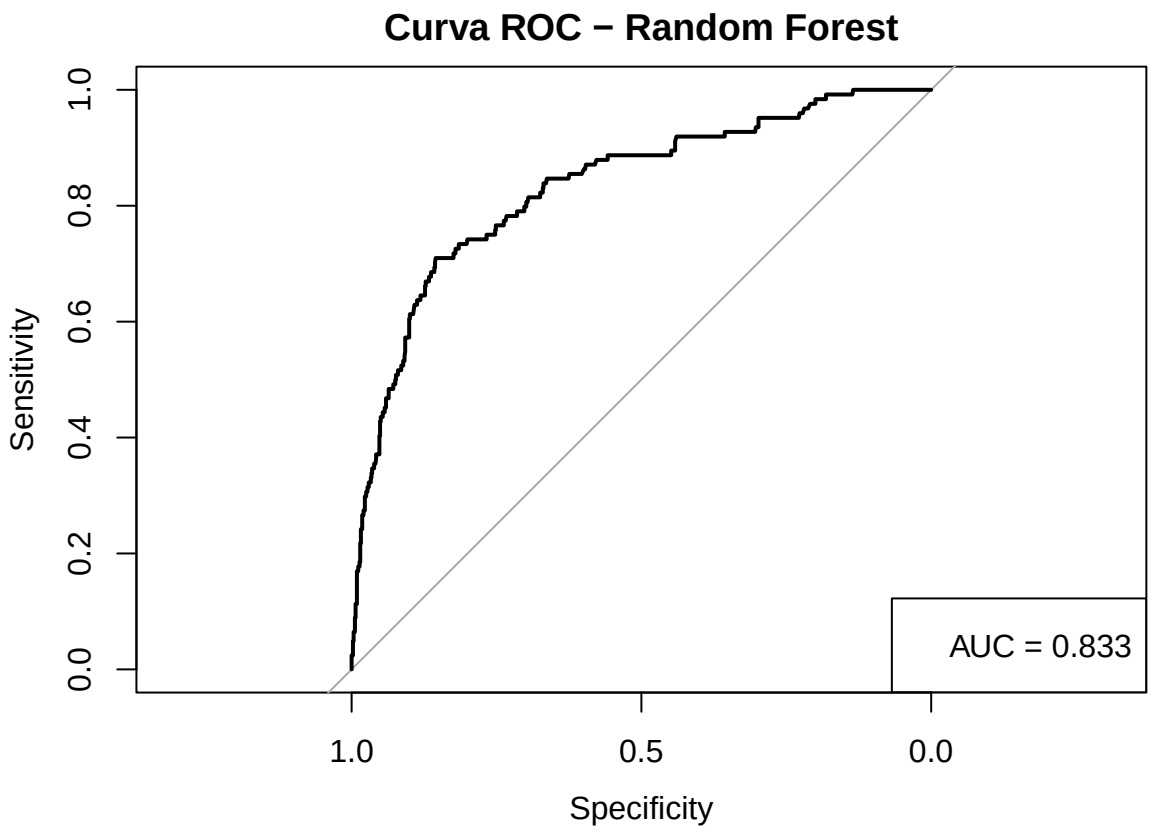
```r
set.seed(42)
modelo_rf <- randomForest(
  y ~ . ,
  data = dados,
  ntree = 500,
  mtry = sqrt(ncol(dados)-1),
  proximity = FALSE
)
```

```
##     0  1 class.error
## 0 867  9  0.01027397
## 1 103 21  0.83064516
```

```
##
## Acurácia: 0.8872539
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
set.seed(42)
svm_model <- svm(y ~ .,
                 data = dados,
                 kernel = "radial",
                 probability = TRUE)

svm_pred <- predict(svm_model, newdata = dados, probability = TRUE)
svm_prob <- attr(svm_pred, "probabilities")[,2]

roc_svm <- roc(dados$y, svm_prob)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
auc_svm <- auc(roc_svm)

best_thresh_svm <- coords(roc_svm, "best", ret = "threshold")$threshold

svm_class <- ifelse(svm_prob > best_thresh_svm, 1, 0) %>% as.factor()
print(confusionMatrix(svm_class, dados$y, positive = "1"))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   0   1
##          0 832   1
##          1  44 123
##
##                Accuracy : 0.955
##                  95% CI : (0.9402, 0.967)
##     No Information Rate : 0.876
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8197
##
##  Mcnemar's Test P-Value : 3.825e-10
##
##             Sensitivity : 0.9919
##             Specificity : 0.9498
##          Pos Pred Value : 0.7365
##          Neg Pred Value : 0.9988
##              Prevalence : 0.1240
##          Detection Rate : 0.1230
##    Detection Prevalence : 0.1670
##       Balanced Accuracy : 0.9709
##
##        'Positive' Class : 1
##
```

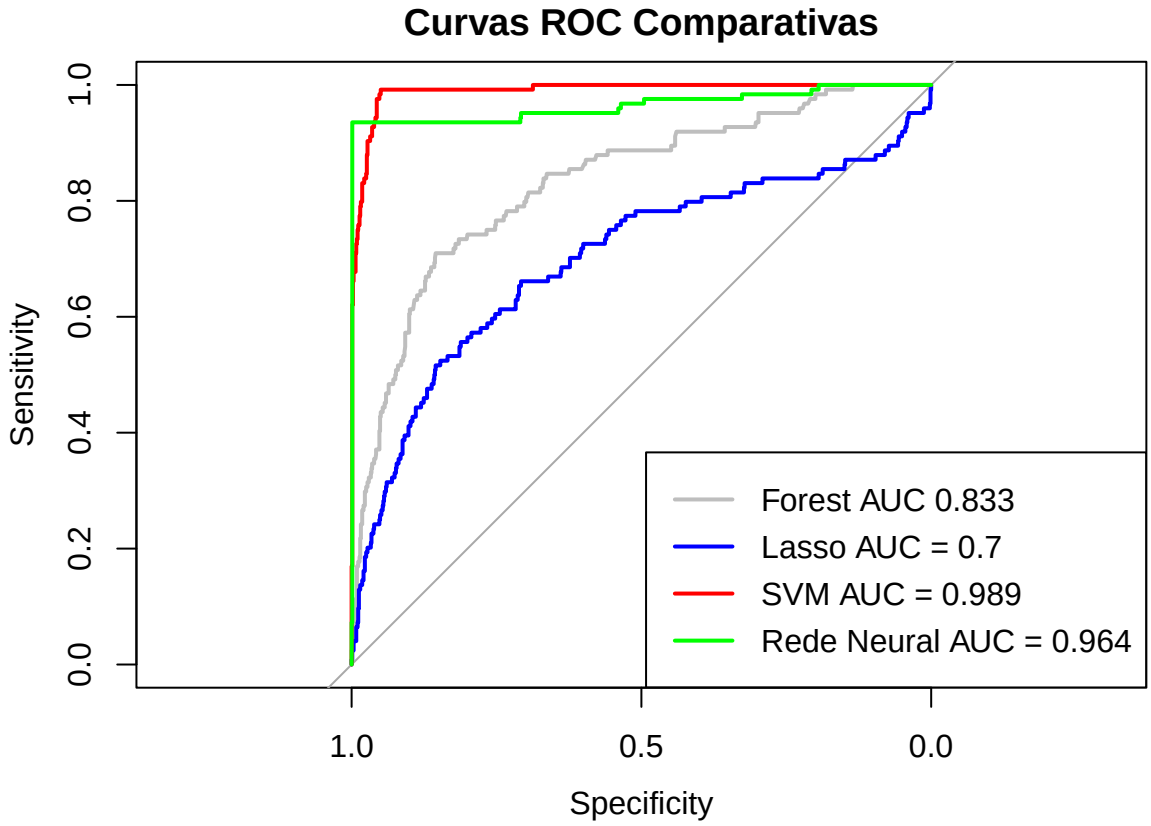
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 875   8
##          1   1 116
##
##                Accuracy : 0.991
##                  95% CI : (0.983, 0.9959)
##     No Information Rate : 0.876
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9575
##
##  Mcnemar's Test P-Value : 0.0455
##
##             Sensitivity : 0.9355
##             Specificity : 0.9989
##          Pos Pred Value : 0.9915
##          Neg Pred Value : 0.9909
##              Prevalence : 0.1240
##          Detection Rate : 0.1160
##    Detection Prevalence : 0.1170
```

```
##        Balanced Accuracy : 0.9672
##
##         'Positive' Class : 1
##
```

Modelo	AUC	Sensibilidade	Especificidade
Random Forest	0.8325968	0.7096774	0.7096774
Lasso	0.6995415	0.5241935	0.5241935
SVM	0.9894038	0.9919355	0.9919355
Rede Neural	0.9644554	0.9354839	0.9354839




```r
set.seed(42)
  final_model <-svm(y ~ .,
                    data = dados,
                    kernel = "radial",
                    probability = TRUE)
```

2. Interpretação : Ajuste de Modelos • Testamos alguns modelos diferentes para prever a chance de um produto estar com defeito. A ideia foi ver qual deles funcionava melhor com base em métricas como acurácia, AUC (que mede a qualidade geral da classificação), sensibilidade e especificidade(é importante ressaltar que Acúracia e sensitividade andam juntos). Random Forest: Teve uma acurácia de quase 89%. Acertou muito bem os produtos bons, mas não foi tão bom em identificar os com defeito. • Lasso: Foi mais simples e teve desempenho inferior, com uma acurácia de 80% e uma sensibilidade

bem baixa. • SVM(Singular Vector Machines): Foi o melhor muito bom, com 95% de acurácia e um AUC de 0.989. Conseguiu detectar quase todos os produtos com defeito. • Rede Neural: Foi muito bom. Acertou quase tudo, com uma AUC de 0.964 e acurácia de 99%.

Em termos de desempenho o modelo SVM e o de Rede Neural apresentaram resultados muito parecidos. O SVM detectou melhor os produtos com defeito, por isso apresentou um AUC maior. Já a Rede Neural teve uma acurácia geral maior, porém detectou pior os produtos com defeito. Por detectar melhor os produtos defeituosos o modelo SVM foi escolhido como modelo final.

3. Predição e Avaliação

Utilize o modelo final para gerar predições no conjunto de validação (`produtos_valida.csv`).

Tarefas:

- Calcule as probabilidades previstas de `y = 1` para os produtos da validação.
- Escolha um critério de classificação para converter as probabilidades em predições binárias.
- Construa a matriz de confusão e comente os resultados.

```
probabilidades1 <- predict(final_model, newdata = dados_validacao, probability =TRUE)
probabilidades = attr(probabilidades1, "probabilities")[,2]

roc1 <- roc(response = dados_validacao$y,
               predictor = probabilidades,
               levels = c("0", "1"))
```

```
## Setting direction: controls < cases
```

```
best_thresh1 <- coords(roc1, "best", best.method = "youden")$threshold


predicoes <- ifelse(probabilidades > best_thresh1, 1, 0) %>%
  factor(levels = c("1", "0"))
```

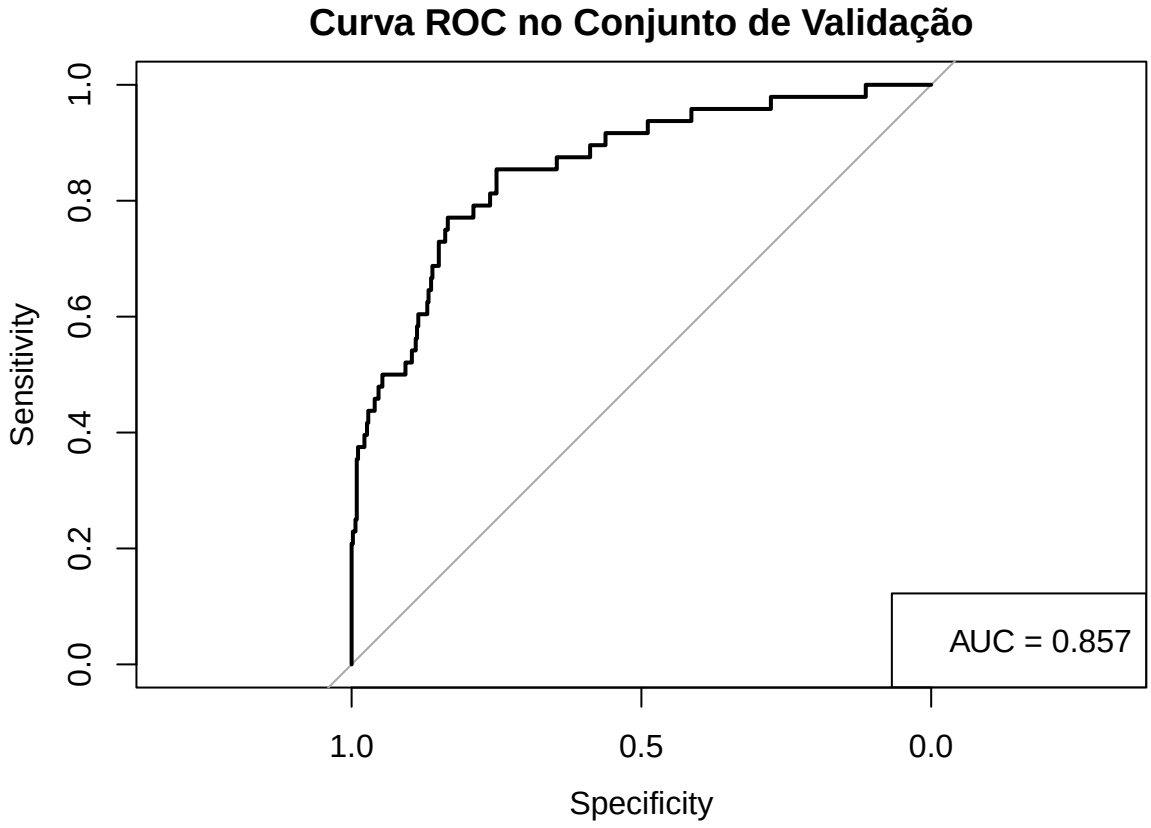
```
## Warning in confusionMatrix.default(data = predicoes, reference =
## dados_validacao$y, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 377  11
##          1  75  37
##
##                Accuracy : 0.828
##                  95% CI : (0.792, 0.8601)
##     No Information Rate : 0.904
```

```
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.379
##
##  Mcnemar's Test P-Value : 1.095e-11
##
##             Sensitivity : 0.7708
##             Specificity : 0.8341
##          Pos Pred Value : 0.3304
##          Neg Pred Value : 0.9716
##              Prevalence : 0.0960
##          Detection Rate : 0.0740
##    Detection Prevalence : 0.2240
##       Balanced Accuracy : 0.8025
##
##        'Positive' Class : 1
##
```

	Acurácia	Sensibilidade	Especificidade	Valor_Preditivo_Positivo	Valor_Preditivo_Negativo
Accuracy	0.828	0.7708333	0.8340708	0.3303571	0.9716495

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



modelo se sai diante de exemplos que nunca viu antes.

Para transformar as probabilidades previstas em decisões (defeituoso ou não), testamos vários thresholds (limiares de corte) — valores a partir dos quais dizemos que a previsão é 1 (defeito). O objetivo era encontrar um ponto que equilibrasse dois fatores importantes:

Sensibilidade (detectar os produtos defeituosos).

Especificidade (não classificar produtos bons como ruins).

Ou seja, queríamos evitar ao máximo:

Comprar produtos defeituosos (prejuízo!),

Deixar de comprar produtos bons (perda de oportunidade).

O threshold escolhido foi aquele que maximizou tanto a sensibilidade quanto a especificidade.

Resultados no Conjunto de Validação Aplicamos o modelo nos dados de validação e avaliamos seu desempenho com as principais métricas de classificação. Observamos o seguinte:

Acurácia: 82,8% dos produtos foram corretamente classificados.

Sensibilidade: 77% dos produtos com defeito foram corretamente identificados.

Especificidade: 83,8% dos produtos funcionando foram corretamente reconhecidos como bons.

AUC (Área sob a Curva ROC): 0.857, indicando ótima capacidade do modelo em distinguir produtos bons dos defeituosos.

O Que Isso Significa Mesmo sem ter visto esses produtos antes, o modelo conseguiu manter uma boa performance. Isso mostra que o SVM generalizou bem o aprendizado: ele não apenas "decorou" os exemplos de treino, mas captou padrões que se repetem em novos dados.

A sensibilidade um pouco menor sugere que ainda podemos deixar passar alguns produtos defeituosos, mas o alto valor de especificidade mostra que raramente erramos ao comprar produtos bons.

A AUC de 0.857 é especialmente encorajadora — ela resume bem o desempenho geral do modelo, e valores acima de 0.85 indicam um classificador sólido.

Sendo assim o modelo SVM ajustado se mostrou bastante eficaz em prever a qualidade dos produtos. Com a escolha cuidadosa do threshold, conseguimos um bom equilíbrio entre evitar prejuízos e aproveitar boas oportunidades de compra. A validação com novos dados confirmou que a abordagem tem potencial para ser aplicada em cenários reais.

4. Decisão Baseada em Custos

Suponha que:

- Cada produto pode ser comprado pelo valor especificado na variável `cost`;
- Produtos funcionando (`y = 0`) podem ser revendidos por **110**;
- Não é possível saber previamente se um produto está quebrado.

Tarefa:

- Determine uma regra de decisão, com base nas probabilidades preditas, que maximize o lucro esperado.
- Sua regra deve indicar, para cada produto, se ele deve ser comprado ou não.

```r
prob_defeito1 <- predict(final_model, newdata = dados, probability = TRUE)
prob_defeito = attr(prob_defeito1,"probabilities")[,2]

predicoes1 <- ifelse(prob_defeito > best_thresh1, 1, 0) %>%
  factor(levels = c("1", "0"))

lucro_esperado <- (1 - prob_defeito) * preco_revenda - custo_compra
```

```r
decisao <- ifelse(lucro_esperado > 0, "Comprar", "Não Comprar")

resultados_treino <- dados_treino %>%
  mutate(
    prob_defeito = prob_defeito,
    lucro_esperado = lucro_esperado,
    decisao = factor(decisao)
  )
```

```
##
##     Comprar Não Comprar
##        0.86        0.14
```

```r
lucro_total <- sum(lucro_esperado[decisao == "Comprar"])
cat("Lucro Total Esperado:", lucro_total, "\n")
```

```
## Lucro Total Esperado: 9365.434
```

Interpretação : Decisão Baseada em Lucro Esperado Com o modelo ajustado, a ideia foi decidir se vale a pena comprar um produto ou não. Pra isso, foi calculado o lucro esperado: se o produto tiver pouca chance de estar com defeito e for barato, compensa comprar. Se não, é melhor evitar. Nos dados de treino: - Recomendamos a compra de 86% dos produtos. - O lucro total esperado foi de cerca de R$ 9 mil.

Como a escolha do modelo levou em consideração minimizar o número de produtos defeitusos comprados é esperado que caso que se queria vender mais caro os produtos mantendo a demanda constante, o lucro aparanta crescer em uma escala quádratica

5. Validação da Regra de Decisão

Aplique sua regra de decisão ao conjunto de validação (`produtos_valida.csv`).

Responda às perguntas:

- Quantos produtos seriam comprados? Quantos não seriam?
- Qual seria o lucro (ou prejuízo) total ao final do processo de compra e revenda?
- Comente os resultados, avaliando se a decisão parece vantajosa.

```
## Produtos que SERIAM comprados: 385

## Produtos que NÃO seriam comprados: 115

## Lucro Total Esperado: 4340.205

## Lucro Real Observado: 5539.68

##          Decisão
## Real      Comprar Não Comprar
##   Bom         373          79
##   Defeito      12          36

##
## Precisão das Compras: 0.969
## Recall dos Produtos Bons: 0.825
## Especificidade: 0.75

##
## --- Análise de Lucro por Decisão ---

## Média de lucro para 'Comprar': 11.27

## Média de lucro para 'Não Comprar': -25.03
```

5.Interpretação : Validação da Regra de Compra

Depois de treinar o modelo e definir a melhor regra de decisão com base nas probabilidades previstas, aplicamos essa lógica ao conjunto de validação. Cada produto foi avaliado individualmente, e a decisão foi simples: comprar ou não, com base no lucro esperado.

Resultado da Simulação 385 produtos seriam comprados com base na regra.

115 produtos seriam rejeitados, considerados de risco ou com baixo lucro esperado.

O lucro estimado inicialmente era de R$ 4.340,21.

No entanto, ao aplicar de fato a decisão aos dados com a variável resposta real, o lucro observado foi maior: R$ 5.539,68.

Ou seja, a estimativa foi conservadora — o modelo previu um lucro mais baixo do que realmente se obteve. Isso indica que a regra de decisão foi eficaz e que o modelo está bem ajustado para o problema.

Avaliando a Qualidade das Decisões A regra de decisão foi muito precisa:

96,9% dos produtos comprados estavam realmente bons. Isso mostra uma taxa altíssima de acerto nas decisões de compra.

A capacidade de evitar produtos defeituosos foi de 75%, indicando que, em 3 de cada 4 vezes, conseguimos evitar um prejuízo potencial.

A estratégia adotada — priorizar a redução de compras erradas (produtos defeituosos) — parece ter sido acertada. Afinal, o prejuízo de comprar um produto com defeito é mais alto do que o ganho ao revender um produto bom. Nesse tipo de decisão, é mais seguro errar por excesso de cautela do que correr riscos desnecessários.

O modelo foi treinado com esse princípio em mente, e os resultados mostram que ele aprendeu bem a fazer distinção dos produtos com maior potencial de retorno positivo.

Sendo assim a aplicação da regra de decisão baseada em lucro esperado foi um sucesso. Mesmo com uma estimativa inicial mais baixa, o lucro final foi ainda maior, e a taxa de erro nas compras foi mínima. Esse desempenho reforça que o modelo pode ser uma ferramenta útil em contextos de decisão financeira, ajudando a maximizar ganhos e reduzir perdas de forma inteligente.

6. Aplicação a Novos Dados

O arquivo `produtos_teste.csv` contém um novo conjunto de produtos, **sem a variável resposta `y`**.

Tarefa:

- Aplique o modelo treinado e sua regra de decisão a esse conjunto.
- Gere um arquivo `.csv` contendo apenas uma coluna chamada `d`, com as seguintes codificações:
  - `d = 1`: decisão de **comprar** o produto;
  - `d = 0`: decisão de **não comprar**.

Exemplo de saída:

d
1
0
0

d

1

---

6. Novos Produtos (Teste) Por último foi aplicado o modelo em um novo conjunto de dados, onde nãos abemos se o produto está com defeito. A regra foi a mesma: calcular a chance de defeito, o lucro esperado e decidir se compensa comprar. O resultado final foi salvo em um arquivo .csv , com 1 para "comprar" e 0 para "não comprar".

Conclusão O SVM foi o modelo que melhor se saiu e por isso foi escolhida. A estratégia de compra com base em lucro esperado funciona bem e ajuda a tomar decisões mais racionais. Mas é bom ter cuidado, pois mesmo um modelo bom pode errar bastante quando aplicado em dados novos. Vale a pena testar mais formas de validação ou ajustar melhor os parâmetros para evitar perdas no futuro.

7. Entrega

- Submeta no Moodle:

  1. Seu **relatório completo** (em PDF gerado a partir deste Rmd).
  2. O **arquivo `.csv` com as decisões** (`d`) para o conjunto de teste.