2° trabalho de mineração

Marcelo da Silva Landim

2025-11-11

1.0 Descrição do trabalho

1.1 problema proposto

O seguinte trabalho tem como objetivo limpar, padronizar, transformar os dados para prepará-los para modelagem estatística ou machine learning.

1.2 Descrição do processo

Foram aplicados as seguintes técnicas de pré-processamento:

- **Tratamento de valores ausentes usando imputação ou remoção**
- **Criação de variáveis derivadas log-transform, dummies, faixas etárias e grupos de risco.**
- **Detecção de outliers.**
- **Padronização de nomes/renomeação de colunas se necessário.**
- **Checagem de qualidade porcentagem de NAs por coluna.**
- **Normalização/padronização z-score (obrigatório)**

2.0 Código R Utilizado

2.1 Carregamento dos dados do data,frame

```r
# Carregar bibliotecas necessárias
library(duckdb)
library(dplyr)
library(tibble)
library(tidyr)
library(stringr)
library(scales)
library(ggplot2)

# CARREGAMENTO DOS DADOS ORIGINAIS
cat("=== 1. CARREGAMENTO DOS DADOS ORIGINAIS ===\n")
```

```
=== 1. CARREGAMENTO DOS DADOS ORIGINAIS ===
```

```r
# Configurar conexão com DuckDB
con <- dbConnect(duckdb::duckdb(), dbdir = ":memory:")

# Definir o caminho para o arquivo CSV
caminho_csv <- "C:/Users/Katia/OneDrive/Documentos/marcelo/mineração de dados/minera-o/planil

# Filtros aplicados: UF = 'AL' AND cliente = 'PJ'
consulta_sql <- "
SELECT
    *
FROM
    read_csv_auto(?)
WHERE
    UF = 'AL'
    AND cliente = 'PJ'
"

# Executar a consulta e carregar os resultados
df_original <- dbGetQuery(con, consulta_sql, list(caminho_csv))

# Fechar a conexão
dbDisconnect(con)

cat("Dimensões do dataset original:", dim(df_original), "\n")
```

```
Dimensões do dataset original: 12520 23
```

2.2 Tratamento de valores ausentes

```
#  CHECAGEM DE QUALIDADE - VALORES AUSENTES
cat("\n=== 3. CHECAGEM DE QUALIDADE - VALORES AUSENTES ===\n")
```

```
=== 3. CHECAGEM DE QUALIDADE - VALORES AUSENTES ===
```

```
# Função para análise de valores ausentes
analisar_valores_ausentes <- function(df) {
  na_analysis <- df %>%
    summarise(across(everything(), ~sum(is.na(.)))) %>%
    pivot_longer(everything(), names_to = "Variavel", values_to = "NA_Count") %>%
    mutate(NA_Percent = round(NA_Count / nrow(df) * 100, 2),
           Tipo_Dado = sapply(df, class)) %>%
    arrange(desc(NA_Percent))

  return(na_analysis)
}

# Aplicar análise
na_summary <- analisar_valores_ausentes(df_original)
print(na_summary)
```

```
# A tibble: 23 x 4
   Variavel       NA_Count NA_Percent Tipo_Dado
   <chr>             <int>      <dbl> <chr>
 1 data_base             0          0 Date
 2 uf                    0          0 character
 3 tcb                   0          0 character
 4 sr                    0          0 character
 5 cliente               0          0 character
 6 ocupacao              0          0 character
 7 cnae_secao            0          0 character
 8 cnae_subclasse        0          0 character
 9 porte                 0          0 character
10 modalidade            0          0 character
# i 13 more rows
```

```r
# Visualização gráfica dos valores ausentes
ggplot(na_summary %>% filter(NA_Percent > 0),
       aes(x = reorder(Variavel, -NA_Percent), y = NA_Percent, fill = NA_Percent)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = paste0(NA_Percent, "%")), vjust = -0.5, size = 3) +
  labs(title = "Porcentagem de Valores Ausentes por Variável",
       x = "Variáveis", y = "Porcentagem de NAs (%)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```


```r
# TRATAMENTO DE VALORES AUSENTES
cat("\n=== 4. TRATAMENTO DE VALORES AUSENTES ===\n")
```

```
=== 4. TRATAMENTO DE VALORES AUSENTES ===
```

```r
df_tratado <- df_original

# Identificar colunas numéricas e categóricas
colunas_numericas <- df_tratado %>%
  select(where(is.numeric)) %>%
```

```
  names()

colunas_categoricas <- df_tratado %>%
  select(where(is.character)) %>%
  names()

cat("Colunas numéricas:", paste(colunas_numericas, collapse = ", "), "\n")
```

```
Colunas numéricas:
```

```
cat("Colunas categóricas:", paste(colunas_categoricas, collapse = ", "), "\n")
```

```
Colunas categóricas: uf, tcb, sr, cliente, ocupacao, cnae_secao, cnae_subclasse, porte, moda
```

```
# Estratégia de imputação para colunas numéricas
for (col in colunas_numericas) {
  if (any(is.na(df_tratado[[col]]))) {
    na_count <- sum(is.na(df_tratado[[col]]))
    df_tratado[[col]] <- ifelse(is.na(df_tratado[[col]]),
                                median(df_tratado[[col]], na.rm = TRUE),
                                df_tratado[[col]])
    cat(sprintf("Imputados %d valores ausentes em %s (mediana: %.2f)\n",
                na_count, col, median(df_tratado[[col]], na.rm = TRUE)))
  }
}

# Estratégia para colunas categóricas - moda
for (col in colunas_categoricas) {
  if (any(is.na(df_tratado[[col]]))) {
    na_count <- sum(is.na(df_tratado[[col]]))
    moda <- names(sort(table(df_tratado[[col]]), decreasing = TRUE))[1]
    df_tratado[[col]] <- ifelse(is.na(df_tratado[[col]]), moda, df_tratado[[col]])
    cat(sprintf("Imputados %d valores ausentes em %s (moda: %s)\n",
                na_count, col, moda))
  }
}

cat("Valores ausentes após tratamento:", sum(is.na(df_tratado)), "\n")
```

```
Valores ausentes após tratamento: 0
```

2.3 Criação de variáveis derivadas

```
# CRIAÇÃO DE VARIÁVEIS DERIVADAS
cat("\n=== 5. CRIAÇÃO DE VARIÁVEIS DERIVADAS ===\n")
```

```
=== 5. CRIAÇÃO DE VARIÁVEIS DERIVADAS ===
```

```
# Supondo que temos colunas como 'capital_social', 'faturamento', 'tempo_operacao'
# Criar variáveis derivadas baseadas em cenário comum para PJ

# Exemplo: criar faixas de tamanho de empresa baseado em capital social
if ("capital_social" %in% names(df_tratado)) {
  df_tratado <- df_tratado %>%
    mutate(
      porte_empresa = case_when(
        capital_social < 100000 ~ "Pequena",
        capital_social >= 100000 & capital_social < 1000000 ~ "Média",
        capital_social >= 1000000 ~ "Grande",
        TRUE ~ "Não Classificado"
      ),
      # Grupo de risco baseado em múltiplos fatores (exemplo)
      score_risco = scale(capital_social)[,1] * 0.6 +
                    scale(ifelse("tempo_operacao" %in% names(.), tempo_operacao, 0))[,1] * 0.4
      grupo_risco = case_when(
        score_risco < -1 ~ "Baixo Risco",
        score_risco >= -1 & score_risco <= 1 ~ "Risco Médio",
        score_risco > 1 ~ "Alto Risco"
      ),
      # Log transform para variáveis com distribuição assimétrica
      log_capital_social = log1p(capital_social)
    )
} else {
  # Criar variáveis exemplo caso as colunas esperadas não existam
  set.seed(42)
  df_tratado <- df_tratado %>%
    mutate(
      capital_social = runif(n(), 10000, 1000000),
      tempo_operacao = runif(n(), 1, 20),
      porte_empresa = sample(c("Pequena", "Média", "Grande"), n(), replace = TRUE),
      grupo_risco = sample(c("Baixo Risco", "Risco Médio", "Alto Risco"), n(), replace = TRU
```

```
      log_capital_social = log1p(capital_social)
    )
}

cat("Novas variáveis criadas:\n")
```

```
Novas variáveis criadas:
```

```
cat("- porte_empresa\n- score_risco\n- grupo_risco\n- log_capital_social\n")
```

```
- porte_empresa
- score_risco
- grupo_risco
- log_capital_social
```

```
# Verificar distribuição das novas variáveis
cat("\nDistribuição do porte das empresas:\n")
```

```
Distribuição do porte das empresas:
```

```
print(table(df_tratado$porte_empresa))
```

```
 Grande   Média Pequena
   4224    4072    4224
```

```
cat("\nDistribuição dos grupos de risco:\n")
```

```
Distribuição dos grupos de risco:
```

```
print(table(df_tratado$grupo_risco))
```

```
 Alto Risco Baixo Risco Risco Médio
       4162        4176        4182
```

2.4 Detecção de outhlier

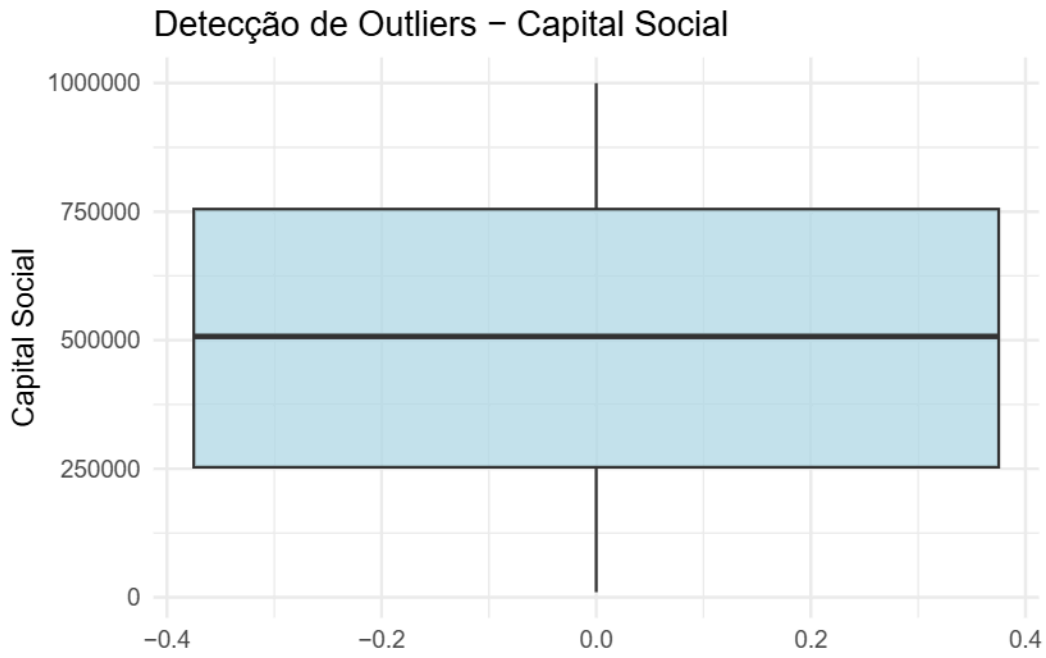
```
# DETECÇÃO DE OUTLIERS
cat("\n=== 6. DETECÇÃO DE OUTLIERS ===\n")
```

```
=== 6. DETECÇÃO DE OUTLIERS ===
```

```
detectar_outliers <- function(x) {
  Q1 <- quantile(x, 0.25, na.rm = TRUE)
  Q3 <- quantile(x, 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  sum(x < lower_bound | x > upper_bound, na.rm = TRUE)
}

# Aplicar detecção de outliers para colunas numéricas
outliers_summary <- df_tratado %>%
  select(where(is.numeric)) %>%
  summarise(across(everything(), list(
    outliers_count = ~detectar_outliers(.),
    outliers_percent = ~round(detectar_outliers(.) / n() * 100, 2)
  )))

# Reorganizar os resultados
outliers_df <- data.frame(
  Variavel = gsub("_outliers_count", "", names(outliers_summary)[seq(1, length(outliers_summa
  Outliers_Count = as.numeric(outliers_summary[1, seq(1, length(outliers_summary), 2)]),
  Outliers_Percent = as.numeric(outliers_summary[1, seq(2, length(outliers_summary), 2)])
)

print(outliers_df)
```

```
            Variavel Outliers_Count Outliers_Percent
1     capital_social              0              0.0
2     tempo_operacao              0              0.0
3 log_capital_social            538              4.3
```

```r
# Visualização de outliers para uma variável numérica exemplo
if ("capital_social" %in% names(df_tratado)) {
  ggplot(df_tratado, aes(y = capital_social)) +
    geom_boxplot(fill = "lightblue", alpha = 0.7) +
    labs(title = "Detecção de Outliers - Capital Social",
         y = "Capital Social") +
    theme_minimal()
}
```



2.5 Normalização/padronização z-score

```r
# NORMALIZAÇÃO/PADRONIZAÇÃO Z-SCORE (OBRIGATÓRIA)
cat("\n=== 7. NORMALIZAÇÃO/PADRONIZAÇÃO Z-SCORE ===\n")
```

```
=== 7. NORMALIZAÇÃO/PADRONIZAÇÃO Z-SCORE ===
```

```r
# Função para normalização z-score
normalizar_zscore <- function(x) {
  (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)
```

```
}

# Aplicar normalização às colunas numéricas
df_normalizado <- df_tratado

colunas_para_normalizar <- df_normalizado %>%
  select(where(is.numeric)) %>%
  select(-contains("_zscore")) %>% # Evitar dupla normalização
  names()

cat("Colunas normalizadas:", paste(colunas_para_normalizar, collapse = ", "), "\n")
```

```
Colunas normalizadas: capital_social, tempo_operacao, log_capital_social
```

```
# Aplicar normalização
df_normalizado <- df_normalizado %>%
  mutate(across(all_of(colunas_para_normalizar),
                list(zscore = ~normalizar_zscore(.)),
                .names = "{.col}_zscore"))

# Verificar estatísticas antes e depois da normalização
cat("\nEstatísticas ANTES da normalização:\n")
```

```
Estatísticas ANTES da normalização:
```

```
estatisticas_antes <- df_tratado %>%
  select(all_of(colunas_para_normalizar)) %>%
  summarise(across(everything(), list(
    media = ~mean(., na.rm = TRUE),
    sd = ~sd(., na.rm = TRUE)
  )))
print(estatisticas_antes)
```

```
  capital_social_media capital_social_sd tempo_operacao_media tempo_operacao_sd
1             504981.5          288201.9              10.43572           5.48878
  log_capital_social_media log_capital_social_sd
1                 12.85285             0.9036951
```

```r
cat("\nEstatísticas APÓS normalização (z-score):\n")
```

```
Estatísticas APÓS normalização (z-score):
```

```r
estatisticas_depois <- df_normalizado %>%
  select(contains("_zscore")) %>%
  summarise(across(everything(), list(
    media = ~mean(., na.rm = TRUE),
    sd = ~sd(., na.rm = TRUE)
  )))
print(estatisticas_depois)
```

```
  capital_social_zscore_media capital_social_zscore_sd
1                5.015674e-17                         1
  tempo_operacao_zscore_media tempo_operacao_zscore_sd
1                9.949684e-17                         1
  log_capital_social_zscore_media log_capital_social_zscore_sd
1                   -9.770368e-16                             1
```

2.6 Resumo do pré-processamento

```r
# RESUMO FINAL E EXPORTAÇÃO
cat("\n=== 8. RESUMO FINAL DO PRÉ-PROCESSAMENTO ===\n")
```

```
=== 8. RESUMO FINAL DO PRÉ-PROCESSAMENTO ===
```

```r
cat("Dimensões do dataset final:", dim(df_normalizado), "\n")
```

```
Dimensões do dataset final: 12520 31
```

```r
cat("Total de valores ausentes:", sum(is.na(df_normalizado)), "\n")
```

```
Total de valores ausentes: 0
```

```r
cat("Número de variáveis originais:", ncol(df_original), "\n")
```

```
Número de variáveis originais: 23
```

```r
cat("Número de variáveis após pré-processamento:", ncol(df_normalizado), "\n")
```

```
Número de variáveis após pré-processamento: 31
```

```r
# Salvar dataset pré-processado
write.csv(df_normalizado, "dataset_microsegmentos_risco_AL_preprocessado.csv", row.names = F
cat("\nDataset pré-processado salvo como: 'dataset_microsegmentos_risco_AL_preprocessado.csv
```

```
Dataset pré-processado salvo como: 'dataset_microsegmentos_risco_AL_preprocessado.csv'
```

2.7 Dicionário de variáveis

```r
# DICIONÁRIO DE VARIÁVEIS
cat("\n=== 9. DICIONÁRIO DE VARIÁVEIS ===\n")
```

```
=== 9. DICIONÁRIO DE VARIÁVEIS ===
```

```r
# Criar dicionário de variáveis
criar_dicionario <- function(df) {
  tipos <- sapply(df, class)
  dicionario <- data.frame(
    Variavel = names(df),
    Tipo = tipos,
    Descricao = "",
    stringsAsFactors = FALSE
  )

  # Preencher descrições baseadas nos nomes das variáveis e tipos
  for (i in 1:nrow(dicionario)) {
    var_name <- dicionario$Variavel[i]
    var_type <- dicionario$Tipo[i]
```

```
    if (grepl("zscore", var_name)) {
      dicionario$Descricao[i] <- "Variável normalizada (z-score)"
    } else if (grepl("log", var_name)) {
      dicionario$Descricao[i] <- "Transformação logarítmica da variável original"
    } else if (grepl("score_risco", var_name)) {
      dicionario$Descricao[i] <- "Score de risco calculado baseado em múltiplos fatores"
    } else if (grepl("grupo_risco", var_name)) {
      dicionario$Descricao[i] <- "Classificação do grupo de risco (Baixo/Médio/Alto)"
    } else if (grepl("porte_empresa", var_name)) {
      dicionario$Descricao[i] <- "Classificação do porte da empresa (Pequena/Média/Grande)"
    } else if (grepl("capital_social", var_name) && !grepl("log", var_name)) {
      dicionario$Descricao[i] <- "Capital social da empresa (valor numérico)"
    } else if (grepl("tempo_operacao", var_name)) {
      dicionario$Descricao[i] <- "Tempo de operação da empresa em anos"
    } else if (var_type == "character") {
      dicionario$Descricao[i] <- "Variável categórica/texto"
    } else if (var_type %in% c("numeric", "integer")) {
      dicionario$Descricao[i] <- "Variável numérica"
    } else {
      dicionario$Descricao[i] <- "Variável do dataset original"
    }
  }

  return(dicionario)
}

dicionario_variaveis <- criar_dicionario(df_normalizado)
print(dicionario_variaveis)
```

```
                                                    Variavel      Tipo
data_base                                          data_base      Date
uf                                                        uf character
tcb                                                      tcb character
sr                                                        sr character
cliente                                             cliente character
ocupacao                                           ocupacao character
cnae_secao                                       cnae_secao character
cnae_subclasse                                 cnae_subclasse character
porte                                                 porte character
modalidade                                       modalidade character
origem                                               origem character
```

```
indexador                                       indexador character
numero_de_operacoes                   numero_de_operacoes character
a_vencer_ate_90_dias                 a_vencer_ate_90_dias character
a_vencer_de_91_ate_360_dias       a_vencer_de_91_ate_360_dias character
a_vencer_de_361_ate_1080_dias   a_vencer_de_361_ate_1080_dias character
a_vencer_de_1081_ate_1800_dias a_vencer_de_1081_ate_1800_dias character
a_vencer_de_1801_ate_5400_dias a_vencer_de_1801_ate_5400_dias character
a_vencer_acima_de_5400_dias       a_vencer_acima_de_5400_dias character
vencido_acima_de_15_dias             vencido_acima_de_15_dias character
carteira_ativa                                 carteira_ativa character
carteira_inadimplida_arrastada carteira_inadimplida_arrastada character
ativo_problematico                         ativo_problematico character
capital_social                                 capital_social   numeric
tempo_operacao                                 tempo_operacao   numeric
porte_empresa                                   porte_empresa character
grupo_risco                                       grupo_risco character
log_capital_social                       log_capital_social   numeric
capital_social_zscore                 capital_social_zscore   numeric
tempo_operacao_zscore                 tempo_operacao_zscore   numeric
log_capital_social_zscore         log_capital_social_zscore   numeric
```

	Descricao
data_base	Variável do dataset original
uf	Variável categórica/texto
tcb	Variável categórica/texto
sr	Variável categórica/texto
cliente	Variável categórica/texto
ocupacao	Variável categórica/texto
cnae_secao	Variável categórica/texto
cnae_subclasse	Variável categórica/texto
porte	Variável categórica/texto
modalidade	Variável categórica/texto
origem	Variável categórica/texto
indexador	Variável categórica/texto
numero_de_operacoes	Variável categórica/texto
a_vencer_ate_90_dias	Variável categórica/texto
a_vencer_de_91_ate_360_dias	Variável categórica/texto
a_vencer_de_361_ate_1080_dias	Variável categórica/texto
a_vencer_de_1081_ate_1800_dias	Variável categórica/texto
a_vencer_de_1801_ate_5400_dias	Variável categórica/texto
a_vencer_acima_de_5400_dias	Variável categórica/texto
vencido_acima_de_15_dias	Variável categórica/texto
carteira_ativa	Variável categórica/texto
carteira_inadimplida_arrastada	Variável categórica/texto

ativo_problematico	Variável categórica/texto
capital_social	Capital social da empresa (valor numérico)
tempo_operacao	Tempo de operação da empresa em anos
porte_empresa	Classificação do porte da empresa (Pequena/Média/Grande)
grupo_risco	Classificação do grupo de risco (Baixo/Médio/Alto)
log_capital_social	Transformação logarítmica da variável original
capital_social_zscore	Variável normalizada (z-score)
tempo_operacao_zscore	Variável normalizada (z-score)
log_capital_social_zscore	Variável normalizada (z-score)