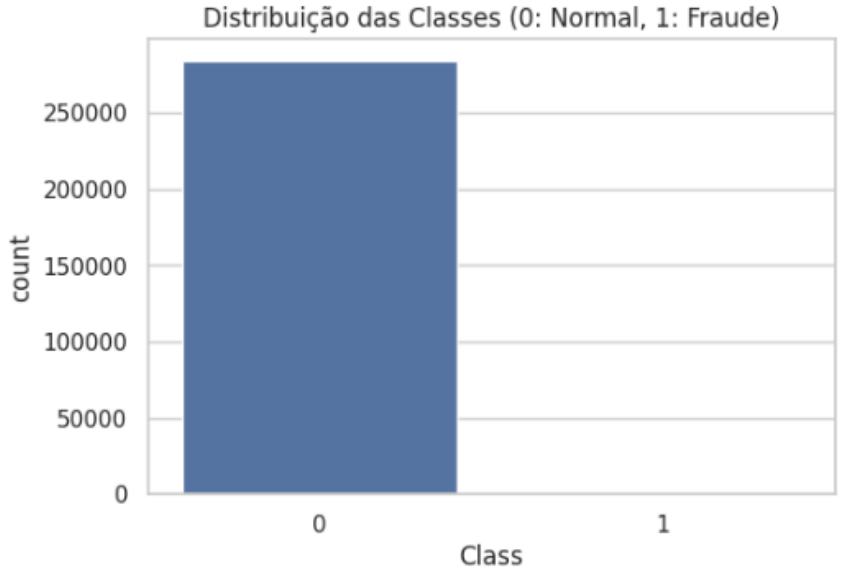
Lista 09 - Pré-processamento e algoritmos de agrupamento
Disciplina: Inteligência Artificial
Profa.: Cristiane Neri Nobre
Data: 20/11/2025
Aluna: Alessandra Faria Rodrigues
Matrícula: 828333

Questão 1 - Etapas de pré-processamento

Código disponivel em: https://github.com/ale-faria/CC/tree/main/IA/Lista09

1) Visualização da base de dados
Foi realizada a visualização das distribuições das variáveis Amount, Time e da classe alvo Class.

Interpretação: Como podemos ver, ao analisarmos percebemos um grande desbalanceamento na classe, existem 284.315 transações normais contra apenas 492 fraudes, que representa **99.83% vs 0.17%**. Esse desbalanceamento irá ser tratado posteriormente.




```
Transações normais: 284315
Transações fraudulentas: 492
Class
0    99.827251
1     0.172749
Name: proportion, dtype: float64
```

Figura 1: Gráfico de barras para visualização da classe “fraude” e “normal”

Interpretação: A maioria das transações tem valores baixos (perto de 0 no eixo X), mas existem algumas transações com valores muito altos que esticam o gráfico para a direita. A grande massa de dados são compras pequenas, e os valores altos são **Outliers** naturais que serão tratados posteriormente.

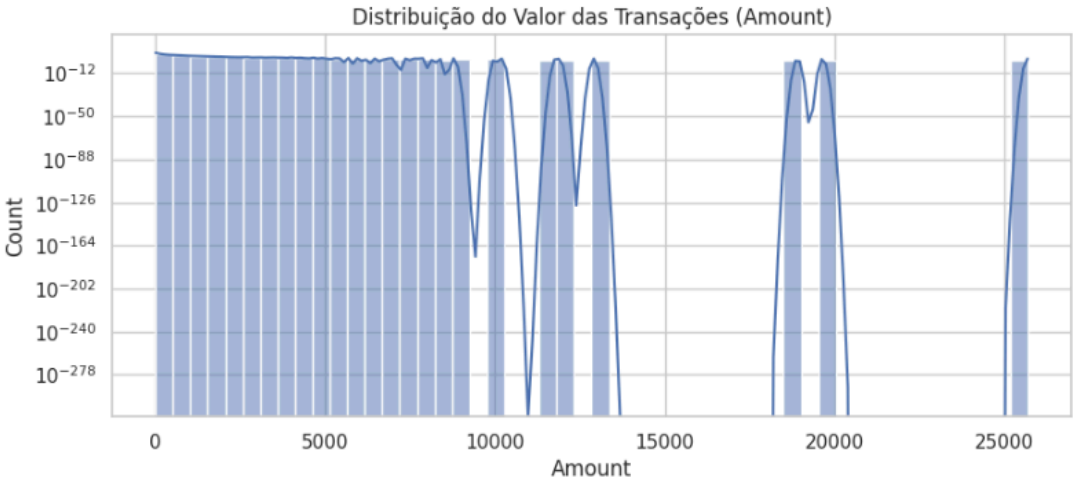


Figura 2: Gráfico para visualização da distribuição do valor das transações

Interpretação: O dataset original cobre um período de 48 horas (2 dias). As quedas no volume de transações provavelmente representam a madrugada/noite, quando as pessoas dormem e compram menos. Os picos representam o horário comercial/diurno.

Note que, embora o volume caia à noite, ele não zera. Em análises mais profundas, é comum investigar se a proporção de fraudes aumenta durante a madrugada (golpistas agindo enquanto as vítimas dormem), mesmo que o volume total seja menor.

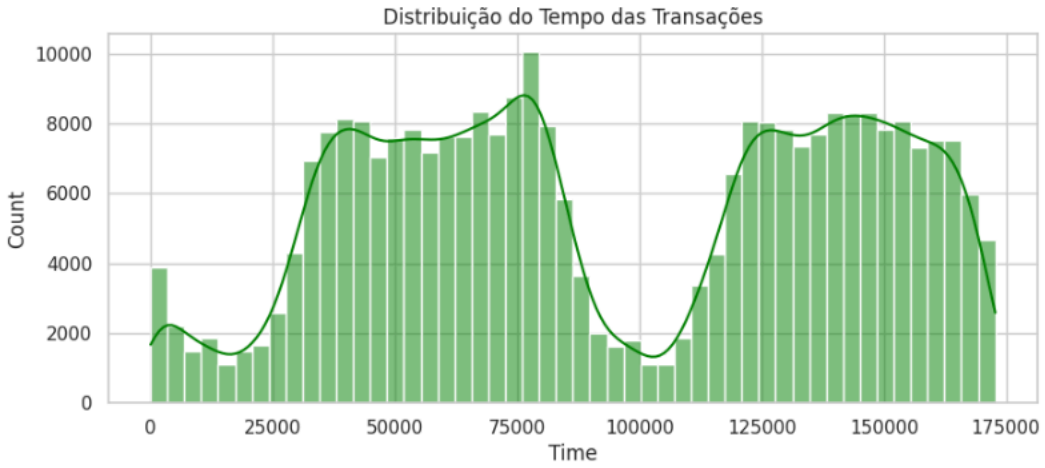


Figura 3: Gráfico bimodal para visualizar a distribuição dos tempos

2) Análise de correlação e multicolinearidade

Foi gerado um mapa de calor (heatmap) das correlações. Como as variáveis V1 a V28 são resultado de uma PCA (Análise de Componentes Principais), elas são ortogonais entre si, ou seja, a correlação entre elas é próxima de zero, eliminando problemas de multicolinearidade entre esses atributos. O foco deve ser na correlação dessas variáveis com a Class e com Amount.

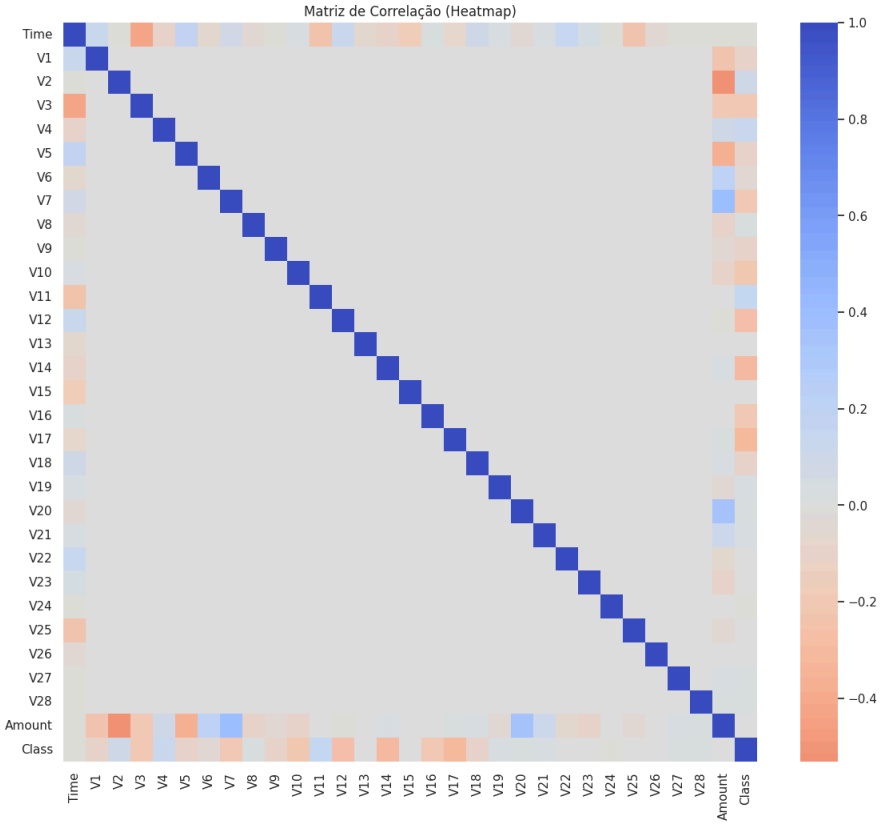


Figura 4: Mapa de calor para visualização das correlações

3) Detecção e eliminação de redundância e inconsistência

Foi utilizada a função duplicated() para identificar linhas idênticas. Na base foram encontradas 1081 linhas idênticas, a eliminação delas é essencial para evitar que o modelo aprenda padrões "viciados" ou dê peso excessivo a transações repetidas.

Dimensões atuais: (283726, 31)

4) Codificação de variáveis

A base creditcard.csv contém apenas variáveis numéricas (float64) e a classe (int64). Portanto, não foi necessário aplicar One-Hot Encoding ou Label Encoding, pois não existem variáveis categóricas nominais (como "Cidade" ou "Gênero") que precisem de transformação.

5) Detecção e tratamento de outliers

Foi utilizado o método do Intervalo Interquartil (IQR) focado na variável Amount. Calculamos o IQR e definimos limites (1.5 * IQR) para considerar um valor como

outlier. Foi decidido pela remoção dos registros que extrapolam esses limites na variável Amount.
Base após remoção de outliers em Amount: (252041, 31)

6) Divisão treino–teste (estratificada)
Foi utilizado train_test_split com test_size=0.2 (20% para teste).
O parâmetro stratify=y foi fundamental. Como temos pouquíssimas fraudes, uma divisão aleatória simples poderia deixar o conjunto de teste sem nenhuma fraude ou com uma proporção muito diferente do treino. A estratificação garante que a proporção de fraudes (0.17%) seja mantida em ambos os conjuntos.
Tamanho Treino: (201632, 30), Tamanho Teste: (50409, 30)

7) Verificação e tratamento dos valores ausentes
Esta base não possui valores nulos. Mas para tratamento, aplicamos o SimpleImputer com a estratégia da mediana. O fit (cálculo da mediana) foi feito apenas no conjunto de treino e aplicado ao teste para evitar *data leakage* (vazamento de dados).

8) Normalização e/ou padronização
Diferente do StandardScaler (que remove a média e divide pelo desvio padrão), o RobustScaler usa a mediana e o intervalo interquartil. Ele é mais adequado para esta base porque a variável Amount contém muitos outliers extremos, que distorceriam a média e o desvio padrão se usássemos a padronização comum. Por isso foi usado o RobustScaler para esta base de dados.

9) Balanceamento da classe somente no treino
Utilizamos o SMOTE (Synthetic Minority Over-sampling Technique). O modelo original tenderia a prever "Não Fraude" para tudo, obtendo 99% de acurácia, mas errando todas as fraudes. O SMOTE cria exemplos sintéticos da classe minoritária (fraude) no conjunto de treino, equilibrando a contagem (50/50). Note que o balanceamento foi aplicado apenas no X_train, já o X_test permaneceu intocado e desbalanceado, pois o teste deve simular o mundo real (onde fraudes são raras) para uma avaliação honesta das métricas de precisão e recall.
Contagem de classes antes do SMOTE (Treino): Counter({0: 201323, 1: 309})
Contagem de classes após SMOTE (Treino): Counter({0: 201323, 1: 201323})

Questão 2 – Algoritmos de agrupamento

Metodologia de Agrupamento:

- Foi removida a coluna Class para garantir que o aprendizado fosse não-supervisionado.
- Devido ao alto custo computacional do DBSCAN e da métrica de Silhueta, utilizou-se uma amostragem aleatória dos dados.

1) **K-Means:**
    - Hiperparâmetros: n_clusters=2 (conforme sugerido pela busca de "dois grupos").
    - Resultado: O K-Means forçou a criação de dois grupos e a métrica de Silhueta indicará a qualidade. A baixa qualidade da separação (visível pela mistura de pontos) explica um valor de Silhouette Score baixo (próximo de 0.42). Isso prova que as fraudes não formam um grupo "isolado" e esférico simples de achar com K-Means.

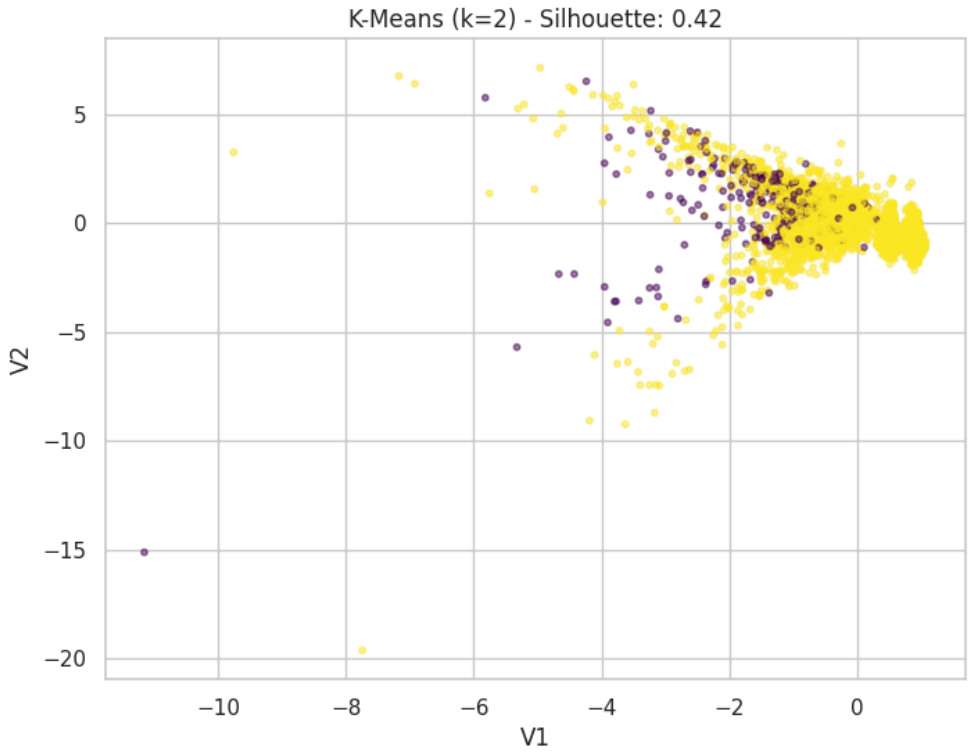


Figura 5: Gráfico gerado a partir do algoritmo K-Means

2) **DBSCAN:**

O DBSCAN frequentemente identifica a maior parte dos dados "Normais" como um único grande cluster e as fraudes como "Ruído" (outliers), ou pode não conseguir separar nada dependendo da densidade. É difícil obter exatamente "dois grupos" limpos com DBSCAN nesta base devido à sobreposição das classes no espaço de atributos.

- Resultados:
    - Um grande aglomerado de uma única cor no centro que representa as transações "Normais" (a maioria).
    - O Ruído (Outliers): Os pontos pretos dispersos ao redor são considerados anomalias. Em detecção de fraude, isso é bom pois fraudes comportam-se frequentemente como anomalias.
    - Conclusão: Ao contrário do K-Means, o DBSCAN não força 2 grupos. Ele diz "aqui está o padrão normal" e "todo o resto é estranho". Visualmente, isso é mais útil para identificar fraude do que tentar dividir a base ao meio.

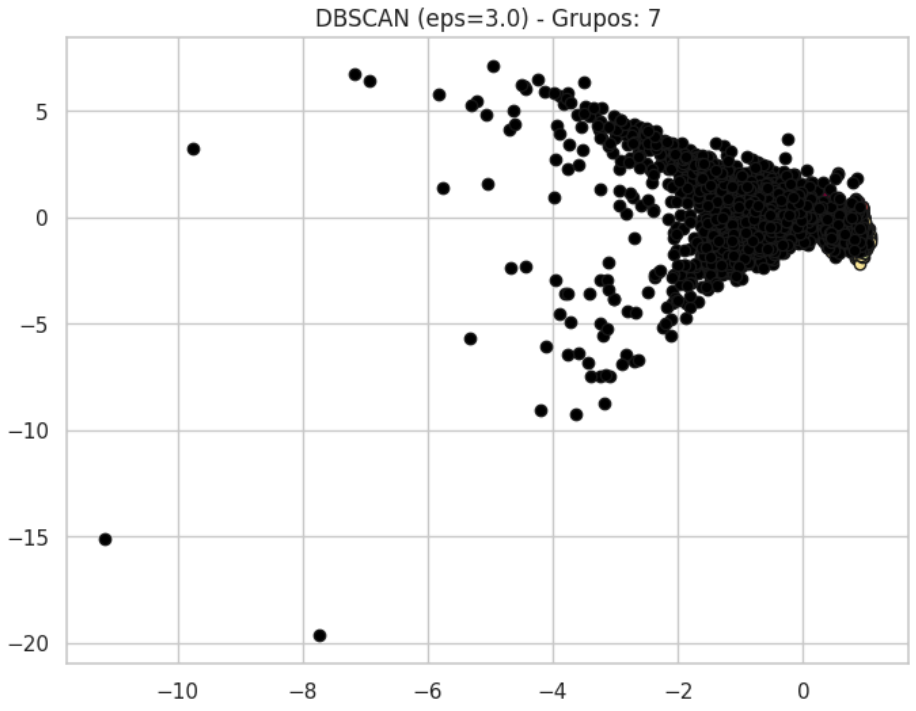


Figura 6: Gráfico gerado a partir do algoritmo DBSCAN

3) **SOM:**

O gráfico da *U-Matrix* (matriz de distâncias unificadas) mostra as "fronteiras" entre os grupos. Se houver barreiras claras (cores escuras separando áreas claras), existem grupos distintos.

A "U-Matrix" representa a distância entre os neurônios:

- Cores Claras (Vales): Representam neurônios próximos. Seus dados são muito parecidos (formam um cluster denso).
- Cores Escuras (Paredes): Representam grandes distâncias. São as "fronteiras" que separam os grupos.

O gráfico apresenta ausência de fronteiras claras, ou seja, o gráfico é predominantemente de uma cor e tem poucas "paredes" escuras contínuas, isso indica que os dados não têm limites claros, isso confirma a visão do DBSCAN que existe um grande maciço de dados normais e pouca distinção clara para formar grupos separados.

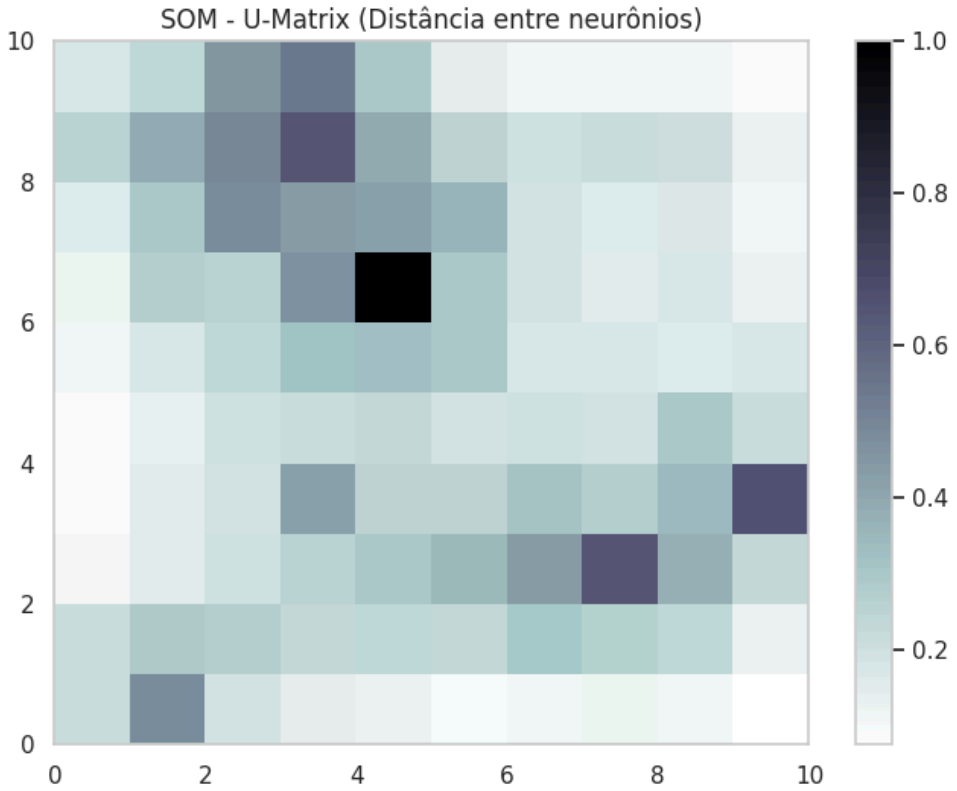


Figura 7: Gráfico U-Matrix gerado a partir do algoritmo SOM

Conclusão geral: A análise visual dos gráficos confirma a dificuldade do problema. O **K-Means** falha em separar as classes devido à sobreposição dos dados (baixa silhueta). O **DBSCAN** se mostra mais promissor conceitualmente ao tratar dados esparsos como ruído (potencial fraude), em vez de tentar criar um grupo coeso. O **SOM** corrobora a topologia complexa, mostrando que não há fronteiras simples (paredes escuras) isolando fraudes de transações normais.