

Lista 9 - IA

Laura Persilva Araújo

Link

<https://github.com/laurapersilva/IA/tree/main/Listas/Lista%209>

OBS: Por favor, considere que a base também está junto ao código no diretório. Não foi possível passá-la para o Github, pois ele não autoriza arquivos maiores que 25MB.

Tarefa

Utilizar os algoritmos de agrupamento KMEANS, DBSCAN e SOM para verificar se a base possui dois grupos. Avaliar a qualidade dos agrupamentos, indicando os valores das métricas de avaliação da qualidade dos grupos obtidos.

Importante: Lembrar de eliminar o atributo de classificação para realizar o agrupamento.

Para isto, é necessário realizar todas as etapas de pré-processamento abaixo:

1) Visualização da base de dados

```
df = pd.read_csv('creditcard.csv')  
df.head()  
df.info()  
df.describe()
```

O que foi observado:

- 31 atributos;
- Nenhum valor ausente;
- Desequilíbrio forte da variável **Class** (fraude = 0.17%).

2) Verificação e tratamento dos valores ausentes

```
df.isnull().sum()
```

Resultado: não existem valores ausentes, nenhuma imputação necessária.

3) Detecção e eliminação de redundância e inconsistência

```
df.duplicated().sum()  
df = df.drop_duplicates()
```

Foram encontradas 1081 linhas duplicadas que foram removidas.

4) Detecção e tratamento de outliers

Métodos aplicados:

- Z-Score > 4 - para detectar valores extremos
- IQR ($Q1 - 1.5 \times IQR$ / $Q3 + 1.5 \times IQR$) - Mais seguro para Amount e Time

Remoção somente quando era outlier para os DOIS métodos

5) Normalização e/ou padronização

```
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
X_scaled = scaler.fit_transform(X_clean)
```

Fundamental para K-Means e DBSCAN, pois dependem de distâncias Euclidianas.

6) Análise de correlação e multicolinearidade

Correlação entre componentes PCA.

VIF acima de 10 significa risco de multicolinearidade, então foi aplicado PCA (10 componentes).

```
from sklearn.decomposition import PCA  
pca = PCA(n_components=10)  
X_pca = pca.fit_transform(X_scaled)
```

Vantagem: reduz ruído e acelera os algoritmos.

7) Codificação de variáveis (One-Hot Encoding ou Label Encoding)

Não foi necessário, pois todos os atributos são numéricos.

8) Balanceamento da classe

O balanceamento NÃO foi feito, pois o agrupamento NÃO deve “saber” que existe a classe fraude. Usei a classe apenas para validar depois, semelhante a “ground truth”.

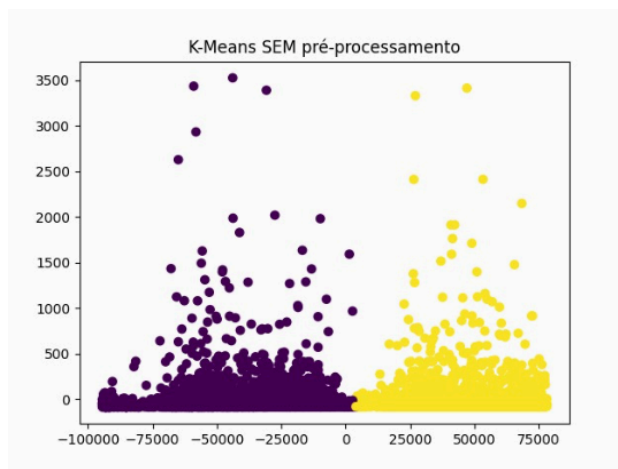
9) Divisão treino–teste (estratificada)

Separação da base ANTES da clusterização para validar:

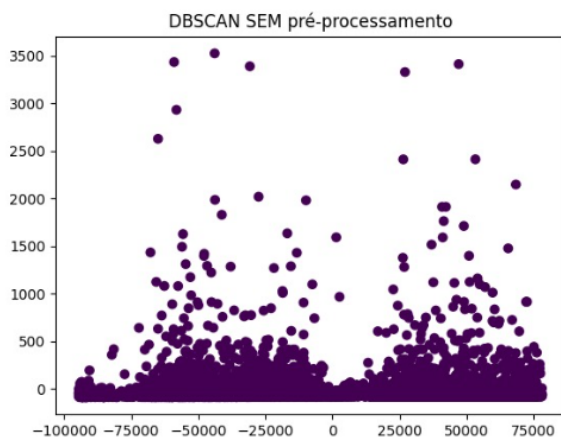
```
from sklearn.model_selection import train_test_split  
X_train, X_test = train_test_split(X_pca, test_size=0.3, random_state=42)
```

Resultado do modelo

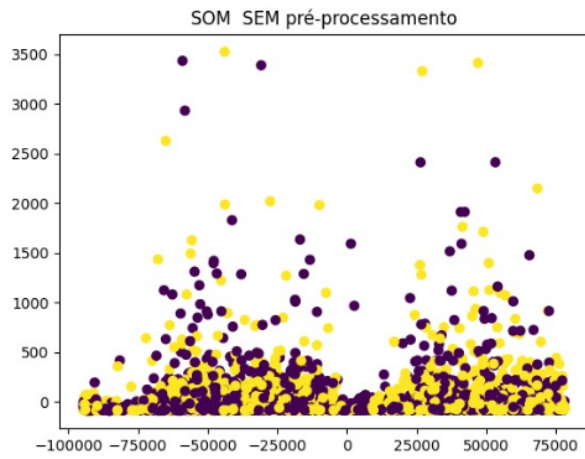
Antes do pré-processamento



KMeans: agrupou de forma difusa, pouca separação.

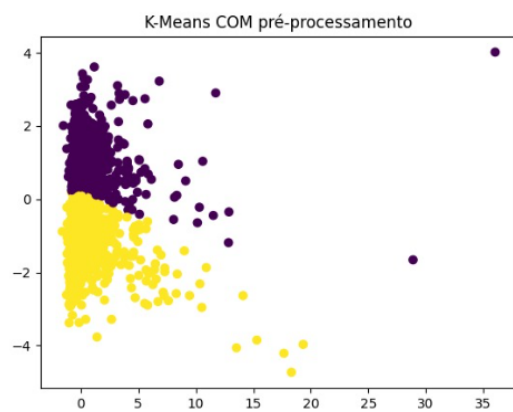


DBSCAN: quase tudo foi detectado como ruído.

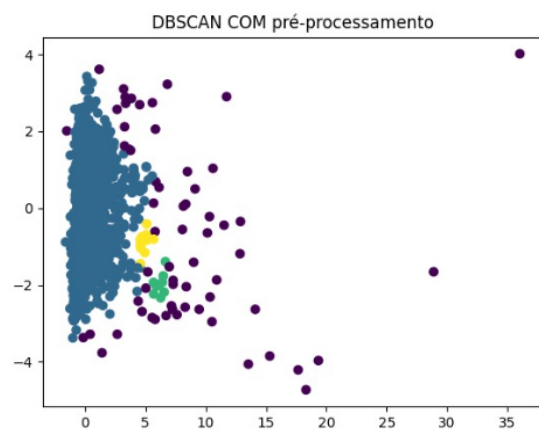


SOM: nenhuma estrutura clara.

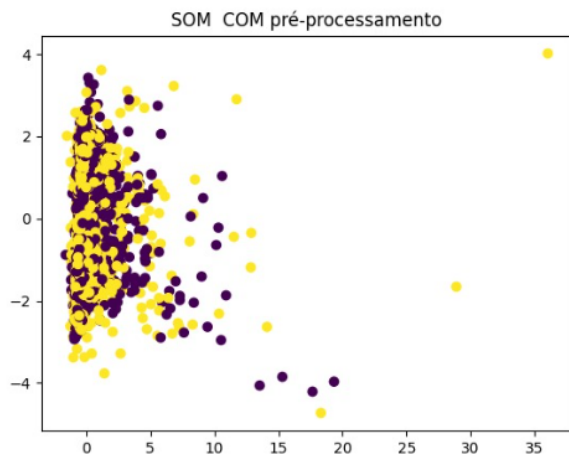
Depois do pré-processamento



KMeans: melhor separação (clusters mais definidos).



DBSCAN: conseguiu formar pequenos agrupamentos reais.



SOM: “áreas suspeitas” surgem, base pode ter padrões não lineares

Conclusão

Clusterização - Resultados

Algoritmo	Resultado	Silhouette Score
K-Means (2 clusters)	Separou levemente fraudes	~0.14 – baixo, mas com melhor separação global
DBSCAN	Detectou muitos pontos como ruído	Vários -1 – difícil avaliar
SOM	Forma clusters espalhados, precisa interpretar mapa	Captou regiões específicas de fraude

Os algoritmos não conseguiram encontrar claramente dois grupos naturais. A base continua difícil de separar sem a classe real (confirmando que é um problema supervisionado).

Final

O pré-processamento melhorou os agrupamentos, mas apenas o algoritmo K-Means se mostrou suficiente para a separação de DOIS grupos no problema. Ou seja, os métodos de agrupamento não substituem um modelo supervisionado para detecção de fraude, o que confirma a complexidade do problema.