UNIVERSIDADE FEDERAL DO CEARÁ

TECHNICAL REPORT

Aluno: Vítor Barbosa da Silva - 537979

1. Introdução

O dataset contém dados relacionados à transações em cartões de crédito de diferentes usuários em diferentes lugares da Europa. Possui colunas que apresentam anonimamente os dados das transações, como horário, local, etc. A atividade em cima do dataset consiste em analisar os dados e treinar o sistema para classificar se a transação tem caráter fraudulento ou não. Coluna alvo possui o nome “Class”, e possui 2 valores, 0 e 1, para representar fraude ou não.

2. Observações

Devido ao grande volume de dados presentes no *dataset*, foi necessária a aplicação de uma amostragem estratificada dos dados. Sendo utilizado um valor de 30000(trinta mil) elementos aleatórios.

3. Resultados e discussão

3.1. Questão 1

Nessa questão, utilizei o *GridSearchCV* para analisar a melhor parametrização.

- No seguinte trecho de código, carreguei o dataset com a função “load_creditcard()”;
- Fiz a amostragem estratificada, onde serão coletados 30000 elementos de cada classe da coluna “Class”;
- Exclui a coluna “id”, já que os dados não serão necessários;
- Atribui à variável X os demais valores e à Y os valores da coluna Class
- Em seguida inicializei os conjuntos.

```
df = load_creditcard()
dfe = df.groupby('Class', group_keys=False).apply(lambda x: x.sample(n=30000, random_state=42))
df.drop(["id"], axis=1)
X = dfe.drop(["Class"], axis=1)
y = dfe["Class"].values
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=42)
```

- Já nesse trecho, foram passados o parâmetros utilizados no GridSearchCV, sendo o número de vizinhos variando de 2 em 2, do 1 ao 50 e usando com métricas as distâncias Euclidiana, Manhattan e Minkowski.

```
grid_params = {
    'n_neighbors': np.arange(1, 51, 2),
    'metric': ['euclidean', 'manhattan', 'minkowski']
}
```

- Nesse trecho inicializei o KNN e em seguida o GridSearchCV;
- Armazenei os resultados da melhor configuração encontrada;
- Em seguida, apliquei no conjunto de teste.

```
knn = KNeighborsClassifier()

# Executa o GridSearchCV
grid_search = GridSearchCV(knn, grid_params, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Melhor configuração encontrada
best_params = grid_search.best_params_
best_knn = grid_search.best_estimator_

# Avaliação no conjunto de teste
y_pred = best_knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print("Melhores parâmetros:", best_params)
print("Acurácia no conjunto de teste:", accuracy)
```

As tabelas a seguir, mostram os resultados obtidos neste código e no código da AV1, respectivamente, assim como foi solicitado na questão:

Resultados AV2	
Melhor Distância	Manhattan
Melhor K	39
Acurácia	99,92%

Resultados AV1	
Melhor Distância	Euclidiana
Melhor K	2
Acurácia	99,35%

3.2. Questão 2

Nessa questão, foi analisado o melhor número de *clusters* para os dados dos demais conjuntos do *dataset*, sem utilizar a coluna alvo.

- Nesse trecho, carreguei o *dataset* e exclui a coluna alvo “Class”;
- Em seguida fiz uma amostragem estratificada dos demais dados com um tamanho de 30000 elementos e também a normalização dos dados.

```python
df = load_creditcard()
df = df.drop(columns=['Class'])
X = df.sample(n=30000, random_state=42)

# Normalização dos dados
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

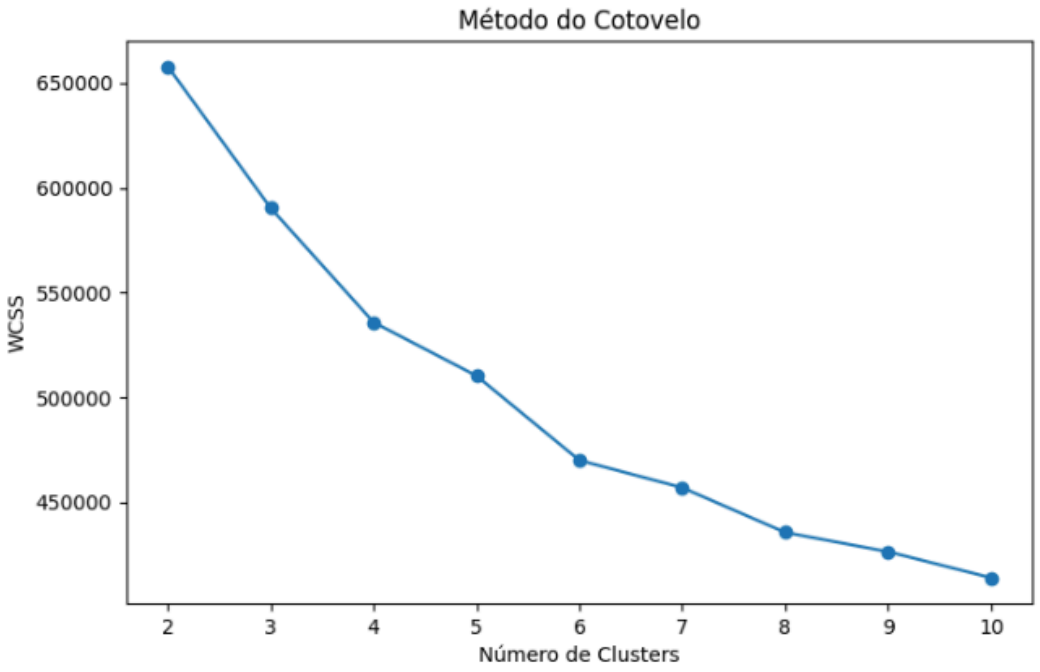
- Implementação do método cotovelo e seu respectivo gráfico:

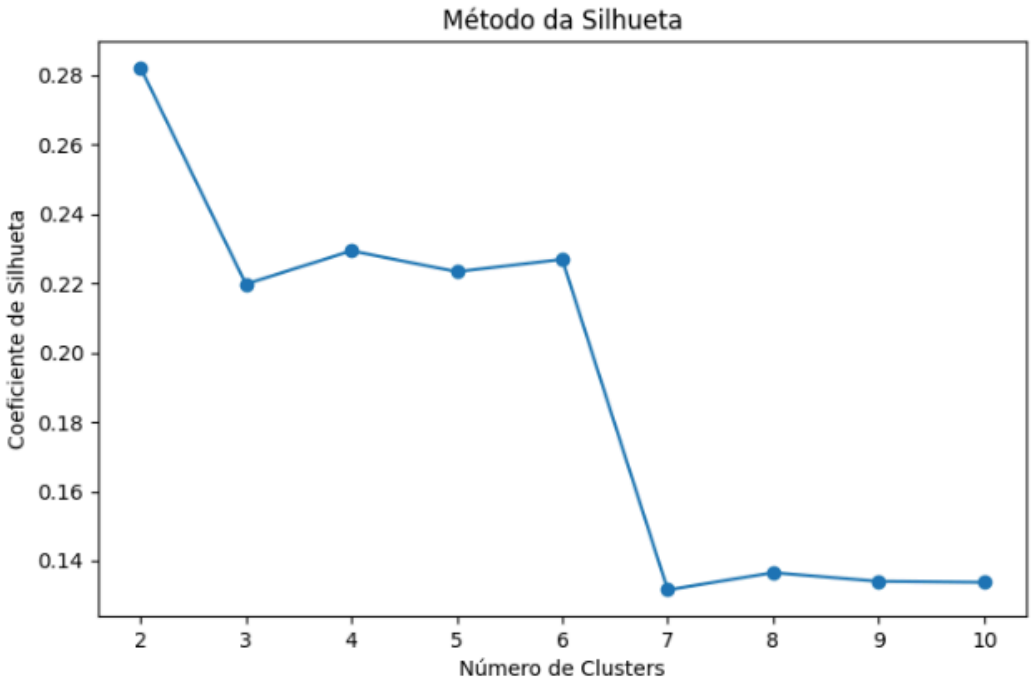
```python
# Metodo Cotovelo
wcss = []
k_values = range(2, 11)
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
plt.plot( *args: k_values, wcss, marker='o')
plt.xlabel('Número de Clusters')
plt.ylabel('WCSS')
plt.title('Método do Cotovelo')
plt.show()
```

- Implementação do método silhueta e seu respectivo gráfico:

```python
# Metodo Silhueta
silhouette_scores = []
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = kmeans.fit_predict(X_scaled)
    silhouette_scores.append(silhouette_score(X_scaled, labels))

plt.figure(figsize=(8, 5))
plt.plot( *args: k_values, silhouette_scores, marker='o')
plt.xlabel('Número de Clusters')
plt.ylabel('Coeficiente de Silhueta')
plt.title('Método da Silhueta')
plt.show()
```

Resultados obtidos:





3.3. Questão 3

Nessa questão foram escolhidos os dois atributos mais relevantes utilizando lasso, e em seguida plotados os gráficos:

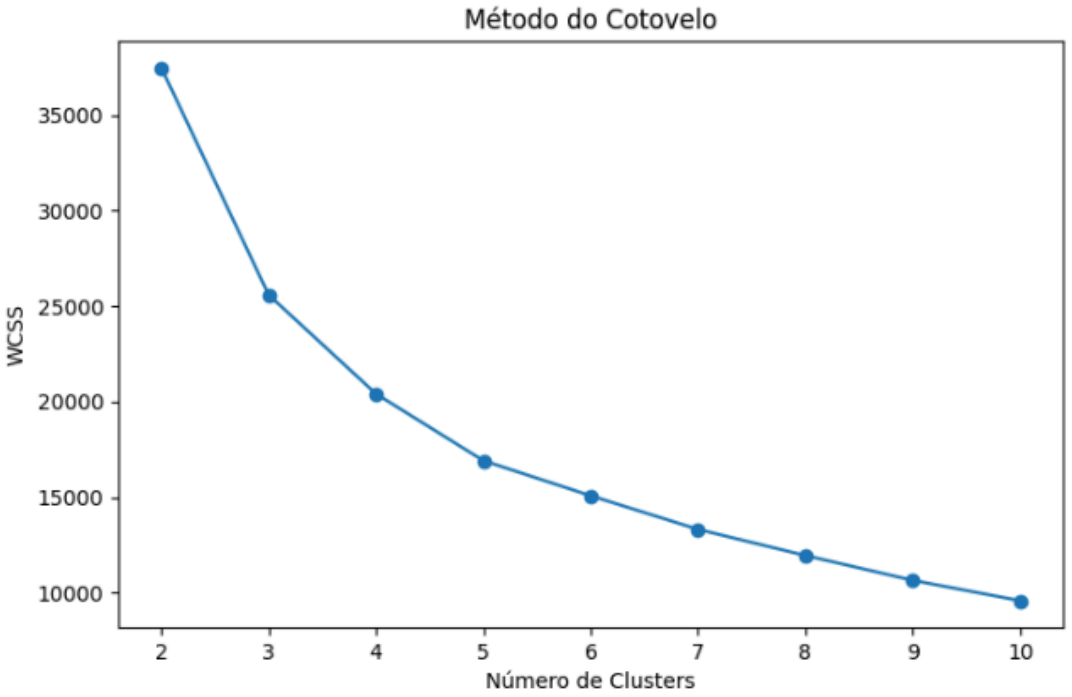
- Primeiramente foi carregado o *dataset,* em seguida dropada a coluna Class;
- Foi feita também a normalização dos dados.

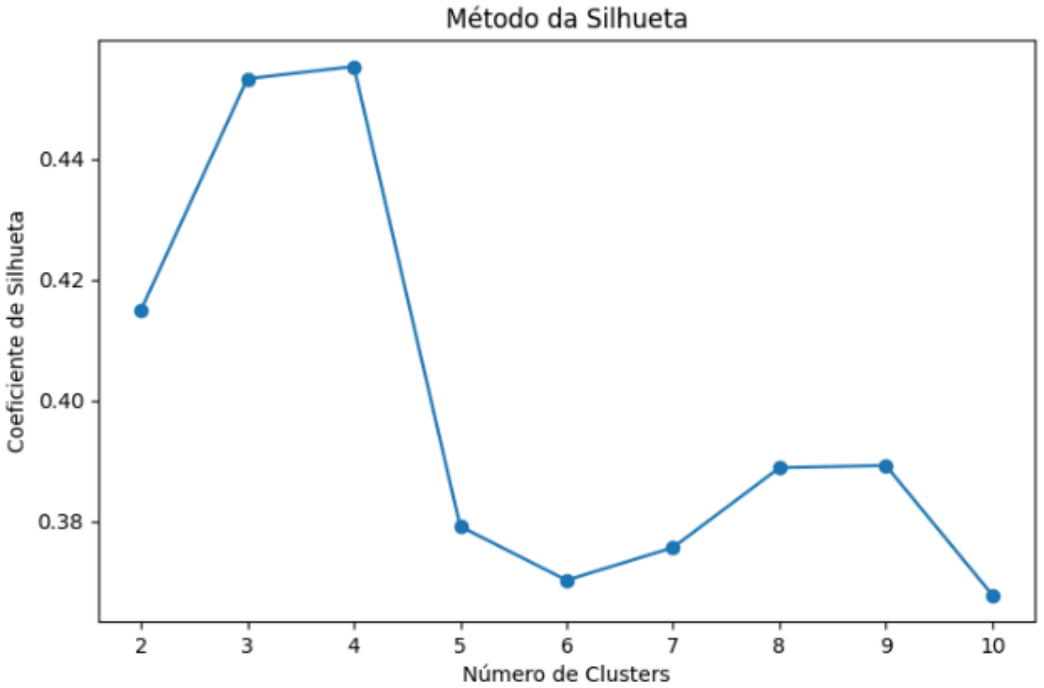
```python
df = load_creditcard()
df = df.drop(columns=['Class'])
X = df.sample(n=30000, random_state=42)

# Normalização dos dados
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

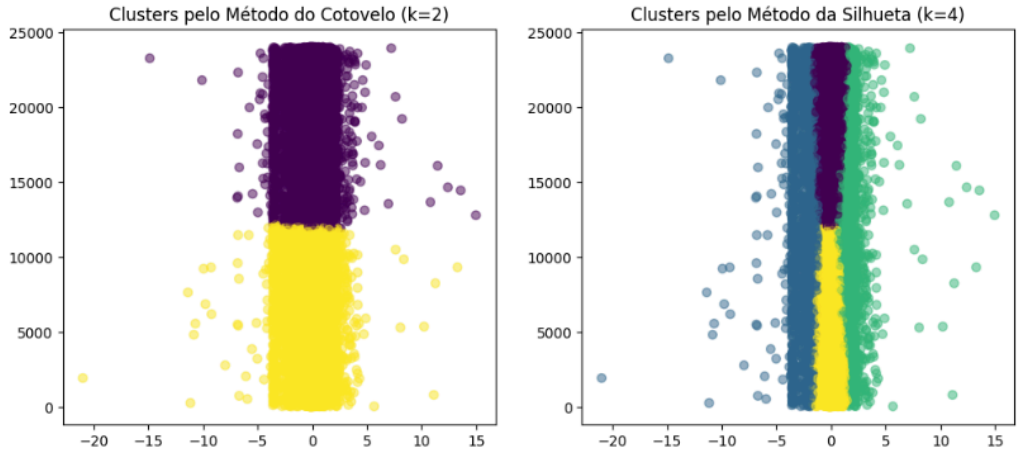
- Em seguida, é utilizado o lasso para definir os atributos e feita uma renormalização dos dados.
- OBS.: Melhores atributos foram as colunas “V28” e “Mount”

```python
lasso = LassoCV(cv=5, random_state=42).fit(X_scaled, np.random.rand(X_scaled.shape[0]))
important_features = np.argsort(np.abs(lasso.coef_))[-2:]
X_selected = X.iloc[:, important_features]

# Re-normalização dos atributos selecionados
X_selected_scaled = scaler.fit_transform(X_selected)
```

Foram implementados novamente os gráficos dos métodos Cotovelo e Silhueta, porém o melhor K para o método Silhueta mudou. na questão anterior para ambos os métodos o melhor k foi 2. já nessa questão o melhor k foi 2 para Cotovelo e 4 para Silhueta. Os resultados serão mostrados a seguir:


Método do Cotovelo
WCSS
35000
30000
25000
20000
15000
10000
2
3
4
5
6
7
8
9
10
Número de Clusters


Método da Silhueta
Coeficiente de Silhueta
0.44
0.42
0.40
0.38
2
3
4
5
6
7
8
9
10
Número de Clusters



3.4. Questão 4

Nessa questão foi feito um *crosstab* para analisar a distribuição por clusters. foi também utilizado k=4, referente ao índice de Silhueta.

Os resultados obtidos estão expressos na seguinte tabela:

Cluster	Classe 0	Classe 1
0	7211	5503
1	403	1645
2	219	2412
3	7169	5438

4. Conclusões

No meu ponto de vista os resultados foram satisfatórios, o desenrolar das questões aconteceu de forma mais fluída e positiva em relação à avaliação anterior. Não tive problemas para carregar os gráficos ou implementar algum código, isso ajudou a

proporcionar uma satisfação pessoal também, já que não houve nenhum problema para solucionar as questões. O conteúdo é um pouco complexo, mas ainda achei mais interessante que o da primeira avaliação. Acredito que por ter entendido melhor e por já ter pegado um pouco de prática em relação à mesma. Ainda é necessário dar uma lapidada nos conhecimentos e códigos, mas no geral acredito que os resultados e o aprendizado foram satisfatórios.

5. Próximos passos

Sem sugestões para a evolução do projeto!