Lista 8

22 DE JUNHO DE 2024 / INTELIGÊNCIA ARTIFICIAL

Aluno: Júlia Nathália Sebastião Pinto - 784397

Questão 1

1. Pré-processamento

Para o pré-processamento, o código inclui a normalização dos dados usando 'StandardScaler'.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, calinski_harabasz_score
import seaborn as sns

# Carregar os dados
iris_data = pd.read_csv('Iris.csv')

# Identificação de outliers
plt.figure(figsize=(12, 8))
sns.boxplot(data=iris_data.drop('class', axis=1))
plt.title('Boxplots of Iris Dataset Features')
plt.show()

# Normalização dos dados
scaler = StandardScaler()
iris_scaled = scaler.fit_transform(iris_data.drop('class', axis=1))
```

2. Encontrar agrupamentos e discutir a qualidade

Para encontrar os agrupamentos e discutir a qualidade, foram usados os métodos Elbow e Silhouette. A seguir, apresento o código com comentários adicionais:

```
# Elbow Method
sse = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(iris_scaled)
    sse.append(kmeans.inertia_)
```

```
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), sse, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('SSE')
plt.show()

# KMeans com 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(iris_scaled)

# Avaliação com Silhouette Score e Calinski-Harabasz Score
silhouette_avg = silhouette_score(iris_scaled, clusters)
calinski_harabasz = calinski_harabasz_score(iris_scaled, clusters)
print(f'Silhouette Score: {silhouette_avg:.4f}')
print(f'Calinski-Harabasz Score: {calinski_harabasz:.4f}')

# Visualização dos clusters
plt.figure(figsize=(10, 6))
plt.scatter(iris_scaled[:, 0], iris_scaled[:, 1], c=clusters,
cmap='viridis', edgecolor='k', s=50)
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75,
marker='x')
plt.title('Visualization of Clustered Data')
plt.xlabel('Normalized Sepal Length')
plt.ylabel('Normalized Sepal Width')
plt.show()
```

3. Explicação das métricas

- Silhouette Score: A métrica é dada por:

  $s(i) = \frac{b(i)-a(i)}{max(a(i),b(i)}$

  onde $a(i)$ é a distância média entre o ponto i e todos os outros pontos no mesmo cluster, e $b(i)$ é a menor distância média entre o ponto i e todos os pontos em qualquer outro cluster.

- Calinski-Harabasz Score: A métrica é dada por:

  $s(i) = \frac{tr(B_k)}{tr(W_k)} \cdot \frac{N-k}{k-1}$

  onde $tr(B_k)$ é a soma das distâncias ao quandrado entre os centros dos clusters e o centroide global, $tr(W_k)$ é a soma das distâncias ao quadrado dentro dos clusters, N é o número total de pontos e k é o número de clusters.

4. Outra métrica de avaliação

A métrica que optei por colocar foi Davies-Bouldin:

```
from sklearn.metrics import davies_bouldin_score

davies_bouldin = davies_bouldin_score(iris_scaled, clusters)
print(f'Davies-Bouldin Score: {davies_bouldin:.4f}')
```

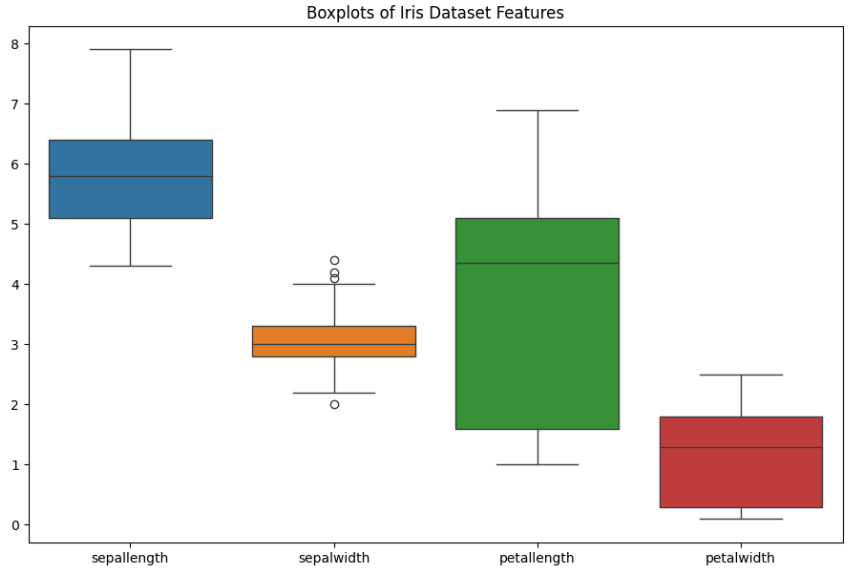
5. Visualização de instâncias incorretamente agrupadas

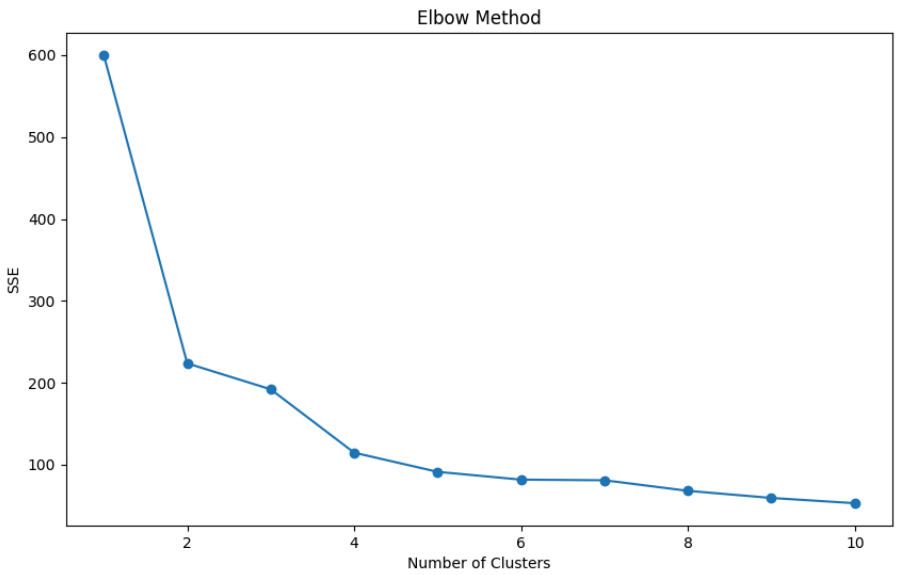
```
import seaborn as sns

# Adicionando os clusters ao dataframe original
iris_data['cluster'] = clusters

# Mapeando as classes para números
class_mapping = {'setosa': 0, 'versicolor': 1, 'virginica': 2}
iris_data['class_num'] = iris_data['class'].map(class_mapping)

# Visualizando os agrupamentos incorretos
incorrect = iris_data[iris_data['cluster'] != iris_data['class_num']]

plt.figure(figsize=(10, 6))
sns.scatterplot(x=iris_data['sepallength'], y=iris_data['sepalwidth'],
hue=iris_data['class'], style=iris_data['cluster'], palette='viridis')
plt.title('Incorrectly Clustered Instances')
plt.show()
```

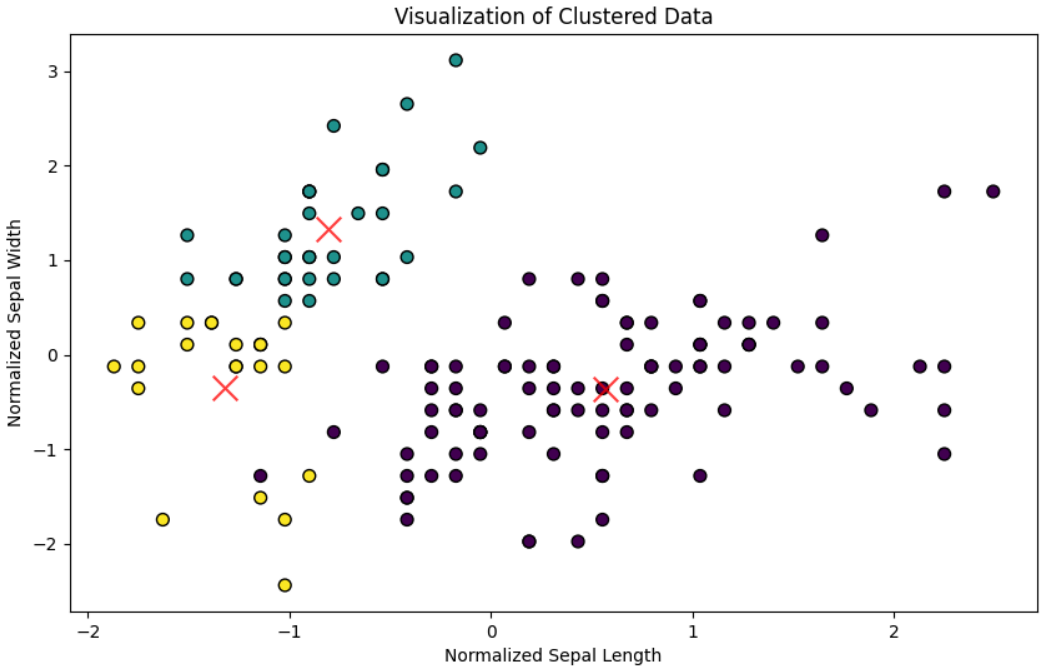
6. Resultados

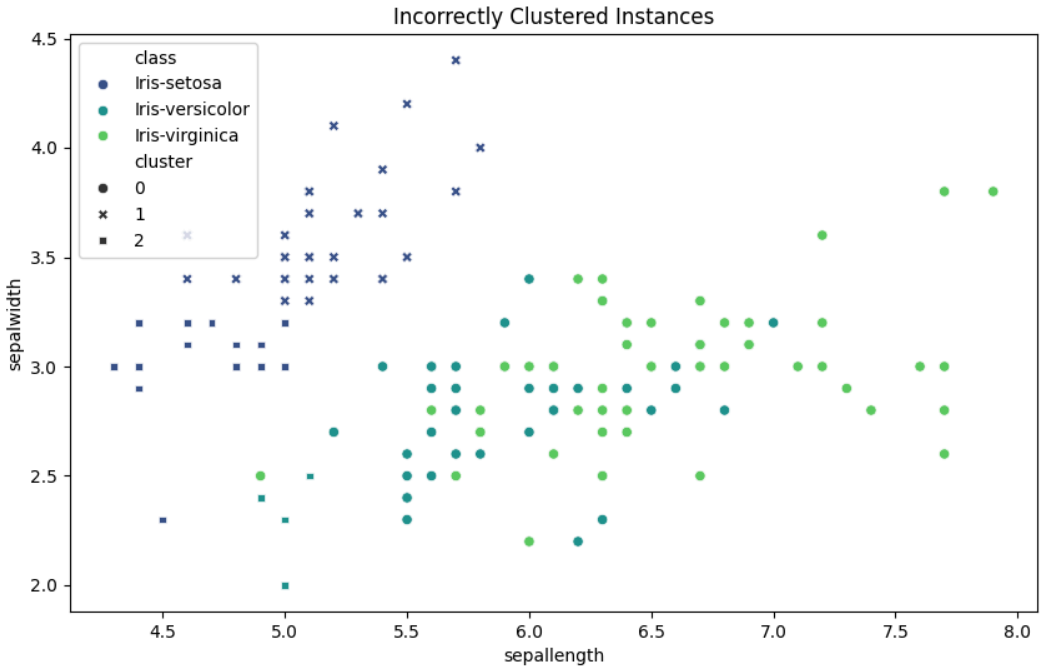
```
PS C:\Users\julia\Documents\progams\lista8> python question1.py
   sepallength  sepalwidth  petallength  petalwidth        class
0          5.1         3.5          1.4         0.2  Iris-setosa
1          4.9         3.0          1.4         0.2  Iris-setosa
2          4.7         3.2          1.3         0.2  Iris-setosa
Returning the number of logical cores instead. You can silence this warning
by setting LOKY_MAX_CPU_COUNT to the number of cores you want to use.
Returning the number of logical cores instead. You can silence this warning
by setting LOKY_MAX_CPU_COUNT to the number of cores you want to use.
  warnings.warn(
  File
"C:\Users\julia\AppData\Local\Programs\Python\Python312\Lib\site-packages\j
oblib\externals\loky\backend\context.py", line 282, in
_count_physical_cores
```

```
    raise ValueError(f"found {cpu_count_physical} physical cores < 
1")Silhouette Score: 0.4787
Calinski-Harabasz Score: 156.1430
Davies-Bouldin Score: 0.7868
PS C:\Users\julia\Documents\progams\lista8>
```






Visualization of Clustered Data
Normalized Sepal Width
Normalized Sepal Length
3
2
1
0
−1
−2
−2
−1
0
1
2


Incorrectly Clustered Instances
class
Iris-setosa
Iris-versicolor
Iris-virginica
cluster
0
1
2
sepalwidth
sepallength
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

Questão 2

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

nltk.download('punkt')
nltk.download('stopwords')

def preprocess(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    tokens = word_tokenize(text)
    stop_words = set(stopwords.words('english'))
    tokens = [token for token in tokens if token not in stop_words]
    stemmer = PorterStemmer()
    tokens = [stemmer.stem(token) for token in tokens]
    processed_text = ' '.join(tokens)
    return processed_text

def load_data(filepath):
    try:
        data = pd.read_csv(filepath, delimiter=';', on_bad_lines='skip',
engine='python')
    except pd.errors.ParserError:
        data = pd.read_csv(filepath, delimiter=';', on_bad_lines='skip',
engine='python')
    return data

train_data = load_data('ReutersGrain-train.csv')
test_data = load_data('ReutersGrain-test.csv')

# Verificando as colunas e renomeando se necessário
print("Train Data Columns:", train_data.columns)
```

```
print("Test Data Columns:", test_data.columns)

if 'Text' in train_data.columns and 'class-att' in train_data.columns:
    train_data = train_data.rename(columns={'Text': 'text', 'class-att':
'label'})
if 'Text' in test_data.columns and 'class-att' in test_data.columns:
    test_data = test_data.rename(columns={'Text': 'text', 'class-att':
'label'})

print(train_data.head())
print(test_data.head())

# Aplicando pré-processamento
if 'text' in train_data.columns and 'text' in test_data.columns:
    train_data['processed_text'] = train_data['text'].apply(preprocess)
    test_data['processed_text'] = test_data['text'].apply(preprocess)
else:
    raise KeyError("A coluna 'text' não está presente nos dados de
treinamento ou teste.")

# Vetorização
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(train_data['processed_text'])
X_test = vectorizer.transform(test_data['processed_text'])

y_train = train_data['label']
y_test = test_data['label']

# Treinando modelos
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
nb_predictions = nb_model.predict(X_test)

svm_model = SVC()
svm_model.fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)

# Avaliando modelos
nb_accuracy = accuracy_score(y_test, nb_predictions)
svm_accuracy = accuracy_score(y_test, svm_predictions)

print(f'Accuracy of Naive Bayes: {nb_accuracy}')
print(f'Accuracy of SVM: {svm_accuracy}')
```

Resultados encontrados:

```
PS C:\Users\julia\Documents\progams\lista8> python question2.py
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\julia\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\julia\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
Train Data Columns: Index(['Text', 'class-att'], dtype='object')
Test Data Columns: Index(['Text', 'class-att'], dtype='object')
                                                text  label
0  'BAHIA COCOA REVIEW Showers continued througho...      0
1  'NATIONAL AVERAGE PRICES FOR FARMER-OWNED RESE...      1
2  'ARGENTINE 1986/87 GRAIN/OILSEED REGISTRATIONS...      1
3  'CHAMPION PRODUCTS &lt.CH> APPROVES STOCK SPLI...      0
4  'COMPUTER TERMINAL SYSTEMS &lt.CPML> COMPLETES...      0
                                                text  label
0  'ASIAN EXPORTERS FEAR DAMAGE FROM U.S.-JAPAN R...      0
1  'CHINA DAILY SAYS VERMIN EAT 7-12 PCT GRAIN ST...      1
2  'JAPAN TO REVISE LONG-TERM ENERGY DEMAND DOWNW...      0
3  'THAI TRADE DEFICIT WIDENS IN FIRST QUARTER Th...      1
4  'INDONESIA SEES CPO PRICE RISING SHARPLY Indon...      0
Accuracy of Naive Bayes: 0.9519867549668874
Accuracy of SVM: 0.9370860927152318
PS C:\Users\julia\Documents\progams\lista8>
```