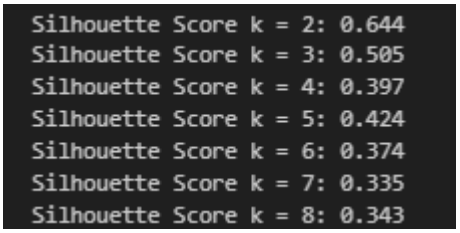
Lista 8

1) Pré-Processamento Realizado:
    - Remoção de outlier utilizando quartis como referência
    - Binarização da classe com OneHotEncoder
    - Normalização com MinMaxScaler
    - Remoção das instâncias mais semelhantes utilizando NearMiss para balancear as classes

Métricas de Avaliação:

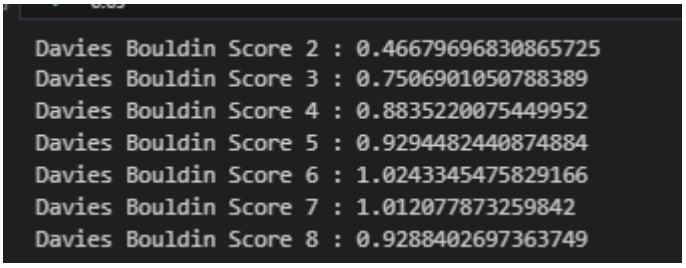
Silhouette Score

A silhueta é calculada utilizando a diferença entre o somatório das distâncias de um registro para outros do seu clusters e o somatório das distâncias dele para os registros de outros grupos e dividido pelo máximo entre esses somatórios



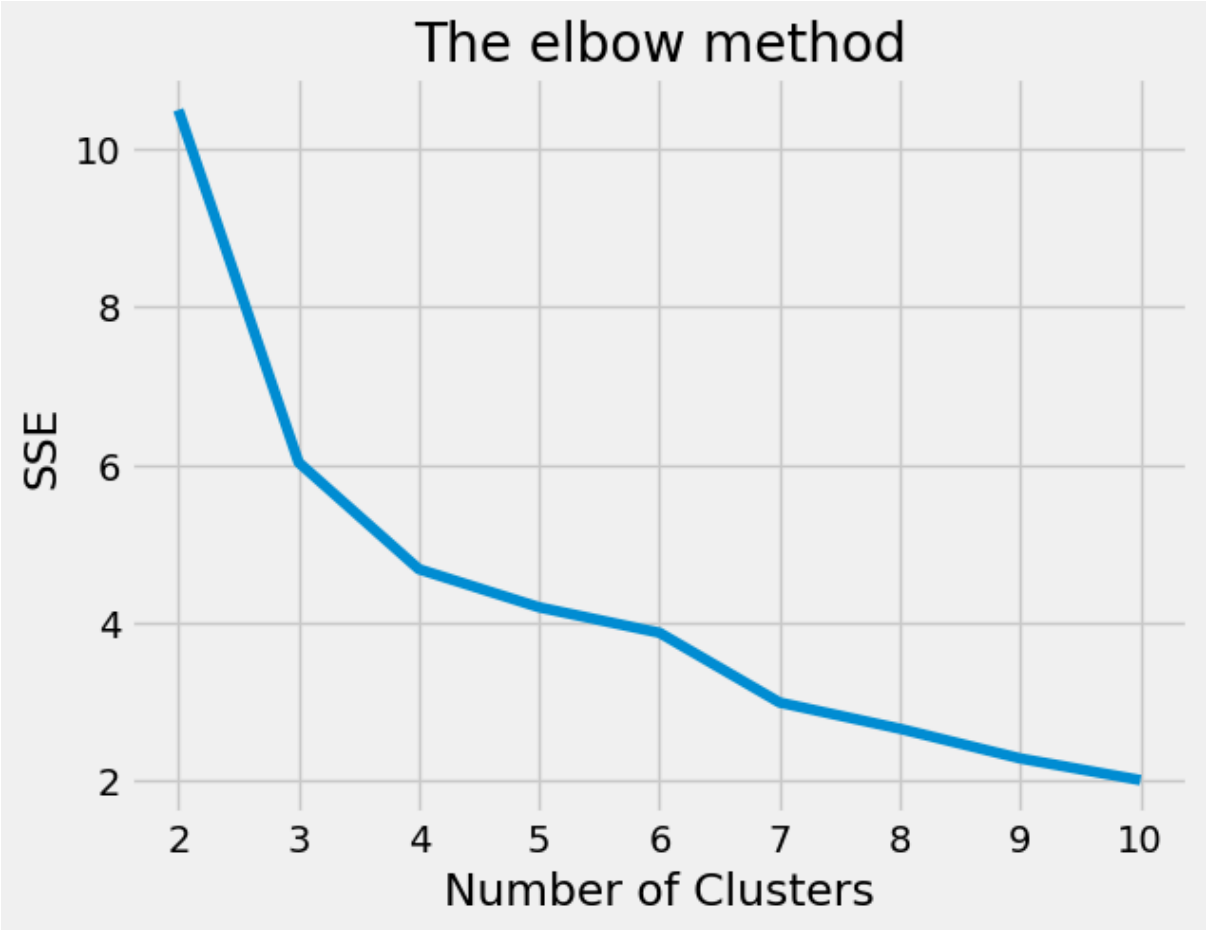
Davies Bouldin Score (DBI)

Esse método avalia os grupos baseado na semelhança média de cada cluster com o cluster mais similar a ele utilizando a distância entre os centroides como referência. Quanto menor o valor (semelhança) melhor o agrupamento

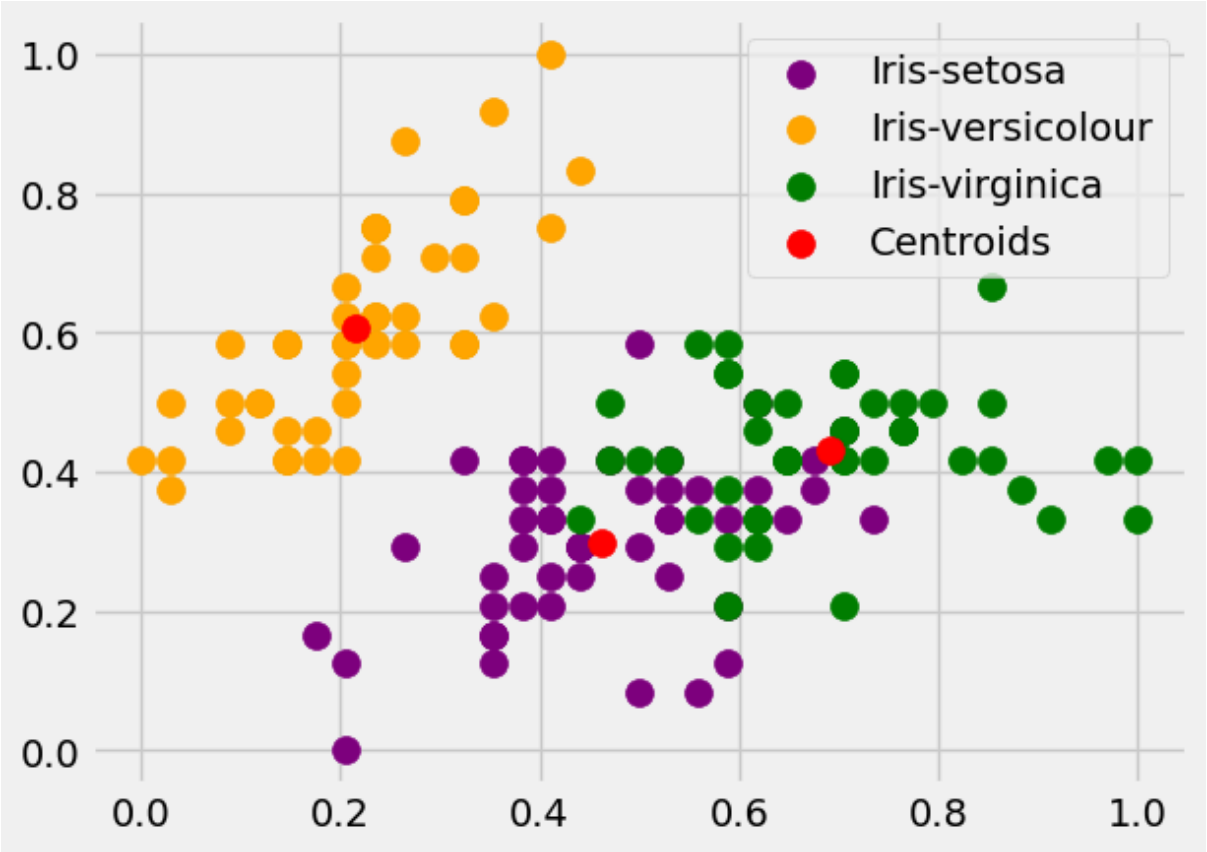


Elbow

O Elbow é um gráfico gerado a partir da soma dos erros quadrados para cada quantidade de grupos, quantos mais grupos menor o erro, pois geram distâncias cada vez menores entre seus membros. O gráfico forma uma espécie de cotovelo, o ponto de inflexão nesse cotovelo costuma ser a quantidade de grupos ideal para o conjunto de dados



Grupos encontrados:
Nota-se que o algoritmo confunde setosas com virgínicas, existem muitas instâncias setosas no grupo das virgínicas e vice-versa mesmo que a métrica elbow diga que a quantidade de grupos ideal seria 3 ou 4. A semelhança pode ser melhor vista ao analisar as métricas de DBI e Silhouette, que retornam suas melhores avaliações quando há apenas 2 grupos. Conclui-se então que as duas classes são semelhantes demais para serem completamente separadas.



2)

Pré-Processamento:

- Remoção de números
- Remoção de marcadores
- Remoção de stopwords
- Stemização

Algoritmos Utilizados:

- KerasNLP

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(base['Text'])
sequences = tokenizer.texts_to_sequences(base['Text'])
data = pad_sequences(sequences,maxlen=100)
model = Sequential()
model.add(Embedding(100000,8))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))  # Sigmoid activation for binary classification


model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(data, base['class-att'], epochs=10, batch_size=2)
```

```
test_seq = tokenizer.texts_to_sequences(test['Text'])
test_data = pad_sequences(test_seq, maxlen=100)
prev = model.predict(test_data)
loss, accuracy = model.evaluate(test_data, test['class-att'])
accuracy
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 611us/step - accuracy: 0.9627 - loss: 0.1639

0.9552980065345764
```

- MultinomialNB

```
vectorizer = CountVectorizer(analyzer = "word")
textos = vectorizer.fit_transform(base['Text'])
modelo = MultinomialNB()
modelo.fit(textos, base['class-att'])
```

```
freq_testes = vectorizer.transform(test['Text'])
resultados = modelo.predict(freq_testes)
```

```
metrics.accuracy_score(test['class-att'], resultados)
```

```
0.9486754966887417
```

Link para os códigos: https://github.com/Tadleao/IA-Lista8