Aprendizagem de Máquina:
Atividade 09 – Árvores de Decisão

Carlos Emmanuel Pereira Alves
Curso de Bacharelado em Ciência da Computação
Universidade Federal do Agreste de Pernambuco (UFAPE)
Garanhuns, Brasil
carlos.emmanuel.236@gmail.com

1) Utilizando a Balance Scale:
   a) Avalie a taxa de erro para uma árvore na qual a raíz é uma folha (todos os exemplos do conjunto de treino estão nesta folha).

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

cols = ['class', 'left_weight', 'left_distance', 'right_weight', 'right_distance']

data = pd.read_csv('balance-scale.data', header=None, names=cols)

X_train, X_test, y_train, y_test = train_test_split(data[cols[1:]], data['class'], test_size=0.3, random_state=42)

tree = DecisionTreeClassifier(max_depth=1)

tree.fit(X_train, y_train)

predictions = tree.predict(X_test)

error_rate = 1 - accuracy_score(y_test, predictions)

print("Taxa de Erro:", error_rate)
```

```
Taxa de Erro: 0.4042553191489362
```

   b) Calcule qual atributo é o mais adequado para ser a raiz da árvore. Mostre como você calculou estes dados.

   c) Calcule os atributos mais adequados para o próximo nível da árvore. Mostre como você calculou estes dados.

2) Utilizando a base Car Evalution e um Holdout 50/50. Teste pelo menos três valores distintos de número mínimo de exemplos por folha e realize os experimentos PAREADOS. Avalie os efeitos de limitar o número mínimo de exemplos por folha:

a) Na taxa de acerto para o conjunto de treino.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OrdinalEncoder

cols = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']

buying = ['low', 'med', 'high', 'vhigh']
maint = ['low', 'med', 'high', 'vhigh']
doors = ['2', '3', '4', '5more']
persons = ['2', '4', 'more']
lug_boot = ['small', 'med', 'big']
safety = ['low', 'med', 'high']

data = pd.read_csv('car.data', header=None, names=cols, sep=',')

enc = OrdinalEncoder(categories=[buying, maint, doors, persons, lug_boot, safety])
data_transformed = pd.DataFrame(enc.fit_transform(data[cols[:-1]]), columns=cols[:-1])
data_transformed['class'] = data['class']

X_train, X_test, y_train, y_test = train_test_split(data_transformed[cols[:-1]], data_transformed['class'], test_size=0.5, random_state=42)

scores = []
results = {}
values = [1, 6, 12]

for value in values:
  tree = DecisionTreeClassifier(min_samples_leaf=value)
  tree.fit(X_train, y_train)
  predictions = tree.predict(X_train)

  accuracy = accuracy_score(y_train, predictions)
  results[value] = accuracy

for min_samples, accuracy in results.items():
    print(f"Número mínimo de exemplos por folha: {min_samples}\nTaxa de acerto: {accuracy:.4f}\n")
```

```
Número mínimo de exemplos por folha: 1
Taxa de acerto: 1.0000

Número mínimo de exemplos por folha: 6
Taxa de acerto: 0.9699

Número mínimo de exemplos por folha: 12
Taxa de acerto: 0.9178
```

b) Na taxa de acerto e para o conjunto de teste.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OrdinalEncoder

cols = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']

buying = ['low', 'med', 'high', 'vhigh']
maint = ['low', 'med', 'high', 'vhigh']
doors = ['2', '3', '4', '5more']
persons = ['2', '4', 'more']
lug_boot = ['small', 'med', 'big']
safety = ['low', 'med', 'high']

data = pd.read_csv('car.data', header=None, names=cols, sep=',')

enc = OrdinalEncoder(categories=[buying, maint, doors, persons, lug_boot, safety])
data_transformed = pd.DataFrame(enc.fit_transform(data[cols[:-1]]), columns=cols[:-1])
data_transformed['class'] = data['class']

X_train, X_test, y_train, y_test = train_test_split(data_transformed[cols[:-1]], data_transformed['class'], test_size=0.5, random_state=42)

scores = []
results = {}
values = [1, 6, 12]

for value in values:
  tree = DecisionTreeClassifier(min_samples_leaf=value)
  tree.fit(X_train, y_train)
  predictions = tree.predict(X_test)

  accuracy = accuracy_score(y_test, predictions)
  results[value] = accuracy

for min_samples, accuracy in results.items():
    print(f"Número mínimo de exemplos por folha: {min_samples}\nTaxa de acerto: {accuracy:.4f}\n")
```

```
Número mínimo de exemplos por folha: 1
Taxa de acerto: 0.9734

Número mínimo de exemplos por folha: 6
Taxa de acerto: 0.9491

Número mínimo de exemplos por folha: 12
Taxa de acerto: 0.9132
```

c) Matriz de confusão para o conjunto de teste.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import OrdinalEncoder

cols = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']

buying = ['low', 'med', 'high', 'vhigh']
maint = ['low', 'med', 'high', 'vhigh']
doors = ['2', '3', '4', '5more']
persons = ['2', '4', 'more']
lug_boot = ['small', 'med', 'big']
safety = ['low', 'med', 'high']

data = pd.read_csv('car.data', header=None, names=cols, sep=',')

enc = OrdinalEncoder(categories=[buying, maint, doors, persons, lug_boot, safety])
data_transformed = pd.DataFrame(enc.fit_transform(data[cols[:-1]]), columns=cols[:-1])
data_transformed['class'] = data['class']

X_train, X_test, y_train, y_test = train_test_split(data_transformed[cols[:-1]], data_transformed['class'], test_size=0.5, random_state=42)

scores = []
results = {}
values = [1, 6, 12]

for value in values:
  tree = DecisionTreeClassifier(min_samples_leaf=value)
  tree.fit(X_train, y_train)
  predict = tree.predict(X_test)

  results[value] = confusion_matrix(y_test, predict)

for min_samples, confusion_matrix in results.items():
    print(f"Matriz de confusão do número mínimo de exemplos por folha: {min_samples}\n {confusion_matrix}\n")
```

```
Matriz de confusão do número mínimo de exemplos por folha: 1
 [[175   7   2   2]
 [  1  27   0   5]
 [  4   0 612   0]
 [  0   2   0  27]]

Matriz de confusão do número mínimo de exemplos por folha: 6
 [[162  11   9   4]
 [  0  25   1   7]
 [ 11   1 604   0]
 [  0   0   0  29]]

Matriz de confusão do número mínimo de exemplos por folha: 12
 [[168  13   4   1]
 [  8  10   1  14]
 [ 25   1 589   1]
 [  7   0   0  22]]
```

d) Indique se ocorreu overfitting ou underfitting baseado nos resultados obtidos.

Quando o número mínimo de exemplos por folha é igual a 1, o conjunto fica mais propenso a um overfitting. Quando aumentamos esse valor, para 6 ou 12, a tendência é de uma melhor generalização pela árvore.

3) Faça o mesmo da questão anterior para a base Breast Cancer. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer

Primeiro fiz o tratamento da tabela:

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder, OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.impute import SimpleImputer

cols = ['class', 'age', 'menopause', 'tumor-size', 'inv-nodes', 'node-caps', 'deg-malig', 'breast', 'breast-quad', 'irradiat']

age = ['10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80-89', '90-99']
tumor_size = ['0-4', '5-9', '10-14', '15-19', '20-24', '25-29', '30-34', '35-39', '40-44', '45-49', '50-54', '55-59']
inv_nodes = ['0-2', '3-5', '6-8', '9-11', '12-14', '15-17', '18-20', '21-23', '24-26', '27-29', '30-32', '33-35', '36-39']

binary = ['node-caps', 'breast', 'irradiat']

data = pd.read_csv('breast-cancer.data', header=None, names=cols)
data = data.replace('?', np.nan)

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
data_imputer = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)
data = data_imputer

datat = data[['age', 'tumor-size', 'inv-nodes']].copy()
data = data.drop(columns=['age', 'tumor-size', 'inv-nodes'])

for i in range(len(binary)):
  le = LabelEncoder()
  data[binary[i]] = le.fit_transform(data[binary[i]])

enc = OrdinalEncoder(categories=[age, tumor_size, inv_nodes])
data_transformed = pd.DataFrame(enc.fit_transform(datat), columns=['age', 'tumor-size', 'inv-nodes'])
data['age'] = data_transformed['age']
data['tumor-size'] = data_transformed['tumor-size']
data['inv-nodes'] = data_transformed['inv-nodes']

transformer = make_column_transformer((OneHotEncoder(), ['menopause', 'breast-quad']), remainder='passthrough', verbose_feature_names_out=False)
transformed = transformer.fit_transform(data)
transformed_data = pd.DataFrame(transformed, columns=transformer.get_feature_names_out())
column_to_move = transformed_data.pop('class')
transformed_data.insert(0, 'class', column_to_move)

cols = transformed_data.columns.tolist()
```

a)

```python
X_train, X_test, y_train, y_test = train_test_split(transformed_data[cols[1:]], transformed_data['class'], test_size=0.5, random_state=42)

scores = []
results = {}
values = [1, 6, 12]

for value in values:
  tree = DecisionTreeClassifier(min_samples_leaf=value)
  tree.fit(X_train, y_train)
  predictions = tree.predict(X_train)

  accuracy = accuracy_score(y_train, predictions)
  results[value] = accuracy

for min_samples, accuracy in results.items():
    print(f"Número mínimo de exemplos por folha: {min_samples}\nTaxa de acerto: {accuracy:.4f}\n")
```

```
Número mínimo de exemplos por folha: 1
Taxa de acerto: 0.9930

Número mínimo de exemplos por folha: 6
Taxa de acerto: 0.7972

Número mínimo de exemplos por folha: 12
Taxa de acerto: 0.7832
```

b)

```
X_train, X_test, y_train, y_test = train_test_split(transformed_data[cols[1:]], transformed_data['class'], test_size=0.5, random_state=42)

scores = []
results = {}
values = [1, 6, 12]

for value in values:
  tree = DecisionTreeClassifier(min_samples_leaf=value)
  tree.fit(X_train, y_train)
  predictions = tree.predict(X_test)

  accuracy = accuracy_score(y_test, predictions)
  results[value] = accuracy

for min_samples, accuracy in results.items():
    print(f"Número mínimo de exemplos por folha: {min_samples}\nTaxa de acerto: {accuracy:.4f}\n")
```

```
Número mínimo de exemplos por folha: 1
Taxa de acerto: 0.6154

Número mínimo de exemplos por folha: 6
Taxa de acerto: 0.6923

Número mínimo de exemplos por folha: 12
Taxa de acerto: 0.7273
```

c)

```
X_train, X_test, y_train, y_test = train_test_split(transformed_data[cols[1:]], transformed_data['class'], test_size=0.5, random_state=42)

scores = []
results = {}
values = [1, 6, 12]

for value in values:
  tree = DecisionTreeClassifier(min_samples_leaf=value)
  tree.fit(X_train, y_train)
  predict = tree.predict(X_test)

  results[value] = confusion_matrix(y_test, predict)

for min_samples, confusion_matrix in results.items():
    print(f"Matriz de confusão do número mínimo de exemplos por folha: {min_samples}\n {confusion_matrix}\n")
```

```
Matriz de confusão do número mínimo de exemplos por folha: 1
 [[68 33]
 [22 20]]

Matriz de confusão do número mínimo de exemplos por folha: 6
 [[82 19]
 [27 15]]

Matriz de confusão do número mínimo de exemplos por folha: 12
 [[93  8]
 [31 11]]
```

d)

Quando o número mínimo de exemplos por folha é igual a 1 no conjunto de treino, o conjunto fica mais propenso a um overfitting, o que não acontece no conjunto de teste. Quando aumentamos esse valor, para 6 ou 12, a tendência é de uma melhor generalização pela árvore nos dois conjuntos.

4) Interprete as árvores de decisão para duas das bases abaixo. Explique de forma que uma pessoa qualquer possa utilizar a árvore para tomar uma decisão. Estude o que significa cada atributo para contextualizar a explicação. Escolha um árvore razoavelmente pequena para cada caso.

Utilize a base de dados inteira como conjunto de treino. Pegue um ou dois exemplos da base de dados e classique eles seguindo o fluxo da árvore construída.

a) Balance Scale
b) Car Evaluation
c) Breast Cancer