Iniciado em	sexta, 20 mar 2020, 21:05
Estado	Finalizada
Concluída em	sexta, 20 mar 2020, 21:08
Tempo empregado	3 minutos 35 segundos

Questão **1**

Completo

Vale 2,00 ponto(s).

Considere a base de dados 'vertebralcolumn-2C'. Calcule a acurácia usando uma árvore de decisão que tem como critério a medida de entropia. Use o código abaixo. Arrendonde o valor para uma casa decimal.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
np.random.seed(42) # define the seed (important to reproduce the results)

data = pd.read_csv('data/vertebralcolumn-2C.csv', header=(0))
data = data.dropna(axis='rows') #remove NaN
data = data.to_numpy()
nrow,ncol = data.shape
y = data[:,-1]
X = data[:,0:ncol-1]

scaler = StandardScaler().fit(X)
X = scaler.transform(X)

p = 0.2 # fraction of elements in the test set
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = p, random_state = 42)
```

Escolha uma:

- a. 0.8
- b. 0.1
- c. 0.5
- d. 1.0
- e. 0.2

Questão **2**

Completo

Vale 2,00 ponto(s).

Considere a base de dados 'vertebralcolumn-2C'. Calcule a acurácia na classificação usando o método bagging com 100 estimadores. Considere o código abaixo. Arrendonde o valor para uma casa decimal.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
np.random.seed(42) # define the seed (important to reproduce the results)

data = pd.read_csv('data/vertebralcolumn-2C.csv', header=(0))
data = data.dropna(axis='rows') #remove NaN

data = data.to_numpy()
nrow,ncol = data.shape
y = data[:,-1]
X = data[:,0:ncol-1]

scaler = StandardScaler().fit(X)
X = scaler.transform(X)

p = 0.2 # fraction of elements in the test set
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = p, random_state = 42)
```

Escolha uma:

- a. 0.1
- b. 0.2
- c. 0.5
- d. 0.8
- e. 1.0

Questão **3**

Completo

Vale 2,00 ponto(s).

Considere a base de dados 'vertebralcolumn-3C'. Calcule a acurácia na classificação usando o classificar random forest considerando 100 árvores. Considere o código abaixo. Arrendonde o valor para uma casa decimal.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

np.random.seed(42) # define the seed (important to reproduce the results)

data = pd.read_csv('data/vertebralcolumn-3C.csv', header=(0))
data = data.dropna(axis='rows') #remove NaN

data = data.to_numpy()
nrow,ncol = data.shape
y = data[:,-1]
X = data[:,0:ncol-1]

scaler = StandardScaler().fit(X)
X = scaler.transform(X)

p = 0.2 # fraction of elements in the test set
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = p, random_state = 42)
```

Escolha uma:

- a. 1.0
- b. 0.2
- c. 0.1
- d. 0.8
- e. 0.5

Questão **4**

Completo

Vale 2,00 ponto(s).

Considere a base winequality-red. Realizando a classificação usando o método hold-out, selecionando 20% dos dados no conjunto de teste, qual classificador oferece o melhor resultado?

Escolha uma:

- a. floresta aleatória com n=100 árvores
- b. bagging com n=10 estimadores
- c. Todos oferecem a mesma acurácia (considere duas casas decimais)
- d. árvore de decisão usando o critério entropia
- e. Adaboosting com n=10 estimadores

Questão **5**

Completo

Vale 2,00 ponto(s).

Para os dados gerados com o código abaixo, qual classificar oferece a melhor acurácia? Considere apenas uma casa decimal.

```
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import numpy as np
np.random.seed(42) # define the seed (important to reproduce the results)
plt.figure(figsize=(6,4))

n_samples = 1000
data  = datasets.make_moons(n_samples=n_samples, noise=.5, random_state = 42)
X = data[0]
y = data[1]
plt.scatter(X[:,0], X[:,1], c=y, cmap='viridis', s=50, alpha=0.7)
plt.show(True)

scaler = StandardScaler().fit(X)
X = scaler.transform(X)

p = 0.2 # fraction of elements in the test set
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = p, random_state = 42)
```

Escolha uma:

- a. árvore de decisão usando o critério entropia
- b. floresta aleatória com n=100 árvores
- c. Adaboosting com n=10 estimadores
- d. Todos oferecem a mesma acurácia (considere apenas uma casa decimal)
- e. bagging com n=10 estimadores

◄ Exercícios de Fixação - Soluções

Seguir para...