Деревья решений

***Деревья решений*** – алгоритм обучения с учителем, который используется в задачах классификации данных. Алгоритм основан на построении древовидной структуры на этапе обучения и использования этой структуры на этапе классификации.

Для начала используем небольшой датасет, описывающий склонность к диабету в зависимости от возраста и индекса массы тела.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = [[31,45,'yes'],[29,47,'no'],[27,55,'no'],[35,53,'yes'],[28,51,'no'],[40,50,'yes'],[39,41,'no']]

df = pd.DataFrame(data, columns=['Bmi','Age','Diabetes'])

print(df)

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Контент, сгенерированный ИИ, может содержать ошибки.

x=df.iloc[:,:2].to\_numpy()

y=df.iloc[:,2].to\_numpy()

plt.figure(figsize=(7,5))

for i in range(x.shape[0]):

plt.scatter(x[i,0], x[i,1], c=('blue' if y[i]=='no' else 'red'), marker='\*', s=200)

plt.xlabel('Bmi')

plt.ylabel('Age')

plt.show()

Изображение выглядит как снимок экрана, текст, диаграмма, линия

Контент, сгенерированный ИИ, может содержать ошибки.

from sklearn import tree

from sklearn import inspection

model = tree.DecisionTreeClassifier(criterion="entropy").fit(x, y)

tree.plot\_tree(model)

plt.show()

Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

inspection.DecisionBoundaryDisplay.from\_estimator(model, x, response\_method="predict", cmap=plt.cm.RdYlBu)

for i in range(x.shape[0]):

plt.scatter(x[i,0], x[i,1], c=('blue' if y[i]=='no' else 'red'), marker='\*', s=200)

plt.xlabel('Bmi')

plt.ylabel('Age')

plt.show()

Изображение выглядит как снимок экрана, Прямоугольник, диаграмма, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

***Критерий разбиения на поддеревья.***

Индекс Джини:

– множество объектов;

– индекс класса в множестве;

– вероятность, доля объектов -го класса в множестве ;

Энтропия:

Информационный выигрыш:

S – Множество точек данных

H(S) – Энтропия набора данных

А – Атрибут, признак

– Возможные значения A

– Подмножество набора данных, ограниченное значением атрибута

Конкретно, для искомого набора данных

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Контент, сгенерированный ИИ, может содержать ошибки.

получаем:

класс ‘no’ –

класс ‘yes’ –

Разобьём на два подмножества по признаку Bmi<=30.

Разобьём на два подмножества по признаку Age<=50.

- плохое разбиение, так как в подмножестве получаем полностью неопределённое состояние.

Разобьём признаку Age<=45.

Выбираем разбиение , имеющее высокий информационный выигрыш.

Продолжаем аналогичную процедуру разбиения для полученных подмножеств до достижения необходимой точности или глубины дерева. См. результат в приведённом примере программы:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

Алгоритм построения дерева решений:

1. На основе множества создаём узел дерева.
2. Завершаем процесс при достижении заданных глубины дерева или распределения элементов по классам.
3. Разбиваем множество на подмножества по критерию, связанному с атрибутами элементов множества. Выбираем лучшее по информационному выигрышу разбиение.
4. Создаём поддеревья и для каждого подмножества повторяем процесс с 1-го пункта.

Пример на реальных данных.

***Ирисы Фишера:***

from sklearn.datasets import load\_iris

iris = load\_iris()

import matplotlib.pyplot as plt

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.decomposition import PCA

from sklearn import tree

#from sklearn import inspection

from sklearn.inspection import DecisionBoundaryDisplay

from sklearn.tree import DecisionTreeClassifier

X = PCA(n\_components=2).fit\_transform(iris.data)

y = iris.target

#model = DecisionTreeClassifier().fit(X, y)

model = DecisionTreeClassifier(max\_depth=4, min\_samples\_leaf=5).fit(X, y)

plt.figure(figsize=(9,7))

tree.plot\_tree(model)

plt.show()

Изображение выглядит как текст, диаграмма, Шрифт, План

Контент, сгенерированный ИИ, может содержать ошибки.

DecisionBoundaryDisplay.from\_estimator(model, X, response\_method="predict", cmap=plt.cm.RdYlBu)

colors = ['blue', 'green', 'purple', 'brown', 'black']

for i in range(X.shape[0]):

plt.scatter(X[i,0], X[i,1], c=colors[y[i]], marker='o', s=40)

plt.show()

Изображение выглядит как снимок экрана, Красочность, Графика, графический дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Дерево «без ограничений»

model = DecisionTreeClassifier().fit(X, y)

Изображение выглядит как текст, диаграмма, Шрифт, План

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как снимок экрана, Красочность, Графика, графический дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

***Датасет Mushroms***:

# Decision Trees - Mushroom

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

dataset = pd.read\_csv('../mushrooms.csv')

X = dataset.drop(['class'],axis=1)

y = dataset['class']

X = pd.get\_dummies(X)

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

y = encoder.fit\_transform(y)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=1)

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

model = DecisionTreeClassifier(criterion='gini', max\_depth=3, random\_state=0).fit(X\_train, y\_train)

plt.figure(figsize=(12,8))

tree.plot\_tree(model)

plt.show()

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Контент, сгенерированный ИИ, может содержать ошибки.

y\_pred = model.predict(X\_test)

from sklearn.metrics import accuracy\_score

print('Model accuracy score: {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))



Для наглядности результатов используем PCA:

# Decision Trees - Mushroom

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

dataset = pd.read\_csv('../mushrooms.csv')

X = dataset.drop(['class'],axis=1)

y = dataset['class']

X = pd.get\_dummies(X)

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

y = encoder.fit\_transform(y)

# ========== PCA ==================

from sklearn.decomposition import PCA

X = PCA(n\_components=2).fit\_transform(X)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1)

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

from sklearn.inspection import DecisionBoundaryDisplay

model = DecisionTreeClassifier(criterion='gini', max\_depth=3, random\_state=0).fit(X\_train, y\_train)

plt.figure(figsize=(12,8))

tree.plot\_tree(model)

plt.show()

Изображение выглядит как диаграмма, зарисовка, План, Технический чертеж

Контент, сгенерированный ИИ, может содержать ошибки.

y\_pred = model.predict(X\_test)

DecisionBoundaryDisplay.from\_estimator(model, X, response\_method="predict", cmap=plt.cm.RdYlBu)

colors = ['blue', 'green', 'purple', 'brown', 'black']

for i in range(X.shape[0]):

plt.scatter(X[i,0], X[i,1], c=colors[y[i]], marker='o')

plt.show()

Изображение выглядит как снимок экрана, карта, Графика, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

from sklearn.metrics import accuracy\_score

print('Model accuracy score: {0:0.4f}'. format(accuracy\_score(y\_test, y\_pred)))



***Дерево решений и случайный лес.***

Случайный лес – совокупность деревьев решений на случайных выборках из исходного набора данных и атрибутов.

Позволяет избежать преобучения и сильной зависимости от «шума» в данных.