**Chương 1:** **Giới thiệu các bài học sẽ được học.**

**Chương 2:** **Hướng dẫn tải và kiểm tra phiên bản của môi trường để dùng cho Machine Learning trong Python.**

**Chương 3:** **Học các bài học cơ bản của Python và SciPy.**

**Các kiểu dữ liệu cơ bản trong Python :**

String

Number

Boolean

Multiple Assignment

No value (‘None’)

**Vòng lặp :**

Hàm điều kiện if then else

Vòng lặp for ( for \_\_ in range (\_\_) :)

Vòng lặp While ( while \_\_ :)

**Cấu trúc dữ liệu :**

Tuple ( a=(1,2,3) )

List ( a=[1,2,3] )

Dictionary ( a={‘a’:1,’b’:2,’c’:3} )

Function

( def mysum(x,y)

return x+y )

**Thư viện Numpy :**

Định nghĩa lại các dãy giá trị :

ví dụ :

import numpy

list =[1,2,3]

array=numpy.array(list)

print (array.shape)

Output :

(3,)

Chú thích : dãy có 3 cột.

Cách truy cập dữ liệu sau khi chuyển về dạng ở thư viện Numpy :

array[0] Nếu là dãy thì sẽ truy cập vị trí đầu tiên trong dãy, nếu là ma trận thi sẽ truy cập dữ liệu dòng đầu tiên.

Ví dụ :

array =[1,2,3] 🡪output: 1

array =[[1,2,3],[4.5.6]]🡪output: 1 2 3

array[-1] tương tự array[0] nhưng truy cập dữ liệu cuối cùng của dãy hoặc dòng cuối cùng của ma trận.

Ví dụ :

array =[1,2,3] 🡪output: 3

array =[[1,2,3],[4.5.6]]🡪output: 4 5 6

array [:,2] truy cập cột thứ tự 2

ví dụ :

array =[[1,2,3],[4.5.6]]🡪output: 3 6

array[0,2] truy cập cụ thể dữ liệu dòng thứ 0 cột thứ 2.

Ví dụ :

array =[[1,2,3],[4.5.6]]🡪output: 3

**Phép toán :**

Ví dụ :

import numpy

array1=numpy.array([2,2,2])

array2=numpy.array([3,3,3] )

print (array1+array2)

print (array1\*array2)

output :

5 5 5

6 6 6

**Thư viện matplotlib :**

Dùng với thư viện numpy để vẽ biểu đồ.

**Biểu đồ dạng đường thẳng :**

Ví dụ :

import numpy as np

import matplotlib.pyplot as plt

array =np.array([1,2,3])

plt.plot(array)

plt.xlabel(‘trục x’) # hiện dòng chú thích trục hoành

plt.ylabel(‘trục y’) # hiện dòng chú thích trụ tung

plt.show() # hiển thị hình biểu đồ

**Biểu đồ dạng chấm :**

Ví dụ :

import numpy as np

import matplotlib.pyplot as plt

array =np.array([1,2,3])

array1 =np.array([4,5,6])

plt.scatter(array,array1) # tọa độ của các chấm

plt.xlabel(‘trục x’) # hiện dòng chú thích trục hoành

plt.ylabel(‘trục y’) # hiện dòng chú thích trụ tung

plt.show() # hiển thị hình biểu đồ

**Thư viện Pandas :**

**Series :**

Ví dụ :

import numpy

import pandas

myarray = numpy.array([1, 2, 3])

rownames = [‘a’,’b’,’c’]

myseries = pandas.Series(myarray, index=rownames)

print(myseries)

output :

a 1

b 2

c 3

Thêm cách gọi khác cho dãy số.

Khi muốn truy cập : myseries[0] bằng với cách gọi myseries[‘a’] đều truy cập dữ liệu đầu tiên.

**DataFrame :**

Ví dụ :

import numpy

import pandas

myarray = numpy.array([[1, 2, 3], [4, 5, 6]])

rownames = [‘a’, ‘b’]

colnames = [‘one’, ‘two’, ‘three’]

mydataframe = pandas.DataFrame(myarray, index=rownames, columns=colnames)

print(mydataframe)

output :

one two three

a 1 2 3

b 4 5 6

Thêm mảng chú thích (cách gọi) : mydataframe[‘one’] bằng cách gọi mydataframe.one đều truy cập các phàn tử trong cột ‘one’ (1 4).

**Chương 4 : Tải file csv bằng python.**

**Dùng thư viện mặc định.**

Ví dụ :

import csv

import numpy

filename = ‘pima-indians-diabetes.data.csv’

raw\_data = open(filename, ‘rb’)

reader =csv.reader(raw\_data,delimiter=’,’,quoting=csv.QUOTE\_NONE)

x = list(reader)

data = numpy.array (x).astype(‘float’)

print(data.shape)

Lấy dữ liệu trong file ‘pima-indians-diabetes.data.csv’ bằng thư viện có sẵn ban đầu.

**Dùng thư viện Numpy :**

*Truy cập file csv được tải xuống sẵn có trong máy :*

from numpy import loadtxt

filename = ‘pima-indians-diabetes.data.csv’

raw\_data = open(filename, ‘rb’)

data = loadtxt(raw\_data, delimiter=",")

print(data.shape)

Hàm loadtxt lấy từ thư viện numpy để lấy dữ liệu từ file csv ra (ưu điểm là không cần định dạng dữ liệu được lấy ra như cách mặc định).

*Truy cập qua đường dẫn url :*

from numpy import loadtxt

from urllib import urlopen

url = ‘https://goo.gl/vhm1eU’

raw\_data = urlopen(url)

dataset = loadtxt(raw\_data, delimiter=",")

print(dataset.shape)

Muốn lấy dữ liệu từ đường dẫn url phải dùng thêm thư viện urllib (hàm urlopen).

**Thư viện Pandas :**

*Truy cập dữ liệu file csv đã tải :*

from pandas import read\_csv

filename = ‘pima-indians-diabetes.data.csv’

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

print(data.shape)

Khi dùng pandas thì có thể chọn các cột cụ thể để truy cập.

*Truy cập đường dẫn url :*

from pandas import read\_csv

url = ‘https://goo.gl/vhm1eU’

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(url, names=names)

print(data.shape)

Truy cập được như cách truy cập file csv đã tải xuống nhưng thay bằng link url.

**Chương 5 :**

from pandas import read\_csv

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

peek = data.head(20)

print(peek)

Trích 20 dòng đầu tiên của 9 cột dữ liệu từ file csv dữ liệu.

from pandas import read\_csv

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

types = data.dtypes

print(types)

Kiểu dữ liệu của 9 cột được lấy ra từ file csv.

from pandas import read\_csv

from pandas import set\_option

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

set\_option(‘display.width’, 100)

set\_option(‘precision’, 3)

description = data.describe()

print(description)

.describe() trong DataFrame để thể hiện các thông số sau :

Count.

Mean.

Standard Deviation.

Minimum Value.

25th Percentile.

50th Percentile (Median).

75th Percentile.

Maximum Value

from pandas import read\_csv

from pandas import set\_option

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

set\_option(‘display.width’, 100)

set\_option(‘precision’, 3)

class\_counts = data.groupby(‘class’).size()

print(class\_counts)

Thống kê số lượng của từng loại có trong cột ‘class’.

from pandas import read\_csv

from pandas import set\_option

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

set\_option(‘display.width’, 100)

set\_option(‘precision’, 3)

correlations = data.corr(method="pearson")

Tính ma trận hệ số tương quan bằng phương pháp Pearson.

from pandas import read\_csv

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

skew = data.skew()

print(skew)

Tính độ lệch của 9 cột trong file csv.

**Chương 6 : Hiểu về trực quan hóa dữ liệu.**

**Biểu đồ Histograms :**

from matplotlib import pyplot

from pandas import read\_csv

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

data.hist()

pyplot.show()

Chuyển các cột dữ liệu thành biểu đồ histograms rồi biểu diễn ra.

**Biểu đồ mật độ :**

from matplotlib import pyplot

from pandas import read\_csv

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

data.plot(kind=’density’, subplots=True, layout=(3,3), sharex=False)

pyplot.show()

Chú thích :

subplots=True : mỗi cột có một biểu đồ riêng.

sharex=False : mỗi biểu độ sẽ có trục x riêng biệt.

kind=’density’ : loại biểu đồ.

**Box and Whisker plots**

Biểu đồ Box and Whisker (hay còn gọi là Box Plot) là một cách hiệu quả để trực quan hóa sự phân phối dữ liệu, và tính các giá trị bất thường (outliers), giá trị trung vị (median), và phạm vi của dữ liệu (từ Q1 đến Q3)

from matplotlib import pyplot

from pandas import read\_csv

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

data.plot(kind=’box’, subplots=True, layout=(3,3), sharex=False,sharey=False)

pyplot.show()

Chú thích :

subplots=True : mỗi cột có một biểu đồ riêng.

sharex=False : mỗi biểu độ sẽ có trục x riêng biệt.

sharey=False : mỗi biểu độ sẽ có trục y riêng biệt.

kind=’box’ : loại biểu đồ.

**Biểu đồ ma trận tương quan :**

from matplotlib import pyplot

from pandas import read\_csv

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

correlations = data.corr()

fig = pyplot.figure()

ax = fig.add\_subplot(111)

cax = ax.matshow(correlations, vmin=-1, vmax=1)

fig.colorbar(cax)

ticks = numpy.arange(0,9,1)

ax.set\_xticks(ticks)

ax.set\_yticks(ticks)

ax.set\_xticklabels(names)

ax.set\_yticklabels(names)

pyplot.show()

**Biểu đồ ma trận phân tán :**

from matplotlib import pyplot

from pandas import read\_csv

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

scatter\_matrix(data)

pyplot.show()

**Chương 7 : Chuẩn bị dữ liệu cho Machine Learning.**

**Tái tỉ lệ dữ liệu :**

from pandas import read\_csv

from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

array = data.values

X = array[:,0:8]

Y = array[:,8]

num\_folds = 10

kfold = KFold(n\_splits=10, random\_state=7)

model = LinearDiscriminantAnalysis()

results = cross\_val\_score(model, X, Y, cv=kfold)

print(results.mean())

**Chuẩn hóa dữ liệu :**

from sklearn.preprocessing import Normalizer

from pandas import read\_csv

from numpy import set\_printoptions

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

array = data.values

X = array[:,0:8]

Y = array[:,8]

scaler = StandardScaler().fit(X)

rescaledX = scaler.transform(X)

set\_printoptions(precision=3)

print(rescaledX[0:5,:])

**Normalize Data :**

from sklearn.preprocessing import Normalizer

from pandas import read\_csv

from numpy import set\_printoptions

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

array = data.values

X = array[:,0:8]

Y = array[:,8]

scaler = Normalizer().fit(X)

normalizedX = scaler.transform(X)

set\_printoptions(precision=3)

print(normalizedX[0:5,:])

**Binarize Data :**

from sklearn.preprocessing import Normalizer

from pandas import read\_csv

from numpy import set\_printoptions

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

array = data.values

X = array[:,0:8]

Y = array[:,8]

binarizer = Binarizer(threshold=0.0).fit(X)

binaryX = binarizer.transform(X)

set\_printoptions(precision=3)

print(binaryX[0:5,:])

**Chương 8 : Feature Selection for Machine Learning.**

**Unvariate selection :**

from pandas import read\_csv

from numpy import set\_printoptions

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

array = data.values

X = array[:,0:8]

Y = array[:,8]

test = SelectKBest(score\_func=chi2, k=4)

fit = test.fit(X, Y)

set\_printoptions(precision=3)

print(fit.scores\_)

features = fit.transform(X)

print(features[0:5,:])

Chọn ra 4 đặc trưng tốt nhất.

**Recursive Feature Elimination :**

from pandas import read\_csv

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

array = data.values

X = array[:,0:8]

Y = array[:,8]

model = LogisticRegression()

rfe = RFE(model, 3)

fit = rfe.fit(X, Y)

print("Num Features: %d") % fit.n\_features\_

print("Selected Features: %s") % fit.support\_

print("Feature Ranking: %s") % fit.ranking\_

Dùng Logistic Regression, giữ lại 3 đặc trưng quan trọng nhất.

**Principal component analysis :**

Kỹ thuật giảm chiều dữ liệu.

from pandas import read\_csv

from sklearn.decomposition import PCA

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

array = data.values

X = array[:,0:8]

Y = array[:,8]

pca = PCA(n\_components=3)

fit = pca.fit(X)

print("Explained Variance: %s") % fit.explained\_variance\_ratio\_

print(fit.components\_)

**Feature importance :**

from pandas import read\_csv

from sklearn.ensemble import ExtraTreesClassifier

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

array = data.values

X = array[:,0:8]

Y = array[:,8]

model = ExtraTreesClassifier()

model.fit(X, Y)

print(model.feature\_importances\_)

Tính mức độ đóng góp của từng đặc trưng gốc vào thành phần chính.

**Chương 9 : Đánh giá hiệu suất của các thuật toán học máy bằng phương pháp resampling.**

**K-Fold Cross Validation :**

Dữ liệu được chia thành k phần (folds). Mô hình sẽ huấn luyện trên k-1 phần và kiểm tra trên phần còn lại. Quá trình này lặp lại k lần với mỗi phần được dùng làm tập kiểm tra một lần.

Ưu điểm: Đánh giá mô hình trên toàn bộ dữ liệu, giảm thiểu sai sót do phân chia dữ liệu ngẫu nhiên.

Nhược điểm: Tốn thời gian tính toán khi k lớn.

**Leave-One-Out Cross Validation (LOOCV) :**

Một dạng của k-fold, trong đó k = n (n là số lượng mẫu trong dữ liệu). Mỗi lần mô hình được huấn luyện trên tất cả các điểm dữ liệu trừ một và kiểm tra trên điểm còn lại.

Ưu điểm: Cung cấp đánh giá mô hình chính xác nhất, nhưng tốn rất nhiều tài nguyên tính toán.

**Repeated Random Test-Train Splits**

from pandas import read\_csv

from sklearn.model\_selection import ShuffleSplit

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

filename = "pima-indians-diabetes.data.csv"

names = [‘preg’, ‘plas’, ‘pres’, ‘skin’, ‘test’, ‘mass’, ‘pedi’, ‘age’, ‘class’]

data = read\_csv(filename, names=names)

array = data.values

X = array[:,0:8]

Y = array[:,8]

n\_splits = 10

test\_size = 0.33

seed = 7

kfold = ShuffleSplit(n\_splits=n\_splits, test\_size=test\_size, random\_state=seed)

model = LogisticRegression()

results = cross\_val\_score(model, X, Y, cv=kfold)

print("Accuracy: %.3f%% (%.3f%%)") % (results.mean()\*100.0, results.std()\*100.0)

Tính ra một danh sách các điểm số của mô hình trong mỗi lần chia và đánh giá trung bình để thấy được hiệu suất tổng thể của mô hình.

**Chapter 10**

**Machine Learning Algorithm Performance Metrics**

10.1 Algorithm Evaluation Metrics

Trong bài học này, chúng ta sẽ cùng tìm hiểu và áp dụng nhiều chỉ số đánh giá thuật toán khác nhau cho cả hai loại bài toán học máy: phân loại và hồi quy.

10.2 Classification Metrics

Trong phần này, chúng ta sẽ cùng tìm hiểu cách sử dụng các chỉ số đánh giá sau:

Độ chính xác phân loại (Classification Accuracy)

Hàm mất mát logarit (Logarithmic Loss / Log Loss)

Diện tích dưới đường cong ROC (Area Under ROC Curve / AUC)

Ma trận nhầm lẫn (Confusion Matrix)

Báo cáo phân loại (Classification Report)

10.2.1 Classification Accuracy

Độ chính xác phân loại là tỷ lệ giữa số lần dự đoán đúng trên tổng số dự đoán được thực hiện. Đây là chỉ số đánh giá phổ biến nhất cho các bài toán phân loại — nhưng đồng thời cũng là chỉ số dễ bị lạm dụng nhất.

Chỉ số này chỉ thực sự phù hợp khi số lượng mẫu trong mỗi lớp là tương đương nhau (điều này hiếm khi xảy ra trong thực tế) và khi tất cả các loại dự đoán đúng và sai đều có mức độ quan trọng như nhau — điều này cũng thường không đúng trong các bài toán thực tế.

10.2.2 Logarithmic Loss

Hàm mất mát logarit (logarithmic loss hay logloss) là một chỉ số dùng để đánh giá độ chính xác của các dự đoán dưới dạng xác suất đối với việc một mẫu thuộc vào một lớp nào đó.

Xác suất đầu ra (giá trị nằm trong khoảng từ 0 đến 1) thể hiện mức độ tự tin của mô hình đối với dự đoán của mình. Logloss sẽ "thưởng" cho những dự đoán đúng với độ tin cậy cao và "phạt" nặng hơn nếu mô hình tự tin nhưng lại đoán sai.

Nói cách khác:

Dự đoán đúng với xác suất cao → điểm logloss thấp (tốt).

Dự đoán sai với xác suất cao → điểm logloss cao (xấu).

Giá trị logloss càng nhỏ thì càng tốt, trong đó 0 là điểm logloss lý tưởng — tức là mô hình dự đoán hoàn toàn chính xác với độ tin cậy tuyệt đối.

10.2.3 Area Under ROC Curve

Diện tích dưới đường cong ROC (AUC - Area Under the ROC Curve) là một chỉ số đánh giá hiệu suất dành cho các bài toán phân loại nhị phân.

AUC thể hiện khả năng phân biệt giữa hai lớp (dương và âm) của mô hình.

Nếu AUC = 1.0 → mô hình dự đoán hoàn toàn chính xác.

Nếu AUC = 0.5 → mô hình đoán ngẫu nhiên, không có khả năng phân biệt giữa hai lớp.

Đường cong ROC dựa trên hai khái niệm chính: Độ nhạy (Sensitivity) và Độ đặc hiệu (Specificity). Khi xây dựng mô hình phân loại nhị phân, luôn tồn tại sự đánh đổi giữa hai yếu tố này.

Sensitivity (Độ nhạy) còn gọi là Recall hay True Positive Rate: là tỷ lệ các trường hợp thuộc lớp dương (lớp thứ nhất) được mô hình dự đoán đúng.

Specificity (Độ đặc hiệu) còn gọi là True Negative Rate: là tỷ lệ các trường hợp thuộc lớp âm (lớp thứ hai) được mô hình dự đoán đúng.

Nói cách khác, một mô hình tốt là mô hình vừa phát hiện đúng các trường hợp dương (độ nhạy cao), vừa tránh nhầm lẫn các trường hợp âm là dương (độ đặc hiệu cao).

Có thể thấy rằng giá trị AUC khá gần với 1 và lớn hơn 0.5, điều này cho thấy mô hình có khả năng phân biệt nhất định giữa hai lớp — tức là mô hình có hiệu quả trong việc dự đoán, chứ không phải chỉ đoán ngẫu nhiên.

10.2.4 Confusion Matrix

Ma trận nhầm lẫn (Confusion Matrix) là một công cụ hữu ích để trình bày độ chính xác của mô hình phân loại với hai hoặc nhiều lớp. Ma trận này thể hiện các dự đoán trên trục hoành (x-axis) và kết quả thực tế trên trục tung (y-axis). Các ô trong ma trận chứa số lượng dự đoán được mô hình học máy đưa ra.

10.2.5 Classification Report

Thư viện scikit-learn cung cấp một báo cáo tiện lợi khi làm việc với các bài toán phân loại, giúp bạn nhanh chóng có cái nhìn tổng quan về độ chính xác của mô hình thông qua một số chỉ số.

Hàm classification\_report() hiển thị các chỉ số như precision (độ chính xác), recall (độ nhạy), F1-score, và support (sự hỗ trợ) cho từng lớp trong bài toán phân loại.

10.3 Regression Metrics

Trong phần này, chúng ta sẽ cùng tìm hiểu 3 chỉ số đánh giá phổ biến nhất dùng để đánh giá các dự đoán trong bài toán hồi quy của học máy:

Mean Absolute Error (MAE) - Lỗi tuyệt đối trung bình

Mean Squared Error (MSE) - Lỗi bình phương trung bình

R² (Hệ số xác định)

10.3.1 Mean Absolute Error

Lỗi tuyệt đối trung bình (MAE - Mean Absolute Error) là tổng của các sai số tuyệt đối giữa các dự đoán và giá trị thực tế. MAE cho bạn một cái nhìn tổng quan về mức độ sai lệch của các dự đoán, nhưng không cho biết hướng sai lệch (ví dụ: dự đoán quá cao hay quá thấp).

Giá trị MAE bằng 0 nghĩa là không có sai sót, tức là các dự đoán là chính xác hoàn toàn.

10.3.2 Mean Squared Error

Lỗi bình phương trung bình (MSE - Mean Squared Error) tương tự như lỗi tuyệt đối trung bình (MAE) trong việc cung cấp một cái nhìn tổng quan về mức độ sai lệch của dự đoán, nhưng MSE làm mạnh hơn sai số lớn bằng cách bình phương các sai số. Điều này có nghĩa là các sai số lớn sẽ có ảnh hưởng lớn hơn đến kết quả tính toán.

Khi bạn lấy căn bậc hai của MSE, bạn sẽ có Lỗi bình phương trung bình căn (RMSE - Root Mean Squared Error), giúp chuyển đơn vị về lại đơn vị ban đầu của biến đầu ra và dễ hiểu hơn khi trình bày và mô tả kết quả.

Lưu ý rằng chỉ số này cũng bị đảo dấu trong hàm cross\_val\_score() để sắp xếp kết quả theo thứ tự tăng dần.

Nếu bạn muốn tính RMSE, đừng quên lấy giá trị tuyệt đối trước khi tính căn bậc hai nếu sai số có thể là số âm.

10.3.3 *R*2 Metric

*R*2 (Hệ số xác định) là một chỉ số dùng để đánh giá mức độ phù hợp của một tập hợp các dự đoán với các giá trị thực tế. Trong lý thuyết thống kê, chỉ số này được gọi là hệ số xác định.

*R*2 có giá trị nằm trong khoảng từ 0 đến 1:

0 đại diện cho một mô hình không có khả năng phù hợp với dữ liệu.

1 đại diện cho một mô hình phù hợp hoàn hảo với dữ liệu.

Nếu giá trị *R*2 gần 0 và nhỏ hơn 0.5, bạn có thể thấy rằng các dự đoán có sự phù hợp kém với các giá trị thực tế.

**Chapter 11**

**Spot-Check Classification Algorithms**

11.1 Algorithm Spot-Checking

Bạn không thể biết trước thuật toán nào sẽ hoạt động tốt nhất trên tập dữ liệu của mình. Thay vào đó, bạn cần sử dụng phương pháp thử và sai (trial and error) để tìm ra một danh sách các thuật toán hoạt động hiệu quả, sau đó tiếp tục tinh chỉnh những thuật toán này. Tôi gọi quá trình này là kiểm tra nhanh (spot-checking).

Dưới đây là một vài gợi ý khi kiểm tra nhanh các thuật toán trên tập dữ liệu của bạn:

Thử nghiệm nhiều loại biểu diễn thuật toán (ví dụ: các thuật toán dựa trên cá thể (instances) và cây (trees)).

Thử nhiều thuật toán học khác nhau, kể cả khi chúng học cùng một kiểu mô hình.

Thử cả mô hình tuyến tính và phi tuyến, hoặc mô hình tham số và phi tham số.

11.2 Algorithms Overview

Chúng ta sẽ cùng tìm hiểu sáu thuật toán phân loại mà bạn có thể áp dụng để kiểm tra nhanh (spot-check) trên tập dữ liệu của mình. Bắt đầu với hai thuật toán học máy tuyến tính:

Logistic Regression (Hồi quy logistic)

Linear Discriminant Analysis (Phân tích phân biệt tuyến tính)

Sau đó là bốn thuật toán học máy phi tuyến:

k-Nearest Neighbors (K láng giềng gần nhất)

Naive Bayes (Bayes ngây thơ)

Classification and Regression Trees (Cây phân loại và hồi quy - CART)

Support Vector Machines (Máy vector hỗ trợ - SVM)

11.3 Linear Machine Learning Algorithms

11.3.1 Logistic Regression

Hồi quy logistic giả định rằng các biến đầu vào dạng số tuân theo phân phối chuẩn (Gaussian) và có thể được sử dụng để mô hình hóa các bài toán phân loại nhị phân.

11.4 Nonlinear Machine Learning Algorithms

11.4.1 k-Nearest Neighbors

Thuật toán k-Nearest Neighbors (KNN) sử dụng hàm đo khoảng cách để tìm ra k mẫu gần nhất trong tập huấn luyện so với một mẫu mới, sau đó dự đoán kết quả dựa trên giá trị trung bình (hoặc đa số) của các hàng xóm này.

11.4.2 Naive Bayes

Naive Bayes tính xác suất của từng lớp và xác suất có điều kiện của mỗi lớp dựa trên từng giá trị đầu vào. Các xác suất này được ước lượng cho dữ liệu mới và nhân lại với nhau, với giả định rằng tất cả các biến đầu vào là độc lập với nhau — một giả định đơn giản nên được gọi là “naive”.

Khi làm việc với dữ liệu dạng số thực, thuật toán thường giả định phân phối chuẩn (Gaussian) để dễ dàng ước lượng xác suất của các biến đầu vào, thông qua hàm mật độ xác suất Gaussian.

11.4.3 Classification and Regression Trees

Classification and Regression Trees (CART) — hay thường gọi là cây quyết định — xây dựng một cây nhị phân dựa trên dữ liệu huấn luyện.

Tại mỗi nút, thuật toán sẽ chọn điểm chia (split point) một cách tham lam (greedy) bằng cách đánh giá tất cả các thuộc tính và giá trị có thể của chúng trong dữ liệu, nhằm tối thiểu hóa một hàm chi phí, chẳng hạn như chỉ số Gini (Gini index).

11.4.4 Support Vector Machines

Support Vector Machines (SVM) là thuật toán tìm kiếm một đường ranh giới (đường thẳng hoặc siêu phẳng) sao cho tách biệt hai lớp dữ liệu một cách tốt nhất.

Những điểm dữ liệu nằm gần nhất với đường phân cách này được gọi là support vectors (vector hỗ trợ) và chúng quyết định vị trí của đường phân cách.

**Chapter 12**

**Spot-Check Regression Algorithms**

12.1 Algorithms Overview

Trong bài học này, chúng ta sẽ tìm hiểu về bảy thuật toán hồi quy mà bạn có thể dùng để thử nghiệm (spot-check) trên bộ dữ liệu của mình.

Bắt đầu với bốn thuật toán hồi quy tuyến tính:

Linear Regression (Hồi quy tuyến tính)

Ridge Regression (Hồi quy Ridge)

LASSO Linear Regression (Hồi quy tuyến tính LASSO)

Elastic Net Regression (Hồi quy Elastic Net)

Sau đó là ba thuật toán hồi quy phi tuyến:

k-Nearest Neighbors (KNN)

Classification and Regression Trees (Cây phân loại và hồi quy)

Support Vector Machines (Máy vector hỗ trợ - SVM)

12.2 Linear Machine Learning Algorithms

12.2.1 Linear Regression

Hồi quy tuyến tính (Linear Regression) giả định rằng các biến đầu vào tuân theo phân phối chuẩn (Gaussian).

Thuật toán cũng giả định rằng các biến đầu vào có liên quan đến biến đầu ra và không bị tương quan chặt chẽ với nhau — nếu có, đó là vấn đề gọi là đa cộng tuyến (collinearity).

12.2.2 Ridge Regression

Hồi quy Ridge (Ridge Regression) là một sự mở rộng của hồi quy tuyến tính, trong đó hàm mất mát được điều chỉnh để giảm thiểu độ phức tạp của mô hình. Độ phức tạp này được đo bằng tổng bình phương các giá trị hệ số (L2-norm), giúp ngăn ngừa hiện tượng overfitting và cải thiện khả năng tổng quát của mô hình.

12.2.3 LASSO Regression

LASSO (Least Absolute Shrinkage and Selection Operator) là một sự điều chỉnh của hồi quy tuyến tính, tương tự như hồi quy Ridge, nhưng thay vì sử dụng tổng bình phương các giá trị hệ số, hàm mất mát trong LASSO được điều chỉnh để giảm thiểu độ phức tạp của mô hình bằng tổng giá trị tuyệt đối của các hệ số (hay còn gọi là L1-norm).

Điều này giúp chọn lọc các biến quan trọng và có thể dẫn đến lựa chọn biến tự động bằng cách làm cho một số hệ số trở thành 0.

12.2.4 ElasticNet Regression

ElasticNet là một hình thức hồi quy điều chuẩn hóa kết hợp các đặc tính của cả hồi quy Ridge và hồi quy LASSO.

ElasticNet nhằm mục đích giảm thiểu độ phức tạp của mô hình hồi quy (bao gồm cả độ lớn và số lượng các hệ số hồi quy) bằng cách phạt mô hình sử dụng cả L2-norm (tổng bình phương các giá trị hệ số) và L1-norm (tổng giá trị tuyệt đối của các hệ số).

Điều này giúp ElasticNet kết hợp lợi ích của cả hai phương pháp Ridge và LASSO, cho phép mô hình xử lý tốt hơn trong các tình huống có nhiều biến giải thích hoặc mối quan hệ không hoàn toàn tuyến tính giữa các biến đầu vào.

12.3 Nonlinear Machine Learning Algorithms

12.3.1 K-Nearest Neighbors

Thuật toán k-Nearest Neighbors (KNN) tìm ra k mẫu gần nhất trong tập huấn luyện cho một mẫu dữ liệu mới. Từ k hàng xóm, một giá trị trung bình hoặc giá trị trung vị của biến đầu ra sẽ được chọn làm dự đoán.

Một điểm cần lưu ý là hàm đo khoảng cách sử dụng (thông qua tham số metric). Mặc định, khoảng cách Minkowski được sử dụng, đây là một tổng quát của cả khoảng cách Euclidean (sử dụng khi tất cả các đầu vào có cùng đơn vị đo) và khoảng cách Manhattan (dùng khi các biến đầu vào có đơn vị đo khác nhau).

12.3.2 Classification and Regression Trees

Cây quyết định hay Cây phân loại và hồi quy (CART) sử dụng dữ liệu huấn luyện để chọn các điểm phân chia tốt nhất nhằm tối thiểu hóa một chỉ số chi phí.

Chỉ số chi phí mặc định cho cây quyết định hồi quy là sai số bình phương trung bình (Mean Squared Error - MSE), được chỉ định thông qua tham số criterion.

12.3.3 Support Vector Machines

Máy vector hỗ trợ (SVM) được phát triển ban đầu cho phân loại nhị phân. Tuy nhiên, kỹ thuật này đã được mở rộng để dự đoán các bài toán với giá trị thực, gọi là Hồi quy máy vector hỗ trợ (Support Vector Regression - SVR).

Giống như ví dụ phân loại, SVR được xây dựng dựa trên thư viện LIBSVM.

**Chapter 13**

**Compare Machine Learning Algorithms**

13.1 Choose The Best Machine Learning Model

Khi làm việc trên một dự án học máy, bạn thường sẽ có nhiều mô hình tốt để lựa chọn. Mỗi mô hình sẽ có các đặc điểm hiệu suất khác nhau. Sử dụng các phương pháp tái mẫu như cross-validation, bạn có thể ước tính độ chính xác của từng mô hình đối với dữ liệu chưa thấy. Bạn cần phải sử dụng những ước tính này để chọn ra một hoặc hai mô hình tốt nhất từ các mô hình mà bạn đã tạo ra.

13.2 Compare Machine Learning Algorithms Consistently

Chìa khóa để so sánh công bằng các thuật toán học máy là đảm bảo rằng mỗi thuật toán được đánh giá theo cùng một cách trên cùng một tập dữ liệu. Bạn có thể đạt được điều này bằng cách yêu cầu mỗi thuật toán được đánh giá trên một bộ kiểm tra (test harness) nhất quán.

Khi có một tập dữ liệu mới, việc hình dung dữ liệu bằng nhiều kỹ thuật khác nhau là một ý tưởng hay để nhìn nhận dữ liệu từ nhiều góc độ. Ý tưởng tương tự cũng áp dụng cho việc chọn mô hình. Bạn nên sử dụng một số cách khác nhau để đánh giá độ chính xác ước tính của các thuật toán học máy nhằm chọn ra một hoặc hai thuật toán để hoàn thiện. Một cách để làm điều này là sử dụng các phương pháp hình dung để hiển thị độ chính xác trung bình, biến thiên và các đặc tính khác của phân phối độ chính xác của mô hình.

**Chapter 14**

**Automate Machine Learning Workflows with Pipelines**

14.1 Automating Machine Learning Workflows

Có những quy trình làm việc chuẩn trong học máy ứng dụng. Những quy trình này giúp khắc phục các vấn đề phổ biến như rò rỉ dữ liệu (data leakage) trong bộ kiểm tra của bạn. Scikit-learn của Python cung cấp một công cụ gọi là Pipeline để giúp tự động hóa quy trình làm việc trong học máy. Pipeline hoạt động bằng cách cho phép chuỗi các bước chuyển đổi dữ liệu được liên kết với nhau, cuối cùng dẫn đến quá trình mô hình hóa có thể được đánh giá.

Mục tiêu là đảm bảo rằng tất cả các bước trong pipeline đều bị giới hạn chỉ với dữ liệu có sẵn cho việc đánh giá, chẳng hạn như tập dữ liệu huấn luyện hoặc mỗi fold trong quy trình cross-validation.

14.2 Data Preparation and Modeling Pipeline

Một cái bẫy dễ rơi vào trong học máy ứng dụng là rò rỉ dữ liệu từ tập huấn luyện vào tập kiểm tra. Để tránh bẫy này, bạn cần một bộ kiểm tra vững chắc với sự phân tách rõ ràng giữa huấn luyện và kiểm tra. Điều này bao gồm cả quá trình chuẩn bị dữ liệu. Việc chuẩn bị dữ liệu là một cách dễ dàng để làm rò rỉ kiến thức về toàn bộ tập huấn luyện vào thuật toán. Ví dụ, chuẩn hóa hoặc chuẩn hoá dữ liệu trên toàn bộ tập huấn luyện trước khi học sẽ không phải là một kiểm tra hợp lệ, vì tập huấn luyện sẽ bị ảnh hưởng bởi quy mô của dữ liệu trong tập kiểm tra.

Pipeline giúp bạn ngăn ngừa rò rỉ dữ liệu trong bộ kiểm tra của mình bằng cách đảm bảo rằng quá trình chuẩn bị dữ liệu như chuẩn hoá được giới hạn trong mỗi fold của quy trình cross-validation.

14.3 Feature Extraction and Modeling Pipeline

Rút trích đặc trưng là một quy trình khác dễ bị rò rỉ dữ liệu. Tương tự như chuẩn bị dữ liệu, các quy trình rút trích đặc trưng phải được giới hạn trong tập huấn luyện. Pipeline cung cấp một công cụ tiện lợi gọi là FeatureUnion, cho phép kết hợp kết quả từ nhiều quy trình lựa chọn và rút trích đặc trưng vào một tập dữ liệu lớn hơn mà mô hình có thể được huấn luyện. Quan trọng là, tất cả các bước rút trích đặc trưng và FeatureUnion đều xảy ra trong mỗi fold của quy trình cross-validation, giúp ngăn ngừa rò rỉ dữ liệu.

**Chapter 15**

**Improve Performance with Ensembles**

15.1 Combine Models Into Ensemble Predictions

Ba phương pháp phổ biến nhất để kết hợp các dự đoán từ các mô hình khác nhau là:

Bagging: Xây dựng nhiều mô hình (thường là cùng loại) từ các mẫu con khác nhau của tập huấn luyện.

Boosting: Xây dựng nhiều mô hình (thường là cùng loại), mỗi mô hình học cách sửa chữa các lỗi dự đoán của mô hình trước trong chuỗi các mô hình.

Voting: Xây dựng nhiều mô hình (thường là các loại khác nhau), sau đó sử dụng các thống kê đơn giản (như tính trung bình) để kết hợp các dự đoán.

15.2 Bagging Algorithms

Bootstrap Aggregation (Bagging) là phương pháp lấy mẫu nhiều lần từ tập huấn luyện của bạn (với sự thay thế) và huấn luyện một mô hình cho mỗi mẫu. Dự đoán cuối cùng là trung bình của các dự đoán từ tất cả các mô hình con.

Ba mô hình bagging được đề cập trong phần này là:

Bagged Decision Trees: Là việc sử dụng quyết định cây (Decision Trees) trong một môi trường bagging để giảm thiểu độ phức tạp và tăng cường độ chính xác của mô hình.

Random Forest: Một kỹ thuật tiên tiến của bagging, sử dụng nhiều cây quyết định (decision trees) và chọn lựa ngẫu nhiên các tính năng cho mỗi cây để tránh overfitting và cải thiện hiệu suất.

Extra Trees: Là một biến thể của Random Forest, nhưng sử dụng sự ngẫu nhiên cao hơn trong việc chọn lựa các điểm chia (splits) trong cây quyết định, giúp tăng cường sự đa dạng của mô hình và giảm độ thiên lệch.

15.2.1 Bagged Decision Trees

Bagging hoạt động tốt nhất với các thuật toán có độ biến thiên cao. Một ví dụ phổ biến là cây quyết định (decision trees), thường được xây dựng mà không có bước cắt tỉa (pruning).

15.2.2 Random Forest

Random Forests là một sự mở rộng của phương pháp bagged decision trees. Các mẫu dữ liệu huấn luyện được chọn với sự thay thế (sampling with replacement), nhưng các cây quyết định được xây dựng theo cách giảm bớt sự tương quan giữa các phân loại viên (classifiers) cá nhân.

Cụ thể, thay vì chọn điểm chia (split point) tốt nhất một cách tham lam (greedily) khi xây dựng mỗi cây, Random Forests chỉ xem xét một tập con ngẫu nhiên của các đặc trưng (features) cho mỗi điểm chia. Điều này giúp làm giảm sự tương quan giữa các cây trong rừng, từ đó tăng cường khả năng tổng quát (generalization) và cải thiện hiệu suất của mô hình.

15.2.3 Extra Trees

Extra Trees (Cây cực kỳ ngẫu nhiên) là một biến thể khác của phương pháp bagging, trong đó các cây quyết định được xây dựng từ các mẫu dữ liệu huấn luyện.

15.3 Boosting Algorithms

Các thuật toán Boosting tạo ra một chuỗi các mô hình cố gắng sửa các sai lầm của các mô hình trước đó trong chuỗi. Sau khi được xây dựng, các mô hình này đưa ra dự đoán, có thể được trọng số theo độ chính xác đã được chứng minh của chúng, và kết quả cuối cùng được kết hợp để tạo ra dự đoán đầu ra cuối cùng. Hai thuật toán boosting phổ biến nhất trong học máy là:

AdaBoost (Adaptive Boosting)

Stochastic Gradient Boosting

15.3.1 AdaBoost

AdaBoost là thuật toán boosting ensemble thành công đầu tiên. Nó hoạt động chủ yếu bằng cách gán trọng số cho các mẫu trong bộ dữ liệu dựa trên mức độ dễ dàng hoặc khó khăn khi phân loại, cho phép thuật toán chú ý nhiều hơn hoặc ít hơn đến chúng trong việc xây dựng các mô hình tiếp theo.

15.3.2 Stochastic Gradient Boosting

Stochastic Gradient Boosting (còn gọi là Gradient Boosting Machines) là một trong những kỹ thuật ensemble tinh vi nhất. Đây cũng là một kỹ thuật đang chứng tỏ có thể là một trong những phương pháp tốt nhất hiện có để cải thiện hiệu suất thông qua các bộ ensemble.

15.4 Voting Ensemble

Voting là một trong những cách đơn giản nhất để kết hợp dự đoán từ nhiều thuật toán học máy. Nó hoạt động bằng cách tạo ra hai hoặc nhiều mô hình độc lập từ bộ dữ liệu huấn luyện của bạn. Sau đó, một Voting Classifier có thể được sử dụng để bao bọc các mô hình của bạn và trung bình các dự đoán của các mô hình con khi yêu cầu dự đoán cho dữ liệu mới. Các dự đoán của các mô hình con có thể được trọng số, nhưng việc chỉ định trọng số cho các bộ phân loại thủ công hoặc thậm chí theo cách ước lượng là khá khó khăn. Các phương pháp tiên tiến hơn có thể học cách trọng số dự đoán từ các mô hình con tốt nhất, nhưng điều này được gọi là stacking (gộp chồng) và hiện tại không có sẵn trong scikit-learn.

**Chapter 16**

**Improve Performance with Algorithm Tuning**

16.1 Machine Learning Algorithm Parameters

Tối ưu hóa thuật toán là bước cuối cùng trong quy trình học máy ứng dụng trước khi hoàn thiện mô hình của bạn. Nó đôi khi được gọi là tối ưu hóa siêu tham số, trong đó các tham số của thuật toán được gọi là siêu tham số, trong khi các hệ số được tìm ra bởi thuật toán học máy tự nó được gọi là tham số. Tối ưu hóa ám chỉ tính chất tìm kiếm của vấn đề. Khi được xem như một vấn đề tìm kiếm, bạn có thể sử dụng các chiến lược tìm kiếm khác nhau để tìm ra một tham số hoặc tập hợp tham số tốt và mạnh mẽ cho một thuật toán trên một vấn đề cụ thể. Python scikit-learn cung cấp hai phương pháp đơn giản để điều chỉnh tham số thuật toán:

Tuning tham số bằng Tìm kiếm lưới (Grid Search).

Tuning tham số bằng Tìm kiếm ngẫu nhiên (Random Search).

16.2 Grid Search Parameter Tuning

Tìm kiếm lưới (Grid search) là một phương pháp điều chỉnh tham số mà sẽ xây dựng và đánh giá một mô hình cho mỗi kết hợp tham số thuật toán được chỉ định trong một lưới.

16.3 Random Search Parameter Tuning

Tìm kiếm ngẫu nhiên (Random search) là một phương pháp điều chỉnh tham số mà sẽ lấy mẫu các tham số thuật toán từ một phân phối ngẫu nhiên (ví dụ: phân phối đồng nhất) trong một số lượng vòng lặp cố định. Một mô hình được xây dựng và đánh giá cho mỗi kết hợp tham số đã chọn.

**Chapter 17**

**Save and Load Machine Learning Models**

17.1 Finalize Your Model with pickle

Pickle là cách chuẩn để tuần tự hóa (serialization) các đối tượng trong Python.

17.2 Finalize Your Model with Joblib

Thư viện Joblib là một phần của hệ sinh thái SciPy và cung cấp các tiện ích cho việc xử lý công việc (jobs) trong Python. Nó cung cấp các công cụ để lưu và tải các đối tượng Python sử dụng cấu trúc dữ liệu NumPy một cách hiệu quả. Điều này rất hữu ích cho một số thuật toán học máy yêu cầu nhiều tham số hoặc lưu trữ toàn bộ tập dữ liệu (ví dụ: k-Nearest Neighbors).

17.3 Tips for Finalizing Your Model

Phần này liệt kê một số yếu tố quan trọng khi hoàn thiện các mô hình học máy của bạn.

Phiên bản Python: Lưu ý phiên bản Python bạn đang sử dụng. Bạn gần như chắc chắn cần phải sử dụng cùng một phiên bản chính (và có thể là phiên bản phụ) của Python khi bạn lưu và tải mô hình trong tương lai.

Phiên bản thư viện: Phiên bản của tất cả các thư viện chính được sử dụng trong dự án học máy của bạn gần như chắc chắn cần phải giống nhau khi tải lại mô hình đã lưu. Điều này không chỉ giới hạn ở phiên bản của NumPy và scikit-learn.

Lưu trữ thủ công: Bạn có thể muốn xuất các tham số của mô hình đã học một cách thủ công để có thể sử dụng chúng trực tiếp trong scikit-learn hoặc nền tảng khác trong tương lai. Thường thì các kỹ thuật được sử dụng nội bộ bởi các thuật toán học máy để đưa ra dự đoán đơn giản hơn nhiều so với các phương pháp được sử dụng để học tham số, và chúng có thể dễ dàng triển khai trong mã tùy chỉnh mà bạn kiểm soát.