+

LỜI MỞ ĐẦU

Công nghệ thông tin đang ngày một phát triển mạnh mẽ hơn. Do nhu cầu trao đổi thông tin ngày càng tăng và nhu cầu sử dụng sản phẩm công nghệ cao nhiều tính năng, cấu hình cao, chất lượng tốt, kiểu dáng mẫu mã đẹp, phong phú nên nhà cung cấp phải luôn cải thiện, nâng cao những sản phẩm của mình. Do đó việc xây dựng các website đang là một ngành công nghiệp mới đầy tiềm năng và hứa hẹn nhiều sự phát triển vượt bậc của ngành khoa học kỹ thuật. Cùng với sự phát triển mạnh mẽ của xu hướng lập trình các website. Các nghiệp vụ trên website hiện nay rất đa dạng và phong phú. Hiện nay, việc sở hữu một chiếc điện thoại thông minh không còn là một điều quá khó với nhiều người, khi ai cũng có cho riêng mình một chiếc điện thoại thông minh thì việc sử dụng nó để phục vụ cho cuộc sống là một điều tất yếu. Một trong những framework tốt để xây dựng chương trình. Chính bởi sự hữu dụng này mà ngày nay, các website được viết bằng Pycharm đang dần trở lên phổ biến. Và trong báo cáo này, em xin chọn đề tài “tensorflow serving khái niệm, cơ chế hoạt động và ứng dụng”. Trong quá trình làm báo cáo cũng như xây dựng, sẽ còn rất nhiều nhược điểm, thiếu sót, rất mong được thầy cô chỉ dẫn và bổ sung để bản báo cáo này được hoàn thiện hơn.

Em xin cảm ơn cô!

MỤC LỤC

DANH MỤC HÌNH ẢNH

Hình 1: Tổng quan về Tensorflow Serving

Hình 1.2: Sơ đồ Tensorflow Serving

Hình 3.1.1: Một mạng neuron đơn giản

Hình 3.1.2: Mạng neuron với một lớp ẩn

DANH MỤC TỪ VIẾT TẮT

AI Artificial Intelligence

ANN Artificial Neural Network

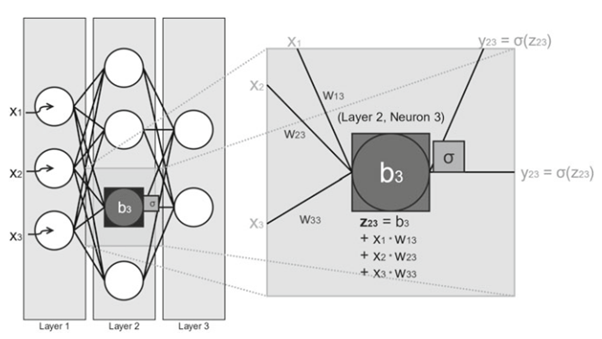
MAE Mean Absolute Error

Chương I: LÝ THUYẾT VỀ MẠNG NEURON SỬ DỤNG TENSORFLOW

# 1.1. Nguyên lý hoạt động của mạng Neuron nhân tạo

## **1.1.1. Mạng neuron nhân tạo**

Mạng neuron nhân tạo mô phỏng cấu trúc của bộ não con người. Một mạng neuron nhân tạo gồm có một lớp đầu vào (input layer), một lớp đầu ra (output layer), một hoặc nhiều lớp ẩn (hidden layer). Cấu trúc của một mạng neuron nhân tạo đơn giản với một lớp ẩn (bên trái) và phóng to một neuron (bên phải) được minh họa trong hình 1.1



Hình 1.1 Một mạng neuron đơn giản

Số lượng các neuron của lớp đầu vào và lớp đầu ra phụ thuộc vào yêu cầu của vấn đề mà mạng neuron đang giải quyết. Số lượng các lớp ẩn và số neuron của mỗi lớp ẩn thường là kết quả của một quá trình thử nghiệm để tìm ra số lượng tối ưu. Việc sử dụng nhiều lớp ẩn trong một mạng neuron thường được gọi là học sâu (deep learning). Mỗi neuron của một lớp sẽ kết nối đến tất cả các neuron của lớp tiếp theo. Các neuron của cùng một lớp sẽ không kết nối với nhau. Mỗi liên kết sẽ được gán một trọng số (weight). Các trọng số này sẽ quyết định lượng thông tin mà một lớp chuyển đến một lớp kế tiếp trong mạng.

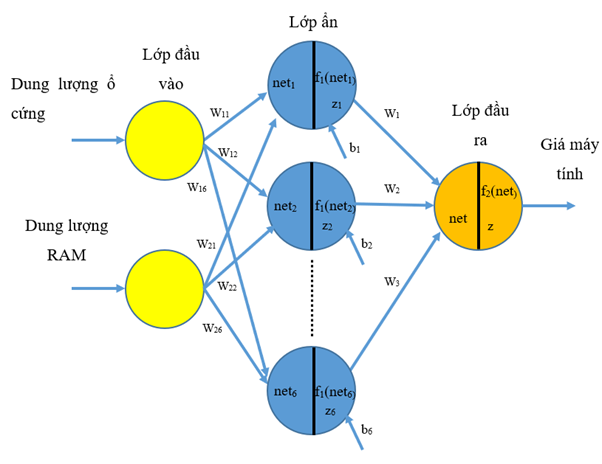
Với mạng neuron trong 1.1, thông tin sẽ được chuyển từ các neuron của lớp thứ nhất (lớp đầu vào), qua các neuron của lớp thứ hai (lớp ẩn) rồi tới các neuron của lớp thứ ba (lớp đầu ra). Lớp đầu vào gồm có 3 neuron, mỗi neuron có thể nhận một giá trị đầu vào. Các giá trị đầu vào được ký hiệu bằng các biến x1, x2, x3. Chức năng duy nhất của lớp đầu vào là nhận các giá trị đầu vào. Số lượng các giá trị đầu vào có thể nhỏ hơn số lượng của các neuron trong lớp đầu vào vì ta có thể truyền giá trị 0 cho các neuron không sử dụng. Tuy nhiên mỗi neuron chỉ có thể nhận một giá trị đầu vào nên số lượng các biến đầu vào không thể vượt quá số lượng neuron của lớp đầu vào.

Trên hình 1.1, neuron 3 của lớp thứ 2 nhận các giá trị đầu vào là x1, x2, x3 và trọng số của các liên kết tương ứng là w13, w23, w33. Mỗi neuron đều có một đại lượng gọi là bias, trong trường hợp neuron này là b3. Tại mỗi neuron tổng các tích giữa các giá trị đầu vào với trọng số tương ứng cộng với giá trị bias sẽ được đưa qua một hàm kích hoạt (activation function) để tăng tính phi tuyến cho mô hình. Có nhiều loại hàm kích hoạt khác nhau nhưng được sử dụng phổ biến nhất là hàm sigmoid có dạng σ(z)=1/(1+e^(-z) ) và hàm Rectifier σ(z)=max⁡(z,0) . Các lớp khác nhau có thể sử dụng các hàm kích hoạt khác nhau nhưng các neuron trong cùng một lớp sẽ sử dụng cùng một hàm kích hoạt giống nhau.

Thông thường các giá trị đầu vào và đầu ra của mạng neuron thường được biểu diễn dưới dạng vector (x1, x2, x3,…, xn) và (y1, y2,…, yn).

Trong nghiên cứu này, mạng neuron sẽ gồm lớp đầu vào, các lớp ẩn, và một lớp đầu ra. Lớp đầu vào gồm có 2 neuron bởi vì ta sẽ có hai tham số đầu vào là dung lượng ổ cứng (GB) và dung lượng bộ nhớ RAM (GB) của máy tính. . Lớp đầu ra của mạng neuron gồm có 1 neuron là giá của máy tính.

Ví dụ sơ đồ của một mạng Neuron gồm có 1 lớp ẩn như hình bên dưới:



Trong đó:

xi là biến đầu vào thứ i

wih là trọng số (weight) của liên kết giữa neuron thứ i của lớp đầu vào với neuron thứ h của lớp ẩn (h nhận các giá trị 1,2,..,6)

bh là giá trị bias tương ứng với neuron thứ h của lớp ẩn

f1 là hàm kích hoạt (activation) của lớp ẩn

zh là đầu ra của neuron thứ h của lớp ẩn

neth là giá trị tổng các tích giữa giá trị đầu vào nhân với trọng số tương ứng của liên kết giữa mỗi neuron đầu vào với neuron thứ h của lớp ẩn cộng với giá trị bias của neuron thứ h. neth cũng chính là đầu vào của hàm kích hoạt f1

〖net〗\_h=∑\_(i=1)^2▒w\_ih x\_i+b\_h

z\_h=f\_1 (〖net〗\_h)

net là giá trị tổng các tích giữa giá trị đầu ra của lớp ẩn với trọng số tương ứng của liên kết giữa mỗi neuron lớp ẩn với neuron đầu ra. z là giá trị đầu ra mạng neuron.

net=∑\_(i=1)^3▒〖z\_i w\_i 〗

z=f\_2 (net)

## 1.1.2. Thuật toán Gradient Descent

Khi giải quyết một vấn đề sử dụng mạng neuron nhân tạo ANN, ta phải giải bài toán tìm các giá trị trọng số cho các liên kết trong mạng sao cho giá trị của một hàm chi phí (cost function) là nhỏ nhất. Hàm chi phí này do ta chọn tùy theo vấn đề cụ thể. Hàm chi phí còn có tên gọi khác là hàm tổn thất (loss function).

Việc tối ưu hàm chi phí trong mạng ANN được thực hiện thông qua thuật toán tối ưu Gradient Descent.

Các bước thực hiện của thuật toán này như sau:

Bước 1: Bắt đầu với một tập các giá trị ngẫu nhiên của các trọng số (weights) của các liên kết trong mạng ANN. Các giá trị ngẫu nhiên này được biểu diễn dưới dạng một vector θ. Giá trị của hàm chi phí phụ thuộc vào các trọng số này ký hiệu là C(θt)

Bước 2: Tại mỗi vòng lặp, ví dụ tại vòng lặp thứ t, giá trị vector các trọng số của mô hình là θt , giá trị của đạo hàm của hàm chi phí theo các trọng số là ΔC(θt). Khi đó giá trị của vector các trọng số tại vòng lặp thứ t+1 tính theo công thức sau:

θt+1=θt – ηΔC(θt)

trong đó η được gọi là tốc độ học (learning rate). Giá trị η đóng một vai trò quan trọng trong quá trình đào tạo. Nếu η lớn thì việc đào tạo có thể nhanh hơn nhưng có thể dẫn đến sự giao động quanh điểm tối ưu của mô hình. Một số thuật toán coi giá trị tốc độ học η là một hằng số. Tuy nhiên cũng có một số thuật toán, ví dụ như thuật toán Adam Gradient Descent coi tốc độ học cũng là một tham số được tự động thay đổi phụ thuộc vào giá trị đạo hàm của vòng lặp trước đó.

Bước 3: Theo lý thuyết, khi hàm chi phí đạt giá trị nhỏ nhất thì đạo hàm của nó ΔC(θt) có giá trị bằng 0. Khi đó quá trình cập nhật các trọng số trong mạng kết thúc và θt+1=θt . Đến lúc này ta sẽ có được các giá trị trọng số tối ưu của mô hình. Tuy nhiên trong thực tế do lỗi làm tròn và các giới hạn của máy tính, có thể ta sẽ không bao giờ đạt tới được điểm làm cho đạo hàm hàm chi phí bằng 0. Do đó ta cần phải đưa ra cách để dừng quá trình lặp khi ta tin rằng đã tìm được điểm cực tiểu hoặc ít nhất là đạt đến điểm đủ gần với giá trị cực tiểu. Việc này cũng có thể được thực hiện bằng cách đưa ra giới hạn về số vòng lặp để dừng quá trình đào tạo.

Quá trình cập nhật các trọng số của mạng ANN sử dụng thuật toán gradient descent còn được gọi là quá trình lan truyền ngược (backpropagation).

Khi thực hiện xây dựng mạng ANN ta cần lưu ý đến những vấn đề sau :

Giá trị ban đầu của các trọng số

Giá trị khởi tạo của các trọng số của các liên kết trong mạng ANN rất quan trọng, có thể ảnh hưởng đến tốc độ hội tụ của mạng tức là ảnh hưởng đến thời gian đào tạo mạng ANN.

Tốc độ học (learning rate)

Việc lựa chọn giá trị tốc độ học cũng làm ảnh hưởng đến kết quả cuối cùng. Giá trị tốc độ học quá lớn có thể làm kết quả kém chính xác, giá trị quá nhỏ sẽ làm kéo dài thời gian đào tạo.

Chuẩn hóa dữ liệu đầu vào

Trong nhiều trường hợp dữ liệu đầu vào có thể có các đơn vị khác nhau. Ví dụ trong nghiên cứu này, cường độ bức xạ có đơn vị là (W/m2), còn nhiệt độ có đơn vị là 0C. Do đó ta cần chuẩn hóa các giá trị đầu vào để cho các giá trị đầu vào nằm trong đoạn [0,1]. Có nhiều cách chuẩn hóa dữ liệu đầu vào nhưng trong phạm vi của nghiên cứu này nhóm tác giả chuẩn hóa dữ liệu bằng công thức sau:

x^'=(x-min⁡(x))/(max⁡(x)-min⁡(x))

Trong đó x là giá trị ban đầu, x^' là giá trị đã được chuẩn hóa. x^' nằm trong khoảng đoạn [0,1]. max(x) là giá trị lớn nhất của các giá trị trong tập dữ liệu, min(x) là giá trị nhỏ nhất của các giá trị trong tập dữ liệu.

Việc này cũng có thể thực hiện bằng cách sử dụng MinMaxScaler trong thư viện sklearn của python.

Số lần thực hiện đưa đầy đủ dữ liệu đào tạo vào mô hình (epoch)

Một Epoch được tính là khi chúng ta đưa xong tất cả dữ liệu vào mạng neuron 1 lần. Thuật toán Gradient Descent đòi hỏi chúng ta phải đem toàn bộ dữ liệu qua mạng một vài lần để tìm được kết quả tối ưu. Vì vậy, dùng 1 epoch thật sự không đủ để tìm được kết quả tốt nhất. Giá trị epoch này cũng ảnh hưởng rất lớn đến độ chính xác của kết quả đầu ra. Thông thường ta phải thử nhiều giá trị epoch khác nhau để tìm được giá trị phù hợp.

Số lượng mẫu dữ liệu trong cùng một mẻ dữ liệu (batch size)

Trong nhiều trường hợp, dữ liệu đầu vào quá lớn nên ta không thể truyền toàn bộ dữ liệu vào mô hình một lần được. Khi đó ta phải chia dữ liệu thành nhiều mẻ (batch). Số lượng mẫu dữ liệu trong một mẻ được gọi là kích thước mẻ (batch size). Như vậy một epoch có thể gồm nhiều batch.

# 2. Xây dựng chương trình dự báo giá máy tính

Nhóm tác giả đưa ra giả định rằng giá máy tính sẽ phụ thuộc vào hai tham số là dung lượng ổ cứng (GB) và dung lượng bộ nhớ RAM (GB).

Nhóm tác giả xây dựng chương trình phần mềm dự báo dựa vào thư viện Keras chạy trên nền tensorflow của ngôn ngữ lập trình Python. Đây là thư viện phổ biến được dùng để xây dựng các mạng neuron nhân tạo.

Các bước để xây dựng chương trình như sau:

Bước 1: Thu thập dữ liệu giá máy tính laptop từ trang web thương mại điện tử aliexpress. Dữ liệu gồm ba giá trị là dung lượng ổ cứng (GB), dung lượng bộ nhớ RAM (GB), giá máy tính (USD).

Bước 2: Chia dữ liệu thành hai tập, dữ liệu đào tạo (training data) chiếm 80% và dữ liệu kiểm tra (test data) chiếm 20%.

Bước 3: Chuẩn hóa dữ liệu đào tạo đầu vào để dữ liệu nằm trong đoạn [0,1]. Sử dụng MinMaxScaler của thư viện sklearn của ngôn ngữ lập trình Python.

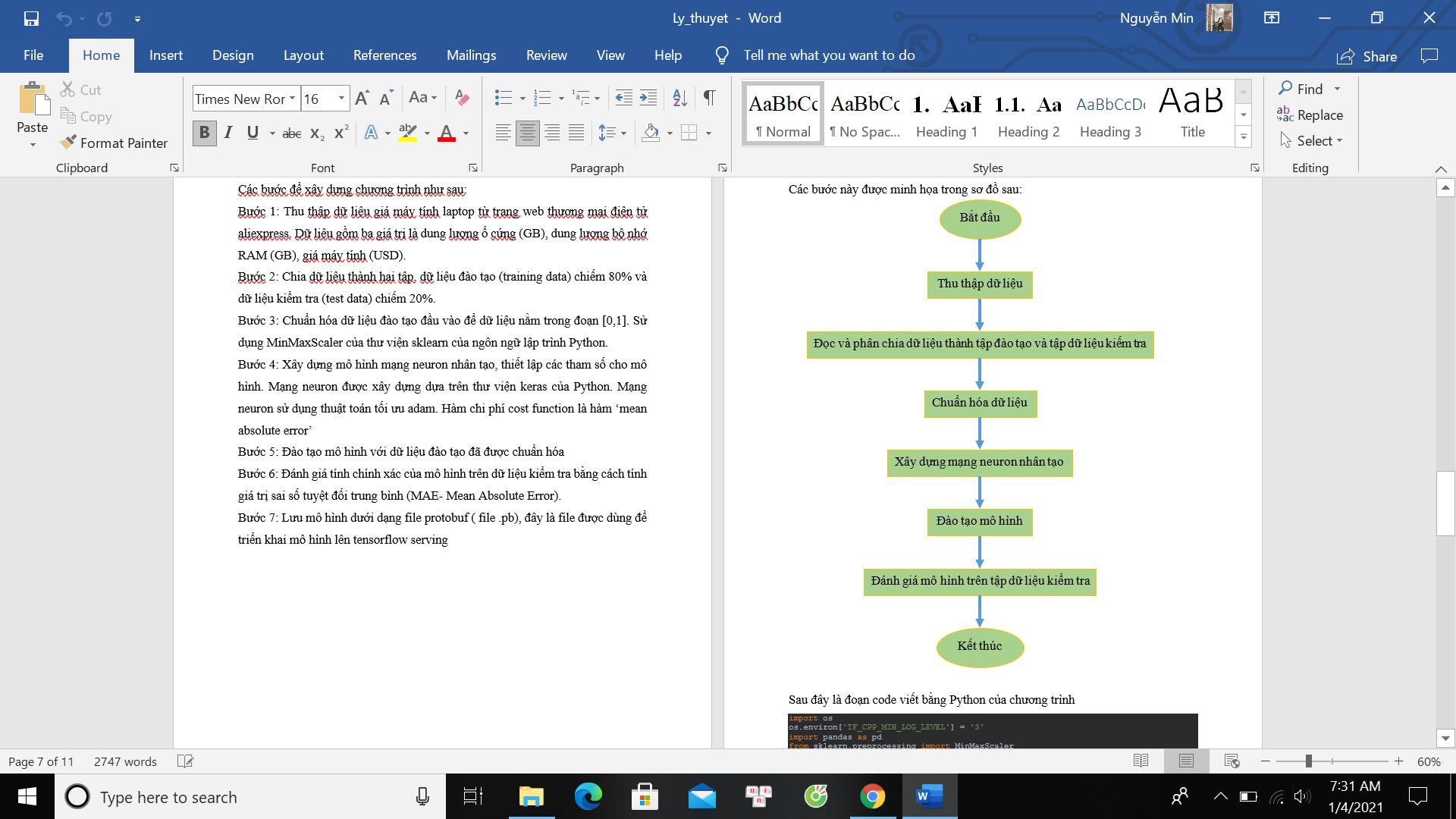
Bước 4: Xây dựng mô hình mạng neuron nhân tạo, thiết lập các tham số cho mô hình. Mạng neuron được xây dựng dựa trên thư viện keras của Python. Mạng neuron sử dụng thuật toán tối ưu adam. Hàm chi phí cost function là hàm ‘mean absolute error’

Bước 5: Đào tạo mô hình với dữ liệu đào tạo đã được chuẩn hóa

Bước 6: Đánh giá tính chính xác của mô hình trên dữ liệu kiểm tra bằng cách tính giá trị sai số tuyệt đối trung bình (MAE- Mean Absolute Error).

Bước 7: Lưu mô hình dưới dạng file protobuf ( file .pb), đây là file được dùng để triển khai mô hình lên tensorflow serving

Các bước này được minh họa trong sơ đồ sau:



Sau đây là đoạn code viết bằng Python của chương trình:

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '3'

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

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

from sklearn.metrics import mean\_absolute\_error

from keras.models import Sequential

from keras.layers import Dense

import tensorflow as tf

import tempfile

df=pd.read\_csv('laptop.csv')

labels=df['PriceUSD']

features=df.drop('PriceUSD',axis=1)

#Chuyển các trường chữ thành số

features=pd.get\_dummies(features,drop\_first=True)

#Phân tách dữ liệu thành dữ liệu đào tạo và dữ liệu kiểm tra

X\_train,X\_test,y\_train,y\_test=train\_test\_split(features,labels,train\_size=0.8,test\_size=0.2,random\_state=0)

#Chuẩn hóa dữ liệu bằng MinMaxScaler

scaler=MinMaxScaler()

scaler.fit(X\_train)

#Xây dựng mạng neuron

num\_hidden\_layers=int(input('Nhap so luong hidden layer:'))

num\_neuron\_first\_hidden\_layer=int(input('Nhap so luong neuron lop hidden thu nhat:'))

model = Sequential([Dense(num\_neuron\_first\_hidden\_layer, activation='relu', input\_shape=(features.shape[1],))])

i=2

num\_neuron\_previous\_layer=num\_neuron\_first\_hidden\_layer

while (i<=num\_hidden\_layers):

num\_neuron=int(input('Nhap so luong neuron lop hidden thu '+str(i) + ':'))

model.add(Dense(num\_neuron,input\_dim = num\_neuron\_previous\_layer,activation="relu"))

num\_neuron\_previous\_layer=num\_neuron

i=i+1

model.add(Dense(1,activation = "linear"))

num\_epoch=int(input('Nhap so luong vong lap training:'))

#model.compile(optimizer='adam',loss='mean\_squared\_error')

model.compile(optimizer='adam',loss='mean\_absolute\_error')

hist=model.fit(X\_train,y\_train,epochs=num\_epoch)

#Lưu lại các tham số của mô hình

tmpdir = tempfile.mkdtemp()

save\_path = os.path.join(os.getcwd(),'laptop\_model/1/')

print('Mo hinh da duoc luu vao thu muc ' + save\_path + ' , su dung de dua len tensorflow serving')

tf.saved\_model.save(model,save\_path)

pred=model.predict(X\_test)

print('10 gia tri du doan dau tien')

print(pred[:10].tolist())

print('10 gia tri thuc te dau tien')

print(list(y\_test[:10]))

print('Gia tri MAE tren tap du lieu test la: ' + str(mean\_absolute\_error(y\_test,pred)))

# 3. Đánh giá kết quả

## **3.1. Phương pháp đánh giá**

Để đánh giá mô hình nhóm nghiên cứu sử dụng mô hình để tính toán giá trị công suất dự báo đầu ra của mạng neuron trên tập dữ liệu kiểm tra và so sánh với giá trị thực tế dựa trên giá trị sai số bình phương trung bình (MAE-mean absolute error). Công thức tính giá trị này như sau:

MAE=1/N ∑\_(i=1)^N▒〖|P\_i^true 〗-P\_i^predict |

Trong đó:

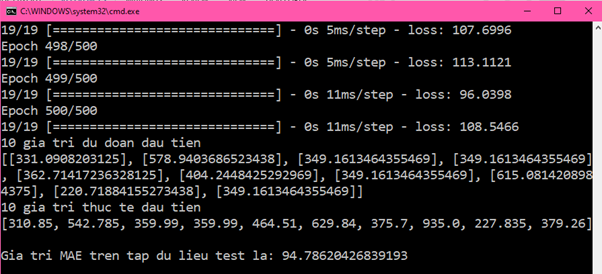
P\_i^true là giá trị thực tế thứ i của giá máy tính

P\_i^predict là giá trị dự đoán thứ i của giá máy tính

N là tổng số bản ghi của dữ liệu kiểm tra

## 3.2. Kết quả

Kết quả chạy mô hình như sau:



Như vậy sau khi chạy mô hình với dữ liệu đào tạo (training data) thì ta có thể dùng mạng Neuron để dự đoán giá máy tính trên tập dữ liệu test.

Khi đánh giá mô hình trên dữ liệu kiểm tra (test data) thì giá trị sai số tuyệt đối trung bình MAE=94.78

Triển khai mô hình lên tensorflow serving

Tensorflow Serving là bộ công cụ mã nguồn mở, dùng để triển khai (deploy) các mô hình được huấn luyện bởi tensorflow lên môi trường production.

Với Tensorflow Serving, quá trình triển khai mô hình lên hệ thống (server) sẽ trở nên nhanh chóng và dễ dàng hơn rất nhiều với việc triển khai và load model 1 cách thông thường. Tensorflow Serving triển khai mô hình 1 cách độc lậpdễ dàng thêm và cập nhật model mới theo từng phiên bản.

Ở phần trước sau khi chạy mô hình ta sẽ có được thư mục chứa file saved\_model.pb và thư mục variables. Trong đó:

saved\_model.pb: serialized model, lưu giữ toàn bộ thông tin graph của mô hình cũng như các meta-data khác như signature, inputs, outputs của model

variables: lưu giữ các serialized variables của graph (learned weights)

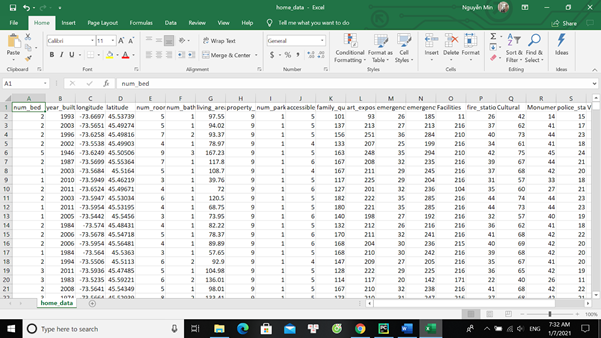
Đến đây ta có thể dùng docker để triển khai mô hình bằng tensorflow serving.

Chương II : Bài toán học máy

# Nguyễn Thùy Miên – 1781310153 – Dự đoán giá nhà Boston bằng bài toán hồi quy tuyến tính

## **1.1. Mô tả dữ liệu**

-Bộ dữ liệu boston.csv gôm có 22 thuộc tính



Hình 1:Mô tả dữ liệu

-Mô tả thuộc tính:

|  |  |  |
| --- | --- | --- |
| STT | Thuộc tính | Mô tả |
| 1 | num\_bed | Số phòng ngủ |
| 2 | year\_built | Năm xây dựng |
| 3 | longitude | kinh độ |
| 4 | latitude | vĩ độ |
| 5 | num\_rum |  |
| 6 | num\_bath | số phòng tắm |
| 7 | living\_area | khu vực sống |
| 8 | property | bất động sản |
| 9 | num\_park | công viên |
| 10 | accessible | ôn hòa |
| 11 | family\_quality | chất lượng gia đình |
| 12 | art\_expos | triển lãm nghệ thuật |
| 13 | emergency\_shelters | nơi trú ẩn khẩn cấp |
| 14 | emergency\_water | nước khẩn cấp |
| 15 | facilities | cơ sở vật chất |
| 16 | fire\_stations | trạm cứu hỏa |
| 17 | cultural | văn hóa |
| 18 | monuments | tượngđài |
| 19 | police\_stations | đồn cảnh sát |
| 20 | vacant | chỗ trống |
| 21 | free\_parking | bãi đỗ xe |
| 22 | askprice | giá |

- Xác định bài toán học máy.

- Với bộ dữ liệu này, ta sử dụng bài toán hồi quy tuyến tính.

+ Đầu vào X gồm 21 thuộc tính(cột) đầu.

+ Đầu ra Y là thuộc tính (cột) cuối cùng.

- Sử dụng kĩ thuật LinearRegression.

## 1.**2.Tiền xử lý dữ liệu**

- Đang nhập thư viện máy học sklearn , numpy và gấu trúc .

import pandas as pd

import numpy as np

from sklearn import linear\_model

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

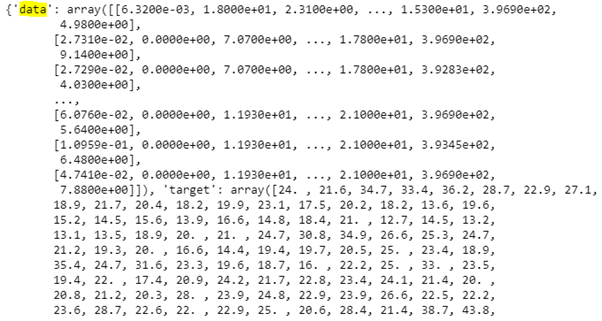
#Load the Boston Housing Data Set from sklearn.datasets and print it

from sklearn.datasets import load\_boston

boston = load\_boston()

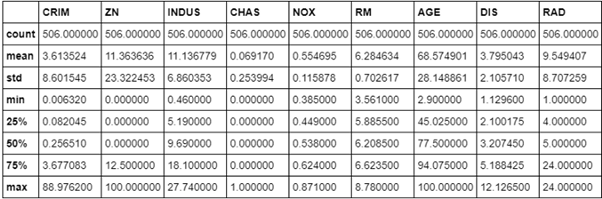
print(boston)

Kết quả:



hình 2:chuyển đổi tập dữ liệu thành khung dữ liệu

Chia dữ liệu thành 67% dữ liệu đào tạo và 33% dữ liệu thử nghiệm



Khởi tạo mô hình hồi quy tuyến tính

#Initialize the linear regression model

reg = linear\_model.LinearRegression()

#Split the data into 67% training and 33% testing data

#NOTE: We have to split the dependent variables (x) and the target or independent variable (y)

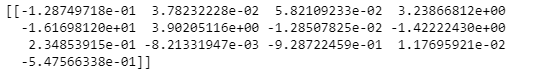
x\_train, x\_test, y\_train, y\_test = train\_test\_split(df\_x, df\_y, test\_size=0.33, random\_state=42)

#Train our model with the training data

reg.fit(x\_train, y\_train)

#Print the coefecients/weights for each feature/column of our model

print(reg.coef\_)

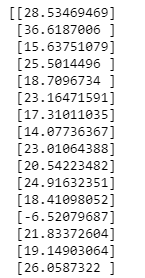


In dự đoán giá trên dữ liệu thử nghiệm

#print our price predictions on our test data

y\_pred = reg.predict(x\_test)

print(y\_pred)



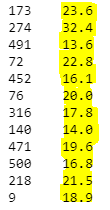
In dự đoán cho hàng thứ ba của dữ liệu thực nghiệm giá thực tế= 13.6

#Print the the prediction for the third row of our test data actual price = 13.6

y\_pred[2]

#print the actual price of houses from the testing data set

y\_test[0]



Lỗi bình phương trung bình (MSE)

# Two different ways to check model performance/accuracy using,

# mean squared error which tells you how close a regression line is to a set of points.

# 1. Mean squared error by numpy

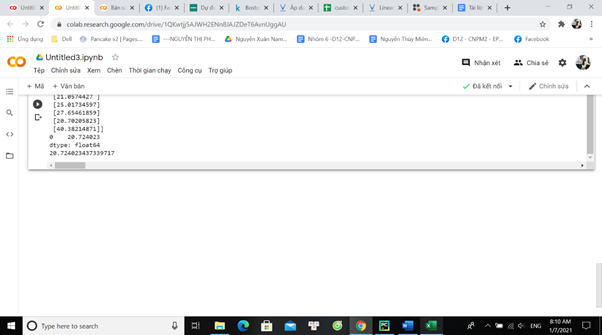
print(np.mean((y\_pred-y\_test)\*\*2))

# 2. Mean squared error by sklearn

# Resource: https://stackoverflow.com/questions/42453875/precision-score-and-accuracy-score-showing-value-error?rq=1

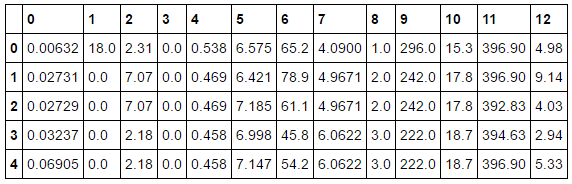
from sklearn.metrics import mean\_squared\_error

print(mean\_squared\_error(y\_test, y\_pred))



-kích thước của tập dữ liệu với kích thước column.

print(boston.data.shape)



-kích thước của tập dữ liệu với kích thước column

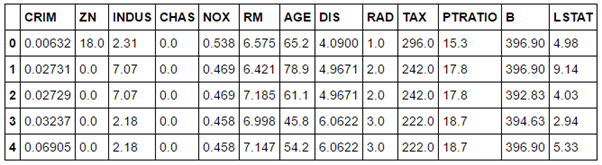
print(boston.data.shape)



-show được tên cột

bos.columns = boston.feature\_names

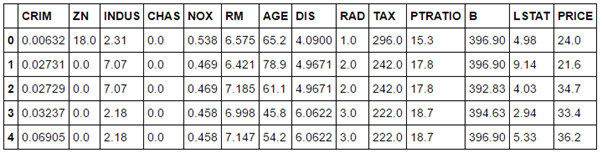
print(bos.head())



-column tên là PRICE

bos['PRICE'] = boston.target

print(bos.head())



-chạy hồi quy tuyến tính

from sklearn.linear\_model import LinearRegression

lm = LinearRegression()

lm.fit(X\_train, Y\_train)

Y\_pred = lm.predict(X\_test)

plt.scatter(Y\_test, Y\_pred)

plt.xlabel("Prices: $Y\_i$")

plt.ylabel("Predicted prices: $\hat{Y}\_i$")

plt.title("Prices vs Predicted prices: $Y\_i$ vs $\hat{Y}\_i$")



## 1.**3.Kết luận**

Linear regression chạy khá nhanh và chuẩn xác, tuy nhiên nhược điểm của nó cũng như các thuật toán hồi quy khác Đơn giản để thực hiện,Được sử dụng để dự đoán các giá trị số,Dễ bị overfitting,Không thể sử dụng khi mối quan hệ giữa các biến độc lập và phụ thuộc là phi tuyến.Chính vì thế, để đảm bảo dữ liệu được huấn luyện tốt và cho ra dự đoán chính xác, cần tiến hành xử lý mất cân bằng ngay từ khâu ban đầu trước khi chạy mô hình dự đoán kết quả

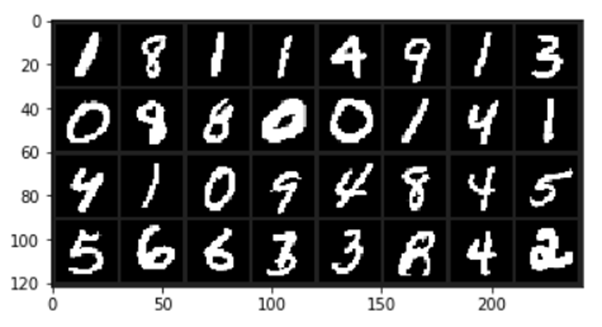
# Vũ Thị Thanh Bình – 1781310112 – Dự đoán chữ số viết tay bằng bài toán phân lớp

## **2.1. Mô tả tập dữ liệu**

MNIST được giới thiệu năm 1998 bởi Yann Lecun và cộng sự nhằm đánh giá các mô hình phân lớp.

MNIST là tập dữ liệu về chữ số viết tay từ 0 đến 9, bao gồm 2 tập con: training set gồm 60.000 ảnh các chữ số viết tay và test set gồm 10.000 ảnh các chữ số. Trong đó, mỗi hình là một ảnh đen trắng chứa một số được viết tay có kích thước là 28x28.

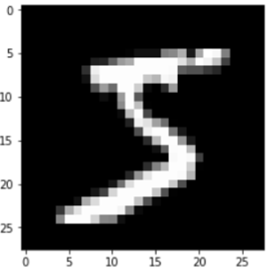
Website chính thức của tập dữ liệu: http://yann.lecun.com/exdb/mnist/



Tập dữ liệu MNIST chữ số viết tay từ 0 đến 9

## **2.2. Mô tả bài toán**

Ban đầu bài toán có ảnh xám kích thước 28\*28 của chữ số từ 0 đến 9 và muốn dự đoán số đấy là số mấy. Ví dụ:



Dữ liệu đầu tiên trong MNIST dataset

Bài toán phân lớp được áp dụng để dự đoán chữ số viết tay qua ảnh.

Kỹ thuật học máy được áp dụng: Neural Network (Convolutional Neural Network)

## **2.3. Áp dụng bài toán học máy**

# 1. Thêm các thư viện cần thiết

import numpy as np

import matplotlib.pyplot as plt

from keras.models import Sequential

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Conv2D, MaxPooling2D

from keras.utils import np\_utils

from keras.datasets import mnist

## **2.3.1. Mô tả về xử lý dữ liệu (Training set, Validation set, Test set)**

Trong bài toán trên:

50.000 dữ liệu trong tổng số 60.000 dữ liệu được gọi là training set, có thể hiểu là dữ liệu dùng để dạy cho model học.

10.000 dữ liệu còn lại gọi là validation set, là để đánh giá xem model hiện tại có tốt không, thường được dùng để chỉnh các tham số của model.

10.000 dữ liệu test set có sẵn là để đánh giá xem model hoạt động với dữ liệu thực tế có tốt không.

Như vậy MNIST dataset có 60.000 dữ liệu ở training set ở trong MNIST, ta sẽ chia ra 50.000 dữ liệu cho training set và 10.000 dữ liệu cho validation set. Vẫn giữ nguyên 10.000 dữ liệu của test set.

# 2. Load dữ liệu MNIST

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

X\_val, y\_val = X\_train[50000:60000,:], y\_train[50000:60000]

X\_train, y\_train = X\_train[:50000,:], y\_train[:50000]

print(X\_train.shape)

#show 25 dữ liệu

plt.figure(figsize=(10, 10))

for i in range(25):

plt.subplot(5, 5, i+1)

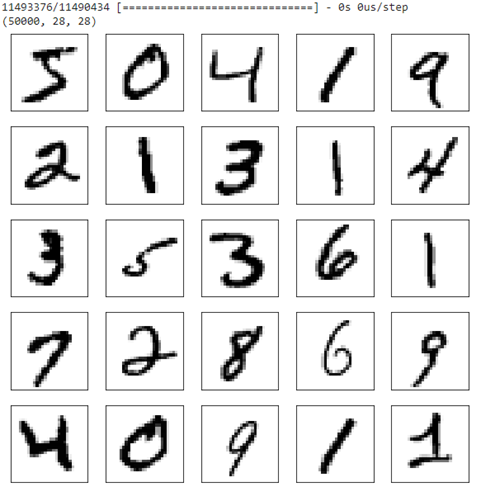
plt.xticks([])

plt.yticks([])

plt.imshow(X\_train[i], cmap='binary')

plt.show()

OUT:



Dữ liệu input cho mô hình Convolutional Neural Network là 1 tensor 4 chiều (N, W, H, D), trong bài này là ảnh xám nên W = H = 28, D = 1, N là số lượng ảnh cho mỗi lần training.

Do dữ liệu ảnh ở trên có kích thước là (N, 28, 28) tức là (N, W, H) nên cần reshape lại thành kích thước N 28 28 \* 1 để giống kích thước mà Keras yêu cầu.

# 3. Reshape lại dữ liệu cho đúng kích thước mà keras yêu cầu

X\_train = X\_train.reshape(X\_train.shape[0], 28, 28, 1)

X\_val = X\_val.reshape(X\_val.shape[0], 28, 28, 1)

X\_test = X\_test.reshape(X\_test.shape[0], 28, 28, 1)

Bước này chuyển đổi one-hot encoding label Y của ảnh

Ví dụ số 5 thành vector [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]

# 4. One hot encoding label (Y)

Y\_train = np\_utils.to\_categorical(y\_train, 10)

Y\_val = np\_utils.to\_categorical(y\_val, 10)

Y\_test = np\_utils.to\_categorical(y\_test, 10)

print('Dữ liệu y ban đầu ', y\_train[0])

print('Dữ liệu y sau one-hot encoding ',Y\_train[0])

OUT:



## **2.3.2. Mô tả về kỹ thuật học máy**

Xây dựng model:

Vì input của model là ảnh nên sử dụng convolutional neural network (CNN).

Mô hình chung bài toán CNN:

Input image -> Convolutional layer (Conv) + Pooling layer (Pool) -> Fully connected layer (FC) -> Output.

Thay vì chỉ muốn kết quả là ảnh là số mấy, ta muốn dự đoán phần trăm của ảnh là số nào. Ví dụ: 90% ảnh là số 5, 1% ảnh là số 1,…

# 5. Định nghĩa model

model = Sequential()

# Thêm Convolutional layer với 32 kernel, kích thước kernel 3\*3

# dùng hàm sigmoid làm activation và chỉ rõ input\_shape cho layer đầu tiên

model.add(Conv2D(32, (3, 3), activation='sigmoid', input\_shape=(28,28,1)))

# Thêm Convolutional layer

model.add(Conv2D(32, (3, 3), activation='sigmoid'))

# Thêm Max pooling layer

model.add(MaxPooling2D(pool\_size=(2,2)))

# Flatten layer chuyển từ tensor sang vector

model.add(Flatten())

# Thêm Fully Connected layer với 128 nodes và dùng hàm sigmoid

model.add(Dense(128, activation='sigmoid'))

# Output layer với 10 node và dùng softmax function để chuyển sang xác suất.

model.add(Dense(10, activation='softmax'))

# 6. Compile model, chỉ rõ hàm loss\_function nào được sử dụng, phương thức

# dùng để tối ưu hàm loss function.

model.compile(loss='categorical\_crossentropy',

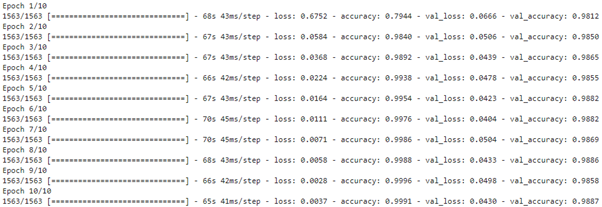
optimizer='adam',

metrics=['accuracy'])

# 7. Thực hiện train model với data

H = model.fit(X\_train, Y\_train, validation\_data=(X\_val, Y\_val), batch\_size=32, epochs=10, verbose=1)

OUT:



## **2.3.3. Mô tả phương pháp đánh giá**

# 8. Đánh giá model với dữ liệu test set

score = model.evaluate(X\_test, Y\_test, verbose=0)

print(score)

**OUT: [0.03442202161136956, 0.9892]**

Ta sẽ dùng kết quả đánh giá của mode với Test set để làm kết quả cuối cùng của model. Tức model của chúng ta dự đoán chữ số có độ chính xác 98.92% với MNIST dataset cũng có nghĩa là dự đoán khoảng 100 ảnh thì sai 1 ảnh.

Thử dự đoán 1 ảnh từ tập dữ liệu MNIST

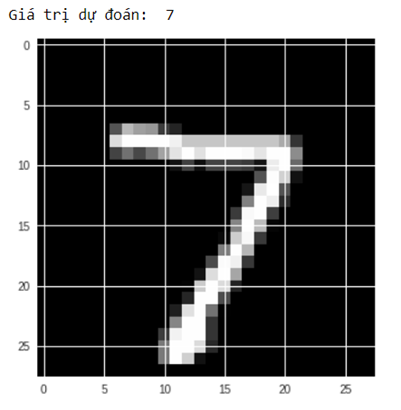
# 9. Dự đoán ảnh

plt.imshow(X\_test[0].reshape(28,28), cmap='gray')

y\_predict = model.predict(X\_test[0].reshape(1,28,28,1))

print('Giá trị dự đoán: ', np.argmax(y\_predict))

**OUT:**

****

Vậy dự đoán đúng giá trị X\_test[0] là chữ số 7.

## **2.4. Kết luận**

Chúng ta đã hoàn thành bài toán phân lớp từ load data, tạo model và tiến hành huấn luyện mô hình CNN cho tập dữ liệu MNIST với độ chính xác là 98.92 % rất cao . Chính vì thế mà ngày nay việc áp dụng kỹ thuật này đang trở nên vô cùng phổ biến cũng như tầm quan trọng của nó

Chuẩn đoán ảnh X-ray của bệnh nhân có bị ung thư hay không

Phân loại, nhận diện được các chữ, số viết tay => tự động đọc được biển số xe, văn bản.

Phân loại được các biển báo giao thông => hỗ trợ cho ô tô tự lái

…

**C. Dự đoán hoa dựa vào hồi quy tuyến tính (Nguyễn Hữu Đạt – 1781310119)**

**3.1.** **Mô tả tập dữ liệu**

Bộ dữ liệu về hoa Iris được thu thập bởi Edgar Anderson – 1 nhà thực vật học người mỹ - trên bán đảo Gaspe (là một bán đảo dọc theo bờ biển phía nam của sông Saint Lawrence ở Quebec , Canada , mở rộng vào vịnh Saint Lawrence) Sau đó bộ dữ liệu trên được Ronald Aylmer Fisher – 1 nhà thống kê người Anh đồng thời là 1 nhà di truyền học rút gọn lại, bỏ đi các thuộc tính mang tính chất mô tả,không đo đạt được giá trị theo số học (như thời điểm gieo trồng, thời điểm nở hoa, màu sắc hoa, tốc độ tăng trưởng, điều kiện ánh sáng, đất nước và độ ẩm,…) Bộ dữ liệu sau khi được rút gọn bao gồm 5 thuộc tính : 4 thuộc tính kiểu số: chiều dài đài hoa, chiều rộng đài hoa, chiều dài cánh hoa, chiều rộng cánh hoa (đơn vị là cm) 1 thuộc tính còn lại là tên của loài hoa Iris (có 3 loài tất cả : Iris Setosa, Iris Versicolour, Iris Virginica)



Tỉ lệ phân chia loài hoa

**3.2 Làm sạch dữ liệu:**

Thiếu giá trị: khi xảy ra sự thiếu thông tin ở 1 thuộc tính nào đó trong 1 bản ghi của bộ dữ liệu thu thập, khi mà tính đảm bảo số bản ghi chia đều cho 3 loài trên đã có ( 33,3%) và số lượng bản ghi thiếu là ít ta có thể áp dụng phương pháp loại bỏ.(thực tế trong bộ dữ liệu trên, không tồn tại giá trị thuộc tính bị thiếu hay không được điền).mặt khác khi cần phải điền vào các giá trị thiếu, ta có thể áp dụng phương pháp điền tự động, thay thế các giá trị thiếu bằng giá trị trung bình đã được thống kê ở trên Nhiễu dữ liệu: khi xuất hiện 1 giá trị bất ngờ nào đó, đột nhiên vượt quá các giá trị biên đã được thống kê, ta có thể sửa lại giá trị đó thành các giá trị ở vùng biên theo bảng thống kê ở trên

**3.3. Chọn lọc dữ liệu:**

Tích hợp và dư thừa dữ liệu: do dữ liệu trên thu thập từ 1 nguồn duy nhất nên việc tích hợp là không cần thiết, các thuộc tính của dữ liệu là độc lập nhau, không có mối quan hệ tương quan nào, các thuộc tính đã được rút gọn chọn lọc nên không cần việc phân tích dư thừa dữ liệu

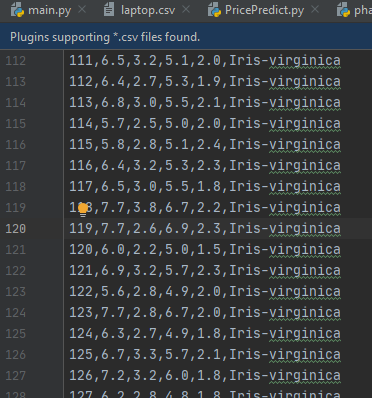
Biến đổi dữ liệu**:**

Biến đổi dữ liệu: ta sử dụng phương pháp chuẩn hóa dữ liệu về khoảng giá trị [0,1], phần này sẽ được nói rõ hơn ở phần sau tùy vào phương pháp khai phá dữ liệu được lựa chọn.

**3.4. Rút gọn dữ liệu:**

:Như đã nói từ đầu, bộ dữ liệu đã được loại bỏ đi các thuộc tính mang tính chất mô tả, chỉ giữ lại 1 thuộc tính mô tả duy nhất là tên loài Iris cụ thể.các thuộc tính mang tính số học còn lại vẫn đảm bảo được tính phân loại (đã được Ronald Aylmer Fisher – 1 nhà di truyền học người Anh – chọn lọc)

**3.5 Dữ liệu:**

****

Hình 3.5.1: Dữ liệu hoa

**3.6 Mô tả về kỹ thuật học máy:**

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import seaborn as sns # data visualisation and plotting

import matplotlib.pyplot as plt # data plotting

import warnings

# Seaborn default configuration

sns.set\_style("darkgrid")

# set the custom size for my graphs

sns.set(rc={'figure.figsize':(8.7,6.27)})

# filter all warnings

warnings.filterwarnings('ignore')

# set max column to 999 for displaying in pandas

pd.options.display.max\_columns=999

data = pd.read\_csv('Iris-1.csv')

data.head()

data.info()

data.describe()

data['Species'].value\_counts()

rows, col = data.shape

print("Rows : %s, column : %s" % (rows, col))

#Trực quan hóa dữ liệu

# snsdata = data.drop(['Id'], axis=1)

# g = sns.pairplot(snsdata, hue='Species', markers='x')

# g = g.map\_upper(plt.scatter)

# g = g.map\_lower(sns.kdeplot)

#

# sns.violinplot(x='SepalLengthCm', y='Species', data=data, inner='stick', palette='autumn')

# plt.show()

# sns.violinplot(x='SepalWidthCm', y='Species', data=data, inner='stick', palette='autumn')

# plt.show()

# sns.violinplot(x='PetalLengthCm', y='Species', data=data, inner='stick', palette='autumn')

# plt.show()

# sns.violinplot(x='PetalWidthCm', y='Species', data=data, inner='stick', palette='autumn')

# plt.show()

#Multivariate Linear Regression Model(Mô hình hồi quy tuyến tính đa biến)

mapping = {

'Iris-setosa' : 1,

'Iris-versicolor' : 2,

'Iris-virginica' : 3

}

X = data.drop(['Id', 'Species'], axis=1).values # Input Feature Values

y = data.Species.replace(mapping).values.reshape(rows,1) # Output values

X = np.hstack(((np.ones((rows,1))), X))# Adding one more column for bias

np.random.seed(0) # Let's set the zero for time being

theta = np.random.randn(1,5) # Setting values of theta randomly

print("Theta : %s" % (theta))

iteration = 10000

learning\_rate = 0.003 # If you are going by formula, this is actually alpha.

J = np.zeros(iteration) # 1 x 10000 maxtix

# Let's train our model to compute values of theta

for i in range(iteration):

J[i] = (1/(2 \* rows) \* np.sum((np.dot(X, theta.T) - y) \*\* 2 ))

theta -= ((learning\_rate/rows) \* np.dot((np.dot(X, theta.T) - y).reshape(1,rows), X))

prediction = np.round(np.dot(X, theta.T))

ax = plt.subplot(111)

ax.plot(np.arange(iteration), J)

ax.set\_ylim([0,0.15])

plt.ylabel("Cost Values", color="Green")

plt.xlabel("No. of Iterations", color="Green")

plt.title("Mean Squared Error vs Iterations")

plt.show()

ax = sns.lineplot(x=np.arange(iteration), y=J)

plt.show()

ax = plt.subplot(111)

ax.plot(np.arange(1, 151, 1), y, label='Orignal value', color='red')

ax.scatter(np.arange(1, 151, 1), prediction, label='Predicted Value')

plt.xlabel("Dataset size", color="Green")

plt.ylabel("Iris Flower (1-3)", color="Green")

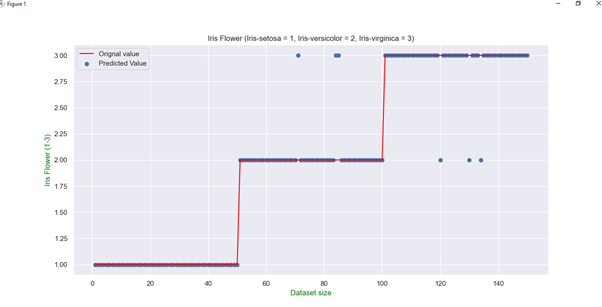
plt.title("Iris Flower (Iris-setosa = 1, Iris-versicolor = 2, Iris-virginica = 3)")

ax.legend()

plt.show()

accuracy = (sum(prediction == y)/float(len(y)) \* 100)[0]

print("The model predicted values of Iris dataset with an overall accuracy of %s" % (accuracy))



Hình 3.6.1 : Kết quả

**3.7 Kết luận:**

Chúng ta đã hoàn thành bài toán dự đoán hoa dựa vào các thuộc tính độ dài, rộng của đài hoa, cánh hoa. Qua đó ta cũng nhận thấy được tầm quan trọng cũng như các ứng dụng vô cùng cần thiết của hồi quy tuyến tính trong việc xử lí các bài toán liên quan tới dự đoán.