TRƯỜNG ĐẠI HỌC ĐIỆN LỰC

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CHUYÊN ĐỀ MÔN HỌC MÁY NÂNG CAO**

**ĐỀ TÀI: TÌM HIỂU KIẾN TRÚC MẠNG RESNET, ỨNG DỤNG VÀ GIẢI CÁC BÀI TOÁN HỒI QUY, PHÂN LỚP**

|  |  |
| --- | --- |
|  |  |
| |  |  | | --- | --- | | **Sinh viên thực hiện** | **: NGUYỄN ĐỨC PHÚ**  **: PHẠM THẾ DUYỆT**  **: NGUYỄN TRÍ TUỆ** | | **Giảng viên hướng dẫn** | **: PHẠM THỊ KIM DUNG** | | | **Ngành** | **: CÔNG NGHỆ THÔNG TIN** | | | **Chuyên ngành** | **: CÔNG NGHỆ PHẦN MỀM** | | | **Lớp** | **: D12CNPM2** | | | **Khóa** | **: 2017-2022** | |   ***Hà Nội, tháng 01 năm 2021***  **PHIẾU CHẤM ĐIỂM**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **STT** | **Họ và tên sinh viên** | **Nội dung thực hiện** | **Điểm** | **Chữ ký** | | 1 | Nguyễn Đức Phú | Tìm tài liệu học tập  Viết báo cáo |  |  | | 2 | Phạm Thế Duyệt | Tìm tài liệu học tập |  |  | | 3 | Nguyễn Trí Tuệ | Tìm tài liệu học tập |  |  |  |  |  |  | | --- | --- | --- | | **Họ và tên giảng viên** | **Chữ ký** | **Ghi chú** | | Giảng viên chấm 1: |  |  | | Giảng viên chấm 2: |  |  | |  |

**MỤC LỤC**

[LỜI MỞ ĐẦU 1](#_Toc61440737)

[CHƯƠNG 1: TÌM HIỂU KIẾN TRÚC MẠNG RESNET VÀ ỨNG DỤNG 2](#_Toc61440738)

[1.1. Giới thiệu chung về mạng ResNet : 2](#_Toc61440739)

[1.2. Ý tưởng của ResNet : 3](#_Toc61440740)

[1.3. Tổng quan và cách cài đặt tensorflow : 4](#_Toc61440741)

[1.3.1. Giới thiệu: 4](#_Toc61440742)

[1.3.2. Cài đặt: 5](#_Toc61440743)

[1.4. Bài toán minh họa ResNet: 5](#_Toc61440744)

[1.4.1.Giới thiệu chung về ResNet-50 : 6](#_Toc61440745)

[1.4.2.Thực hiện bài toán : 7](#_Toc61440746)

[CHƯƠNG 2: BÀI TOÁN HỌC MÁY 13](#_Toc61440747)

[2.1. Bài toán hồi quy dự đoán giá xe ô tô Toyota Corolla cũ : 13](#_Toc61440748)

[2.1.1. Mô tả bài toán : 13](#_Toc61440749)

[2.1.2. Thu thập và mô tả dữ liệu 13](#_Toc61440750)

[2.1.3. Mô tả thuộc tính 14](#_Toc61440751)

[2.1.4. Số lượng mẫu 14](#_Toc61440752)

[2.1.5. Xây dựng mô hình thuật toán và giới thiệu chung Linear Regression: 15](#_Toc61440753)

[2.1.6. Huấn luyện mô hình 16](#_Toc61440754)

[2.1.7. Áp dụng kỹ thuật, thuật toán vào bài toán 17](#_Toc61440755)

[2.1.7. Đánh giá mô hình 23](#_Toc61440756)

[2.2. Bài toán phân lớp giá điện thoại : 24](#_Toc61440757)

[2.2.1. Mô tả bài toán : 24](#_Toc61440758)

[2.2.2. Thu thập và mô tả dữ liệu 24](#_Toc61440759)

[2.2.3. Mô tả thuộc tính 24](#_Toc61440760)

[2.2.4. Giới thiệu Logistic Regression và mô hình giải bài toán : 25](#_Toc61440761)

[2.2.5. Áp dụng kỹ thuật, thuật toán vào bài toán 38](#_Toc61440762)

[2.3. Bài toán phân lớp loại thú : 41](#_Toc61440763)

[2.3.1. Mô tả bài toán : 41](#_Toc61440764)

[2.3.2. Thu thập và mô tả dữ liệu 41](#_Toc61440765)

[2.3.3. Mô tả thuộc tính 41](#_Toc61440766)

[2.3.4. Giới thiệu về kỹ thuật cây quyết định và mô hình giải bài toán: 42](#_Toc61440767)

[2.3.5. Áp dụng kỹ thuật, thuật toán vào bài toán : 45](#_Toc61440768)

[CHƯƠNG 3 : KẾT LUẬN 49](#_Toc61440769)

[3.1. Bài toán hồi quy dự đoán giá xe ô tô Toyota Corolla cũ: 49](#_Toc61440770)

[3.2. Bài toán phân lớp giá điện thoại: 49](#_Toc61440771)

[3.3.Bài toán phân lớp thú: 49](#_Toc61440772)

[DANH MỤC TÀI LIỆU THAM KHẢO 50](#_Toc61440773)

# LỜI MỞ ĐẦU

Những năm gần đây, AI - Artificial Intelligence (Trí Tuệ Nhân Tạo), và cụ thể hơn là Machine Learning (Học Máy hoặc Máy Học) nổi lên như một bằng chứng của cuộc cách mạng công nghiệp lần thứ tư (1 - động cơ hơi nước, 2 - năng lượng điện, 3 - công nghệ thông tin). Trí Tuệ Nhân Tạo đang len lỏi vào mọi lĩnh vực trong đời sống mà có thể chúng ta không nhận ra. Xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trong ảnh của Facebook, trợ lý ảo Siri của Apple, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix, máy chơi cờ vây AlphaGo của Google DeepMind, …, chỉ là một vài trong vô vàn những ứng dụng của AI/Machine Learning.

Thị giác máy tính là một lĩnh vực trong Artificial Intelligence và Computer Science (Trí tuệ nhân tạo và Khoa học máy tính) nhằm giúp máy tính có được khả năng nhìn và hiểu giống như con người. Tạo ra một chiếc máy nhìn được như cách con người nhìn là không đơn giản, không chỉ vì khó tạo ra một chiếc máy như vậy mà ngay cả chúng ta cũng chưa thực sự hiểu cách thức hoạt động của quá trình nhìn.

Vậy nên để tìm hiểu xâu về thị giác máy tính, nhóm chúng em xin chọn đề tài :”**Tìm hiểu kiến trúc mạng ResNet và ứng dụng**” để làm đề tài báo cáo cuối kỳ cho năm học này. Vì thời gian ngắn chuẩn bị nên báo cáo của chúng em còn nhiều hạn chế. Chúng em mong thầy cô giáo góp ý để chúng em xây dựng báo cáo lần sau hoàn thiện hơn

**Chúng em xin trân thành cảm ơn !**

# CHƯƠNG 1: TÌM HIỂU KIẾN TRÚC MẠNG RESNET VÀ ỨNG DỤNG

## 1.1. Giới thiệu chung về mạng ResNet :

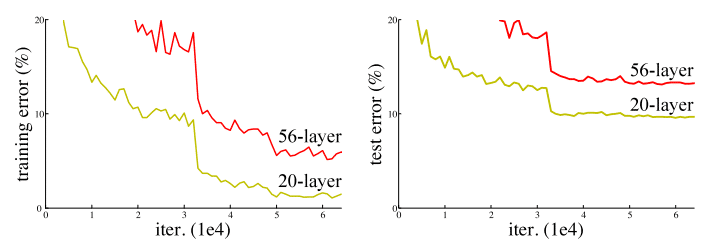
ResNet (viết tắt của residual network) là mạng học sâu nhận được quan tâm từ những năm 2012 sau cuộc thi [LSVRC2012](http://www.image-net.org/challenges/LSVRC/2012/) và trở nên phổ biến trong lĩnh vực thị giác máy. ResNet khiến cho việc huấn luyện hàng trăm thậm chí hàng nghìn lớp của mạng [nơ ron](https://trituenhantao.io/tu-dien-thuat-ngu/no-ron/) trở nên khả thi và hiệu quả.

Nhờ khả năng biểu diễn mạnh mẽ của ResNet, hiệu suất của nhiều ứng dụng thị giác máy, không chỉ các ứng dụng phân loại hình ảnh được tăng cường. Một số ví dụ có thể kể đến là các ứng dụng phát hiện đồ vật và nhận dạng khuôn mặt.

Theo định lý gần đúng phổ quát, về mặt kiến trúc, một mạng [nơ ron](https://trituenhantao.io/tu-dien-thuat-ngu/no-ron/) truyền thẳng có khả năng xấp xỉ mọi hàm với dữ liệu huấn luyện được cung cấp, miễn là không vượt quá sức chứa của nó. Tuy nhiên, xấp xỉ tốt dữ liệu không phải là mục tiêu duy nhất, chúng ta cần một mô hình có khả năng tổng quát hóa dữ liệu. Đó là lý do các kiến trúc sâu trở thành xu hướng của cộng đồng nghiên cứu.

Kể từ AlexNet, các kiến trúc CNN ngày càng sâu hơn. Trong khi AlexNet chỉ có 5 lớp tích chập, mạng VGG và GoogleNet (còn gọi là Inception\_v1) có đến 19 và 22 lớp tương ứng.

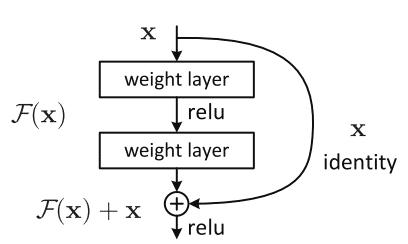
Tuy nhiên, tăng độ sâu mạng không chỉ đơn giản là xếp chồng các lớp lại với nhau. Mạng sâu rất khó huấn luyện vì vấn đề [vanishing gradient](https://trituenhantao.io/tu-dien-thuat-ngu/vanishing-gradient/) – vì độ dốc được truyền ngược trở lại các lớp trước đó, phép nhân lặp đi lặp lại có thể làm cho độ dốc cực nhỏ. Kết quả là, hiệu suất của mạng bị bão hòa hoặc giảm hiệu quả nhanh chóng.



Hình 1.1.Mối tương quan giữa độ sâu và hiệu suất mạng

## 1.2. Ý tưởng của ResNet :

Ý tưởng chính của ResNet là sử dụng kết nối tắt đồng nhất để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một residual block như trong hình sau:



Hình 1.2. ResNet sử dụng kết nối tắt xuyên qua một hay nhiều lớp

Việc xếp chồng các lớp sẽ không làm giảm hiệu suất mạng. Chúng ta có thể đơn giản xếp chồng các ánh xạ đồng nhất lên mạng hiện tại và hiệu quả của kiến trúc không thay đổi. Điều này giúp cho kiến trúc sâu ít nhất là không kém hơn các kiến trúc nông. Hơn nữa, với kiến trúc này, các lớp ở phía trên có được thông tin trực tiếp hơn từ các lớp dưới nên sẽ điều chỉnh trọng số hiệu quả hơn.

Thực tế, ResNet không phải là kiến trúc đầu tiên sử dụng các kết nối tắt, Highway Network sử dụng kiến trúc cổng với kết nối tắt. Các cổng được tham số hóa có nhiệm vụ kiểm soát lượng thông tin được phép truyền qua kết nối tắt. Ý tưởng tương tự cũng được sử dụng trong [LSTM](https://trituenhantao.io/tu-dien-thuat-ngu/lstm/), cổng được tham số hóa kiểm soát lượng thông tin sẽ truyền đến bước tiếp theo. Do đó, ResNet có thể được coi là một trường hợp đặc biệt của Highway Network.

Tuy nhiên, các thử nghiệm cho thấy Highway Network hoạt động không tốt hơn ResNet, điều này có vẻ khó giải thích vì Highway Network đưa ra một giải pháp tổng quát hơn ResNet. Từ đó có thể thấy rằng một giải pháp tổng quát chưa chắc đã đem đến hiệu suất tốt hơn. Giải pháp ResNet là một giải pháp đơn giản tập trung vào cải tiến thông tin phản hồi thông qua độ dốc của mạng.

Sau ResNet, hàng loạt những biến thể của kiến trúc này được giới thiệu. Thực nghiệm cho thấy những kiến trúc sau này có thể được huấn luyện mạng [nơ ron](https://trituenhantao.io/tu-dien-thuat-ngu/no-ron/) với độ sâu hàng nghìn lớp. ResNet nhanh chóng trở thành kiến trúc phổ biến nhất trong thị giác máy tính.

## 1.3. Tổng quan và cách cài đặt tensorflow :

### 1.3.1. Giới thiệu:

Tensorflow là một thư viện mã nguồn mở do nhóm Google Drain nghiên cứu và phát triển. Sau đó được phát hành theo giấy phép mã nguồn mở Apache 2.0 vào ngày 9/11/2015. Nó được sử dụng nhiều trong xử lý trí tuệ nhân tạo – AI, machine learning,…

Tensorflow hỗ trợ CPU, GPU và đặt biệt là TPU – (Tensor Processing Unit : bộ xử lý Tensor được Google nghiên cứu và phát triển nhằm tối ưu cho machine learning nói riêng và xử lý trí tuệ nhân tạo nói chung). Nền tảng Google Cloud Platform đã hỗ trợ đầy đủ cho Tensorflow với cả CPU, GPU và TPU.

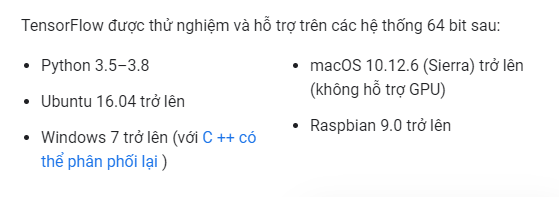
Tensorflow hỗ trợ cho cả PC và mobile, đồng thời nó hỗ trợ một số ngôn ngữ lập trình như ở bên dưới:

* Python
* C/C++
* Java
* Go
* JavaScript

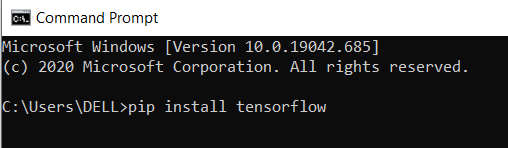
Trong những ngôn ngữ lập trình mà Tensorflow hỗ trợ thì Python sẽ có tính tương thích cao nhất và có cả tương thích ngược giữa các phiên bản. Có nghĩa là code được implement với Tensorlfow 2.x có thể chạy tốt trên Tensorflow 1.x. Còn các ngôn ngữ còn lại thì không tương thích ngược.

### 1.3.2. Cài đặt:

Vào trang chủ của Python và chọn phiên bản Python 3.5 trở lên để tải về và cài đặt. Bởi vì hiện tại thì Tensorflow 2 chỉ hỗ trờ từ phiên bản Python 3.5 trở lên.



Sau khi đã cài xong Python thì mở command line lên và gõ: pip install tensorflow để tiến hành cài đặt Tensorflow



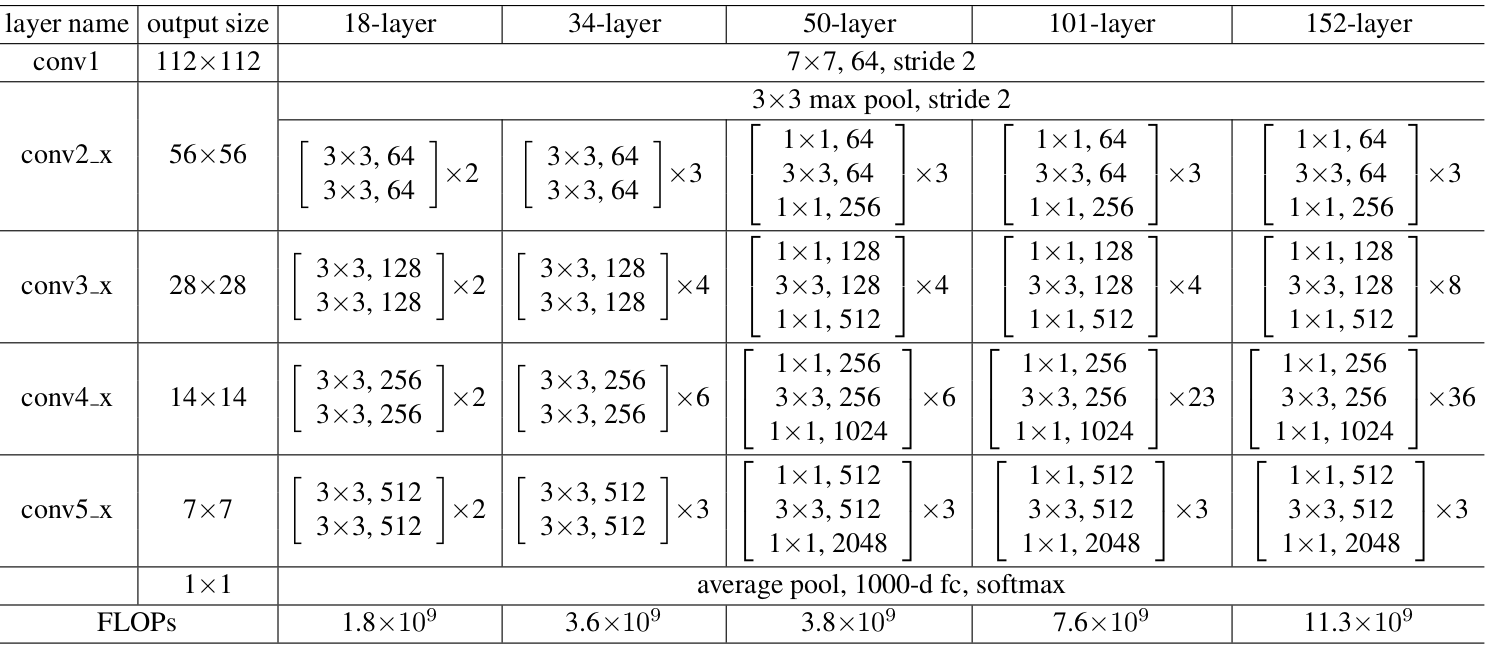
## 1.4. Bài toán minh họa ResNet:

Tập dữ liệu được lấy trên : <https://www.kaggle.com/a03102030/plant-pathology-2020-resnet50/notebook>.

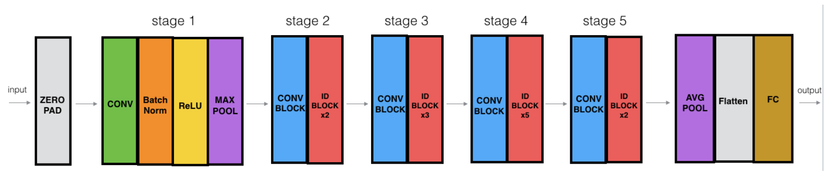
Tập dữ liệu gồm 1 file train.csv, 1file test.csv và 1 file ảnh gồm 200 bức ảnh về lá của các thực vật.

Kiến trúc sử dụng là ResNet-50

### 1.4.1.Giới thiệu chung về ResNet-50 :



Hình dưới đây mô tả chi tiết kiến trúc mạng nơ ron ResNet :



"ID BLOCK" trong hình trên là viết tắt của từ Identity block và ID BLOCK x3 nghĩa là có 3 khối Identity block chồng lên nhau. Nội dung hình trên như sau :

* Zero-padding : Input với (3,3)
* Stage 1 : Tích chập (Conv1) với 64 filters với shape(7,7), sử dụng stride (2,2). BatchNorm, MaxPooling (3,3).
* Stage 2 : Convolutiontal block sử dụng 3 filter với size 64x64x256, f=3, s=1. Có 2 Identity blocks với filter size 64x64x256, f=3.
* Stage 3 : Convolutional sử dụng 3 filter size 128x128x512, f=3,s=2. Có 3 Identity blocks với filter size 128x128x512, f=3.
* Stage 4 : Convolutional sử dụng 3 filter size 256x256x1024, f=3,s=2. Có 5 Identity blocks với filter size 256x256x1024, f=3.
* Stage 5 :Convolutional sử dụng 3 filter size 512x512x2048, f=3,s=2. Có 2 Identity blocks với filter size 512x512x2048, f=3.
* The 2D Average Pooling : sử dụng với kích thước (2,2).
* The Flatten.
* Fully Connected (Dense) : sử dụng softmax activation.

### 1.4.2.Thực hiện bài toán :

**Bước 1**: Khởi tạo thư viện

from keras.utils import np\_utils  
from keras.models import Sequential  
from keras.layers import Convolution2D,Dense,MaxPool2D,Activation,Dropout,Flatten  
from keras.layers import GlobalAveragePooling2D  
from keras.optimizers import Adam  
from sklearn.model\_selection import train\_test\_split  
from keras.layers.normalization import BatchNormalization  
import os  
import pandas as pd  
import plotly.graph\_objs as go  
import matplotlib.ticker as ticker  
import matplotlib.pyplot as plt  
import seaborn as sns  
import plotly.express as px  
import cv2  
import numpy as np

**Bước 2**: Xử lý dữ liệu

print(os.listdir('databaseResNet/plant-pathology-2020'))  
  
train=pd.read\_csv("databaseResNet/plant-pathology-2020/train.csv")  
test=pd.read\_csv("databaseResNet/plant-pathology-2020/test.csv")  
train['image\_id']=train['image\_id']+'.jpg'  
test['image\_id']=test['image\_id']+'.jpg'  
print(train.head())  
  
img=[]  
filename=train.image\_id  
for file in filename:  
 image=cv2.imread("databaseResNet/plant-pathology-2020/images/"+file)  
 res=cv2.resize(image,(256,256))  
 img.append(res)  
img=np.array(img)  
  
print(img.shape)  
  
plt.figure(figsize=(15,15))  
for i in range(9):  
 plt.subplot(3,3,i+1)  
 plt.imshow(img[i])  
  
train\_labels = np.float32(train.loc[:, 'healthy':'scab'].values)  
  
train, val = train\_test\_split(train, test\_size = 0.5)  
  
  
from keras.preprocessing.image import ImageDataGenerator  
train\_datagen = ImageDataGenerator( horizontal\_flip=True,  
 vertical\_flip=True,  
 rotation\_range=10,  
 width\_shift\_range=0.1,  
 height\_shift\_range=0.1,  
 zoom\_range=.1,  
 fill\_mode='nearest',  
 shear\_range=0.1,  
 rescale=1/255,  
 brightness\_range=[0.5, 1.5])  
  
train\_generator=train\_datagen.flow\_from\_dataframe(train,directory='databaseResNet/plant-pathology-2020/images/',  
 target\_size=(384,384),  
 x\_col="image\_id",  
 y\_col=['healthy','multiple\_diseases','rust','scab'],  
 class\_mode='raw',  
 shuffle=False,  
 subset='training',  
 batch\_size=32)  
  
val\_generator=train\_datagen.flow\_from\_dataframe(val,directory='databaseResNet/plant-pathology-2020/images/',  
 target\_size=(384,384),  
 x\_col="image\_id",  
 y\_col=['healthy','multiple\_diseases','rust','scab'],  
 class\_mode='raw',  
 shuffle=False,  
 batch\_size=32,  
 )  
  
test\_generator=train\_datagen.flow\_from\_dataframe(test,directory='databaseResNet/plant-pathology-2020/images/',  
 target\_size=(384,384),  
 x\_col="image\_id",  
 y\_col=None,  
 class\_mode=None,  
 shuffle=False,  
 batch\_size=32)

**Bước 3**: Thực hiện với kiến trúc ResNet-50

from keras.applications.resnet50 import ResNet50  
from keras.models import Model  
import keras  
from keras import optimizers  
model\_finetuned = ResNet50(include\_top=False, weights='imagenet', input\_shape=(384,384,3))  
x = model\_finetuned.output  
x = GlobalAveragePooling2D()(x)  
x = Dense(128, activation="relu")(x)  
x = Dense(64, activation="relu")(x)  
predictions = Dense(4, activation="softmax")(x)  
model\_finetuned = Model(inputs=model\_finetuned.input, outputs=predictions)  
model\_finetuned.compile(optimizer='adam',  
 loss = 'categorical\_crossentropy',  
 metrics=['accuracy'])  
model\_finetuned.summary()  
  
from keras.callbacks import ReduceLROnPlateau  
  
history\_1 = model\_finetuned.fit\_generator(train\_generator,  
 steps\_per\_epoch=2,  
 epochs=5,validation\_data=val\_generator,validation\_steps=10  
 ,verbose=1,callbacks=[ReduceLROnPlateau(monitor='val\_loss', factor=0.3,patience=3, min\_lr=0.000001)],use\_multiprocessing=False,  
 shuffle=True)  
  
fig = go.Figure(data=[  
 go.Line(name='train\_acc', x=history\_1.epoch, y=history\_1.history['accuracy']),  
 go.Line(name='Val\_acc', x=history\_1.epoch, y=history\_1.history['val\_accuracy'])])  
  
fig.update\_layout(  
 title="Accuracy",  
 xaxis\_title="epoch",  
 yaxis\_title="accuracy",  
 font=dict(  
 family="Courier New, monospace",  
 size=18,  
 color="#7f7f7f"  
 ))  
  
fig.show()  
SUB\_PATH = "databaseResNet/plant-pathology-2020/sample\_submission.csv"  
  
sub = pd.read\_csv(SUB\_PATH)  
probs\_RESNET = model\_finetuned.predict(test\_generator, verbose=1)  
sub.loc[:, 'healthy':] = probs\_RESNET

**Bước 4**: Hiển thị kết quả ra 1 file csv riêng

sub.to\_csv('resultResNet/submission\_RESNET.csv', index=False)  
print(sub.head())

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| image\_id | healthy | multiple\_diseases | rust | scab |
| Test\_0 | 0.21757704 | 0.001319926 | 0.758147776 | 0.022955241 |
| Test\_1 | 0.21234405 | 0.001451361 | 0.76270467 | 0.023499897 |
| Test\_2 | 0.219856322 | 0.00129359 | 0.756073415 | 0.02277676 |
| Test\_3 | 0.22040011 | 0.001299632 | 0.755606651 | 0.022693653 |
| Test\_4 | 0.218642846 | 0.001292591 | 0.757475674 | 0.022588897 |

# CHƯƠNG 2: BÀI TOÁN HỌC MÁY

## 2.1. Bài toán hồi quy dự đoán giá xe ô tô Toyota Corolla cũ :

### 2.1.1. Mô tả bài toán :

Trong xu thế phát triển mạnh mẽ của xã hội như hiện nay, có rất nhiều người lựa chọn mua xe ô tô để phục vụ di chuyển, vừa để khẳng định đẳng cấp, vị thế của mình. Thế nhưng việc quyết định nên mua xe cũ hay xe mới lại trở thành một bài toán khiến cho nhiều người phải đau đầu suy nghĩ. Vấn đề được – mất trong việc mua xe ô tô cũ có lẽ đã trở thành vấn đề được rất nhiều người quan tâm. Bỏ qua những điểm hạn chế khi mua xe ô tô cũ so với ô tô mới như là các vấn đề bảo hành, nội thất, tiện nghi, ta có thể kể ra được một số ưu điểm khi mua ô tô cũ. Đầu tiên là việc tiết kiệm chi phí. Có thể nói, một trong những lợi thế lớn nhất khi mua ô tô cũ đó là tiết kiếm được một khoản chi phí tương đối lớn. Một ưu điểm nữa là ta có rất nhiều mẫu mã và chủng loại để lựa chọn ô tô sao cho phù hợp với túi tiền của bản thân và cho gia đình.

Trước những vấn đề đó, trong báo cáo học tập này, em đã sử dụng thuật toán **Hồi quy tuyến tính (Linear Regression)** để giải quyết bài toán “hồi quy dự đoán giá bán xe ô tô Toyota Corolla cũ” thông qua các thuộc tính cơ bản của xe ô tô.

### 2.1.2. Thu thập và mô tả dữ liệu

Trong bài toán dự đoán giá bán xe ô tô cũ này, em đã tìm hiểu và thu thập đươc tập dữ liệu các thuộc tính của xe ô tô cũ mà ở đây cụ thể là của nhãn hiệu Toyota Corolla. Tập dữ liệu nói trên được chúng em thu thập thông qua trang web **https://github.com/datailluminations/PredictingToyotaPricesBlog.** Tập dữ liệu bao gồm 1437 mẫu với 10 thuộc tính của xe ô tô.

### 2.1.3. Mô tả thuộc tính

|  |  |  |
| --- | --- | --- |
| **STT** | **Thuộc tính** | **Mô tả** |
| 1 | Price | Giá bán |
| 2 | Age | Tuổi thọ |
| 3 | KM | Số lượng km xe đã đi |
| 4 | HP | Horse Power: Mã lực |
| 5 | Met color | Màu kim loại |
| 6 | Automatic | Xe tự động |
| 7 | CC | Dung tích động cơ |
| 8 | Doors | Số cửa |
| 9 | Weight | Trọng lượng |
| 10 | FuelType | Loại nhiên liệu |

Bảng 2.1. Các thuộc tính của tập dữ liệu

Với 10 thuộc tính của xe trên đây, để phù hợp với yêu cầu là dự đoán giá bán xe ô tô cũ, chúng em chọn thuộc tính **Price** ở đây làm biến mục tiêu cần tìm, các thuộc tính còn lại làm biến giải thích.

### 2.1.4. Số lượng mẫu

Tập dữ liệu của chúng em bao gồm 1437 dòng tương ứng với số lượng mẫu. Dưới đây là một số mẫu có trong tập dữ liệu:

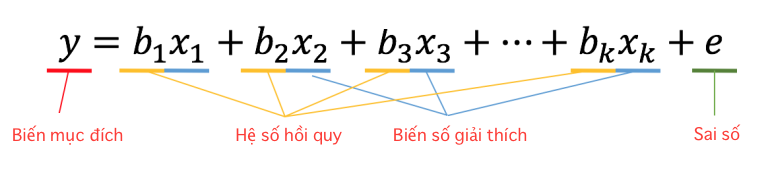
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Price** | **Age** | **KM** | **HP** | **MetColor** | **Automatic** | **CC** | **Doors** | **Weight** | **FuelType** |
| 13500 | 23 | 46986 | 90 | 1 | 0 | 2000 | 3 | 1165 | Diesel |
| 13750 | 23 | 72937 | 90 | 1 | 0 | 2000 | 3 | 1165 | Diesel |
| 13950 | 24 | 41711 | 90 | 1 | 0 | 2000 | 3 | 1165 | Diesel |
| 14950 | 26 | 48000 | 90 | 0 | 0 | 2000 | 3 | 1165 | Diesel |
| 13750 | 30 | 38500 | 90 | 0 | 0 | 2000 | 3 | 1170 | Diesel |
| 12950 | 32 | 61000 | 90 | 0 | 0 | 2000 | 3 | 1170 | Diesel |
| 16900 | 27 | 94612 | 90 | 1 | 0 | 2000 | 3 | 1245 | Diesel |

Bảng 2.2. Mô tả chi tiết tập dữ liệu

### 2.1.5. Xây dựng mô hình thuật toán và giới thiệu chung Linear Regression:

Học có giám sát (Supervised Learning) được chia ra làm 2 dạng lớn là Hồi quy (Regression) và Phân loại (Classification) dựa trên tập dữ liệu mẫu - tập huấn luyện (training data). Ở đây chúng em sẽ tìm hiểu về Hồi quy mà cụ thể là Hồi quy tuyến tính (Linear Regression).

**Khái niệm:** Mô hình hồi quy tuyến tính (Linear Regression) được biểu diễn dưới dạng công thức toán học, là loại mô hình dự đoán giá trị của biến mục tiêu dựa trên giá trị của biến giải thích.



Hình 2.1. Mô hình thuật toán Hồi quy tuyến tính

Ví dụ: dự đoán giá nhà ở Hà Nội dựa vào thông tin về diện tích, vị trí, năm xây dựng của ngôi nhà thì biến mục tiêu ở đây sẽ là giá nhà và **x = (x1,x2,x3)** với **x1**là diện tích, **x2** là vị trí và **x3** là năm xây dựng.

Có hai dạng hồi quy tuyến tính là đơn biến và đa biến. Hồi quy tuyến tính biến đơn là một kỹ thuật được sử dụng để mô hình hóa mối quan hệ giữa một biến độc lập đầu vào (biến giải thích) và biến phụ thuộc đầu ra bằng mô hình tuyến tính, ví dụ như một đường. Trường hợp tổng quát hơn là Hồi quy tuyến tính đa biến trong đó một mô hình được tạo cho mối quan hệ giữa nhiều biến đầu vào độc lập (biến giải thích) và biến phụ thuộc đầu ra. Mô hình vẫn tuyến tính trong đó đầu ra là sự kết hợp tuyến tính của các biến đầu vào.

Có một trường hợp tổng quát thứ ba được gọi là Hồi quy đa thức trong đó mô hình bây giờ trở thành tổ hợp phi tuyến tính của các biến giải thích, có thể có các biến số mũ, sin và cos, v.v. Tuy nhiên, điều này đòi hỏi kiến thức về cách dữ liệu liên quan đến đầu ra. Các mô hình hồi quy có thể được đào tạo bằng cách sử dụng thuật toán giảm dần độ dốc Stochastic Gradient Descent (SGD).

Ưu điểm và nhược điểm của thuật toán Hồi quy tuyến tính:

**Ưu điểm:**

* Mô hình đơn giản, dễ hiểu
* Dễ diễn giải hệ số hồi quy
* Nhận được kết quả khả thi khi dữ liệu nhỏ
* Nhiều cải tiến/mở rộng

**Nhược điểm:**

* Mô hình đơn giản nên khó dự đoán chính xác với dữ liệu có miền giá trị rộng
* Nhạy cảm với dữ liệu ngoại lai – do sử dụng phương pháp bình phương nhỏ nhất

Như đã trên trình bày ở trên, mô hình đơn giản nhất được sử dụng trong thuật toán Hồi quy tuyến tính có dạng:

Áp dụng vào bài toán này ta có thể thấy được **y** (biến mục tiêu) cần tìm ở đây là **Price** (giá nhà), còn **x1** là Age, **x2**là KM, **x3** là FuelType,… ở đây là độ lệch chuẩn, ,… là các tham số của mô hình. Các tham số này sẽ đánh giá mức độ quan trọng của các thuộc tính đến kết quả cuối cùng.

### 2.1.6. Huấn luyện mô hình

Sau khi có mô hình hồi quy, ta thực hiện đến bước huấn luyện chúng. Trước khi bắt đầu huấn luyện, ta chia tập dữ liệu thành hai phần: một phần sử dụng cho việc huấn luyện (Training) và phần còn lại sử dụng trong việc kiểm thử mô hình (Test). Huấn luyện mô hình là việc tìm ra các tham số tối ưu nhất thông qua bộ dữ liệu huấn luyện. Để làm được điều này chúng ta phái xác định được thước đo để biết được mô hình tốt hay không tốt (tham số có tối ưu hay không).

Một trong những phương pháp được sử dụng để ước lượng tham số là dùng biểu thức Root Mean Square Error (RMSE). Mục tiêu của việc huấn luyện đó là giảm thiểu giá trị của Mean Square Error (MSE).

MSE của mô hình hồi quy tuyến tính trên tập huấn luyện X được tính như sau:

Để tìm giá trị θ để phương trình MSE(***θ***) đạt cực tiểu có thể tìm được nhờ công thức sau:

Trong đó:

* là giá trị của ***θ*** *mà tại đó MSE(****θ)*** *đạt cực tiểu*
* là vector giá trị cần tìm bao gồm

### 2.1.7. Áp dụng kỹ thuật, thuật toán vào bài toán

Sau khi huấn luyện mô hình, ta sử dụng ngôn ngữ lập trình Python trong ứng dụng PyCharm IDE để thực hiện khởi tạo mô hình trên máy tính:

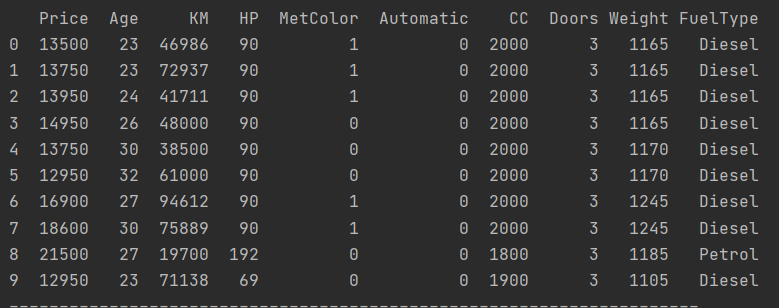
**Bước 1:** Khởi tạo thư viện

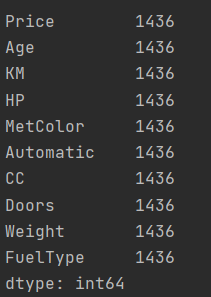
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.metrics import mean\_squared\_error  
from sklearn.model\_selection import cross\_val\_score

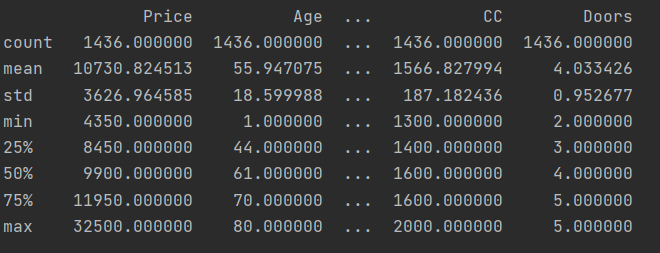
**Bước 2:** Xử lý dữ liệu

**Đọc file dữ liệu đã tải về**

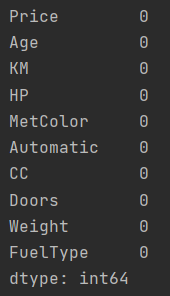
dataset = pd.read\_csv('database/ToyotaCorolla.csv')  
print(dataset.head(10))  
print("---------------------------------------------------------------------")

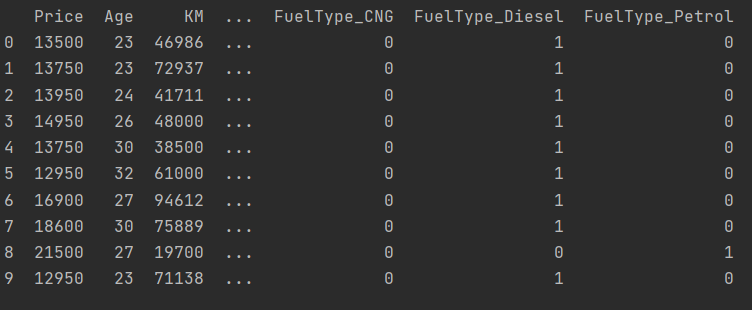
  
print(dataset.count())  
print("---------------------------------------------------------------------")

  
print(dataset.describe())  
print("---------------------------------------------------------------------")

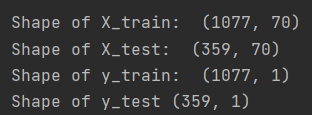


print(dataset.isnull().sum())  
print("---------------------------------------------------------------------")

  
  
# dataset = pd.get\_dummies(dataset, columns=['FuelType'])  
dataset = pd.get\_dummies(dataset)  
print(dataset.head(10))

  
X = dataset.drop(['Price'], axis = 1).values  
y = dataset.iloc[:, 0].values.reshape(-1,1)  
  
print("---------------------------------------------------------------------")

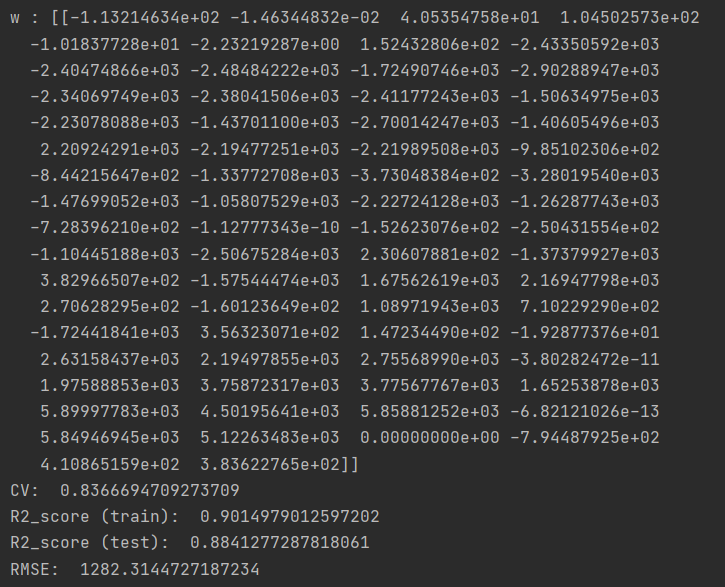
**Chia tập dữ liệu thành tập Train và tập Test**  
  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 42)  
print("Shape of X\_train: ", X\_train.shape)  
print("Shape of X\_test: ", X\_test.shape)  
print("Shape of y\_train: ", y\_train.shape)  
print("Shape of y\_test", y\_test.shape)  
print("---------------------------------------------------------------------")

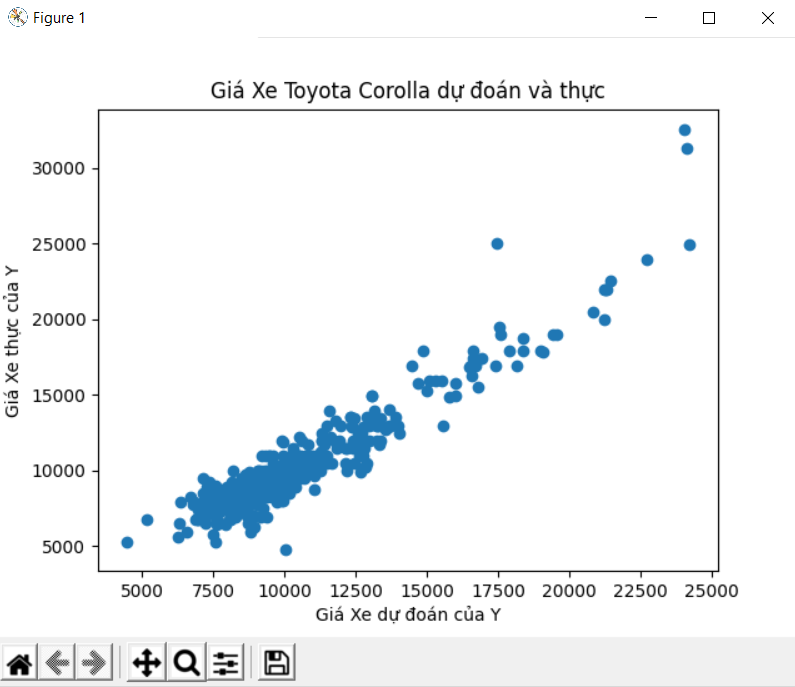


**Bước 3:** Áp dụng thuật toán Linear Regression giải quyết bài toán

#Linear Regression  
from sklearn.linear\_model import LinearRegression  
regressor\_linear = LinearRegression()  
regressor\_linear.fit(X\_train, y\_train)  
  
from sklearn.metrics import r2\_score  
  
# Predicting Cross Validation Score the Test set results  
#Dự đoán kết quả giao của tập Test  
cv\_linear = cross\_val\_score(estimator = regressor\_linear, X = X\_train, y = y\_train, cv = 10)  
  
# Predicting R2 Score the Train set results  
y\_pred\_linear\_train = regressor\_linear.predict(X\_train)  
r2\_score\_linear\_train = r2\_score(y\_train, y\_pred\_linear\_train)  
  
# Predicting R2 Score the Test set results  
y\_pred\_linear\_test = regressor\_linear.predict(X\_test)  
r2\_score\_linear\_test = r2\_score(y\_test, y\_pred\_linear\_test)  
  
# Predicting RMSE the Test set results  
rmse\_linear = (np.sqrt(mean\_squared\_error(y\_test, y\_pred\_linear\_test)))

print("w :",regressor\_linear.coef\_)#Sai so  
print("CV: ", cv\_linear.mean())  
print('R2\_score (train): ', r2\_score\_linear\_train)  
print('R2\_score (test): ', r2\_score\_linear\_test)  
print("RMSE: ", rmse\_linear)#Trung binh sai so

  
  
plt.scatter(y\_pred\_linear\_test,y\_test)  
plt.xlabel("Giá Xe dự đoán của Y")  
plt.ylabel("Giá Xe thực của Y")  
plt.title("Giá Xe Toyota Corolla dự đoán và thực")  
plt.show()



### 2.1.7. Đánh giá mô hình

Sau khi việc huấn luyện hoàn tất, ta cần xem mô hình có tốt không bằng cách dùng bước Đánh giá. Việc đánh giá cho phép chúng ta kiểm tra mô hình với những dữ liệu chưa từng được dùng cho huấn luyện.

CV: 0.8366694709273709

R2\_score (train): 0.9014979012597202

R2\_score (test): 0.8841277287818061

RMSE: 1282.3144727187234

## 2.2. Bài toán phân lớp giá điện thoại :

### 2.2.1. Mô tả bài toán :

Bài toán sử dụng bộ dữ liệu các thuộc tính vật lý điện thoại để phân loại ra 4 loại giá điện thoại. Sử dụng **bài toán học máy phân lớp**. Áp dụng ký thuật **Logistic Regression**

### 2.2.2. Thu thập và mô tả dữ liệu

<https://www.kaggle.com/iabhishekofficial/mobile-price-classification>

**Thông tin tập dữ liệu:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Đặc điểm của Tập dữ liệu:** | Đa biến | **Số phiên bản:** | 1 | **Khu vực:** | Đời sống |
| **Đặc điểm thuộc tính:** | Phân loại | **Số thuộc tính:** | 20 | **Ngày được tặng** | 2017 |
| **Nhiệm vụ liên quan:** | Phân loại | **Giá trị bị mất?** | Đúng | **Số lượt truy cập web:** | 236k |

Bob đã thành lập công ty di động của riêng mình. Anh ta muốn đấu tranh gay gắt với các công ty lớn như Apple, Samsung, v.v. Anh ta không biết làm thế nào để ước tính giá điện thoại di động mà công ty mình tạo ra. Trong thị trường điện thoại di động cạnh tranh này, bạn không thể chỉ đơn giản giả định mọi thứ. Để giải quyết vấn đề này, anh ta thu thập dữ liệu bán điện thoại di động của nhiều công ty khác nhau. Bob muốn tìm hiểu một số mối quan hệ giữa các tính năng của điện thoại di động (ví dụ: - RAM, Bộ nhớ trong, v.v.) và giá bán của nó. Nhưng anh ấy không giỏi về Máy học. Vì vậy, anh ấy cần sự giúp đỡ của bạn để giải quyết vấn đề này. Trong bài toán này, bạn không phải dự đoán giá thực tế mà là một khoảng giá cho biết giá cao như thế nào

### 2.2.3. Mô tả thuộc tính

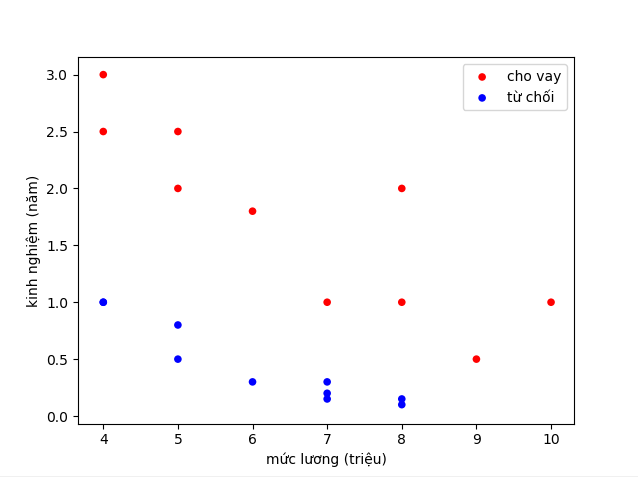
|  |  |  |
| --- | --- | --- |
| STT | Thuộc tính | Mô tả |
| 1 | Battery | Interger (Lượng pin) |
| 2 | Bluetooth | Boolean (Có bluetooth hay không) |
| 3 | Clock speed | Float (Tốc độ bộ vi xử lý) |
| 4 | Dual sim | Boolean ( Có sim đôi hay không ) |
| 5 | Front camera | Boolean (Có sữa hay không) |
| 6 | 4G | Boolean (Có sim 4G hay không) |
| 7 | Internal memory | Integer (Bộ nhớ trong) |
| 8 | Mobile depth | Integer (Chiều sâu) |
| 9 | Weight of mobile | Float (trọng lượng) |
| 10 | Number of core | Integer (Số nhân) |
| 11 | Primary camera | Integer (Cam sau) |
| 12 | Pixel Height | Integer |
| 13 | Pixel Width | Integer |
| 14 | Ram | Integer |
| 15 | Screen Height | Float |
| 16 | Screen Width | Float |
| 17 | 3G | Boolean |
| 18 | Talk time | Integer |
| 19 | Touch Screen | Boolean |
| 20 | Wifi | Boolean |
| 21 | Price Range | Integer |

Bảng 2.3. Các thuộc tính của tập dữ liệu

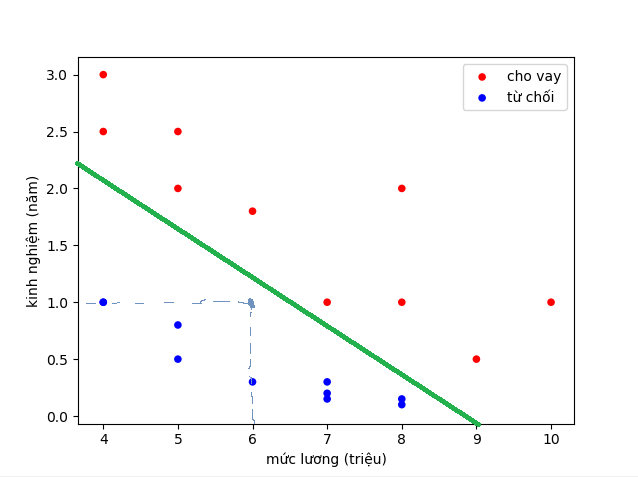
### 2.2.4. Giới thiệu Logistic Regression và mô hình giải bài toán :

Trong bài viết này, chúng ta sẽ thảo luận các khái niệm Logistic Regression và xem nó có thể giúp chúng ta xử lý các vấn đề thế nào.

Logistic Regression là 1 thuật toán phân loại được dùng để gán các đối tượng cho 1 tập hợp giá trị rời rạc (như 0, 1, 2, ...). Một ví dụ điển hình là phân loại Email, gồm có email công việc, email gia đình, email spam, ... Giao dịch trực tuyến có là an toàn hay không an toàn, khối u lành tính hay ác tình. Thuật toán trên dùng hàm sigmoid logistic để đưa ra đánh giá theo xác suất. Ví dụ: Khối u này 80% là lành tính, giao dịch này 90% là gian lận, ...

Ngân hàng bạn đang làm có chương trình cho vay ưu đãi cho các đối tượng mua chung cư. Tuy nhiên gần đây có nhiều chung cư hấp dẫn nên lượng hồ sơ người nộp cho chương trình ưu đãi tăng lên nhiều. Bình thường bạn có thể duyệt 10-20 hồ sơ một ngày để quyết định hồ sơ có được cho vay hay không, tuy nhiên gần đây bạn nhận được 1000-2000 hồ sơ mỗi ngày. Bạn không thể xử lý hết hồ sơ và bạn cần có một giải pháp để có thể dự đoán hồ sơ mới là có nên cho vay hay không. Sau khi phân tích, bạn nhận thấy có 2 yếu tố quyết định đến việc hồ sơ có được chấp nhận hay không, đó là mức lương và kinh nghiệm làm việc. Dưới đây là 1 đồ thị ví dụ

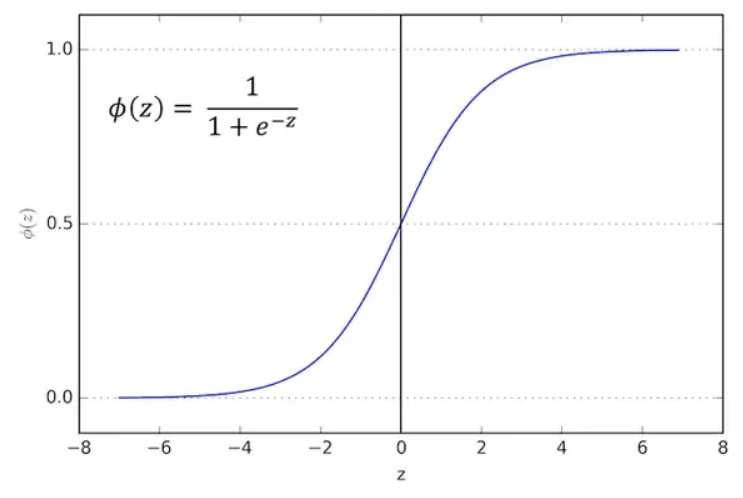
Về mặt logic, chúng ta có thể nghĩ ngay đến việc vẽ 1 đường thẳng phân chia các điểm xanh và đó, rồi đưa ra quyết định cho 1 điểm mới dựa vào đường thẳng đó. Ví dụ thế này:



Ví dụ đường xanh là đường phân chia. Dự đoán cho hồ sơ của người có mức lương 6 triệu và 1 năm kinh nghiệm là không chấp nhận

Tuy nhiên, do ngân hàng đang gặp khó khăn nên hạn chế cho vay, ngân hàng yêu cầu hồ sơ đạt trên 80% mới cho vay. Bây giờ không chỉ dừng lại ở việc quyết định cho vay hay không, mà phải tìm xác suất hồ sơ đó cho vay là bao nhiêu.

**Hàm sigmoid**

Giờ phải tìm xác suất cho vay của 1 hồ sơ, đương nhiên là giá trị trong đoạn [0, 1] rồi. Hàm mà luôn có giá trị trong đoạn [0, 1], liên tục mà lại dễ sử dụng thì đó là hàm sigmoid.

Nhận xét:

* Hàm liên tục và luôn đưa ra giá trị trong khoảng (0, 1)
* Có đạo hàm tại mọi điểm nên có thể dùng gradient descent

**Thiết lập bài toán**

Về cơ bản thì chúng ta sẽ có các bước sau cho 1 bài toán Machine learning:

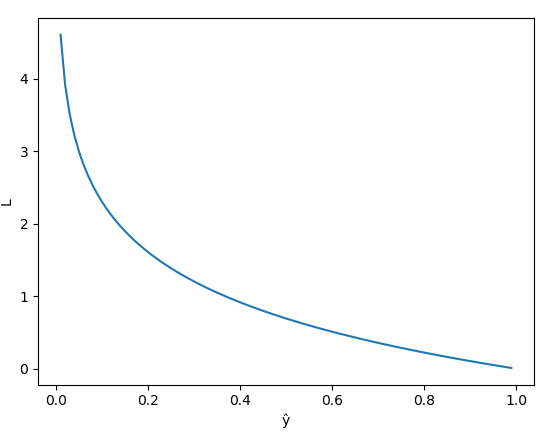
* Thiết lập model
* Thiết lập hàm mất mát Loss Function
* Tìm tham số bằng việc tối ưu loss function
* Dự đoán dữ liệu mới dựa vào loss function mới tìm được

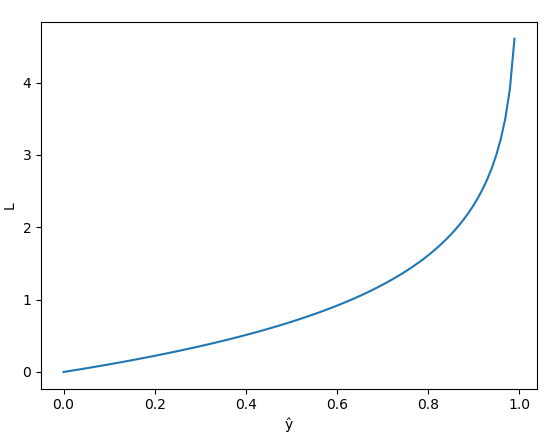
**Model**

* Với dòng thức i trong dữ liệu, gọi https://images.viblo.asia/d9f97821-75be-401c-b635-0aeef1507415.png là lương và https://images.viblo.asia/e3d9e4d2-ce8d-4e6b-a23c-984d542207cc.png là kinh nghiệm làm việc của hồ sơ thứ i
* https://images.viblo.asia/8c0f8d7c-824e-43a0-8a2d-02a58a4ace0c.pnglà xác suất mà model dự đoán hồ sơ thứ i cho vay
* https://images.viblo.asia/792b0e6b-a144-4659-bdf7-bd87208050e7.pnglà xác suất mà model dự đoán hồ sơ thứ i không cho vay.
* Ta có ngay https://images.viblo.asia/81a18a8e-7c71-43e6-91f6-fd3225be6b90.png
* Hàm sigmoid là: https://images.viblo.asia/77820f3f-0631-4b07-9f01-541d4bc0a223.png
* Tương tự như hàm dự đoán trong Linear Regression là https://images.viblo.asia/5388405c-2ddb-4dea-8c6c-4db68e12892a.png, thì trong Logistic Regression ta có hàm dự đoán như sau: https://images.viblo.asia/43965a36-f47e-4f23-8d05-695a315f7ce5.png

**Loss Function - Hàm mất mát**

* Bây giờ chúng ta cần 1 hàm để đánh giá độ tốt của model (tức làm dự đoán).
* Ta có nhận xét như sau: + Nếu hồ sơ thứ i là cho vay, tức https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png = 1 thì ta mong muốn https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png càng gần 1 càng tốt hay model dự đoán xác suất cho hồ sơ thứ i vay càng cao càng tốt. + Nếu hồ sơ thứ i là không cho vay, tức https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png = 0 thì ta mong muốn https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png càng gần 0 càng tốt hay model dự đoán xác suất cho hồ sơ thứ i vay càng thấp càng tốt.
* Với mỗi điểm (https://images.viblo.asia/96db60b5-c191-43f1-873e-0b685263a34d.png, https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png), ta gọi hàm loss function https://images.viblo.asia/ff288b90-04cc-417e-9b8f-5d18335f1446.png (Trong Machine learning, Deep leaning thì chúng ta hiểu log là ln nhé)
* Thử đánh giá hàm L nhé. Nếu https://images.viblo.asia/078a4431-8732-4724-a6f8-416c689488a7.png. Đây là đồ thị hàm loss trong trường hợp https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png = 1

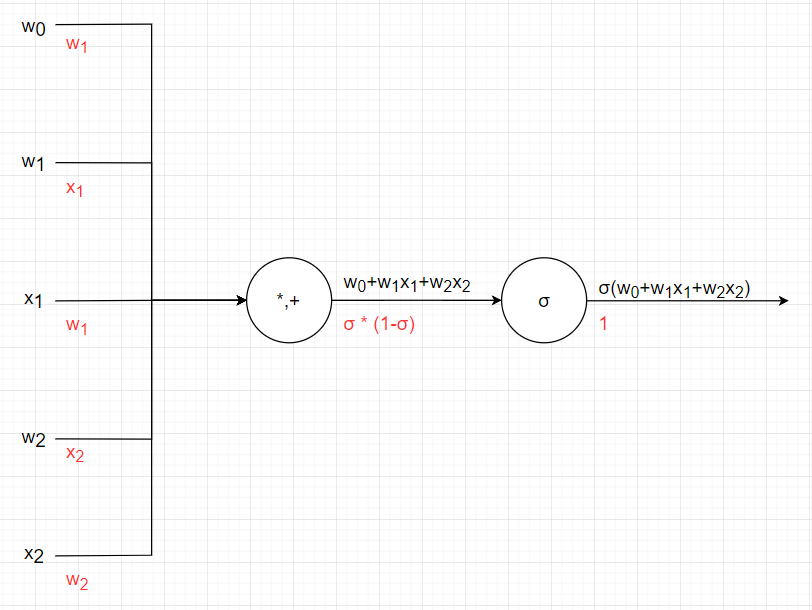
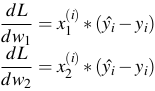


* Nhận xét: + Hàm L giảm từ 0 đến 1 + Khi model dự đoán https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png = 1, tức giá trị dự đoán gần với giá trị thật https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png thì L nhỏ, xấp xỉ 0. + Khi model dự đoán https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png = 0, tức giá trị dự đoán ngược lại với giá trị thật https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png thì L rất lớn.
* Ngược lại, nếu https://images.viblo.asia/43f2576b-0272-46af-847b-23fcf7d499be.png, ta có đồ thị sau 
* Nhận xét: + Hàm L tăng từ 0 đến 1 + Khi model dự đoán https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png gần 0, tức giá trị dự đoán gần với giá trị thật https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png thì L nhỏ, xấp xỉ 0. + Khi model dự đoán https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png gần 1, tức giá trị dự đoán ngược lại với giá trị thật https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png thì L rất lớn => Hàm L nhỏ khi giá trị model gần với giá trị thật và rất lớn khi model dự đoán sai, hay nói cách khác L càng nhỏ thì model dự đoán càng gần với giá trị thật. => Bài toán toán quy về tìm giá trị nhỏ nhất của L.
* Ta có hàm mất mát trên tất cả bộ dữ liệu như sau: https://images.viblo.asia/39f2e8f2-d03b-4fc3-917b-6fa8cae579b4.png

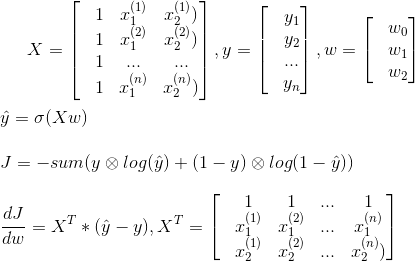
**Tính đạo hàm phức tạp bằng kỹ thuật Chain Rule**

* Chain rule là gì? Nếu z = f(y) và y = g(x) hay z = f(g(x)) thì https://images.viblo.asia/9134d3a9-1936-45a9-a95d-493a101fbe10.png
* Thử áp dụng tính đạo hàm của hàm sigmoid https://images.viblo.asia/77820f3f-0631-4b07-9f01-541d4bc0a223.png. https://images.viblo.asia/f085633c-27de-4eb6-83f3-7b5b79bc94e4.png =https://images.viblo.asia/d3ae2abf-fef7-45d3-a5f4-571d3ea8affc.png

**Áp dụng gradient descent**

* Với mỗi điểm (https://images.viblo.asia/96db60b5-c191-43f1-873e-0b685263a34d.png, https://images.viblo.asia/13db9712-1c4a-4510-8fd5-22584bf57492.png), gọi hàm mất mát https://images.viblo.asia/ff288b90-04cc-417e-9b8f-5d18335f1446.png trong đó https://images.viblo.asia/51a71410-c471-41d0-907d-4c92ab8f559f.png là giá trị mà model dự đoán, còn yi là giá trị thật của dữ liệu.
* Áp dụng Chain rule ta có: https://images.viblo.asia/71e160e4-9ae0-48cd-9452-5c36456f1e56.png https://images.viblo.asia/3c00f0a7-0f33-4fcb-8033-2192e6db6291.png 
* Từ đồ thị ta thấy: https://images.viblo.asia/b7e9d3df-22bd-47f6-92ea-5b2e0ce08922.png https://images.viblo.asia/b7e9d3df-22bd-47f6-92ea-5b2e0ce08922.png https://images.viblo.asia/fff84b5c-446f-4991-b1db-2f82fd50d22d.png Do đó: https://images.viblo.asia/e725863b-eb0a-4e1f-809a-8f1dc0b0c89f.png
* Tương tự: 
* Đấy là trên 1 điểm dữ liệu, còn trên toàn bộ dữ liệu: https://images.viblo.asia/b4bfc1f8-9da3-44b4-9083-6c05e469c00b.png https://images.viblo.asia/9d7065a6-73b5-4703-935e-4e27db70d902.png https://images.viblo.asia/80d71951-a1a4-42ba-aef1-a1884b96e141.png

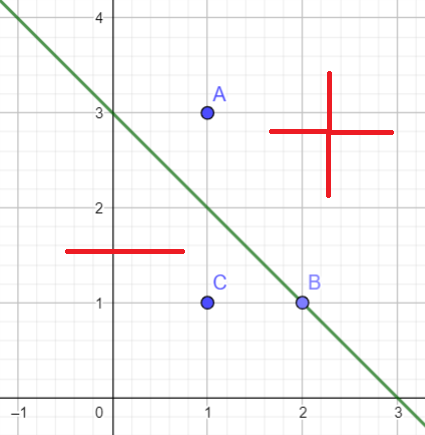
**Biểu diễn bằng ma trận**

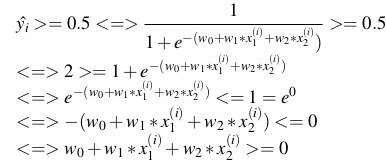


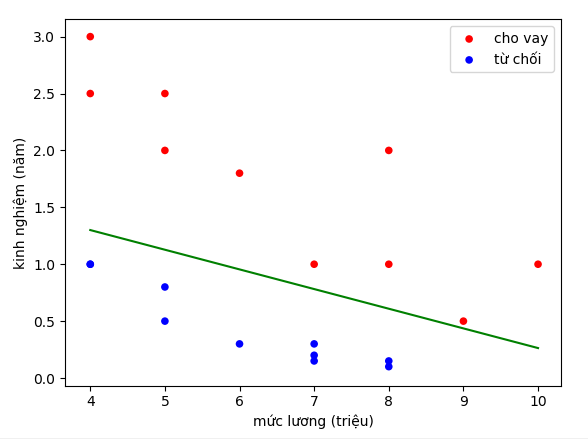
* Sau khi thực hiện gradient descent ta tìm được w0, w1, w2. Với mỗi hồ sơ mới https://images.viblo.asia/54785d11-3317-45df-9d2b-d52028e1784c.png ta tính được phần trăm nên cho vay https://images.viblo.asia/d6f3634c-bdbb-4c8d-8fc2-2ffe7ec785da.png rồi so sánh với ngưỡng cho vay của công ty t (thường là 0.5, hoặc cao hơn là 0.8), nếu https://images.viblo.asia/36b5d293-6fea-42c0-9cc2-9a5c87647f72.png thì cho vay, không thì không cho vay.

**Xây dựng đường thẳng phân chia**

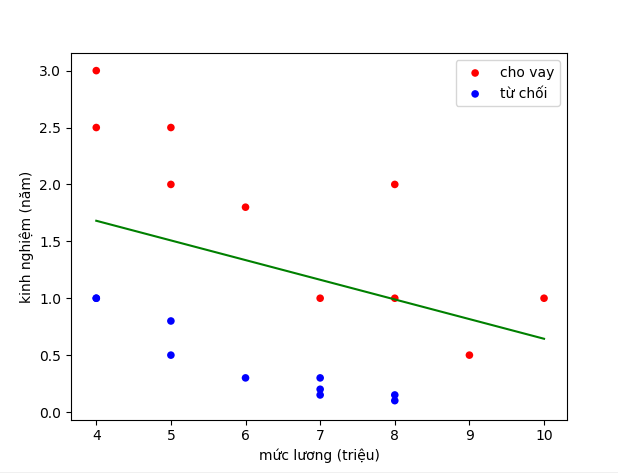
* Xét đường thẳng y = ax + b, thì f = y - (ax + b), ta có được 1 đường thẳng chia mặt phẳng là 2 phần, 1 phần f > 0, 1 phần f < 0 và các điểm trên đường thẳng thì f = 0.



* Giả sử mốc chính giữa là 0.5 thì https://images.viblo.asia/ed726ad9-c92c-43f4-afd6-6f0713d5320c.png >= 0.5 thì cho vay, ngược lại thì không cho vay. 
* Tương tựhttps://images.viblo.asia/224c574e-b9cd-4afa-8c77-3bc8964ac0a5.png => đường thẳng https://images.viblo.asia/e16b7eb3-7175-4fac-b88b-a2d3f27d6b83.png là đường phân cách giữa các điểm cho vay và từ chối.



* Trong trường hợp tổng quát t bất kỳ, https://images.viblo.asia/9c22a37c-4208-4437-ac34-18b42bfcfb1c.png <=> https://images.viblo.asia/4f87b5be-8038-4ce4-bfb0-291a80294b0b.png



* Ta thấy khi t = 0.8 thì đường phân cách gần các điểm màu đỏ hơn so với t = 0.5, thậm chí 2 điểm màu đỏ trước đó được chấp nhận thì bây giờ lại bị loại bỏ.

**Ứng dụng**

* Dự đoán email có phải spam hay không
* Dự đoán giao dịch ngân hàng là gian lận hay không
* Dự đoán khối u lành hay ác tính
* Dự đoán khoản vay có trả được không
* Dự đoán khoản đầu tư vào start-up có sinh lãi hay không.

### 2.2.5. Áp dụng kỹ thuật, thuật toán vào bài toán

**Bước 1**: Khởi tạo thư viện

import pandas as pd

import numpy as np

**Bước 2**: Xử lý dữ liệu

df = pd.read\_csv('train.csv')

df.head()

features = list(df.columns)

print(features)

features.remove('price\_range')

print(features)

X = df[features].values.astype(np.float32)

Y = df.price\_range

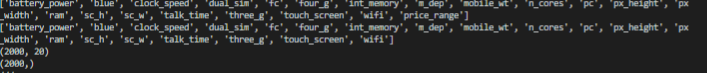
print(X.shape)

print(Y.shape)

print("///")

Chia X, Y với :

* X : Là các thuộc tính Battery đến Wifi
* Y : Là thuộc tính Price\_range



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

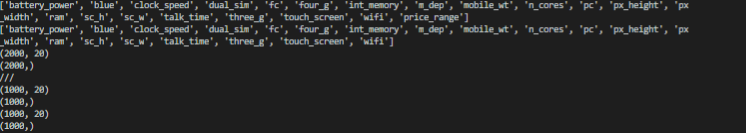
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.5, random\_state = 0)

print(X\_train.shape)

print(Y\_train.shape)

print(X\_test.shape)

print(Y\_test.shape)



# cân bằng

print('Before OverSampling:')

print('Prire\_range 0 =', sum(Y\_train == 0))

print(' Prire\_range 1 =', sum(Y\_train == 1))

print('Prire\_range 2 =', sum(Y\_train == 2))

print(' Prire\_range 3 =', sum(Y\_train == 3))

sm = SMOTE(random\_state=2)

X\_train\_res, y\_train\_res = sm.fit\_sample(X\_train, Y\_train)

print("After OverSampling, counts of label 0: {}".format(sum(y\_train\_res == 0)))

print("After OverSampling, counts of label 1: {}".format(sum(y\_train\_res == 1)))

print("After OverSampling, counts of label 2: {}".format(sum(y\_train\_res == 2)))

print("After OverSampling, counts of label 3: {}".format(sum(y\_train\_res == 3)))

**Bước 3**: Áp dụng Logistic Regression

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

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

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

#print("training accuracy :", model.score(X\_train, Y\_train))

#print("testing accuracy :", model.score(X\_test, Y\_test))

**Bước 4**: Đánh giá

# Đánh giá

from sklearn.metrics import accuracy\_score

print("Accuracy Score:", accuracy\_score(Y\_test, y\_pred), "\n")

from sklearn.metrics import classification\_report

print(classification\_report(Y\_test,y\_pred))

from sklearn.metrics import confusion\_matrix

cnf\_matrix = confusion\_matrix(Y\_test, y\_pred)

print('Confusion matrix:') print(cnf\_matrix)

## 2.3. Bài toán phân lớp loại thú :

### 2.3.1. Mô tả bài toán :

Bài toán phân loại các lớp động vật trong vườn bách thú

Bài toán sử dụng là phân lớp nhiều lớp với **kỹ thuật cây quyết định** và **phương pháp đánh giá Accuracy score và Confusion matrix.**

### 2.3.2. Thu thập và mô tả dữ liệu

Trong bài toán phân loại thu vật này, em đã tìm hiểu và thu thập đươc tập dữ liệu các đặc điểm của con vật. Tập dữ liệu nói trên được em thu thập từ **https://www.kaggle.com/uciml/zoo-animal-classification.** Tập dữ liệu bao gồm 101 dòng mẫu với 18 cột đặc điểm của con vật.

### 2.3.3. Mô tả thuộc tính

|  |  |  |
| --- | --- | --- |
| STT | Thuộc tính | Mô tả |
| 1 | Animal\_name | Tên của con vật |
| 2 | Hair | Boolean (Có lông hay không) |
| 3 | Feathers | Boolean (Có lông vũ hay không) |
| 4 | eggs | Boolean ( Có đẻ trứng hay không) |
| 5 | milk | Boolean (Có sữa hay không) |
| 6 | airborne | Boolean (Trên không có hay không) |
| 7 | aquatic | Boolean ( Sống dưới nước có hay không ) |
| 8 | predator | Boolean ( kẻ săn mồi có haykhông ) |
| 9 | toothed | Boolean ( Có răng hay không) |
| 10 | backbone | Boolean( có xương sống hay không ) |
| 11 | Breathes | Boolean |
| 12 | venomous | Boolean ( có nọc độc hay không) |
| 13 | fins | Boolean ( có vảy hay không ) |
| 14 | legs | Numeric (bộ giá trị: {0,2,4,5,6,8}) |
| 15 | tail | Boolearn (có đuôi hay không) |
| 16 | domestic | Boolearn ( có trong nước) |
| 17 | catsize | Boolearn |
| 18 | class\_type | Có 7 lớp động vật |

Bảng 2.4. Các thuộc tính của tập dữ liệu

Tập dữ liệu này bao gồm 101 loài động vật từ vườn thú.  
Có 16 biến với các đặc điểm khác nhau để mô tả các loài động vật.  
7 Loại Lớp là: Động vật có vú, Chim, Bò sát, Cá, Lưỡng cư, Bọ và Động vật không xương sống.

Mục đích của tập dữ liệu này là có thể dự đoán phân loại của các loài động vật, dựa trên các biến số.

### 2.3.4. Giới thiệu về kỹ thuật cây quyết định và mô hình giải bài toán:

Cây quyết định ([Decision Tree](https://trituenhantao.io/tu-dien-thuat-ngu/decision-tree/)) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượngncó thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary) , Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal.

Cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết.

Thuật toán ID3 sử dụng phương pháp tham lam tìm kiếm từ trên xuống thông qua không gian của các nhánh có thể không có backtracking. ID3 sử dụng [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) và Information Gain để xây dựng một cây quyết định.

Entropy là thuật ngữ thuộc Nhiệt động lực học, là thước đo của sự biến đổi, hỗn loạn hoặc ngẫu nhiên. Năm 1948, Shannon đã mở rộng khái niệm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) sang lĩnh vực nghiên cứu, thống kê với công thức như sau:

Với một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau x1,x2,…,xn.

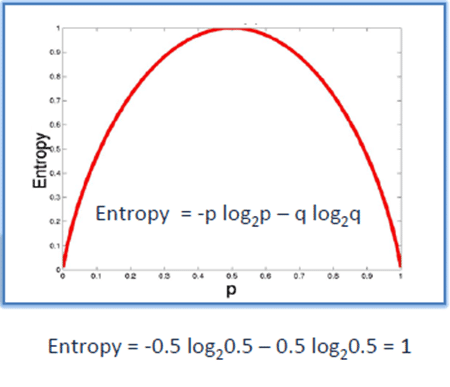
Giả sử rằng xác suất để x nhận các giá trị này là pi=p(x=xi).

Ký hiệu phân phối này là p=(p1 ,p2 ,…,pn). [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) của phân phối này được định nghĩa là:

H(p)=  – ∑nn=1 pi log(pi)

Giả sử bạn tung một đồng xu, [entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) sẽ được tính như sau:

H = -[0.5 ln(0.5) + 0.5 ln(0.5)]



Hàm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)

Hình vẽ trên biểu diễn sự thay đổi của hàm [entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/). Ta có thể thấy rằng, [entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) đạt tối đa khi xác suất xảy ra của hai lớp bằng nhau.

* P tinh khiết: pi = 0 hoặc pi = 1
* P vẩn đục: pi = 0.5, khi đó hàm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) đạt đỉnh cao nhất

**Information Gaintrong Cây quyết định (Decision Tree)**

Information Gain dựa trên sự giảm của hàm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) khi tập dữ liệu được phân chia trên một thuộc tính. Để xây dựng một cây quyết định, ta phải tìm tất cả thuộc tính trả về Infomation gain cao nhất.

Để xác định các nút trong mô hình cây quyết định, ta thực hiện tính Infomation Gain tại mỗi nút theo trình tự sau:

**Bước 1**: Tính toán hệ số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) của biến mục tiêu S có N phần tử với Nc phần tử thuộc lớp c cho trước:

H(S)=  – ∑cc=1 (Nc/N) log(Nc/N)

**Bước 2**: Tính hàm số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) tại mỗi thuộc tính: với thuộc tính x, các điểm dữ liệu trong S được chia ra K child node S1, S2, …, SK với số điểm trong mỗi child node lần lượt là m1, m2 ,…, mK , ta có:

H(x, S) = ∑Kk=1 (mk / N) \* H(Sk )

**Bước 3**: Chỉ số Gain Information được tính bằng:

**G(x, S) = H(S) – H(x,S)**

Với ví dụ 2 trên, ta tính được hệ số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) như sau:

[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)Parent= -(0.57\*ln(0.57) + 0.43\*ln(0.43)) = 0.68

Hệ số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) theo phương pháp chia thứ nhất:

[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)left= -(.75\*ln(0.75) + 0.25\*ln(0.25))  = 0.56[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)right= -(.33\*ln(0.33) + 0.67\*ln(0.67)) = 0.63

Ta có thể tính hệ số Information Gain như sau:

Information Gain = 0.68 – (4\*0.56 + 3\*0.63)/7 = 0.09

Hệ số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) với phương pháp chia thứ hai như sau:

[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)left= -(.67\*ln(0.67) + 0.33\*ln(0.33))  = 0.63[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)middle= -(.5\*ln(0.5) + 0.5\*ln(0.5))  = 0.69[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)right= -(.5\*ln(0.5) + 0.5\*ln(0.5))  = 0.69

Cách đơn giản và hay được sử dụng nhất là *accuracy* (độ chính xác). Cách đánh giá này đơn giản tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử.

Các thông số

**Precision**

Xem xét trên tập dữ liệu kiểm tra xem có bao nhiêu dữ liệu được mô hình dự đoán đúng. Đây chính là chỉ số **accuracy - độ chính xác** như chúng ta sử dụng bên trên. Một cách toán học thì **precision** được biểu diển như sau:

Precision= Y(true)/Y(tổng)

**Recall**

Chỉ số này còn được gọi là **độ bao phủ** tức là xem xét xem mô hình tìm được có khả năng **tổng quát hóa** như thế nào. Từ hai yếu tố **độ chính xác** và **độ bao phủ** phía trên người ta đưa ra một chỉ số khác gọi là **F1-Score**

**Recall= [Y(true) giao Y(tổng)]/Y(true)**

**F1-Score**

Đây được gọi là một **trung bình điều hòa**(harmonic mean) của các tiêu chí Precision và Recall. Nó có xu hướng lấy giá trị gần với giá trị nào nhỏ hơn giữa 2 giá trị **Precision** và **Recall** và đồng thời nó có giá trị lớn nếu cả 2 giá trị **Precision** và **Recall** đều lớn. Chính vì thế **F1-Score** thể hiện được một cách khách quan hơn **performance** của một mô hình học máy.

F1- Score = 2x [(Precision.Recall)/(Precision+Recall)]

### 2.3.5. Áp dụng kỹ thuật, thuật toán vào bài toán :

**Bước 1** : Khai báo các thư viện

import pandas as pd

import numpy as np

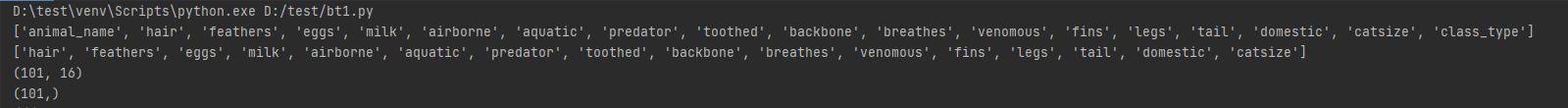
**Bước 2** : Xử lý dữ liệu

Xử lý các cột không sử dụng : Animal\_name

Chia X, Y với :

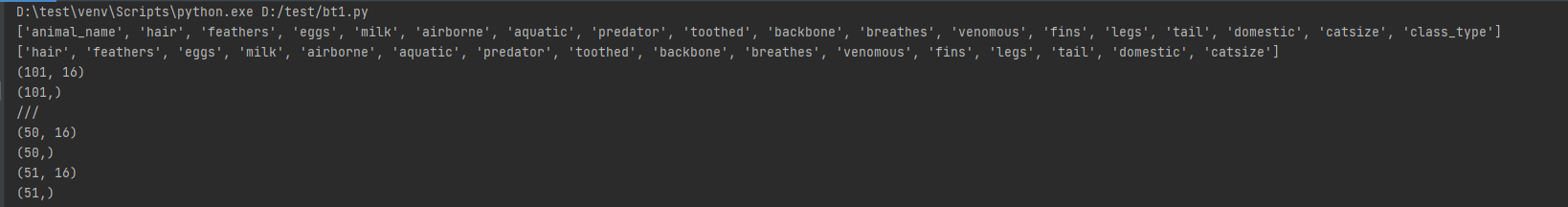
* X : Là các thuộc tính Hair đến Catsize
* Y : Là thuộc tính class\_type

df = pd.read\_csv('zoo.csv')  
df.head()  
features = list(df.columns)  
print(features)  
features.remove('class\_type')  
features.remove('animal\_name')  
  
print(features)  
X = df[features].values.astype(np.float32)  
Y = df.class\_type  
  
print(X.shape)  
print(Y.shape)  
print("///")



**Chia tập train với tập test : chia tập dữ liệu ra thành 2 phần với tỷ lệ 50:50**

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 = 0)  
  
print(X\_train.shape)  
print(Y\_train.shape)  
print(X\_test.shape)  
print(Y\_test.shape)



**Bước 3:** Áp dụng thuật toán Decision Tree giải quyết bài toán

from sklearn.tree import DecisionTreeClassifier  
  
model = DecisionTreeClassifier()  
model.fit(X\_train, Y\_train)  
print("training accuracy :", model.score(X\_train, Y\_train))  
print("testing accuracy :", model.score(X\_test, Y\_test))

#from sklearn.linear\_model import LogisticRegression

#model = LogisticRegression()

#model.fit(X\_train, Y\_train)

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

#print("training accuracy :", model.score(X\_train, Y\_train))

#print("testing accuracy :", model.score(X\_test, Y\_test))

**Bước 4**: Đánh giá

# Đánh giá

from sklearn.metrics import accuracy\_score

print("Accuracy Score:", accuracy\_score(Y\_test, y\_pred), "\n")

from sklearn.metrics import classification\_report

print(classification\_report(Y\_test,y\_pred))

from sklearn.metrics import confusion\_matrix

cnf\_matrix = confusion\_matrix(Y\_test, y\_pred)

print('Confusion matrix:')

print(cnf\_matrix)

# CHƯƠNG 3 : KẾT LUẬN

## 3.1. Bài toán hồi quy dự đoán giá xe ô tô Toyota Corolla cũ:

Sau khi thực hiện xong bài toán, ta thấy được tất cả dữ liệu đã được xử lý và đưa ra dự đoán khá đúng.

Bài toán có thể áp dụng nhiều phương pháp học máy để giải và cũng có thể cải tiến nó thành các bài toán dạng khác.

Trong tương lai, em sẽ cố gắng cải tiến và phát triển bài toán để có thể giải được nhiều tập dữ liệu phức tạp hơn nữa.

## 3.2. Bài toán phân lớp giá điện thoại:

Với bài toán, ta có thể phân chia các loại giá điện thoại một cách khá là chính xác.

Ta có thể kết hợp các kỹ thuật học máy lại với nhau để đưa ra kết quả tốt nhất có thể

## 3.3.Bài toán phân lớp thú:

Với bài toán ta có thể phân chia các loại động vật một cách khá là chính xác, với bài toán ta có thể chia các loại động vật thành các lớp động vật với các đặc tính gần giống nhau.

Ta có thể kết hợp các kỹ thuật học máy lại với nhau để đưa ra kết quả tốt nhất có thể

# DANH MỤC TÀI LIỆU THAM KHẢO

**[1]** https://github.com/datailluminations/PredictingToyotaPricesBlog

**[2]** https://vi.wikipedia.org/ (định nghĩa hồi quy tuyến tính)

**[3]** Lập trình Python cơ bản – Giảng viên Trần Duy Thanh – Đại học Quốc gia TP-Hồ Chí Minh

**[4]** https://classroom.google.com/w/NTMzMzMwNTk1MDVa/t/all