# 

# CHƯƠNG I : ĐỀ TÀI CHUNG CỦA NHÓM

## 1.1.Giới thiệu chung

***Tensorflow là gì*** – Với sự bùng nổ của lĩnh vực [Trí Tuệ Nhân Tạo – A.I.](https://topdev.vn/blog/20-tai-lieu-python-hoc-thiet-thuc-nhat-de-tro-thanh-lap-trinh-vien-chuyen-nghiep/) trong thập kỷ vừa qua, machine learning và deep learning rõ ràng cũng phát triển theo cùng. Và ở thời điểm hiện tại, [TensorFlow](https://www.tensorflow.org/) chính là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.

Các hàm được dựng sẵn trong thư viện cho từng bài toán cho phép TensorFlow xây dựng được nhiều neural network. Nó còn cho phép bạn tính toán song song trên nhiều máy tính khác nhau, thậm chí trên nhiều CPU, GPU trong cùng 1 máy hay tạo ra các dataflow graph – đồ thị luồng dữ liệu để dựng nên các model. Nếu bạn muốn chọn con đường sự nghiệp trong lĩnh vực A.I. này, nắm rõ những điều cơ bản của TensorFlow thực sự rất quan trọng.

Được viết bằng C++ và thao tác interface bằng Python nên phần performance của TensorFlow cực kỳ tốt. Đối tượng sử dụng nó cũng đa dạng không kém: từ các nhà nghiên cứu, nhà khoa học dữ liệu và dĩ nhiên không thể thiếu các lập trình viên.

Lịch sử ra đời TensorFlow

Vài năm trước, khi phải xử lý lượng dữ liệu khổng lồ, deep learning bắt đầu cho thấy hiệu năng vượt trội so với tất cả các thuật toán machine learning khác. Google sớm nhận ra tiềm năng này và nghĩ rằng họ nên sử dụng deep neural network để cải thiện các dịch vụ của mình, trong đó có:  
  
– Gmail.  
– Hình ảnh  
– Google search engine

Thế là họ dựng 1 framework có tên là TensorFlow để các nhà nghiên cứu cũng như lập trình viên có thể làm việc cùng nhau trên model A.I. 1 khi đã được phát triển và scale hoàn chỉnh, rất nhiều người đã có thể sử dụng được nó.

Ra mắt lần đầu vào cuối năm 2015, phiên bản TensorFlow ổn định cuối cùng cũng xuất hiện vào năm 2017. Là mã nguồn mở dưới sự cho phép của [Apache Open Source](https://www.apache.org/), giờ đây bạn có thể sử dụng, điều chỉnh và tái đóng góp phiên bản được điều chỉnh đó, đổi lại không cần phải trả bất cứ đồng nào cho Google.

### **Kiến trúc của TensorFlow**

Kiến trúc TensorFlow hoạt động được chia thành 3 phần:  
  
– Tiền xử lý dữ liệu  
– Dựng model  
– Train và ước tính model

### **Cách TensorFlow hoạt động**

TensorFlow cho phép các lập trình viên tạo ra dataflow graph, cấu trúc mô tả làm thế nào dữ liệu có thể di chuyển qua 1 biểu đồ, hay 1 sê-ri các node đang xử lý. Mỗi node trong đồ thị đại diện 1 operation toán học, và mỗi kết nối hay edge giữa các node là 1 mảng dữ liệu đa chiều, hay còn được gọi là ‘tensor’.   
  
TensorFlow cung cấp tất cả những điều này cho lập trình viên theo phương thức của ngôn ngữ Python. Vì Python khá dễ học và làm việc, ngoài ra còn cung cấp nhiều cách tiện lợi để ta hiểu được làm thế nào các high-level abstractions có thể kết hợp cùng nhau. Node và tensor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow bản thân chúng cũng là các ứng dụng Python.  
  
Các operation toán học thực sự thì không được thi hành bằng Python. Các thư viện biến đổi có sẵn thông qua TensorFlow được viết bằng các binary C++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các high-level abstraction lập trình để nối chúng lại với nhau.

*Các ứng dụng TensorFlow có thể chạy được trên hầu hết các đối tượng thông dụng: máy local, cluster trong các đám mây, thiết bị di động iOS và Android, CPU hay GPU. Nếu bạn dùng đám mây của riêng Google, bạn có thể chạy TensorFlow trên silicon của TensorFlow Processing Unit (TPU) tùy chỉnh từ Google để tăng tốc hiệu quả hơn nữa. Các model được tạo ra bởi TensorFlow, có thể deploy trên hầu hết các thiết bị nơi chúng được sử dụng để phục vụ các dự đoán.*

TensorFlow 2.0, được ra mắt vào tháng 10 năm 2019, cải tiến framework theo nhiều cách dựa trên phản hồi của người dùng, để dễ dàng và hiệu quả hơn khi làm việc cùng nó (ví dụ: bằng cách sử dụng các Keras API liên quan đơn giản cho việc train model). Train phân tán dễ chạy hơn nhờ vào API mới và sự hỗ trợ cho TensorFlow Lite cho phép triển khai các mô hình trên khá nhiều nền tảng khác nhau. Tuy nhiên, nếu đã viết code trên các phiên bản trước đó của TensorFlow thì bạn phải viết lại, đôi lúc 1 ít, đôi lúc cũng khá đáng kể, để tận dụng tối đa các tính năng mới của TensorFlow 2.0. 

#### **Lợi ích từ TensorFlow**

Lợi ích dễ thấy nhưng quan trọng nhất mà TensorFlow cung cấp cho việc lập trình machine learning chính là abstraction. Thay vì phải đối phó với những tình huống rườm rà từ việc thực hiện triển khai các thuật toán, hay tìm ra cách hợp lý để chuyển output của 1 chức năng sang input của 1 chức năng khác, giờ đây bạn có thể tập trung vào phần logic tổng thể của 1 ứng dụng hơn. TensorFlow sẽ chăm sóc phần còn lại thay cho bạn.

Ngoài ra TensorFlow còn ung cấp các tiện ích bổ sung cho các lập trình viên cần debug cũng như giúp bạn tự suy xét các ứng dụng TensorFlow. Chế độ [eager execution](https://www.tensorflow.org/programmers_guide/eager) cho phép bạn đánh giá và sửa đổi từng operation của biểu đồ 1 cách riêng biệt và minh bạch, thay vì phải dựng toàn bộ biểu đồ dưới dạng 1 đối tượng độc lập vốn khá mơ hồ hay phải đánh giá chung tổng thể. Cuối cùng, 1 tính năng khá độc đáo của TensorFlow là [TensorBoard](https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard). TensorBoard cho phép bạn quan sát 1 cách trực quan những gì TensorFlow đang làm.

TensorFlow còn có nhiều cải tiến từ sự hậu thuẫn từ các ekíp thương mại hạng A tại Google. Google không những tiếp lửa cho tiến độ nhanh chóng cho sự phát triển đằng sau dự án, mà còn tạo ra nhiều phục vụ độc đáo xung quanh TensorFlow để nó dễ dàng deploy và sử dụng: như silicon TPU mình đã nói ở trên để tăng tốc hiệu suất đám mây Google, 1 online hub cho việc chia sẻ các model được tạo với framework, sự hiện diện của [in-browser](https://js.tensorflow.org/) và [gần gũi với mobile](https://www.tensorflow.org/mobile/tflite/) của framework, và [nhiều hơn thế nữa…](https://siliconangle.com/2018/04/02/new-google-investments-tensorflow-now-ais-leading-development-framework/)

Lưu ý: Trong 1 số công việc training, vài chi tiết về việc triển khai của TensorFlow làm cho nó khó có thể quyết định được hoàn toàn kết quả training model . Đôi khi 1 model được train trên 1 hệ thống này sẽ có thay đổi 1 chút so với 1 model được train trên hệ thống khác, ngay cả khi chúng được cung cấp dữ liệu như nhau. Các nguyên nhân cho điều này cũng xê xích hay 1 số hành vi khi không được xác định khi sử dụng GPU. Điều này nói rằng, các vấn đề đó có thể giải quyết được, và đôi ngũ của TensorFlow cũng đang xem xét việc kiểm soát nhiều hơn để ảnh hưởng đến tính quyết định trong quy trình làm việc.

Giới thiệu các Component của TensorFlow

### Tensor

Tên của TensorFlow được đưa ra trực tiếp là nhờ vào framework cốt lõi của nó: Tensor. Trong TensorFlow, tất cả các tính toán đều liên quan tới các tensor. 1 tensor là 1 **vector** hay **ma trận** của n-chiều không gian đại diện cho tất cả loại dữ liệu. Tất cả giá trị trong 1 tensor chứa đựng loại dữ liệu giống hệt nhau với 1 **shape** đã biết (hoặc đã biết 1 phần). Shape của dữ liệu chính là chiều của ma trận hay mảng.  
  
1 tensor có thể được bắt nguồn từ dữ liệu input hay kết quả của 1 tính toán. Trong TensorFlow, tất cả các hoạt động được tiến hành bên trong 1 **graph** – biểu đồ. Biểu đồ là 1 tập hợp tính toán được diễn ra liên tiếp. Mỗi operation được gọi là 1 **op node**  (operation node) và được kết nối với nhau.  
  
Biểu đồ phát thảo các op và kết nối giữa các node. Tuy nhiên, nó không hiển thị các giá trị. Phần edge của các node chính là tensor, 1 cách để nhập operation với dữ liệu.

### Graph

TensorFlow sử dụng framework dạng biểu đồ. Biểu đồ tập hợp và mô tả tất cả các chuỗi tính toán được thực hiện trong quá trình training. Biểu đồ cũng mang rất nhiều lợi thế:  
  
– Nó được làm ra để chạy trên nhiều CPU hay GPU, ngay cả các hệ điều hành trên thiết bị điện thoại.  
– Tính di động của biểu đồ cho phép bảo toàn các tính toán để bạn sử dụng ngay hay sau đó. Biểu đồ có thể được lưu lại để thực thi trong tương lai.  
– Tất cả tính toán trong biểu đồ được thực hiện bằng cách kết nối các tensor lại với nhau. 1 **tensor** có 1 **node** và 1 **edge**. Node mang operation toán học và sản xuất các output ở đầu cuối. Các edge giải thích mối quan hệ input/output giữa các node.

## Danh sách các thuật toán nổi bật được hỗ trợ bởi TensorFlow

Tính tới thời điểm phiên bản ra mắt TensorFlow 1.10, nó đã sở hữu built-in API cho:

– **Linear regression:** tf.estimator.LinearRegressor  
– **Classification:** tf.estimator.LinearClassifier  
– **Deep learning classification:** tf.estimator.DNNClassifier  
– **Deep learning wipe and deep:** tf.estimator.DNNLinearCombinedClassifier  
– **Boosted tree regression:** tf.estimator.BoostedTreesRegressor  
– **Boosted tree classification:** tf.estimator.BoostedTreesClassifier

## 1.2.Hướng dẫn cài đặt Tensorflow

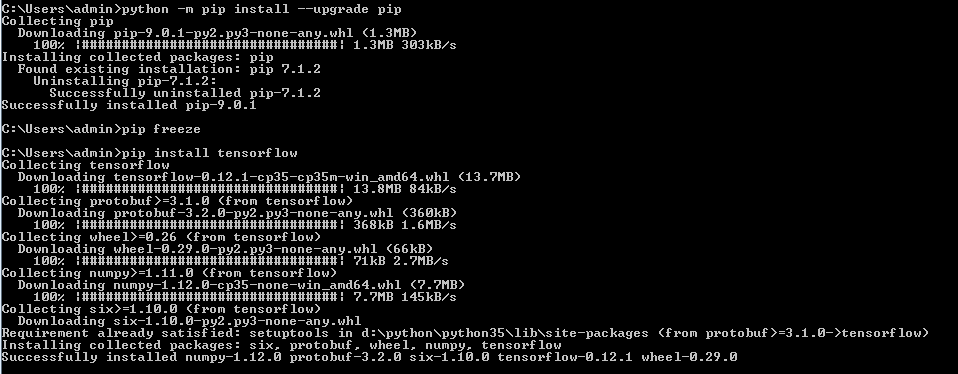
Ở đây, ta sẽ cài đặt Tensorflow với ngôn ngữ lập trình là Python. Trước khi cài đặt thì ta kiểm cần phải kiểm tra version của Python hiện tại.

Tensorflow 2.x trên Window OS với ngôn ngữ lập trình phổ biến là Python. Trên Window OS thì Tensorflow 2.x chỉ hỗ trợ cho Python 3.x từ 3.5.x đến 3.7.x và không hỗ trợ Python 2.x.

Tensorflow có 3 phiên bản:

* tensorflow —Phiên bản mới nhất – hiện là 2x
* tf-nightly —Phiên bản đang phát triển.
* tensorflow==1.15 —Chỉ định phiên bản mà mình muốn cài đặt. Ở đây là phiên bản TensorFlow 1.15.

Ta cài phiên bản mới nhất bằng : **pip install tensorflow**



Tiếp theo, ta nhập :**pip install tensorflow\_gpu**

Sau đó, quá trình cài đặt tensorflow được hoàn thành

## 1.3.Hướng dẫn sử dụng

### 1.3.1.Boosted tree Classifier

tf.estimator.BoostedTreesClassifier(

feature\_columns, n\_batches\_per\_layer, model\_dir=None, n\_classes=2,

weight\_column=None, label\_vocabulary=None, n\_trees=100, max\_depth=6,

learning\_rate=0.1, l1\_regularization=0.0, l2\_regularization=0.0,

tree\_complexity=0.0, min\_node\_weight=0.0, config=None,center\_bias=False,

pruning\_mode='none', quantile\_sketch\_epsilon=0.01,

train\_in\_memory=False

)

|  |  |
| --- | --- |
| Args |  |
| feature\_columns | Có thể lặp lại chứa tất cả các cột tính năng được mô hình sử dụng. Tất cả các mục trong tập hợp phải là thể hiện của các lớp có nguồn gốc từ FeatureColumn. |
| n\_batches\_per\_layer | số lượng lô để thu thập số liệu thống kê trên mỗi lớp. Tổng số lô là tổng số dữ liệu chia cho kích thước lô. |
| model\_dir | Thư mục để lưu các tham số mô hình, đồ thị, v.v. Điều này cũng có thể được sử dụng để tải các điểm kiểm tra từ thư mục vào một công cụ ước lượng để tiếp tục đào tạo mô hình đã lưu trước đó. |
| n\_classes | số lớp nhãn. Mặc định là phân loại nhị phân. |
| weight\_column | Một chuỗi hoặc một NumericColumn được tạo bằng cách xác định cột tính năng đại diện cho trọng số. Nó được sử dụng để giảm cân hoặc tăng cường các ví dụ trong quá trình đào tạo. Nó sẽ được nhân lên với sự mất mát của ví dụ. Nếu nó là một chuỗi, nó được sử dụng như một khóa để lấy tensor trọng lượng từ . Nếu nó là a , tensor thô được lấy bằng khóa , sau đó được áp dụng trên nó để lấy tensor trọng lượng. tf.fc\_old.numeric\_columnfeaturesNumericColumnweight\_column.keyweight\_column.normalizer\_fn |
| label\_vocabulary | Danh sách các chuỗi biểu thị các giá trị nhãn có thể có. Nếu được cung cấp, nhãn phải là kiểu chuỗi và có bất kỳ giá trị nào . Nếu nó không được cung cấp, điều đó có nghĩa là các nhãn đã được mã hóa dưới dạng số nguyên hoặc float bên trong for và được mã hóa dưới dạng giá trị số nguyên trong {0, 1, ..., n\_classes-1} cho . Ngoài ra, sẽ có lỗi nếu từ vựng không được cung cấp và nhãn là chuỗi. label\_vocabulary[0, 1]n\_classes=2n\_classes>2 |
| n\_trees | số cây sẽ được tạo. |
| max\_depth | độ sâu tối đa của cây để phát triển. |
| learning\_rate | tham số co rút sẽ được sử dụng khi một cây được thêm vào mô hình. |
| l1\_regularization | hệ số chính quy áp dụng cho trọng lượng tuyệt đối của lá cây. Đây là một giá trị cho mỗi phiên bản. Mặc định tốt là 1./(n\_batches\_per\_layer batch\_size). |
| l2\_regularization | hệ số chính quy áp dụng cho trọng lượng bình phương của lá cây. Đây là một giá trị cho mỗi phiên bản. Mặc định tốt là 1./(n\_batches\_per\_layerbatch\_size). |
| tree\_complexity | yếu tố chính quy để phạt cây có nhiều lá. Đây là một giá trị cho mỗi phiên bản. Mặc định tốt là 1./(n\_batches\_per\_layer\*batch\_size). |
| min\_node\_weight | min\_node\_weight: hessian tối thiểu mà một nút phải có để phân tách được xem xét. Đây là một giá trị cho mỗi phiên bản. Giá trị sẽ được so sánh với . sum(leaf\_hessian)/(batch\_size \* n\_batches\_per\_layer) |
| config | RunConfig đối tượng để cấu hình cài đặt thời gian chạy. |
| center\_bias | Liệu sự định tâm thiên lệch có cần xảy ra hay không. Căn giữa phân biệt đề cập đến nút đầu tiên trong cây đầu tiên trả về dự đoán phù hợp với phân phối nhãn ban đầu. Ví dụ, đối với các bài toán hồi quy, nút đầu tiên sẽ trả về giá trị trung bình của các nhãn. Đối với các bài toán phân loại nhị phân, nó sẽ trả về một logit cho xác suất trước của nhãn 1. |
| pruning\_mode | một trong những none, pre, pos tđể chỉ không cắt tỉa, cắt tỉa trước (không tách một nút nếu không đủ tăng được quan sát) và sau khi tỉa (xây dựng cây lên đến độ sâu tối đa và chi nhánh sau đó mận với tăng âm). Để cắt tỉa trước và sau, bạn PHẢI cung cấp . tree\_complexity >0 |
| quantile\_sketch\_epsilon | float giữa 0 và 1. Lỗi ràng buộc đối với tính toán lượng tử. Điều này chỉ được sử dụng cho các cột tính năng float và số lượng nhóm được tạo cho mỗi tính năng float là . 1/quantile\_sketch\_epsilon |
| train\_in\_memory | bool, khi đúng, nó giả sử tập dữ liệu nằm trong bộ nhớ, tức là, phải trả về toàn bộ tập dữ liệu dưới dạng một lô duy nhất, nên được đặt là 1, phải là 1 và phải là 0 trong . input\_fnn\_batches\_per\_layernum\_worker\_replicasnum\_ps\_replicastf.Estimator.RunConfig |

### 1.3.2.Boosted Trees Regressor

tf.estimator.BoostedTreesRegressor(

feature\_columns, n\_batches\_per\_layer, model\_dir=None, label\_dimension=1,

weight\_column=None, n\_trees=100, max\_depth=6, learning\_rate=0.1,

l1\_regularization=0.0, l2\_regularization=0.0, tree\_complexity=0.0,

min\_node\_weight=0.0, config=None, center\_bias=False,

pruning\_mode='none', quantile\_sketch\_epsilon=0.01,

train\_in\_memory=False

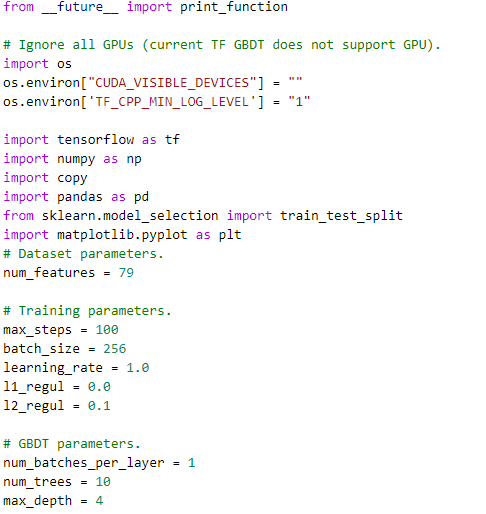
)

|  |  |
| --- | --- |
| Args |  |
| feature\_columns | Có thể lặp lại chứa tất cả các cột tính năng được mô hình sử dụng. Tất cả các mục trong tập hợp phải là thể hiện của các lớp có nguồn gốc từ FeatureColumn. |
| n\_batches\_per\_layer | số lượng lô để thu thập số liệu thống kê trên mỗi lớp. Tổng số lô là tổng số dữ liệu chia cho kích thước lô. |
| model\_dir | Thư mục để lưu các tham số mô hình, đồ thị, v.v. Điều này cũng có thể được sử dụng để tải các điểm kiểm tra từ thư mục vào một công cụ ước lượng để tiếp tục đào tạo mô hình đã lưu trước đó. |
| label\_dimension | Số lượng mục tiêu hồi quy cho mỗi ví dụ.. |
| weight\_column | Một chuỗi hoặc một NumericColumn được tạo bằng cách xác định cột tính năng đại diện cho trọng số. Nó được sử dụng để giảm cân hoặc tăng cường các ví dụ trong quá trình đào tạo. Nó sẽ được nhân lên với sự mất mát của ví dụ. Nếu nó là một chuỗi, nó được sử dụng như một khóa để lấy tensor trọng lượng từ . Nếu nó là a , tensor thô được lấy bằng khóa , sau đó được áp dụng trên nó để lấy tensor trọng lượng. tf.fc\_old.numeric\_columnfeaturesNumericColumnweight\_column.keyweight\_column.normalizer\_fn |
| n\_trees | số cây sẽ được tạo. |
| max\_depth | độ sâu tối đa của cây để phát triển. |
| learning\_rate | tham số co rút sẽ được sử dụng khi một cây được thêm vào mô hình. |
| l1\_regularization | hệ số chính quy áp dụng cho trọng lượng tuyệt đối của lá cây. Đây là một giá trị cho mỗi phiên bản. Mặc định tốt là 1./(n\_batches\_per\_layer batch\_size). |
| l2\_regularization | hệ số chính quy áp dụng cho trọng lượng bình phương của lá cây. Đây là một giá trị cho mỗi phiên bản. Mặc định tốt là 1./(n\_batches\_per\_layerbatch\_size). |
| tree\_complexity | yếu tố chính quy để phạt cây có nhiều lá. Đây là một giá trị cho mỗi phiên bản. Mặc định tốt là 1./(n\_batches\_per\_layer\*batch\_size). |
| min\_node\_weight | min\_node\_weight: hessian tối thiểu mà một nút phải có để phân tách được xem xét. Đây là một giá trị cho mỗi phiên bản. Giá trị sẽ được so sánh với . sum(leaf\_hessian)/(batch\_size \* n\_batches\_per\_layer) |
| config | RunConfig đối tượng để cấu hình cài đặt thời gian chạy. |
| center\_bias | Liệu sự định tâm thiên lệch có cần xảy ra hay không. Căn giữa phân biệt đề cập đến nút đầu tiên trong cây đầu tiên trả về dự đoán phù hợp với phân phối nhãn ban đầu. Ví dụ, đối với các bài toán hồi quy, nút đầu tiên sẽ trả về giá trị trung bình của các nhãn. Đối với các bài toán phân loại nhị phân, nó sẽ trả về một logit cho xác suất trước của nhãn 1. |
| pruning\_mode | một trong những none, pre, pos tđể chỉ không cắt tỉa, cắt tỉa trước (không tách một nút nếu không đủ tăng được quan sát) và sau khi tỉa (xây dựng cây lên đến độ sâu tối đa và chi nhánh sau đó mận với tăng âm). Để cắt tỉa trước và sau, bạn PHẢI cung cấp . tree\_complexity >0 |
| quantile\_sketch\_epsilon | float giữa 0 và 1. Lỗi ràng buộc đối với tính toán lượng tử. Điều này chỉ được sử dụng cho các cột tính năng float và số lượng nhóm được tạo cho mỗi tính năng float là . 1/quantile\_sketch\_epsilon |
| train\_in\_memory | bool, khi đúng, nó giả sử tập dữ liệu nằm trong bộ nhớ, tức là, phải trả về toàn bộ tập dữ liệu dưới dạng một lô duy nhất, nên được đặt là 1, phải là 1 và phải là 0 trong . input\_fnn\_batches\_per\_layernum\_worker\_replicasnum\_ps\_replicastf.Estimator.RunConfig |

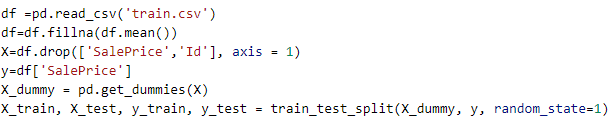
## 1.4. Ví dụ minh họa

Nguồn dữ liệu :<https://www.kaggle.com/alphaepsilon/housing-prices-dataset>

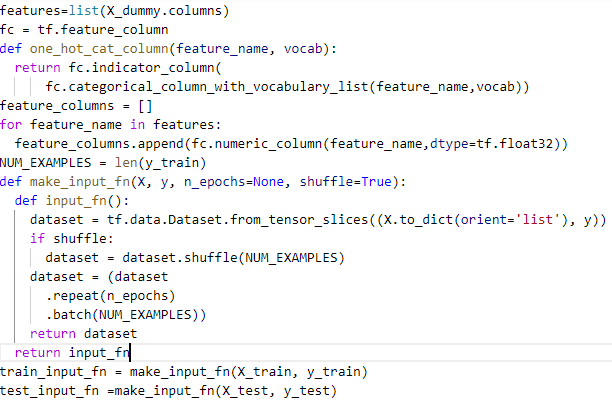
Đầu tiên ta thêm thư viện và chuẩn bị tham số:



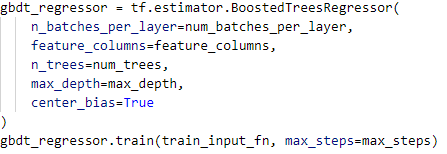
Tiếp theo chia tập train-test



Tạo các hàm dữ liệu đầu vào



Huấn luyện mô hình



Đánh giá mô hình



Kết quả:



# CHƯƠNG II.GIẢI BÀI TOÁN

## 2.1.Dự đoán giá nhà bằng kỹ thuật cây quyết định

### **2.1.1.Chuẩn bị dữ liệu**

#### **2.1.1.1.Thông tin cơ bản về dữ liệu**

-Nguồn dữ liệu :<https://www.kaggle.com/alphaepsilon/housing-prices-dataset>

-Dữ liệu dưới dạng file csv, gồm 81 thuộc tính của một ngôi nhà

MSSubClass: Identifies the type of dwelling involved in the sale.

20 1-STORY 1946 & NEWER ALL STYLES

30 1-STORY 1945 & OLDER

40 1-STORY W/FINISHED ATTIC ALL AGES

45 1-1/2 STORY - UNFINISHED ALL AGES

50 1-1/2 STORY FINISHED ALL AGES

60 2-STORY 1946 & NEWER

70 2-STORY 1945 & OLDER

75 2-1/2 STORY ALL AGES

80 SPLIT OR MULTI-LEVEL

85 SPLIT FOYER

90 DUPLEX - ALL STYLES AND AGES

120 1-STORY PUD (Planned Unit Development) - 1946 & NEWER

150 1-1/2 STORY PUD - ALL AGES

160 2-STORY PUD - 1946 & NEWER

180 PUD - MULTILEVEL - INCL SPLIT LEV/FOYER

190 2 FAMILY CONVERSION - ALL STYLES AND AGES

MSZoning: Identifies the general zoning classification of the sale.

A Agriculture

C Commercial

FV Floating Village Residential

I Industrial

RH Residential High Density

RL Residential Low Density

RP Residential Low Density Park

RM Residential Medium Density

LotFrontage: Linear feet of street connected to property

LotArea: Kích thước lô tích bằng mét vuông

Street: Type of road access to property

Grvl Gravel

Pave Paved

Alley: Type of alley access to property

Grvl Gravel

Pave Paved

NA No alley access

LotShape: General shape of property

Reg Regular

IR1 Slightly irregular

IR2 Moderately Irregular

IR3 Irregular

LandContour: Flatness of the property

Lvl Near Flat/Level

Bnk Banked - Quick and significant rise from street grade to building

HLS Hillside - Significant slope from side to side

Low Depression

Utilities: Type of utilities available

AllPub All public Utilities (E,G,W,& S)

NoSewr Electricity, Gas, and Water (Septic Tank)

NoSeWa Electricity and Gas Only

ELO Electricity only

LotConfig: Lot configuration

Inside Inside lot

Corner Corner lot

CulDSac Cul-de-sac

FR2 Frontage on 2 sides of property

FR3 Frontage on 3 sides of property

LandSlope: Slope of property

Gtl Gentle slope

Mod Moderate Slope

Sev Severe Slope

Neighborhood: Physical locations within Ames city limits

Blmngtn Bloomington Heights

Blueste Bluestem

BrDale Briardale

BrkSide Brookside

ClearCr Clear Creek

CollgCr College Creek

Crawfor Crawford

Edwards Edwards

Gilbert Gilbert

IDOTRR Iowa DOT and Rail Road

MeadowV Meadow Village

Mitchel Mitchell

Names North Ames

NoRidge Northridge

NPkVill Northpark Villa

NridgHt Northridge Heights

NWAmes Northwest Ames

OldTown Old Town

SWISU South & West of Iowa State University

Sawyer Sawyer

SawyerW Sawyer West

Somerst Somerset

StoneBr Stone Brook

Timber Timberland

Veenker Veenker

Condition1: Proximity to various conditions

Artery Adjacent to arterial street

Feedr Adjacent to feeder street

Norm Normal

RRNn Within 200' of North-South Railroad

RRAn Adjacent to North-South Railroad

PosN Near positive off-site feature--park, greenbelt, etc.

PosA Adjacent to postive off-site feature

RRNe Within 200' of East-West Railroad

RRAe Adjacent to East-West Railroad

Condition2: Proximity to various conditions (if more than one is present)

Artery Adjacent to arterial street

Feedr Adjacent to feeder street

Norm Normal

RRNn Within 200' of North-South Railroad

RRAn Adjacent to North-South Railroad

PosN Near positive off-site feature--park, greenbelt, etc.

PosA Adjacent to postive off-site feature

RRNe Within 200' of East-West Railroad

RRAe Adjacent to East-West Railroad

BldgType: Type of dwelling

1Fam Single-family Detached

2FmCon Two-family Conversion; originally built as one-family dwelling

Duplx Duplex

TwnhsE Townhouse End Unit

TwnhsI Townhouse Inside Unit

HouseStyle: Style of dwelling

1Story One story

1.5Fin One and one-half story: 2nd level finished

1.5Unf One and one-half story: 2nd level unfinished

2Story Two story

2.5Fin Two and one-half story: 2nd level finished

2.5Unf Two and one-half story: 2nd level unfinished

SFoyer Split Foyer

SLvl Split Level

OverallQual: Rates the overall material and finish of the house,OverallCond: Rates the overall condition of the house

10 Very Excellent

9 Excellent

8 Very Good

7 Good

6 Above Average

5 Average

4 Below Average

3 Fair

2 Poor

1 Very Poor

YearBuilt: Năm xây dựng

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

Flat Flat

Gable Gable

Gambrel Gabrel (Barn)

Hip Hip

Mansard Mansard

Shed Shed

RoofMatl: Roof material

ClyTile Clay or Tile

CompShg Standard (Composite) Shingle

Membran Membrane

Metal Metal

Roll Roll

Tar&Grv Gravel & Tar

WdShake Wood Shakes

WdShngl Wood Shingles

Exterior1st: Exterior covering on house

AsbShng Asbestos Shingles

AsphShn Asphalt Shingles

BrkComm Brick Common

BrkFace Brick Face

CBlock Cinder Block

CemntBd Cement Board

HdBoard Hard Board

ImStucc Imitation Stucco

MetalSd Metal Siding

Other Other

Plywood Plywood

PreCast PreCast

Stone Stone

Stucco Stucco

VinylSd Vinyl Siding

Wd Sdng Wood Siding

WdShing Wood Shingles

Exterior2nd: Exterior covering on house (if more than one material)

AsbShng Asbestos Shingles

AsphShn Asphalt Shingles

BrkComm Brick Common

BrkFace Brick Face

CBlock Cinder Block

CemntBd Cement Board

HdBoard Hard Board

ImStucc Imitation Stucco

MetalSd Metal Siding

Other Other

Plywood Plywood

PreCast PreCast

Stone Stone

Stucco Stucco

VinylSd Vinyl Siding

Wd Sdng Wood Siding

WdShing Wood Shingles

MasVnrType: Masonry veneer type

BrkCmn Brick Common

BrkFace Brick Face

CBlock Cinder Block

None None

Stone Stone

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior,ExterCond: Evaluates the present condition of the material on the exterior

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

Po Poor

Foundation: Type of foundation

BrkTil Brick & Tile

CBlock Cinder Block

PConc Poured Contrete

Slab Slab

Stone Stone

Wood Wood

BsmtQual: Evaluates the height of the basement

Ex Excellent (100+ inches)

Gd Good (90-99 inches)

TA Typical (80-89 inches)

Fa Fair (70-79 inches)

Po Poor (<70 inches

NA No Basement

BsmtCond: Evaluates the general condition of the basement

Ex Excellent

Gd Good

TA Typical - slight dampness allowed

Fa Fair - dampness or some cracking or settling

Po Poor - Severe cracking, settling, or wetness

NA No Basement

BsmtExposure: Refers to walkout or garden level walls

Gd Good Exposure

Av Average Exposure (split levels or foyers typically score average or above)

Mn Mimimum Exposure

No No Exposure

NA No Basement

BsmtFinType1: Rating of basement finished area,BsmtFinType2: Rating of basement finished area (if multiple types)

GLQ Good Living Quarters

ALQ Average Living Quarters

BLQ Below Average Living Quarters

Rec Average Rec Room

LwQ Low Quality

Unf Unfinshed

NA No Basement

BsmtFinSF1: Type 1 finished square feet

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

Floor Floor Furnace

GasA Gas forced warm air furnace

GasW Gas hot water or steam heat

Grav Gravity furnace

OthW Hot water or steam heat other than gas

Wall Wall furnace

HeatingQC: Heating quality and condition

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

Po Poor

CentralAir: Central air conditioning

N No

Y Yes

Electrical: Electrical system

SBrkr Standard Circuit Breakers & Romex

FuseA Fuse Box over 60 AMP and all Romex wiring (Average)

FuseF 60 AMP Fuse Box and mostly Romex wiring (Fair)

FuseP 60 AMP Fuse Box and mostly knob & tube wiring (poor)

Mix Mixed

1stFlrSF: Diện thích tầng 1

2ndFlrSF: Diện thích tầng 2

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

Typ Typical Functionality

Min1 Minor Deductions 1

Min2 Minor Deductions 2

Mod Moderate Deductions

Maj1 Major Deductions 1

Maj2 Major Deductions 2

Sev Severely Damaged

Sal Salvage only

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

Ex Excellent - Exceptional Masonry Fireplace

Gd Good - Masonry Fireplace in main level

TA Average - Prefabricated Fireplace in main living area or Masonry Fireplace in basement

Fa Fair - Prefabricated Fireplace in basement

Po Poor - Ben Franklin Stove

NA No Fireplace

GarageType: Garage location

2Types More than one type of garage

Attchd Attached to home

Basment Basement Garage

BuiltIn Built-In (Garage part of house - typically has room above garage)

CarPort Car Port

Detchd Detached from home

NA No Garage

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

Fin Finished

RFn Rough Finished

Unf Unfinished

NA No Garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

NA No Garage

GarageCond: Garage condition

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

NA No Garage

PavedDrive: Paved driveway

Y Paved

P Partial Pavement

N Dirt/Gravel

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

NA No Pool

Fence: Fence quality

GdPrv Good Privacy

MnPrv Minimum Privacy

GdWo Good Wood

MnWw Minimum Wood/Wire

NA No Fence

MiscFeature: Miscellaneous feature not covered in other categories

Elev Elevator

Gar2 2nd Garage (if not described in garage section)

Othr Other

Shed Shed (over 100 SF)

TenC Tennis Court

NA None

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

WD Warranty Deed - Conventional

CWD Warranty Deed - Cash

VWD Warranty Deed - VA Loan

New Home just constructed and sold

COD Court Officer Deed/Estate

Con Contract 15% Down payment regular terms

ConLw Contract Low Down payment and low interest

ConLI Contract Low Interest

ConLD Contract Low Down

Oth Other

SaleCondition: Condition of sale

Normal Normal Sale

Abnorml Abnormal Sale - trade, foreclosure, short sale

AdjLand Adjoining Land Purchase

Alloca Allocation - two linked properties with separate deeds, typically condo with a garage unit

Family Sale between family members

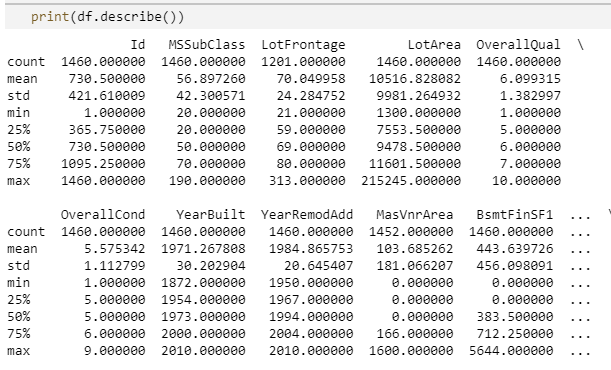
Partial Home was not completed when last assessed (associated with New Homes)

#### **2.1.1.2.Thông tin chi tiết**

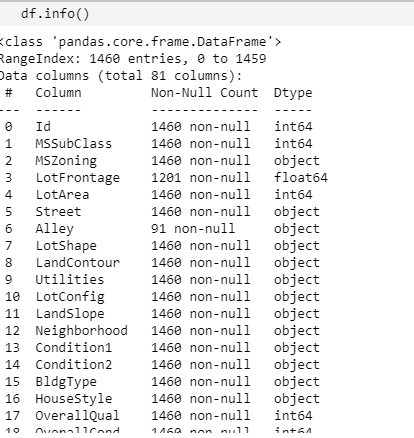
-Ở nguồn trên ta đã chia thành 2 file csv, train.csv chứa 81 thuộc tính (SalePrice) gồm 1460 mẫu và test.csv chứa 80 thuộc tính (không SalePrice)

-Ta sẽ sử dụng train.csv trong chương trình tiếp theo

-Ta sẽ thông kê số lượng ,max,min ,mean của từng cột



-Tiếp theo, thông tin về kiểu dữ liệu cũng số lượng dữ liệu NON NAN của từng cột:



=>dtypes: float64(3), int64(35), object(43)

-Ta chia tập train.csv thành 2 phần gồm

X (1460,79) trừ Id,SalePrices

y (1460, 01) SalePrices

-Sau khi kiểm tra các giá NAN ,ta khử bằng tính trung bình: **df=df.fillna(df.mean())**

-Ta kiểu dữ liệu như object thành số sử dụng Dummy :

**X\_dummy = pd.get\_dummies(X)**

Cuối cùng ta chia X,y thành các (X,y) sử dụng của train /test

X\_train (1095, 288)

y\_train (1095,1)

X\_test (365, 288)

y\_test (365,1)

### **2.1.2.Huấn luyện**

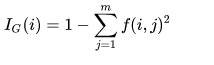
Mục tiêu: Dự đoán giá nhà dựa trên bộ dữ liệu được phân tích ở trên

Cây quyết định:

Gini impurity

Dùng trong thuật toán CART (Classification and Regression Trees). Nó dựa vào việc bình phương các xác suất thành viên cho mỗi thể loại đích trong nút. Giá trị của nó tiến đến cực tiểu (bằng 0) khi mọi trường hợp trong nút rơi vào một thể loại đích duy nhất.

Giả sử *y* nhận các giá trị trong {1, 2,..., *m*} và gọi *f(i,j)* là tần suất của giá trị *j* trong nút *i*. Nghĩa là *f(i,j)* là tỷ lệ các bản ghi với *y*=*j* được xếp vào nhóm *i*.



Hàm số Entropy

Cho một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau x1,x2,....,xn. Giả sử xác suất của x có thể giá trị này la pi=p(x=xi)

Ký hiệu p=(p1,p2,...pn)

Entropy của phân phối là :

H(p)=-

**Information Gain**

Bài toán của ta trở thành, tại mỗi tầng của cây, cần chọn thuộc tính nào để độ giảm Entropy là thấp nhất.

Người ta có khái niệm Information Gain được tính bằng

Gain(S,f) = H(S) - H(f,S)

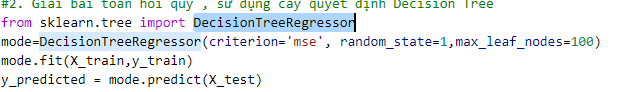
trong đó:

H(S) là Entropy tổng của toàn bộ tập data set SS.

H(f,S) là Entropy được tính trên thuộc tính ff

Do H(S) là không đổi với mỗi tầng, ta chọn thuộc tính f có Entropy nhỏ nhất để thu được Gain(S,f) lớn nhất.

Ta sẽ đi đoán giá trị với tệp dữ liệu đã được xử lý ở trên sử dụng kỹ thuật cây quyết định:



### **2.1.3.Đánh giá**

Ta sử dụng các công thức đánh giá

**MSE=**

**MAE=**

Trong đó:là giá trị ban đầu trong tệp dữ liệu

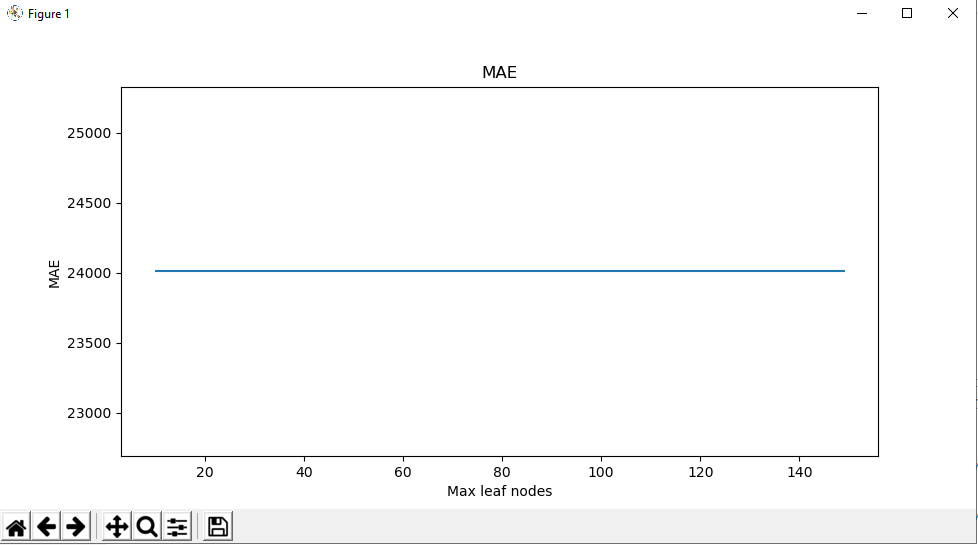
là giá trị dự đoán được

n là số mẫu kiểm thử

Kết quả sau khi chạy với max\_leaf\_nodes=100



Sau đây max\_leaf\_nodes chạy từ 10 đến 150 :



Nhận xét: Nhìn chung MAE có giá trị không thay đổi quá nhiều theo chiều tăng của Max leaf nodes và có xu hướng giảm dần.

### **2.1.4.Sử dụng thư viện Tensorflow**

Ở đâu K là số cây, f là một chức năng trong không gian chức năng F và F là tập hợp của tất cả các CARTS có thể có.

là biến mục tiêu,q là một hàm gán mỗi điểm dữ liệu cho lá tương ứng

Hàm mục tiêu được tối ưu hóa được đưa ra bởi:

***obj(θ)=+***

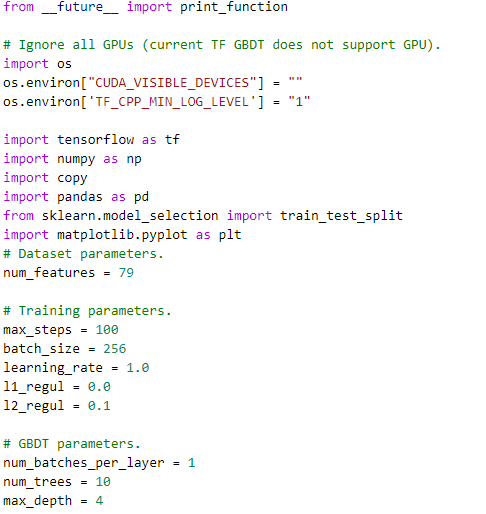
***={ i|q()=j }***

**= và =**

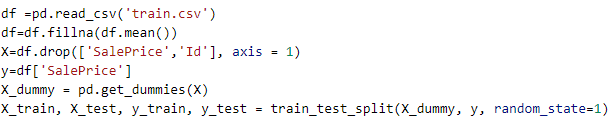
**= và =**

**Gain=[++]−γ**

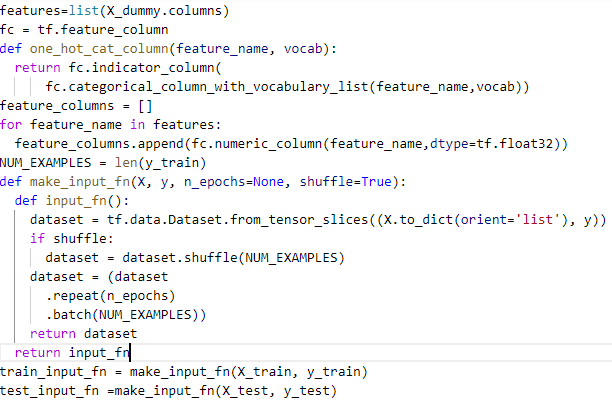
Đầu tiên ta thêm thư viện và chuẩn bị tham số:



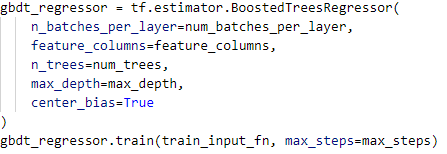
Tiếp theo chia tập train-test



Tạo các hàm dữ liệu đầu vào



Huấn luyện mô hình



Đánh giá mô hình



Kết quả:

## 

## 2.2.Phân loại hoa sử dụng mạng Nơ-rơn

Tập dữ liệu này chứa khoảng 3.700 bức ảnh về hoa. Tập dữ liệu chia làm 5 thư mục con, mỗi thư mục một loại hoa bao gồm :

* daisy
* dandelion
* roses
* sunflower
* tulips

Với dữ liệu này tôi sử dụng bài toán phân lớp cùng với thư viện TensorFlow và mạng nơ-ron để xử lý dữ liệu ảnh, để từ đó đầu vào là ảnh chúng ta có thể dự đoán được nó thuộc loại hoa nào. Bên cạnh đó chúng ta đánh giá mô hình qua Accuracy.

### **2.2.1.Tạo tập dữ liệu**

Xác định một số tham số cho bộ nạp:

batch\_size = 32

img\_height = 180

img\_width = 180

Bạn nên sử dụng phân tách xác thực khi phát triển mô hình của mình. Hãy sử dụng 80% hình ảnh để đào tạo và 20% để xác thực.

train\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory(

data\_dir,

validation\_split=0.2,

subset="training",

seed=123,

image\_size=(img\_height, img\_width),

batch\_size=batch\_size)

Kết quả:

Found 3670 files belonging to 5 classes.

Using 2936 files for training.

val\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory(

data\_dir,

validation\_split=0.2,

subset="validation",

seed=123,

image\_size=(img\_height, img\_width),

batch\_size=batch\_size)

Kết quả:

Found 3670 files belonging to 5 classes.

Using 734 files for validation.

Bạn có thể tìm tên lớp trong thuộc tính class\_names trên các tập dữ liệu này. Chúng tương ứng với tên thư mục theo thứ tự bảng chữ cái.

class\_names = train\_ds.class\_names

print(class\_names)

Kết quả:

['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']

### **2.2.2.Chuẩn hóa dữ liệu**

Các giá trị kênh RGB nằm trong phạm vi [0, 255] . Điều này không lý tưởng cho mạng nơ-ron; nói chung, bạn nên tìm cách giảm giá trị đầu vào của mình. Tại đây, bạn sẽ chuẩn hóa các giá trị trong phạm vi [0, 1] bằng cách sử dụng lớp Thay đổi tỷ lệ.

normalization\_layer =

layers.experimental.preprocessing.Rescaling(1./255)

### **2.2.3.Tạo mô hình**

Mô hình bao gồm ba khối chập với một lớp nhóm tối đa trong mỗi khối. Có một lớp được kết nối đầy đủ với 128 đơn vị trên cùng được kích hoạt bằng chức năng kích hoạt relu . Mô hình này chưa được điều chỉnh để có độ chính xác cao, mục tiêu của hướng dẫn này là thể hiện một cách tiếp cận tiêu chuẩn.

num\_classes = 5

model = Sequential([

layers.experimental.preprocessing.Rescaling(1./255, input\_shape=(img\_height, img\_width, 3)),

layers.Conv2D(16, 3, padding='same', activation='relu'),

layers.MaxPooling2D(),

layers.Conv2D(32, 3, padding='same', activation='relu'),

layers.MaxPooling2D(),

layers.Conv2D(64, 3, padding='same', activation='relu'),

layers.MaxPooling2D(),

layers.Flatten(),

layers.Dense(128, activation='relu'),

layers.Dense(num\_classes)

])

### **2.2.4.Đào tạo mô hình**

epochs=10

history = model.fit(

train\_ds,

validation\_data=val\_ds,

epochs=epochs

)

Kết quả:

Epoch 1/10

92/92 [==============================] - 3s 27ms/step - loss: 1.3816 - accuracy: 0.4077 - val\_loss: 1.0884 - val\_accuracy: 0.5518

Epoch 2/10

92/92 [==============================] - 1s 10ms/step - loss: 1.0222 - accuracy: 0.6039 - val\_loss: 0.9661 - val\_accuracy: 0.5872

Epoch 3/10

92/92 [==============================] - 1s 10ms/step - loss: 0.8417 - accuracy: 0.6778 - val\_loss: 0.8763 - val\_accuracy: 0.6417

Epoch 4/10

92/92 [==============================] - 1s 10ms/step - loss: 0.6234 - accuracy: 0.7691 - val\_loss: 0.8961 - val\_accuracy: 0.6444

Epoch 5/10

92/92 [==============================] - 1s 10ms/step - loss: 0.4066 - accuracy: 0.8580 - val\_loss: 0.9164 - val\_accuracy: 0.6717

Epoch 6/10

92/92 [==============================] - 1s 10ms/step - loss: 0.2379 - accuracy: 0.9234 - val\_loss: 1.1665 - val\_accuracy: 0.6417

Epoch 7/10

92/92 [==============================] - 1s 10ms/step - loss: 0.1372 - accuracy: 0.9571 - val\_loss: 1.3581 - val\_accuracy: 0.6621

Epoch 8/10

92/92 [==============================] - 1s 10ms/step - loss: 0.0802 - accuracy: 0.9789 - val\_loss: 1.5392 - val\_accuracy: 0.6526

Epoch 9/10

92/92 [==============================] - 1s 10ms/step - loss: 0.0405 - accuracy: 0.9918 - val\_loss: 1.7072 - val\_accuracy: 0.6730

Epoch 10/10

92/92 [==============================] - 1s 10ms/step - loss: 0.0311 - accuracy: 0.9925 - val\_loss: 1.7984 - val\_accuracy: 0.6458

### **2.2.5.Kết quả đào tạo mô hình**

Tạo các biểu đồ tổn thất và độ chính xác trên các tập huấn luyện và xác nhận.

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs\_range = range(epochs)

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

plt.subplot(1, 2, 1)

plt.plot(epochs\_range, acc, label='Training Accuracy')

plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(epochs\_range, loss, label='Training Loss')

plt.plot(epochs\_range, val\_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.show()

Kết quả:



Sau khi chạy 10 epoch training độ chính xác khá cao lên đến 99% độ mất mát chỉ có 0,3%, nhưng validation thì ngược lại khá thấp.

## 2.3.Phân loại nấm sử dụng mạng Nơron

### **2.3.1.Chuẩn bị dữ liệu**

#### **2.3.1.1 Thông tin dữ liệu**

-Nguồn dữ liệu:[<https://www.kaggle.com/uciml/mushroom-classification>]

Dữ liệu dưới dạng CSV,gồm 23 thuộc tính của nấm

DATA mushroom.csv

Thông tin thuộc tính: (classes: edible=e, poisonous=p)

-cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s

-cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s

-cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y

-bruises: bruises=t,no=f

-odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s

-gill-attachment: attached=a,descending=d,free=f,notched=n

-gill-spacing: close=c,crowded=w,distant=d

-gill-size: broad=b,narrow=n

-gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, -green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y

-stalk-shape: enlarging=e,tapering=t

-stalk-root: bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=?

-stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s

-stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s

-stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y

-stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y

-veil-type: partial=p,universal=u

-veil-color: brown=n,orange=o,white=w,yellow=y

-ring-number: none=n,one=o,two=t

-ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z

-spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y

-population: abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=yl

-habitat: grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d

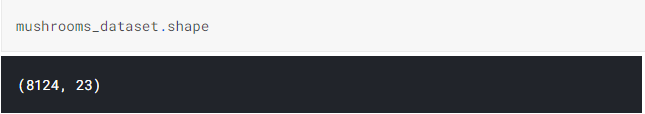
#### **2.3.1.2.Thông tin chi tiết**

Ta sẽ thống kê về tập dữ liệu trong chương trình



Ở đây ta sẽ liệt kê ra 5 dòng với 23 thuộc tính cần phân loại

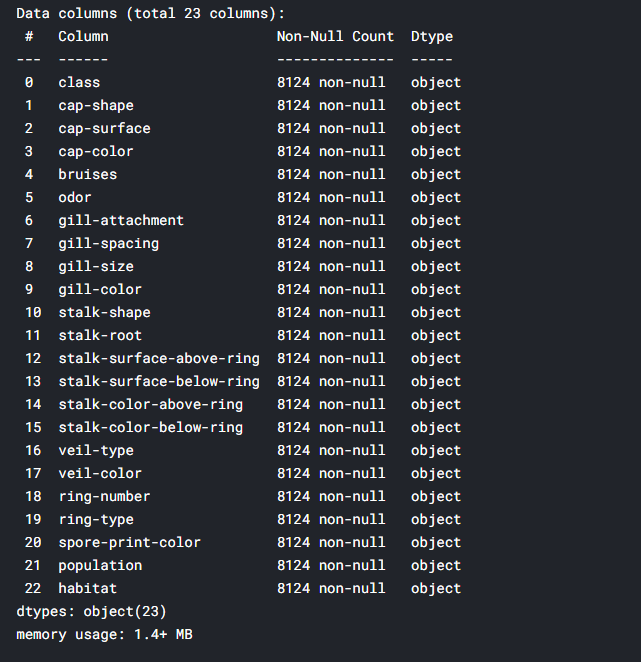
-Tiếp theo ta sẽ xem số lượng mẫu



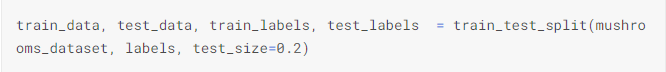
Ở đây chúng ta có 8124 dòng và 23 cột thuộc tính

-Tiếp theo ta sẽ xem thông tin thuộc tính





-Tiếp theo ta sẽ tách tập dữ liệu ra 80% train và 20% test



-Building model

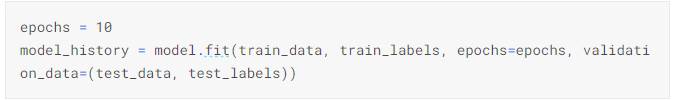


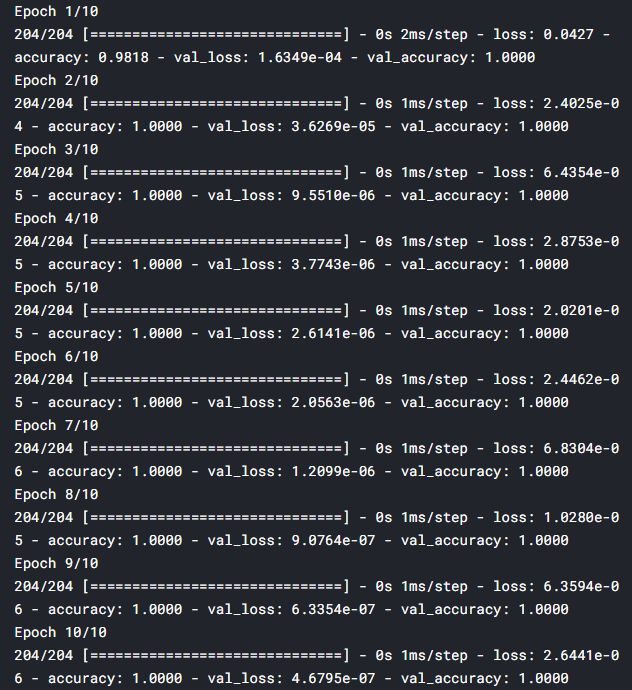
### **2.3.2.Huấn luyện**

Mục tiêu:Phân lớp nấm dựa vào thuộc tính nhận dạng

Ở đây chúng ta dùng mạng nơ-ron để dự đoán

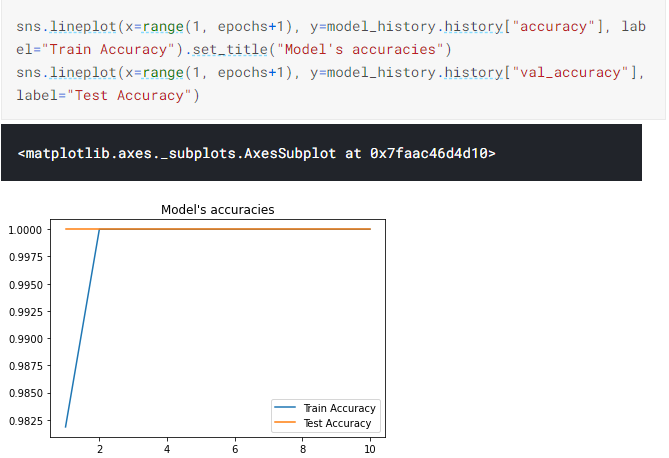
-Model training



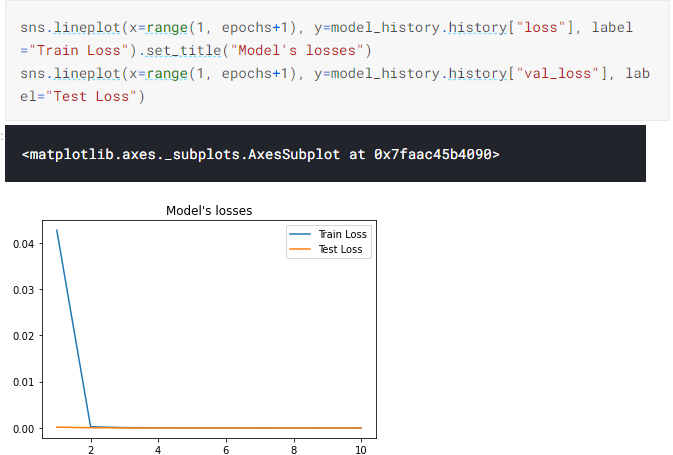


### **2.3.3.Đánh giá**

-Độ chính xác:



-Sai số:



Ở đây độ chính xác có thể lên tới 100% và sai số chỉ 0,0001% cho thấy mô hình này hoàn toàn phù hợp với bài toán phân lớp sử dụng mô hình mạng nơ-ron.

# CHƯƠNG III.KẾT LUẬN

## 3.1.Bài toán dự đoán giá nhà bằng cây quyết định

-Bài toán sử dụng kỹ thuật cây hồi quy (cây quyết định) với tập dữ liệu gồm 81 thuộc tính và 1460 mẫu dữ liệu

-Sau khi huấn luyện, bằng cách cho biến “max leaf nodes” chạy từ 10 đến 150 ,ta được tập giá trị MAE(Trung bình sai số tuyệt đối) có xu hướng giảm nhưng không quá nhiều

-Tiếp theo,bài toán này còn được áp dụng kỹ thuật cây quyết định trong thư viện Tensorflow ,ta được một kết quả tối ưu hơn qua việc so sánh MSE(Trung bình sai số bình phương)

-Trong tương lai, tập dữ liệu sẽ được bổ sung và hoàn thiện hơn để phù hợp với thực tế xu hướng và được sử dụng các kỹ thuật mới hiệu quả hơn thì bài toán sẽ được giải quyết một cách hiệu quả hơn.

## 3.2.Bài toán phân loại hoa sử dụng mạng Nơron

-Bài toán sử dụng bài toán phân lớp với hơn 3700 dữ liệu ảnh.

-Sau khi training độ chính xác khá cao lên đến 99% độ mất mát chỉ có 0,3%, nhưng validation thì ngược lại khá thấp. Chúng ta cần sử dụng thêm kỹ thuật học tăng cường để độ chính xác ở mức cao hơn.

-Trong tương lai, tập dữ liệu sẽ được bổ sung và hoàn thiện hơn để phù hợp với thực tế xu hướng và được sử dụng các kỹ thuật mới hiệu quả hơn thì bài toán sẽ được giải quyết một cách hiệu quả hơn.