**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**



**ĐỒ ÁN III – MACHINE LEARNING**

**ĐỀ TÀI:**

**TÌM HIỂU VỀ DECISION TREE**

Giảng viên hướng dẫn: ThS. Từ Tuyết Hồng

Sinh viên thực hiện: 16149048 – Lê Tiến Hòa

17110123 – Lê Kim Đỉnh

***Thành phố Hồ Chí Minh, tháng 12 năm 2020.***

**MỤC LỤC**

[LỜI MỞ ĐẦU 3](#_Toc57110837)

[I. Decision Tree 6](#_Toc57110838)

[1. Giới thiệu về Decision Tree 6](#_Toc57110839)

[2. Entropy 8](#_Toc57110840)

[3. Information Gain 8](#_Toc57110841)

[4. Gini Impurity (Gini Index) 9](#_Toc57110842)

[5. Xây dựng Decision Tree Classification 9](#_Toc57110843)

[II. Decision Tree trên Python 10](#_Toc57110844)

[1. Dataset 11](#_Toc57110845)

[2. Decision Tree Classification 13](#_Toc57110846)

[3. Code thuật toán: 14](#_Toc57110847)

[4. Kết quả. 19](#_Toc57110848)

[III. Tài liệu tham khảo 23](#_Toc57110849)

**DANH MỤC BẢNG**

[Figure 1:Mô hình Decision Tree 6](#_Toc57112831)

[Figure 2:Decision Tree with Iris Dataset using Sklearn 7](#_Toc57112832)

[Figure 3:Milk Root-node và Child-node 10](#_Toc57112833)

[Figure 4 Đồ thị về Khoảng G3 của df. 19](#_Toc57112834)

[Figure 5 Đồ thị Khoảng G3 của train\_set. 20](#_Toc57112835)

[Figure 6 Đồ thị Khoảng G3 của test\_set. 21](#_Toc57112836)

[Figure 7 Classification Report. 21](#_Toc57112837)

[Figure 8 Sơ đồ của Decision Tree với độ sâu = 2. 22](#_Toc57112838)

[Figure 9 Sơ đồ Decision Tree. 22](#_Toc57112839)

# **LỜI MỞ ĐẦU**

Scikit-learn, Tensorflow từ lâu trở thành thư viện phổ biến trong lĩnh vực máy học và nghiên cứu dữ liệu. Vì vậy, việc tìm hiểu và áp dụng thư viện này cho các thuật toán máy học đã học là điều cần thiết cho sinh viên Công Nghệ Thông Tin ở các trường Đại Học. Dù Scikit-learn cung cấp đầy đủ công cụ, việc hiểu ý nghĩa từng đoạn mã là cần thiết và cũng là mục tiêu quan trọng nhất khi hoàn tất đề tài. Chỉ khi chúng ta thực sự hiểu thì mới hoàn tất được những yêu cầu nâng cao hơn sau này. Ngoài ra, nhóm cũng đã tìm hiểu thêm các kiến thức bên ngoài về tối ưu hóa bộ tham chiếu, giúp kết quả đạt được tốt hơn.

Trong quá trình thực hiện đồ án một, nhóm thực hiện đã được nhận nhiều giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy và bạn bè. Nhóm chúng em xin gửi lời cảm ơn chân thành đến cô Từ Tuyết Hồng, giảng viên phụ trách hướng dẫn người đã tận tình hướng dẫn, chỉ bảo nhóm em trong suốt quá trình làm đồ án. Nhóm đã cố gắng thực hiện đồ án làm sao để được tốt nhất nhưng vì kiến thức còn hạn hẹp và thời gian có hạn nên nhóm chưa thể tối ưu hết các chức năng trong đồ án cũng như không thể tránh khỏi các thiếu sót. Vì vậy rất mong nhận được sự góp ý của thầy để đồ án của nhóm có thể hoàn thiện hơn. Một lần nữa, nhóm thực hiện chân thành cảm ơn cô Từ Tuyết Hồng và các bạn.

**Cam kết không đạo văn**

Đạo văn là một hành động được coi là tối kỵ trong học tập và nghiên cứu, vì nó để lại những hậu quả khá nặng nề cho cả người đạo văn và xã hội. Nó có thể hiểu theo từng phần ý rút gọn như sau:

* Ăn cắp và hình thành những ý tưởng hay ngôn từ mới khởi nguồn từ ý tưởng của ai đó.
* Sử dụng sản phẩm của một ai đó mà không công bố nguồn.
* Giới thiệu một ý tưởng hay sản phẩm mới được chuyển hóa từ một nguồn đã có từ trước.

Để tránh việc đạo văn thì những việc chúng ta cần làm là:

* Hiểu rõ khái niệm về đạo văn:

+ đôi khi chúng ta hiểu rằng đạo văn là xấu và cần tránh những nhiều lúc vô tình chúng ta vi phạm.

+ tránh việc thuê người người khác viết bài luận, cố tình biến ý tưởng, bài của người khác thành của mình, tải và sử dụng những bài báo cáo trên mạng mà chưa xin phép bản quyền.

* Nắm rõ quy định về trích nguồn.

+ chúng ta cần hình thành những thói quen trích nguồn khi lấy thông tin, dữ liệu, biểu đồ hình ảnh minh họa, … của người khác. Ngoài ra cần nắm vững các phương pháp và quy định về trích nguồn dẫn.

* Học cách diễn đạt ý tưởng theo nhiều cách khác nhau như việc thay đổi cấu trúc câu văn, dùng từ đồng nghĩa, thay đổi dạng của câu văn.

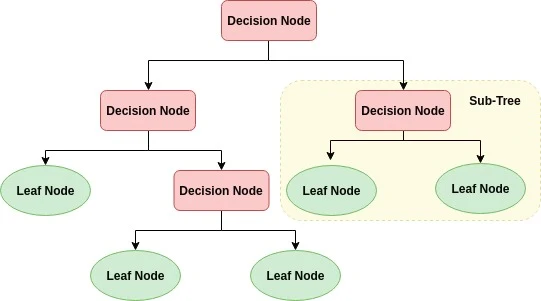
Chúng em xin cam đoan đồ án này do các thành viên nhóm thực hiện. Nếu phạm lỗi đạo văn (như: sử dụng tài liệu, code của người khác mà không ghi rõ nguồn, sao chép trên 30% báo cáo), chúng em xin chịu hoàn toàn trách nhiệm.

# **Decision Tree:**

## **Giới thiệu về Decision Tree:**

Từ cách suy nghĩ và đặt câu hỏi thì con người đưa ra những quyết định của mình cho những vấn đề đó. Machine Learning cũng có một mô hình ra quyết định dựa trên các câu hỏi. Mô hình này có tên là cây quyết định (Decision Tree).

Decision Tree là một thuật toán thuộc loại supervised learning, phương pháp học có giám sát, kết quả hay biến mục tiêu của Decision Tree chủ yếu là biến phân loại. Các thuật toán được xây dựng giống hình dạng một cây có ngọn cây, thân cây, lá cây kết nối bằng các cành cây, và mỗi thành phần đề có ý nghĩa riêng của nó, như các yếu tố tác động lên quyết định sau cùng. Decision Tree được xây dựng trên mô hình cây thể hiện sự kết cấu, sự cấu trúc của một hệ thống ra quyết định và cụ thể ở đây là thể hiện mối quan hệ giữa các yếu tố (các biến độc lập) và sự tác động của nó đến các biến mục tiêu.

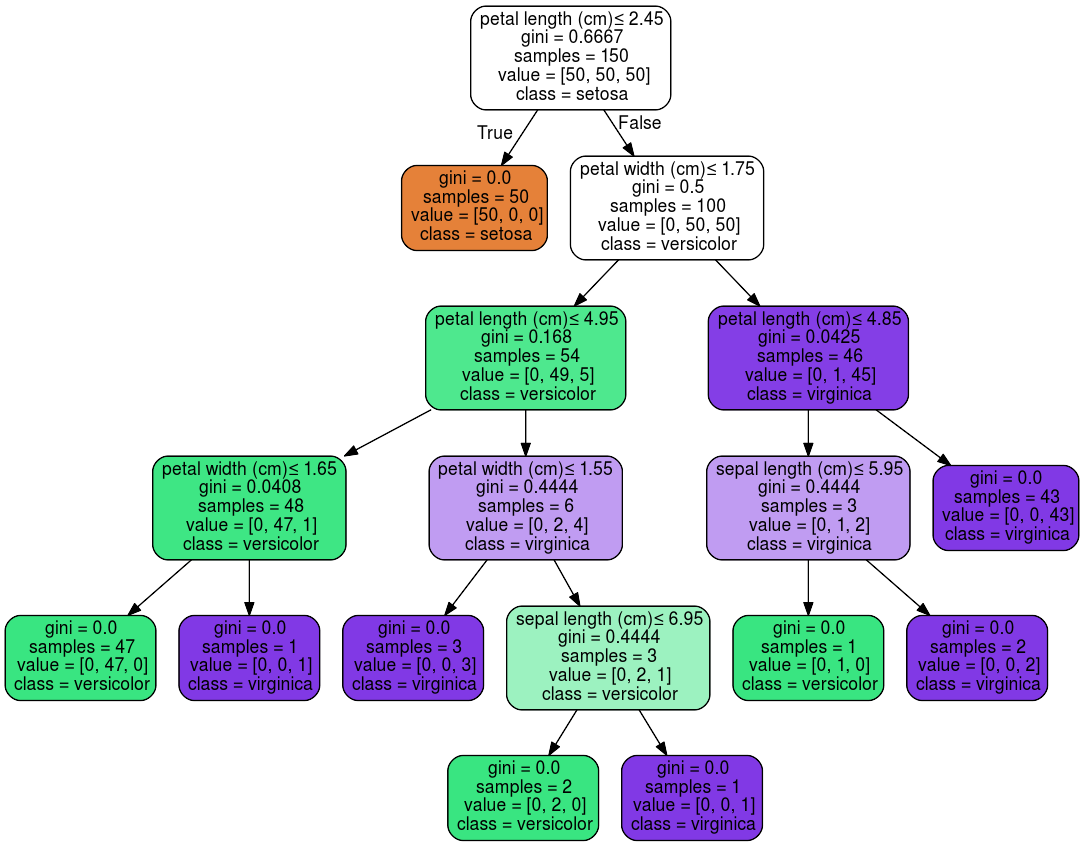


**Figure 1**:Mô hình Decision Tree

Nhìn trên hình sẽ thấy cây quyết định bao gồm:

* Rood node: Điểm ngọn chứa giá trị của biến đầu tiên được dùng để phân nhánh.
* Internal node: Các điểm bên trong thân cây là các biến chứa các thuộc tính, giá trị dữ liệu được dùng để xét cho các phân nhánh tiếp theo.
* Leaf node: là các lá cây node chứa giá trị của biến phân loại sau cùng.
* Branch: là quy luật phân nhánh, là mối quan hệ giữa các giá trị biến độc lập (Internal node) và các giá trị biến mục tiêu (Leaf node).

Ta có ví dụ về Decision Tree trên Iris Dataset:



**Figure 2**:Decision Tree with Iris Dataset using Sklearn

Để xây dựng cấu trúc cây ở trên, thuật toán Decision Tree đơn giản sẽ bao gồm các bước sau:

* Chọn lựa thuộc tính của data để chia data sử dụng Attribute Selection Measures (ASM: Chỉ số đánh giá lựa chọn thuộc tính).
* Tạo decision node với feature và điều kiện ở trên.
* Phân nhánh data tạo các child node và lặp lại tiến trình ở trên cho đến khi một trong các điều kiện sau thoả mãn, ta sẽ có leaf node:
* Tất cả data của node đều thoả mãn điều kiện của decision node.
* Không có feature với điều kiện nào có thể được chọn nữa.
* Không còn data nào thoả mãn điều kiện của decision node.

## **Entropy:**

Entropy là phương pháp lựa chọn cách phân nhánh tối ưu dựa trên cơ sở tối đa hóa lượng thông tin nhận vào, tức là giảm thiểu tối đa độ hỗn độn và nhiễu loạn trong từng node. Các node phân nhánh được lựa chọn theo phương pháp này phải thể hiện tối đa thông tin cần thiết để cây quyết định có thể phân loại chính xác đối tượng dữ liệu vào các tập con có chứa nhãn- là những giá trị, thuộc tính của biến đầu vào:

Trong đó  là xác suất xuất hiện đối tượng dữ liệu mang thuộc tính i của biến mục tiêu trong node của hệ thống. Khi càng lớn thì sẽ mang giá trị âm và tiến về 0 nhân với số âm phía trước sẽ dương và tiến gần đến 0 Giá trị entropy càng cao thì độ hỗn loạn của hệ thống càng cao, còn nếu càng thấp thì hệ thống càng có trật tự. Và một hệ thống có trật tự cũng tương đương với việc data được phân nhánh một cách chuẩn xác.

## **Information Gain:**

Xác định xem trong các cách thức phân nhánh thì cách nào đem lại nhiều thông tin nhất, rõ ràng nhất, đầy đủ cơ sở nhất để chúng ta phân loại đối tượng dữ liệu theo các giá trị, các nhóm, các phân lớp có sẳn của biến mục tiêu.Giá trị entropy thể hiện mức độ hỗn loạn của hệ thống, do đó khi entropy giảm hệ thống có trật tự hơn hay có thể nói có nhiều thông tin hơn. Do vậy, độ giảm của entropy được gọi là Information Gain và có công thức như sau:

Trong đó:

* là giá trị Information Gain.
* là điều kiện để chia data.
* là số nhóm sau khi chia.
* là số lượng data của mỗi nhóm.

## **Gini Impurity (Gini Index):**

Đơn giản hơn so với Entropy và Information Gain, Gini Impurity là chỉ số thể hiện mức độ phân loại sai khi ta chọn ngẫu nhiên một phần tử từ tập data. 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. Gini Impurity có công thức như sau:

Trong đó:

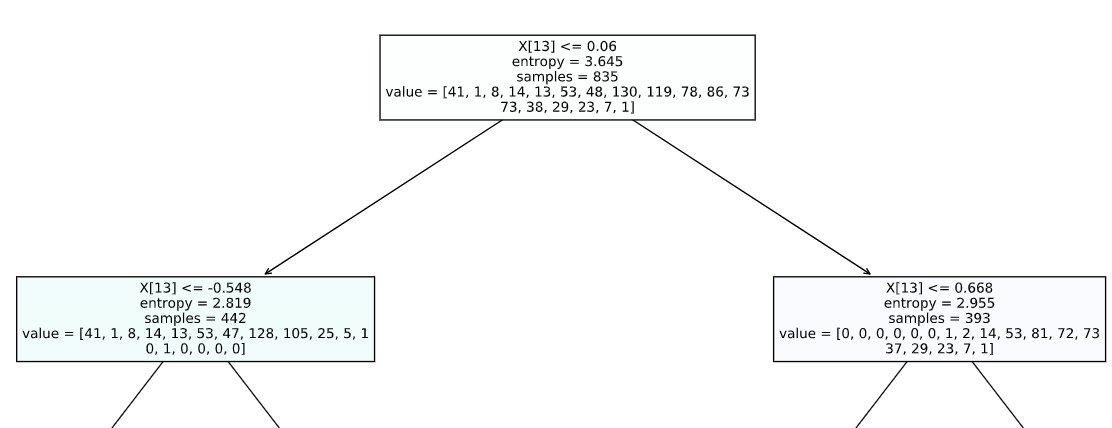
* là giá trị Gini Impurity.
* là số các lớp có trong tập data.
* là xác suất mà một phần tử ngẫu nhiên thuộc lớp .

## **Xây dựng Decision Tree Classification:**

Decision tree Classification được xây dựng như sau:

* Cố gắng tìm ra một tập hợp các “nhánh lá” tối ưu nhất
* Thuật toán bắt đầu với nút gốc là S (thường là tập label).
* Trên mỗi lần lặp của thuật toán, nó lặp đi lặp lại thuộc tính không sử dụng của tập S và tính Entropy(H) và Information Gain (IG) của thuộc tính.
* Nó sẽ chọn thuộc tính có giá trị Entropy nhỏ nhất hoặc giá trị Information Gain lớn nhất.
* Tập S sau đó được phân chia theo thuộc tính được chọn để tạo ra một tập hợp con của dữ liệu.
* Thuật toán tiếp tục lặp lại trên mỗi tập hợp con, chỉ xem xét các thuộc tính chưa từng được chọn trước đó.

Xét ví dụ ở phần sau:



**Figure 3**:Milk Root-node và Child-node

Ở node đầu là node có giá trị entropy cao nhất phân xuống ở Child-node thứ nhất ta tiếp tục xét thuộc tính có giá trị Entropy lớn thứ 2 cứ thể phân tiếp đến khi xác định được label*.* Cứ lặp đi lặp lại việc tính này cho đến khi hết các thuộc tính.

1. **Decision Tree trên Python:**
2. **Dataset:**

Dữ liệu sử dụng Student Performance Data Set [1] để training, dataset này gồm student-mat.csv (math cource) và student-por.csv (Portuguese language course) đều có 32 attribute thông tin về dataset như sau:

***Attribute Information:***

1 -- school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira).  
2 -- sex - student's sex (binary: 'F' - female or 'M' - male).  
3 -- age - student's age (numeric: from 15 to 22).  
4 -- address - student's home address type (binary: 'U' - urban or 'R' - rural).  
5 -- famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3).  
6 -- Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart).  
7 -- Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education).  
8 -- Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education).  
9 -- Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at\_home' or 'other').  
10 -- Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at\_home' or 'other').  
11 -- reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other').  
12 -- guardian - student's guardian (nominal: 'mother', 'father' or 'other').  
13 -- traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour).  
14 -- studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours).  
15 -- failures - number of past class failures (numeric: n if 1<=n<3, else 4).  
16 -- schoolsup - extra educational support (binary: yes or no).  
17 -- famsup - family educational support (binary: yes or no).  
18 -- paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no).  
19 -- activities - extra-curricular activities (binary: yes or no).  
20 -- nursery - attended nursery school (binary: yes or no).  
21 -- higher - wants to take higher education (binary: yes or no).  
22 -- internet - Internet access at home (binary: yes or no).  
23 -- romantic - with a romantic relationship (binary: yes or no).  
24 -- famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent).  
25 -- freetime - free time after school (numeric: from 1 - very low to 5 - very high).  
26 -- goout - going out with friends (numeric: from 1 - very low to 5 - very high).  
27 -- Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high).  
28 -- Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high).  
29 -- health - current health status (numeric: from 1 - very bad to 5 - very good).  
30 -- absences - number of school absences (numeric: from 0 to 93).  
31 -- G1 - first period grade (numeric: from 0 to 20).  
31 -- G2 - second period grade (numeric: from 0 to 20).  
32 -- G3 - final grade (numeric: from 0 to 20, output target).

***Nhận xét:***

Dataset trên có cả dữ liệu số và dữ liệu chữ, đa số là dữ liệu chữ do đó trước khi vào training thì cần phải biến đổi dữ liệu chữ sang số.

1. **Decision Tree Classification:**

*Parameters:*

* **Criterion**: optional (default = ‘gini’) or choose attribute selection measure. Supported criteria are ‘gini’ for the Gini index and “entropy” for the information gain.
* **Splitter**: string, optional (default = ‘best’) or Split Strategy: Supported strategies are ‘best’ to choose the best split and ‘random’ to choose the best random split.
* **Max\_depth**: int or None, optional (default = None) or Maximum Depth of a Tree.

*Methods:*

* **Fit (X, y) – Học dữ liệu dưới dạng mảng, ma trận.**

Parametes:

**X:** Train data ở dạng mảng hoặc ma trận [n\_samples, n\_features].

**y:** Nhãn của dữ liệu ở dạng mảng (nhãn ở dạng Classification là giá trị 0: fail, 1: pass).

*Returns:*

Self: Object – Returns self.

* **Predict (X)** **– Dự đoán giá trị đầu ra y dựa vào đầu vào X.**

*Parameters:*

**X**: Data ở dạng mảng hoặc ma trận [n\_samples, n\_features\_idx]

*Return:*

y: là mảng [n\_samples].

1. **Code thuật toán:**

* Import một số thư viện cần thiết cho bài toán.

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from sklearn import tree  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score  from sklearn.preprocessing import LabelEncoder  from sklearn.pipeline import FeatureUnion  from sklearn.pipeline import Pipeline  from sklearn.preprocessing import StandardScaler  from sklearn.impute import SimpleImputer  from sklearn.preprocessing import OneHotEncoder  from sklearn.base import BaseEstimator, TransformerMixin  from sklearn.model\_selection import cross\_val\_score  train\_test\_split  lb = LabelEncoder() |

* Đọc dữ liệu trên hai file *student-mat.csv* và *student-por.csv* và gộp hai file lại với nhau.

|  |
| --- |
| df\_mat = pd.read\_csv('student-mat.csv')  df\_por = pd.read\_csv('student-por.csv')  df = pd.concat([df\_mat,df\_por], ignore\_index=True) |

* Dùng StratifiedShufffleSplit để chia dataset ra làm hai train\_set và test\_set (theo tỉ lệ 8:2) để cho dữ liệu sau khi chia ra mang tính đại diện. Việc chia này được chia theo feature ‘G3’ với các khoảng được chia như bên dưới. Sau cùng thì dùng plot để show xem các khoảng trên tập dataset, train\_set và test\_set để kiểm tra.

|  |
| --- |
| from sklearn.model\_selection import StratifiedShuffleSplit  df["Khoảng G3"] = pd.cut(df['G3'],                      bins=[-1,5,11,15,20, np.inf],                      labels=[5,11,15,20,25])  splitter = StratifiedShuffleSplit(n\_splits=1, test\_size=0.2, random\_state=42)  for train\_index, test\_index in splitter.split(df, df["Khoảng G3"]):      train\_set = df.loc[train\_index]      test\_set = df.loc[test\_index]  print(df["Khoảng G3"].hist(bins=6,figsize=(5,5)))  plt.show()  print(train\_set["Khoảng G3"].hist(bins=6,figsize=(5,5)))  plt.show()  print(test\_set ["Khoảng G3"].hist(bins=6,figsize=(5,5)))  plt.show() |

* Xóa đi feature ‘Khoảng G3’.

|  |
| --- |
| for \_set\_ in (train\_set, test\_set):      \_set\_.drop(columns="Khoảng G3", inplace=True) |

* Tách label trên hai tập train và test.

|  |
| --- |
| train\_set\_label = train\_set['G3'].copy()  train\_set = train\_set.drop(columns='G3')  test\_set\_label = test\_set['G3'].copy()  test\_set = test\_set.drop(columns='G3') |

* Biến label sang Classification là các giá trị rời rạc với giá trị là 0 – Fail, 1 – Pass.

|  |
| --- |
| train\_set\_label.loc[train\_set\_label[:,] < 10, ] = 0  train\_set\_label.loc[train\_set\_label[:,] >= 10,] = 1  test\_set\_label.loc[test\_set\_label[:,] < 10, ] = 0  test\_set\_label.loc[test\_set\_label[:,] >= 10, ] = 1 |

* Định nghĩa function để có thể tái sử dụng cho những lần chuyển đổi dữ liệu chữ sang số. Tham số truyền vào là 1 mảng các index cần đổi cùng với data muốn chuyển những giá trị của index đó sang số.

|  |
| --- |
| def Fit\_Transform(cat\_array, data):      for i in cat\_array:          data.iloc[:,i] = lb.fit\_transform(data.iloc[:,i])      return data |

* Liệt kê những index mang giá trị số đưa vào mảng có tên là cat\_feat\_index. Sau đó thực hiện việc chuyển dữ liệu bằng gọi hàm đã định nghĩa ở trên.

|  |
| --- |
| cat\_feat\_index = [0,1,2,3,4,5,8,9,10,11,15,16,17,18,19,20,21,22]  processed\_train\_set\_val = Fit\_Transform(cat\_feat\_index, train\_set)  processed\_test\_set = Fit\_Transform(cat\_feat\_index, test\_set) |

* Đi tìm max\_depth cho DecisionTreeClassifier bằng cách chạy vòng lặp với số max\_depths bằng số lượng features (57) sao khi đã transform ở trên với mỗi lần chạy thì thực hiện việc cross\_val\_score để lấy giá trị trung bình accurancy sau đó lưu vào mảng test\_results. Với vòng lặp thứ 2 thì ta dùng để tìm ra được max depth lưu và max\_dep\_index.

|  |
| --- |
| max\_depths = np.linspace(1, 57, 57, endpoint=True)  test\_results = []  for max\_depth in max\_depths:      model = tree.DecisionTreeClassifier(criterion="entroy", max\_depth=max\_depth)      dtc = model.fit(processed\_train\_set\_val,train\_set\_label)      sc = cross\_val\_score(dtc, processed\_test\_set, test\_set\_label, scoring='roc\_auc', cv=5)      test\_results.append(sc.mean())  max\_dep\_value = test\_results[0]  max\_dep\_index = 0  for i in range(0,len(test\_results)):      if(test\_results[i]>max\_dep\_value):          max\_dep\_index = i          max\_dep\_value = test\_results[i]  print(max\_dep\_index) |

* Cho DecisionTreeClassifier học trên tập train và tính cross\_val\_score trên tập test. Sau đó in ra giá trị trung bình.

|  |
| --- |
| model = tree.DecisionTreeClassifier(criterion="entropy", max\_depth = max\_dep\_index+1)  dtc = model.fit(processed\_train\_set\_val,train\_set\_label)  sc = cross\_val\_score(dtc, processed\_test\_set, test\_set\_label, scoring='roc\_auc', cv=5)  print("AUC: %0.2f (+/- %0.2f)" % (sc.mean(), sc.std() \* 2)) |

* Tạo 1 report cho model, được hỗ trợ bởi sklearn là classification\_report.

|  |
| --- |
| from sklearn.metrics import classification\_report  from sklearn import metrics  pd = model.predict(processed\_test\_set)  print(pd)  result1 = metrics.classification\_report(test\_set\_label, pd)  print("Classification Report:",)  print (result1) |

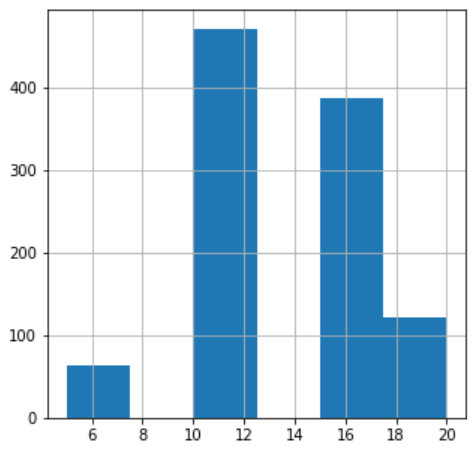
* Vẽ DecisionTree và save vào file tree.png, với model bị giới hạn độ sâu.

|  |
| --- |
| fig, ax = plt.subplots(figsize=(80, 50))  tree.plot\_tree(model,filled=True)  plt.savefig('tree.png',format='png',bbox\_inches = "tight", fontsize=40)  tree.plot\_tree(model, fontsize=40)  plt.show() |

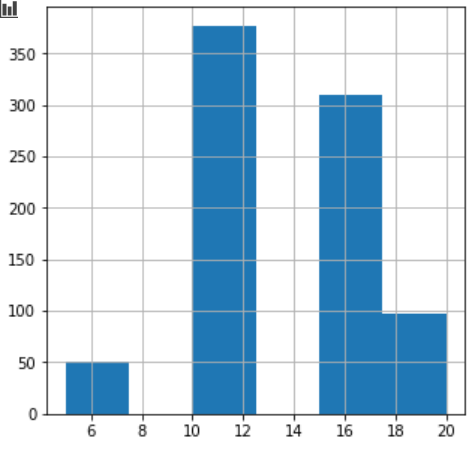
* Vẽ DecisionTree và save vào file tree.png, với model không bị giới hạn độ sâu.

|  |
| --- |
| model = tree.DecisionTreeClassifier()  model.fit(processed\_train\_set\_val, train\_set\_label)  fig, ax = plt.subplots(figsize=(80, 50))  tree.plot\_tree(model,filled=True)  plt.savefig('tree\_all.png',format='png',bbox\_inches = "tight", fontsize=40)  tree.plot\_tree(model, fontsize=40)  plt.show() |

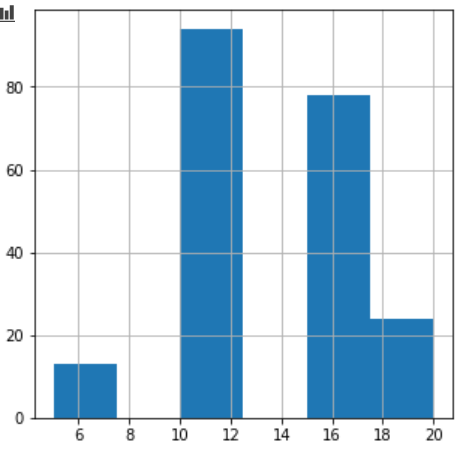
1. **Kết quả.**



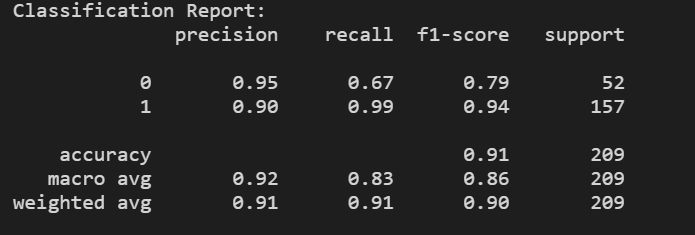
**Figure 4** Đồ thị về Khoảng G3 của df.



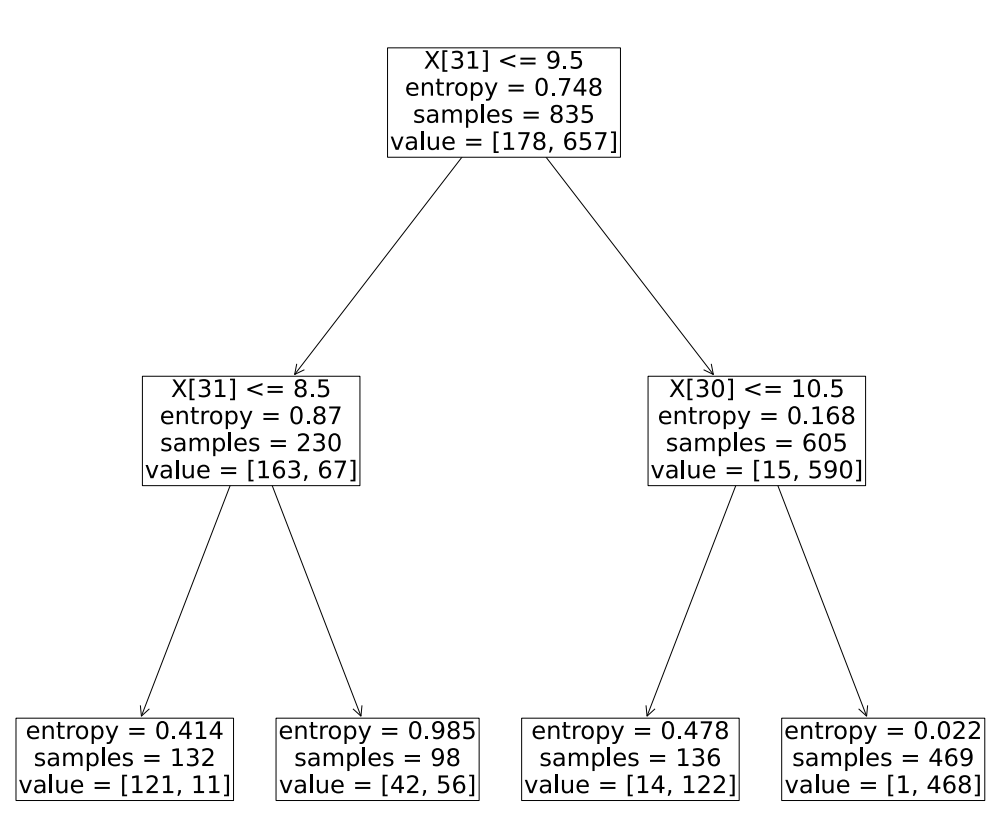
**Figure 5** Đồ thị Khoảng G3 của train\_set.



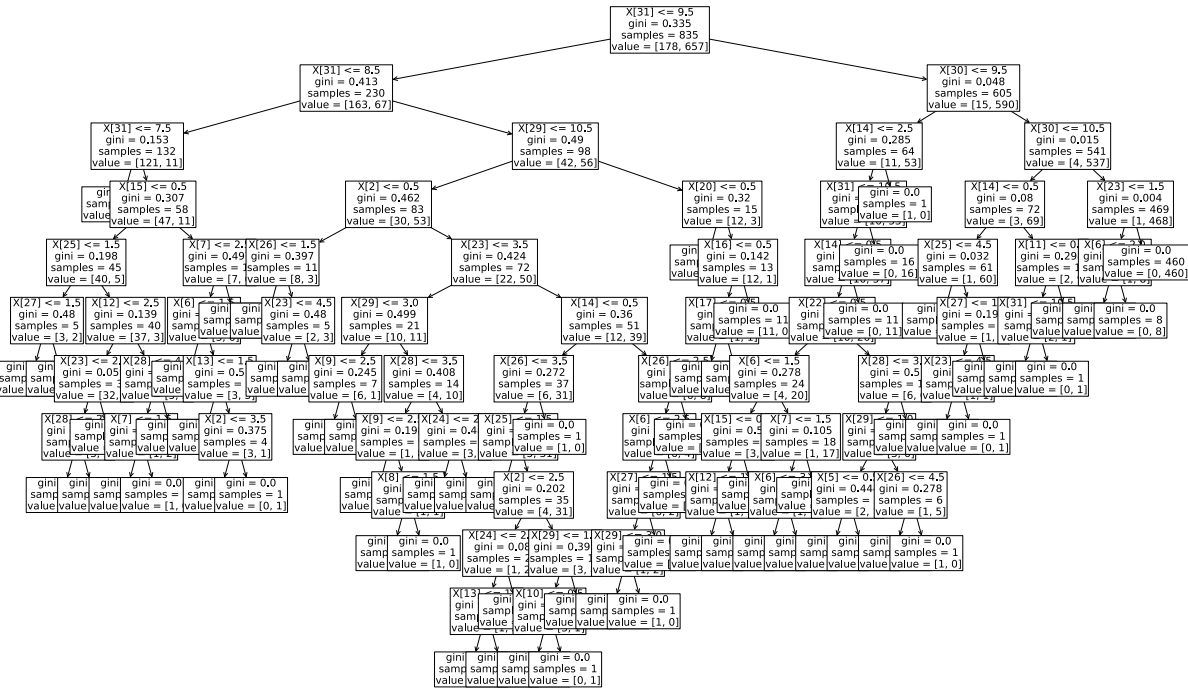
**Figure 6** Đồ thị Khoảng G3 của test\_set.



**Figure 7** Classification Report.



**Figure 8** Sơ đồ của Decision Tree với độ sâu = 2.



**Figure 9** Sơ đồ Decision Tree.

1. **Tài liệu tham khảo:**

[1] Stydent Performance Data Set <https://archive.ics.uci.edu/ml/datasets/student+performance>

[2] Decision Tree Classification <https://www.datacamp.com/community/tutorials/decision-tree-classification-python>