**TRƯỜNG ĐH CÔNG NGHỆ THÔNG TIN – ĐHQG HCM**

CASE STUDY 1

**CLASSIFICATION – TITANIC DATASET**



*Lớp học:*

**Machine learning – CS114.J21.KHTN**

*Giáo viên:*

**Huỳnh Thị Thanh Thương**

*Sinh viên:*

**17520689 – Nguyễn Thùy Linh**

**17520015 – Nguyễn Hữu Phong**

**1. ĐỀ BÀI­**

1.1 CÂU CHUYỆN VỀ TITANIC

Con tàu đã va vào một tảng băng khi đang chở 2.224 hành khách và thủy thủ trong chuyến đi đầu tiên từ Anh đến Mỹ. Nó chìm ở Bắc Đại Tây Dương vào ngày 15.4.1912 mặc dù được phong danh hiệu “không thể chìm” thời đó. Hơn 1.500 người – bao gồm cả phụ nữ và trẻ em – đã chết rong vụ tai nạn. Điều này khiến vụ chìm tàu Titanic trở thành một trong những thảm họa hàng hải nguy hiểm nhất trong lịch sử. Thảm họa Titanic đã là đề tài cho hàng loạt các cuộc điều tra, kiện tụng, giả thuyết ly kỳ đồng thời dần đến các thay đổ lớn trong quy định an toàn hàng hải quốc tế. Titanic cũng trở thành bối cảnh và đề tài cho rất nhiều tác phần văn học, hội họa và điện ảnh.

**Thống kê chung về số người sống sót và tử nạn**

Theo báo cáo của ủy ban điều tra thành lập bởi Thượng viện Hoa Kỳ, trong tổng số 2.223 người trên tàu, chỉ 706 người sống sót, 1.517 người thiệt mạng. Trong số hành khách hạng nhất, 199 người được cứu (60.5%) và 130 người chết. Trong số hành khách hạng hai, 119 (44%) người sống và 166 người chết. Trong số hành khách hạng ba, 174 người được cứu (24.5%) và 536 người chết. Thủy thủy đoàn , 194 người được cứu (22%) và 682 người chết. Đặc biệt, toàn bộ nhân viên kỹ thuật 35 người (trong đó gồm 25 kỹ sư, 6 kỹ sư điện, 2 kỹ sư nồi hơi, một kỹ sư đường ống và một thư ký) đều thiệt mạng.

Đa số người chết do không chịu nổi sự giảm thân nhiệt trong nước có nhiệt độ 28 độ F (-2 độ C). Ngoài ra, tỉ lệ tử vong cao cũng vì các thuyền cứu sinh đã không chở tối đa số người có thể. Cũng theo báo cáo trước Thượng viện Hoa kỳ, Nếu các thuyền cứu sinh chở đủ sức chứa, 1.178 người có thể đã được cứu sống.

1.2 TITANIC DATASET

**Number of observations**: 2.201. Titanic dataset bao gồm thông tin của 2201 người có mặt trên tàu Titanic khi nó đâm phairmoojt tảng băng trôi và chìm xuống biển.

**Number of attributes**: 4

Có 4 thuộc tính (categorical attributes) được xem xét:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên thuộc tính | Các giá trị | Mô tả |
| 1 | Social class | First class, second class, thrid class, crewmember | Phân loại hành khách và thủy thủ đoàn |
| 2 | Age | Child, adult | Độ tuổi |
| 3 | Sex | Male, female | Giới tính |
| 4 | Whether or not the person survived | No, yes | Liệu người đó có sống sót hay không |

**The question of interest**: Câu hỏi quan tâm cho bộ dữ liệu này là “sự sống còn liên quan đến các thuộc tính khác như thế nào?”.

**2. PHÁT BIỂU LẠI BÀI TOÁN:**

2.1 TASK:

Tìm xem với một vector **input**(social class, age, sex) thì xác suất để **label**(whether or not the person survised) nhận lần lượt tập giá trị (yes, no) là bao nhiêu %. Từ đó đưa ra kết luận

2.2 INPUT:

Vector input(social class, age, sex)

2.3 OUTPUT:

Label(whether not the person survised)

**3. TRAINING WITH K-NEAREST NEIGHBORS**

3.1. Thư viện sử dụng

Scikit – learn:

<https://scikit-learn.org/stable/>

3.2. Tách dataset thành training data và test data

**Cách chia:**

Chọn 40 datapoints từ dataset từ dataset làm test\_set

Còn lại 2201 – 40 = 2161 datapoints làm training\_set

> Chia theo social class, lấy chuẩn 10 datapoints cho mỗi social class

**Cách đánh giá:**

> Công thức đánh giá cho mỗi lần training:

P = Số lần dự đoán đúng / 40

> Để dự đoán chính xác hơn, thì training nhiều lần, mỗi lần chọn random datapoints làm test\_set. Được tập P1, P2, ..., Pm với m là số lần training.

Xác suất trung bình P = (P1 + P2 + ... + Pm) / m

3.3. Ý tưởng

**Ý tưởng đơn thuần:** Tìm K datapoints “gần” nhất với datapoint đang xét. Công thức tính “gần” theo chuẩn norm 1.

**Tiếp cận 1:**

Mỗi Input(social class, age, sex) sẽ được xem như 1 điểm trên không gian 3 chiều O(x, y, z). Ta mã hóa các các thông tin này thành dạng số để tiện biểu diễn:

|  |  |
| --- | --- |
| social class | ‘1st’ : ‘0’ | ‘2nd’ : ‘1’ | ‘3rd’ : ‘2’ | ‘crew’ : ‘3’ |
| age | ‘child : ‘0’ | ‘adult : ‘1’ |
| sex | ‘male’ : ‘0 | ‘female’ : ‘1’ |

> Do đó dữ liệu đưa vào ta sẽ mã hóa thành một chuỗi:

Ví dụ: (1st, adult, female) tương ứng là ‘011’

(3rd, child, male) tương ứng là ‘200’

**Tiếp cận 2:**

Với số lượng thuộc tính và số lượng value của mỗi thuộc tính như đề bài thì có số lượng tổ hợp là : 4x2x2 = 16

> Nên ta sẽ chuyển data đầu vào chỉ cần 16 điểm để biểu diễn, và lưu thêm số lượng tại điểm đó

3.3. PSEUDO CODE

For X in [test\_set]: # duyệt từng điểm từ test\_set

For i in range(0, k): # tìm k điểm gần nhất với X

Y = init # khởi tạo Y khi chưa xét

For j in [training\_set]: # duyệt tập training\_set điểm chưa chọn

Updata Y # cập nhập Y thỏa mãn

K\_set add Y # thêm Y vào K\_set

Delete Y from training\_set: # Đánh dấu Y được chọn rồi

cnt\_yes = 0 #khởi tạo cnt\_yes, là số lượng câu trả lời yes cnt\_no = 0 #khởi tạo cnt\_no, là số lượng câu trả lời no

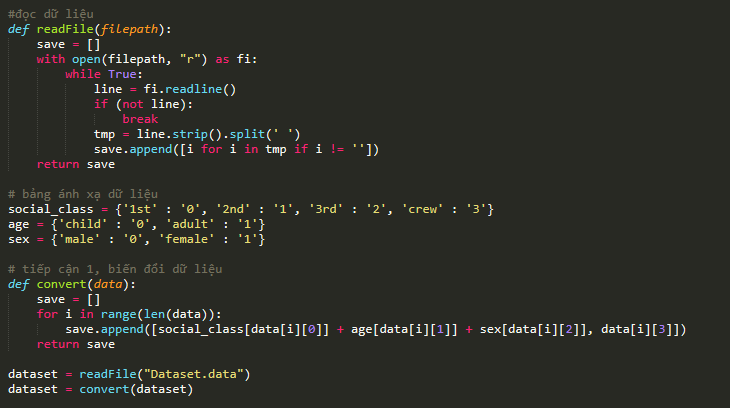
For i in [K\_set]: # duyệt tập K\_set tìm được

Update cnt\_yes, cnt\_no

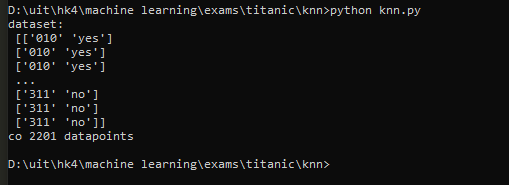
=> Tính kết quả rồi xác nhận nhãn cho X (yes hay no)

3.4. Code không dùng thư viện

> Đọc dữ liệu và xử lý data ban đầu:



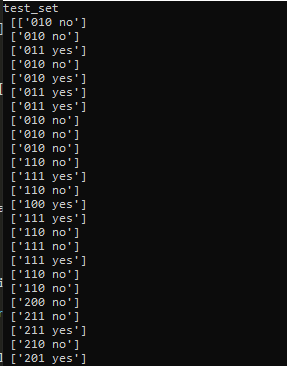
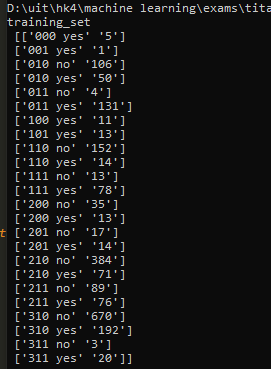
Kết quả:



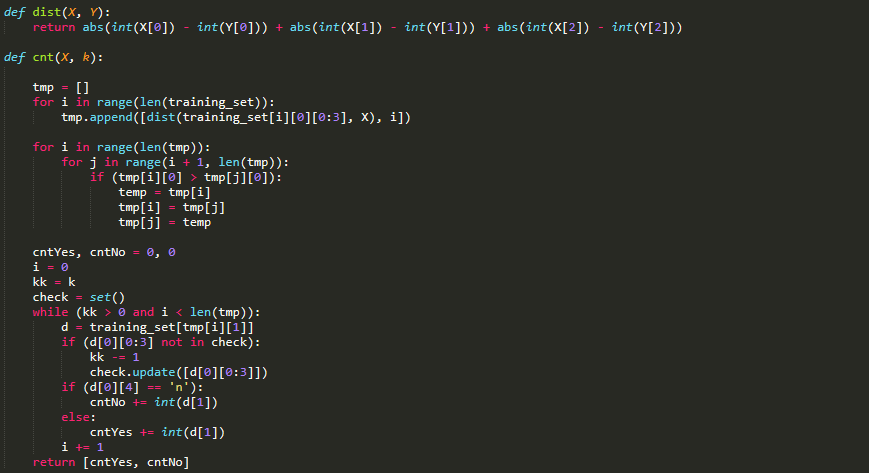
> Tách ra làm training\_set và test\_set:

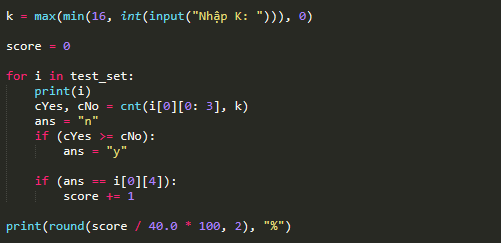


Kết quả:

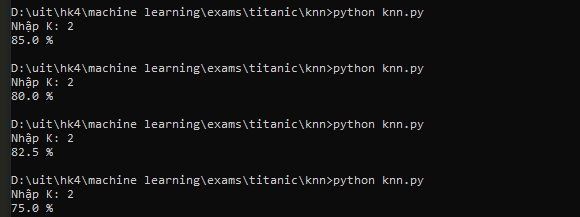


> training:





Kết quả sau vài lần training với 2-NN:



**4. TRAINING WITH NAIVE BAYES**

4.1. Cách thức thực nghiệm và đánh giá

**Chia:** tập dữ liệu thu thập được chia thành 2 phần theo tỉ lệ 9:1

Training set: 1979 mẫu

Test set: 222 mẫu

**Cách chia:** <chia theo social class>

Bước 1: Lập bảng thống kê số lần xuất hiện của các giá trị của thuộc tính social class

Bước 2: Với mỗi giá trị, lấy test set với bước nhảy là 10

**Cách dánh giá:**

Công thức: tỉ lệ giữa số điểm được dự doán đúng và tổng số điểm trong tập dữ liệu kiểm thử

Độ đo: accuracy (độ chính xác)

**Kết quả đánh giá:**

Độ chính xác: 77.02702702702703%

**Những hạn chế của phương pháp:**

Trong thế giới thực, hầu như các feature của dữ liệu không độc lập với nhau

Chính vì vậy, độ chính xác của naive bayes so với các thuật toán khác, chẳng hạn như decision tree thì không được cao

4.2. Cài đặt

**Các tool, thử viện đã sử dụng:**

Sklearn

**Cách cài đặt tool và chạy chương trình:**

Đầu tiên ta đọc file và cho dataset vào mảng t.

Sau đó chuyển mảng t từ kiểu string sang integer

Ta tiếp tục phân chia mảng t thành mảng data và mảng label

Sau đó ta sử dụng hàm train\_test\_split của thư viện sklearn để phân tách data và label thành data(train, test) và label(train, test) với tỉ lệ 9 train: 1 test

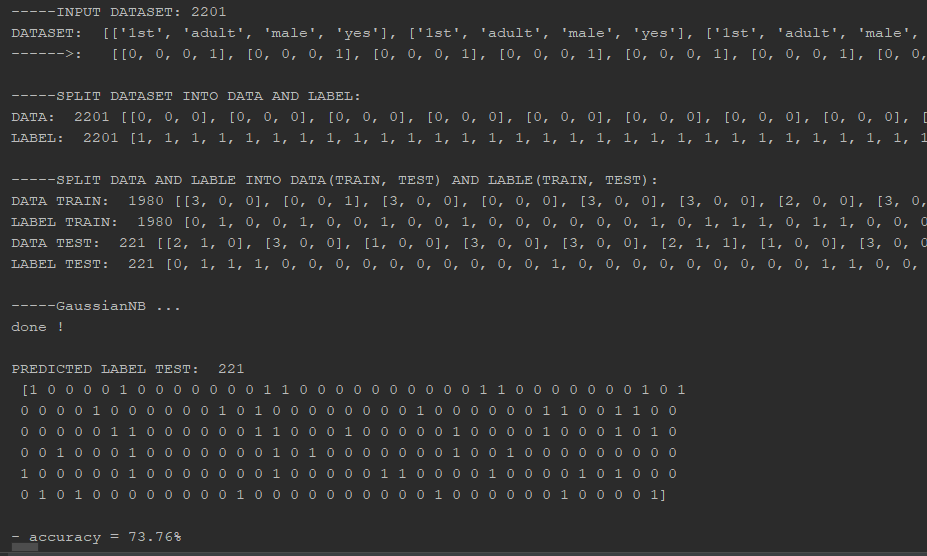
Tiếp theo ta sử dụng hàm GaussianNB của thư viện sklearn để fit data\_train và label\_train

Cuối cùng ta dự đoán label của data\_test và đánh giá độ chính xác bằng hàm accuracy\_score của thư viện sklearn

**Source code:**

* def readFile(path):  
   arr=[]  
   file = open(path,'r',encoding='utf-8')  
   for line in file:  
   data = line.strip()  
   a = data.split()  
   arr.append(a)  
   file.close()  
   return arr  
    
  t = readFile('TitanicDataset')  
  print('-----INPUT DATASET: {0}'.format(len(t)))  
  print('DATASET: ',t)  
  def convert(t):  
   for i in range(len(t)):  
   if t[i][0] == '1st': t[i][0]=0  
   elif t[i][0] == '2nd': t[i][0]=1  
   elif t[i][0] == '3rd': t[i][0]=2  
   else: t[i][0] = 3  
    
   if t[i][1] == 'adult': t[i][1]=0  
   else: t[i][1]=1  
    
   if t[i][2] == 'male': t[i][2]=0  
   else: t[i][2] = 1  
    
   if t[i][3] == 'yes': t[i][3]=1  
   else: t[i][3]=0  
   return t  
    
  convert(t)  
  print('------>: ',t)  
    
  print('\n-----SPLIT DATASET INTO DATA AND LABEL:')  
  def split(t):  
   data = []  
   label = []  
   for i in range(len(t)):  
   data.append([t[i][0],t[i][1],t[i][2]])  
   label.append(t[i][3])  
   return data,label  
    
  data,label = split(t)  
  print('DATA: ',len(data),data)  
  print('LABEL: ',len(label),label)  
    
  print('\n-----SPLIT DATA AND LABLE INTO DATA(TRAIN, TEST) AND LABLE(TRAIN, TEST):')  
  from sklearn.model\_selection import train\_test\_split  
    
  data\_train,data\_test,label\_train,label\_test = train\_test\_split(data,label,test\_size=0.1,random\_state=1)  
  print('DATA TRAIN: ',len(data\_train),data\_train)  
  print('LABEL TRAIN: ',len(label\_train),label\_train)  
  print('DATA TEST: ',len(data\_test),data\_test)  
  print('LABEL TEST: ',len(label\_test),label\_test)  
    
  print('\n-----GaussianNB ...')  
  from sklearn.naive\_bayes import GaussianNB  
  G = GaussianNB()  
  G.fit(data\_train,label\_train)  
    
  predicted\_label = G.predict(data\_test)  
  print('done !\n')  
  print('PREDICTED LABEL TEST: ',len(predicted\_label),'\n',predicted\_label)  
    
  from sklearn.metrics import accuracy\_score  
  print('\n- accuracy = %.2f%%'%(accuracy\_score(label\_test,predicted\_label)\*100))

**Kết quả trả về của hệ thống:**



**5. TRAINING WITH DECISION TREE**

5.1. Pseudo code:

def DecisionTree (dataSet, attributeSet):

if mọi data thuộc dataSet đều nằm trong cùng một lớp:

return leaf\_node được gán nhãn bởi lớp đó

elif atributeSet == NULL:

return leaf\_node được gán nhãn theo số đông

else:

Tính toán E(|dataSet|)

Với mỗi attribute A thuộc attributeSet, tính E(|dataSet|, A)

Chọn attribute A có IG(|dataSet|,A) max làm root cho cây hiện tại

Với mỗi V thuộc attribute A được chọn:

Tạo một nhánh cây gán nhãn V

datasetV = dataSet.phân\_vùng\_theo\_V

DecisionTree(datasetV, attributeSet.xóa\_A)

5.2. Cách thức thực nghiệm và đánh giá

**Chia:** tập dữ liệu thu thập được chia thành 2 phần theo tỉ lệ 9:1

Training set: 1979 mẫu

Test set: 222 mẫu

**Cách chia:** <chia theo social class>

Bước 1: Lập bảng thống kê số lần xuất hiện của các giá trị của thuộc tính social class

Bước 2: Với mỗi giá trị, lấy test set với bước nhảy là 10

**Cách dánh giá:**

Công thức: tỉ lệ giữa số điểm được dự doán đúng và tổng số điểm trong tập dữ liệu kiểm thử

Độ đo: accuracy (độ chính xác)

**Kết quả đánh giá:**

Độ chính xác: 75.11%

**Những hạn chế của phương pháp:**

Nếu một thuộc tính có thể nhận rất nhiều giá trị, decision tree thu được có thể sẽ có rất nhiều node. Xét một ví dụ về các triệu chứng của các bệnh nhân trong một bệnh viện và đàu ra là mắc bệnh hay không? Mỗi bệnh nhân có một mã số (id) khác nhau. Nếu ta sử dụng thuộc tính này cho việc huấn luyện, ta rất có thể sẽ thu được mô hình decision tree mà mỗi leaf node ứng với một bệnh nhân. Lúc đó mô hình này là vô dụng, vì không thể dự đoán được việc mắc bệnh hay không của một bệnh nhân mới

Hạn chế lớn nhất của ID3 và decision tree nói chung là việc nếu một điểm dữ liệu mới rơi vào nhầm nhánh ở ngay những lần phân chia đầu tiên, kết quả cuối cùng sẽ khác rất nhiều. Việc rơi vào nhầm nhánh này rất dễ xảy ra trong trường hợp thuộc tính liên tục được chia thành nhiều nhóm nhỏ, vì hai điểm có thuộc tính tương ứng rất gần nhau có thể rơi vào hai nhóm khác nhau.

5.3. Cài đặt

**Các tool, thư viện đã sử dụng:**

Sklearn

**Cách cài đặt tool và chạy chương trình:**

Đầu tiên ta đọc file và cho dataset vào mảng t

Sau đó chuyển mảng t từ kiểu string sang integer

Ta tiếp tục phân chia mảng t thành mảng data và mảng label

Sau đó ta sử dụng hàm train\_test\_split() của thư viện sklearn để phân tách data và label thành data(train, test) và label(train, test) với tỉ lệ 9 train : 1 test

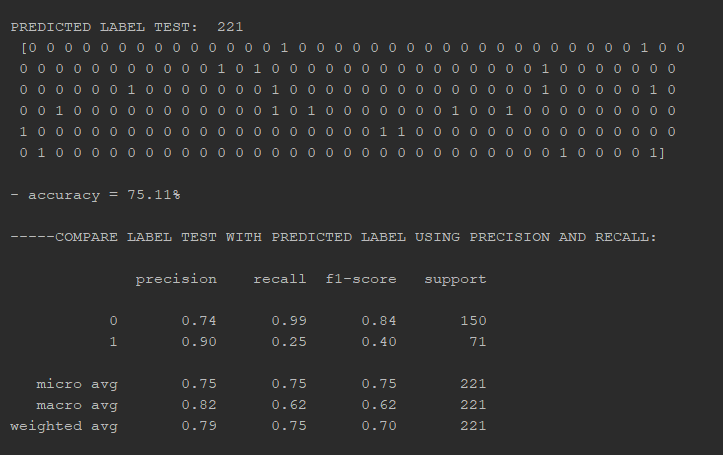
Tiếp theo ta sử dụng hàm tree.DecisionTreeClassifier() của thư viện sklearn để fit data\_train và label\_train

Cuối cùng ta dự đoán label của data\_test và đánh gái độ chính xác bằng hàm accuracy\_score của thư viện sklearn.

**Source code:**

* def readFile(path):  
   arr=[]  
   file = open(path,'r',encoding='utf-8')  
   for line in file:  
   data = line.strip()  
   a = data.split()  
   arr.append(a)  
   file.close()  
   return arr  
    
  t = readFile('TitanicDataset')  
  print('-----INPUT DATASET: {0}'.format(len(t)))  
  print('DATASET: ',t)  
  def convert(t):  
   for i in range(len(t)):  
   if t[i][0] == '1st': t[i][0]=0  
   elif t[i][0] == '2nd': t[i][0]=1  
   elif t[i][0] == '3rd': t[i][0]=2  
   else: t[i][0] = 3  
    
   if t[i][1] == 'adult': t[i][1]=0  
   else: t[i][1]=1  
    
   if t[i][2] == 'male': t[i][2]=0  
   else: t[i][2] = 1  
    
   if t[i][3] == 'yes': t[i][3]=1  
   else: t[i][3]=0  
   return t  
    
  convert(t)  
  print('------>: ',t)  
    
  print('\n-----SPLIT DATASET INTO DATA AND LABEL:')  
  def split(t):  
   data = []  
   label = []  
   for i in range(len(t)):  
   data.append([t[i][0],t[i][1],t[i][2]])  
   label.append(t[i][3])  
   return data,label  
    
  data,label = split(t)  
  print('DATA: ',len(data),data)  
  print('LABEL: ',len(label),label)  
    
  print('\n-----SPLIT DATA AND LABEL INTO DATA(TRAIN, TEST) AND LABEL(TRAIN, TEST):')  
  from sklearn.model\_selection import train\_test\_split  
    
  data\_train,data\_test,label\_train,label\_test = train\_test\_split(data,label,test\_size=0.1,random\_state=1)  
  print('DATA TRAIN: ',len(data\_train),data\_train)  
  print('LABEL TRAIN: ',len(label\_train),label\_train)  
  print('DATA TEST: ',len(data\_test),data\_test)  
  print('LABEL TEST: ',len(label\_test),label\_test)  
    
  print('\n-----Decision Tree Classifier ...')  
  from sklearn import tree  
    
  DS = tree.DecisionTreeClassifier()  
  DS.fit(data\_train,label\_train)  
    
  predicted\_label = DS.predict(data\_test)  
  print('done !\n')  
  print('PREDICTED LABEL TEST: ',len(predicted\_label),'\n',predicted\_label)  
    
  from sklearn.metrics import accuracy\_score  
  print('\n- accuracy = %.2f%%'%(accuracy\_score(label\_test,predicted\_label)\*100))

**Kết quả của hệ thống:**



**6. TÀI LIỆU THAM KHẢO:**

* Phương pháp Accuracy:

<https://machinelearningcoban.com/2017/08/31/evaluation/#-accuracy>

* Phương pháp Precision and Recall:

<https://machinelearningcoban.com/2017/08/31/evaluation/#-precision-va-recall>

* Naïve Bayes:

<https://machinelearningcoban.com/2017/08/08/nbc/>

* Decision Tree:

<https://machinelearningcoban.com/2018/01/14/id3/#-thao-luan>

* sklearn.naive\_bayes.GaussianNB():

<https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html>

* sklearn.tree.DecisionTreeClassifier():

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>