**Introduction to Machine Learning (Spring 2019)**

**Homework #6 (50 Pts, June 23)**

**Student ID 2014313303**

**Name 홍태하**

**Instruction:** We provide all codes and datasets in zip files. Please write your code to complete Decision Tree and Gaussian Naïve Bayesian models. Compress only 4 Files ‘Decision\_Tree\_Answer.py’, ‘NaïveBayesian\_Answer.py’ ‘utils\_Answer.py’ & your report and submit with the filename ‘HW6\_STUDENT\_ ID.zip’.

**1 – (1) [10 pts]** Implement Decision Tree in ‘Decision\_Tree\_Answer.py’, some parts of ‘utils\_Answer.py’

1. **[Gini index]** Implement Gini\_index in ‘utils\_Answer.py’

def Gini\_index(Y\_data):

gini = 0

#========= Edit here ==========

yes = 0

no = 0

total = len(Y\_data)

gini = 1

attribute, count = np.unique(Y\_data, return\_counts=True)

for attr, cnt in zip(attribute, count):

gini -= pow(cnt/total, 2)

#====================================

return gini

1. **[Entropy]** Implement Entropy in ‘utils\_Answer.py’

def Entropy(Y\_data):

entropy = 0

# ===== Edit here ========

yes = 0

no = 0

total = len(Y\_data)

attribute, count = np.unique(Y\_data, return\_counts=True)

for attr, cnt in zip(attribute, count):

entropy -= (cnt/total)\*np.log2(cnt/total)

# ==============================

return entropy

1. **[Find Best feature]** Implement a ‘Find\_Best\_Feature’ in ‘Answer.py’.

def Find\_Best\_Feature(self, df):

"""

This function retuns the feature that can split the data.

in which, the impurity is the least.

you should implement the part of calculating for

impurity (entropy or gini\_index) for given feature.蹂寃쏀븯??

???⑥닔??遺덉닚?꾧? 媛???묎쾶 ?곗씠?곕? 遺꾪븷?????덈뒗 feature瑜?

諛섑솚?섎뒗 ?⑥닔?낅땲??

?щ윭遺꾩씠 援ы쁽???댁슜? 二쇱뼱吏?feature??impurity(entropy or gini\_index)瑜?

怨꾩궛?섎뒗 寃껋엯?덈떎.

[Parameter]:

df : [dataFrame (got by pandas) ][N x D] : Training\_data

[Variables] :

header : [list of string ] [1 x D] the set of attribute\_name, last element is for output\_feature.

input\_feature : [list of string] [1 x (D-1)] the set of attribute name except for last feature(output)

Y\_data : [column vector of label] [N x 1] label\_data, To get the (vector or matrix) not dataFrame,

you can write ' data = df.values '

self.Category\_feature\_idx : [list of integers]: the set of idx only for Category feature.(referred top of the code.)

split\_value : in numeric\_data, You just divide the data two part.

one is the less than split\_value, the other is no less than split\_value.

[Objects]: (the part of implement)

impurity : (float) entropy or gini\_index got by splitting data given attribute.

impurity\_list [list of float] [1 x D] : the list which store the all impurity in order.

[return]:

Best\_feature : (string) feature\_name

feature\_type : (string) the type of feature ('Category' or 'Numeric')

"""

header = df.columns.values

input\_feature = header[:-1]

output\_feature = header[-1]

Y\_data = df[output\_feature].values

impurity\_list = []

# for all features in DataFrame,

for idx, h in enumerate(input\_feature):

# ============ Edit here ==================

# Category Feature Case

if idx in self.Category\_feature\_idx:

impurity = 0

temp = dict()

for i, item in enumerate(df[h].values):

if item in temp:

temp[item] += 1

else:

temp[item] = 1

for key in temp:

temp\_y = []

for i, item in enumerate(df[h].values):

if item == key:

temp\_y.append(Y\_data[i])

val = impurity\_func(temp\_y, self.criterion)

impurity += val\*len(temp\_y)/len(df[h].values)

# Numeric Feature Case

else:

split\_value = Finding\_split\_point(df, h, self.criterion)

impurity = 0

temp\_y\_s = []

temp\_y\_l = []

for i, item in enumerate(df[h].values):

if item < split\_value:

temp\_y\_s.append(Y\_data[i])

else:

temp\_y\_l.append(Y\_data[i])

val\_s = impurity\_func(temp\_y\_s, self.criterion)

val\_l = impurity\_func(temp\_y\_l, self.criterion)

impurity = val\_s\*len(temp\_y\_s)/len(df[h].values) + val\_l\*len(temp\_y\_l)/len(df[h].values)

#=====================================================

impurity\_list.append(np.round(impurity, 6))

idx = np.argmin(impurity\_list)

Best\_feature = input\_feature[idx]

feature\_type = idx in self.Category\_feature\_idx and 'Category' or 'Numeric'

return Best\_feature, feature\_type

**Answer: Fill your code here. You also have to submit your code to i-campus.**

**NOTE 1**: **You should write your codes in ‘EDIT HERE’ signs.** It is not recommended to edit other parts. Once you complete your implementation, run the check codes (‘code\_validation.py’) to check if it is done correctly.

**NOTE 2**: **Read the instructions in template codes VERY CAREFULLY.** Funcionality and input, output format of any function must be the same as what is written.

**1 – (2) [10 Pts]** Experiment results

1. you are given 2 dataset (Heart, Carseats) with Binary classification(Yes or No). Measure the performance of Decision tree given setting environments.

**Answer: Fill the blank in the table. Show the plot of training & test accuracy with a brief explanation.**

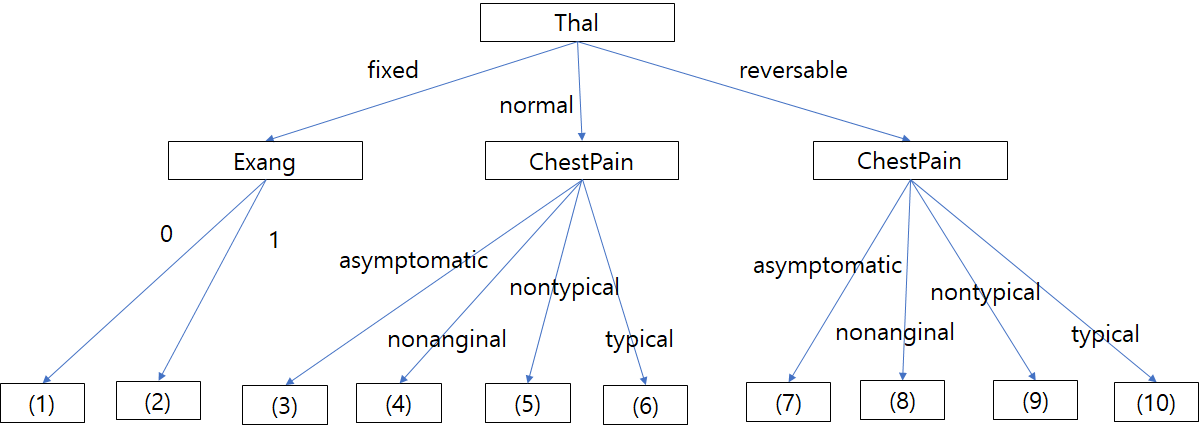
**[Decision Tree]**

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Impurity function | Max depth | Accuracy |
| Heart | Entropy | 2 | 0.6842 |
| Heart | Gini\_index | 3 | 0.7544 |
| Carseats | Entropy | 3 | 0.9125 |
| Carseats | Gini\_index | 3 | 0.8875 |

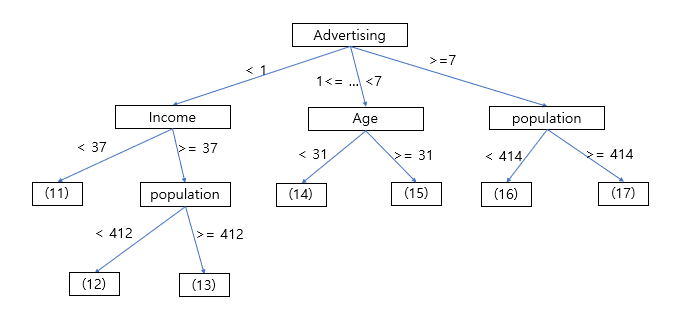
Train accuracy와 test accuracy 모두 heart data보다는 carseats data가 높다. 각각의 data는 모두 train accuracy가 test accuracy보다 높다.

**1 – (3). [10 pts]** Analysis

Heart\_dataset, (Impurity function = entropy, max\_depth = 2)



Carseat\_dataset, (Impurity function = entropy, max\_depth = 3)



Above figure is the result of Decision Tree for the given condition. Please fill the label of leaf node. And Explain which node can be pruned.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (1)No | (2)Yes | (3)No | (4)No | (5)No |
| (6)No | (7)Yes | (8)Yes | (9)No | (10)No |
| (11)No | (12)No | (13)No | (14)No | (15)Yes |
| (16)Yes | (17)Yes |  |  |  |

3,4,5,6은 ChestPain 밑의 node인데 그 값이 모두 No이다. 따라서 3,4,5,6을 쳐내고 Thal에서 normal을 택했을 때 바로 No라고 할 수 있다.

11,12,13도 Income 밑으로 전부 No이므로 쳐낼 수 있고 16,17도 population 밑으로 전부 Yes이기 때문에 쳐낼 수 있다.

2 - (1). **[10 pts]** Implement Gaussian Naïve Bayesian in ‘NaiveBayesian\_Answer.py’ and the rest parts of ‘utils \_Answer.py’ (To prevent that likelihood becomes zero, please apply the Laplacian smoothing technique.).

1. **[likelihood, prior]** Implement ‘fit’ in ‘NaïveBayesian\_Answer.py’ for the following function: (likelihood, prior).

def fit(self, dataFrame, data\_name):

self.df = dataFrame

self.data\_name = data\_name

self.Category\_feature\_idx = (data\_name == 'Heart') \

and Heart\_Category\_feature\_idx or \

(data\_name == 'Carseats') \

and Carseats\_Category\_feature\_idx \

or Tennis\_Category\_feature\_idx

header = self.df.columns.values

input\_feature = header[:-1]

output\_feature = header[-1]

Y\_data = self.df[output\_feature]

# Data split to 'Class = Yes' data and 'Class = No' data.

pos\_idx = (Y\_data == 'Yes')

pos\_data = self.df[pos\_idx]

neg\_data = self.df[~pos\_idx]

# P(Yes), P(No)

# ====== Edit this ======

self.prob\_yes = len(pos\_data)/len(Y\_data)

self.prob\_no = len(neg\_data)/len(Y\_data)

# ==============================

# P(X | Yes), P(X | No)

self.prob\_given\_yes = {}

self.prob\_given\_no = {}

# To deal with avoid zero probability,

# Save the numbers of Category for each column. (only Category Feature)

self.num\_of\_attr = [ 0 ] \* len(input\_feature)

# Setting P(x\_i | Yes)

# To prevent key error, predefine the all dictionary to 0.

self.initialize\_likelihood\_prob(self.prob\_given\_yes)

self.set\_likelihood\_prob(pos\_data, self.prob\_given\_yes)

# Setting P(x\_i | No)

# To prevent key error, predefine the all dictionary to 0.

self.initialize\_likelihood\_prob(self.prob\_given\_no)

self.set\_likelihood\_prob(neg\_data, self.prob\_given\_no)

def set\_likelihood\_prob(self, data, likelihood):

'''

In this function, the process for assigning the probability of likelihood

is done. Because likelihood is managed by dictiorary, you should be careful to the key.

the key format of Category feature is 'feature\_name = value' ('outlook = sunny')

and the Numeric is "'feautre\_name'\_mean" and "'feature\_name\_std'" ('Age\_mean', 'Age\_std')

(Important!) Keep in mind that we use Laplacian correction.

???⑥닔?먯꽌??likelihood ?뺣쪧 媛믪쓣 ?좊떦?섎뒗 怨쇱젙???섑뻾?⑸땲??

likelihood媛 dictionary ?뺥깭濡?愿由щ릺??key 媛믪뿉 二쇱쓽?섏떆湲?諛붾엻?덈떎.

Category data??key 媛믪? 'feature?대쫫 = 媛? ('outlook = sunny')

Numeric data??'feature?대쫫\_mean', 'feature?대쫫\_std' ?낅땲??

(以묒슂!) ?곕━媛 Laplacian correction???ъ슜?섍퀬 ?덈떎???ъ떎??二쇱쓽?섏꽭??

Category: likelihood['%s = %s' %(feature\_name, value)] = ?

Numeric: likelihood['%s\_mean' %(feature\_name)] = ?

Numeric: likelihood['%s\_std' %(feature\_name)] = ?

[Parameter]

data : [dataFrame] label is 'Yes' data xor 'No' data

likelihood : [dictionary] likelihood probability. (don't overwritten!)

[return]

nothing

'''

n = len(data)

input\_feature = data.columns.values[:-1]

# ========== Edit here =============

for idx, f in enumerate(input\_feature):

column\_data = data[f]

if idx in self.Category\_feature\_idx:

attribute, count = np.unique(column\_data, return\_counts=True)

for attr, cnt in zip(attribute, count):

# to avoid key Error in predict fuction.

if type(attr) == np.float64 or type(attr) == np.float32:

attr = str(int(attr))

feature\_name = f

value = attr

likelihood['%s = %s' %(feature\_name, value)] = round((cnt+1)/(len(column\_data)+len(attribute)), 6)

else:

feature\_name = f

likelihood['%s\_mean' %(feature\_name)] = np.mean(data[f].values)

likelihood['%s\_std' %(feature\_name)] = np.std(data[f].values)

# ================================================

return

1. **[Posterior]** Implement ‘predict’ in ‘NaiveBayesian\_Answer.py’ for the posterior probability.

def get\_posterior(self, tuple, likelihood, prior):

'''

In this function return the posterior probability to use maximum a posterior.

(posterior ??likelihood x prior)

we will use the likelihood and prior instead of accurate posterior.

your work is return the posterior for given tuple.

Gaussian prob function is defined at utils.utils\_Answer.py

Keep in mind that if dictionary value is undefined before, the value is 0.

???⑥닔??posterior ?뺣쪧??諛섑솚?섎뒗 ?⑥닔?낅땲??

?ш린?쒕뒗 posterior???뺥솗??媛????likelihood ? prior瑜??댁슜?⑸땲??

?щ윭遺꾩씠 援ы쁽??遺遺꾩? 二쇱뼱吏?tuple data????댁꽌 posterior瑜?諛섑솚?섎뒗 ?⑥닔瑜??묒꽦?섎뒗 寃껋엯?덈떎.

Gaussian prob function??utils.utils\_Answer.py???뺤쓽?섏뼱 ?덉뒿?덈떎.

dictionary 媛믪씠 ?댁쟾???뺤쓽?섏? ?딆븯?쇰㈃ value 媛 0?꾩쓣 紐낆떖?섏꽭??

[ Parameter ]

tuple : one data [1 x (D-1) Dataframe] : a data tuple (exclueded output part)

likelihood [dictionary] : a dictionary for likelihood

prior [float] : probability in [0, 1]

[return]

posterior [float] : a probability of posterior.

'''

feature = tuple.index

posterior = 1

# ============ Edit here ===============

posterior = prior

for idx, f in enumerate(feature):

val = tuple[f]

if idx in self.Category\_feature\_idx:

# to avoid key Error in predict fuction.

if type(val) == np.float64 or type(val) == np.float32:

val = str(int(val))

posterior \*= likelihood['%s = %s' %(f, val)]

else:

posterior \*= Gaussian\_prob(val, likelihood['%s\_mean' % (f)], likelihood['%s\_std' % (f)])

# ===============================================================

return posterior

1. **[Gaussian probability]** Implement Gaussian\_prob in ‘utils\_Answer.py’. for the following function:

def Gaussian\_prob(x, mean, std):

'''

:param x: input value: X

:param mean: the mean of X

:param std: the standard deviation of X

:return: probaility (X) ~ N(關, ?^2)

'''

ret = 0

# ======== Edit here ==========

ret = 1/np.sqrt(2\*np.pi\*pow(std,2))\*np.exp(-pow(x-mean,2)/(2\*pow(std,2)))

# =========================================

return ret

2 - (2). **[10 pts]** Experimental results.

1. you are given 2 dataset (Heart, Carseats) with Binary classification(Yes or No). Measure the performance of Decision tree and fill the blank.

**[Naïve Bayesian]**

|  |  |
| --- | --- |
| Dataset | Accuracy |
| Heart | 0.7719 |
| Carseats | 0.9 |