Web Mining Lab - 8

Aadhitya Swarnesh

[redacted]

- 7 April 2021

Question

Import the heart disease dataset from the UCI machine Learning repository or Use the URL directly in the code and Evaluate the accuracy of any decision tree algorithm(ID3/C4.5/CART).

We will first obtain the dataset from the UCI repository, we here use the **Cleveland Heart Disease Dataset**. After doing this, we can notice that this is with a "data" extension. We can directly import this and begin with our Classification procedure, but we will first observe the dataset, pre-process it and on the way convert it into a **Comma Separated File** format, so as to import it easily.

For the pre-processing part, we had observed that there were a few "**?**" Mark signs which indicated the absence of values or **NULL** values. One remedy is to replace this with the mean of all the values of that column, but we will just be removing such instance from the dataset, as there were a very few such cases. Then we will convert the datatype of all the rows to match their data, we have one column "old peak" which contains float values, and the rest are integers. So we convert them accordingly and then store this modified dataset into a **CSV** file.

Now that the dataset is ready, we will proceed to the actual Classification Process, we will in this lab deal with two algorithms of the **Decision Tree** namely the **ID3** or the **Iterative Dichotomiser 3** which uses **information gain** which in turn depends on the **entropy** calculation in order to find the best set of features in every stage of the algorithm. The other algorithm used here is **CART** or **Classification and Regression Trees** which in turn uses the **Gini Index** for the feature selection process during the phases of the

algorithm. The procedure followed by both these algorithms is similar and they differ only be the feature selection process they use.

The implementation of these algorithms is done using the **Scikit Learn Libraries** in the **Python Programming Language**. We have first performed a **80%-20% train-test** split of the complete dataset, and then we have used the training set to train the decision tree classifiers, and then the test set to evaluate the models. All the functions that have been used here has been imported from the **sk-learn** library as mentioned before.

Result :

We have implemented both the CART and the ID3 model, and the results obtained by these models are as follows :

The **training** phase accuracy obtained by the model is as follows :

```
Train Accuracy of the ID3 Model :  0.7468354430379747
Train Accuracy of the CART Model :  0.7510548523206751
```

The **testing** phase accuracy obtained by the model (55% and 60%) respectively is as follows :

```
Test Accuracy of the ID3 Model :  0.55
Test Accuracy of the CART Model :  0.6
```

We have also used some other metrics, to gain more insight on the results obtained by the models :

This is the Confusion matrix and the scores of precision, recall and F-Measure obtained by the **ID3** model :

```
Confusion Matrix of the ID3 model is :
 [[30  4  1  5  0]
 [ 6  0  1  2  1]
 [ 2  1  1  1  2]
 [ 0  0  1  2  0]
 [ 0  0  0  0  0]]
```

```
Classification Report of the ID3 model is :
              precision    recall  f1-score   support

           0       0.79      0.75      0.77        40
           1       0.00      0.00      0.00        10
           2       0.25      0.14      0.18         7
           3       0.20      0.67      0.31         3
           4       0.00      0.00      0.00         0

    accuracy                           0.55        60
   macro avg       0.25      0.31      0.25        60
weighted avg       0.57      0.55      0.55        60
```

This is the Confusion matrix and the scores of precision, recall and F-Measure obtained by the **CART** model :

```
Confusion Matrix of the CART model is :
 [[33  5  1  1  0]
 [ 6  1  2  1  0]
 [ 2  1  1  2  1]
 [ 0  0  2  1  0]
 [ 0  0  0  0  0]]
```

```
Classification Report of the CART model is :
              precision    recall  f1-score   support

           0       0.80      0.82      0.81        40
           1       0.14      0.10      0.12        10
           2       0.17      0.14      0.15         7
           3       0.20      0.33      0.25         3
           4       0.00      0.00      0.00         0

    accuracy                           0.60        60
   macro avg       0.26      0.28      0.27        60
weighted avg       0.59      0.60      0.59        60
```

Based on all these parameters, we can conclude that the **CART** model which uses the Gini Index performs slightly **better** than the **ID3** model which uses the traditional information gain in this scenario.

Code :

The **code** that has delivered these results are in two notebooks one for the pre-processing and the other for the decision tree building, and they are as follows :

UCI Heart Disease Dataset

In this notebook, we read in the data from the `.data` file and store it in the form of a `csv` file. We also learn more about the parameters in this dataset along the way.

We have used the `Cleveland Dataset` here in this notebook, and the same will be followed in the further notebooks of this lab.

In [1]:

```
# Importing the required libraries

import pandas as pd
```

1. Reading the Raw data

In [2]:

```
# Read the CSV file

df = pd.read_csv('./raw_data/processed.cleveland.data', sep=',', header=None)
df.head()
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0

The Columns names are not present here, so we will refer to the documentation for these info. The column details are :

- `age` : Age of the patient in years
- `gender` : The gender of the patient --

```
-- 1 : male
-- 0 : female
```

- `cp` : Chest Pain Type --

```
-- Value 1: typical angina
-- Value 2: atypical angina
-- Value 3: non-anginal pain
-- Value 4: asymptomatic
```

- `trestbps` : resting blood pressure (in `mm Hg` on admission to the hospital)
- `chol` : serum cholestoral in `mg/dl`
- `fbs` : fasting blood sugar (> 120 mg/dl) --

```
-- 1 : true
-- 0 : false
```

- `restecg` : resting electrocardiographic results --

```
-- Value 0: normal
-- Value 1: having ST-T wave abnormality (T wave inversions and/or S
T elevation or depression of > 0.05 mV)
-- Value 2: showing probable or definite left ventricular hypertroph
y
```

- `thalach` : maximum heart rate achieved
- `exang` : exercise induced angina --

```
-- 1 = yes
-- 0 = no
```

- `oldpeak` : ST depression induced by exercise relative to rest
- `slope` : the slope of the peak exercise ST segment

```
-- Value 1: upsloping
-- Value 2: flat
-- Value 3: downsloping
```

- `ca` : number of major vessels (`0-3`) colored by flourosopy

```
 Coronary arteries are the blood vessels of coronary circulation, wh
ich transports oxygenated blood to the actual heart muscle.
```

- `thal` : Exercise Thallium heart scan (Exercise thallium scintigraphy) --

```
-- 3 = normal
-- 6 = fixed defect
-- 7 = reversable defect
```

- `num` : diagnosis of heart disease (angiographic disease status)

```
-- Value 0: < 50% diameter narrowing
-- Value 1: > 50% diameter narrowing
```

In [3]:

```
# Name the columns into the dataset

df.columns = ['age', 'gender', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thal
ach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'num']
```

In [5]:

```
df.head()
```

Out[5]:

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    float64
 1   gender    303 non-null    float64
 2   cp        303 non-null    float64
 3   trestbps  303 non-null    float64
 4   chol      303 non-null    float64
 5   fbs       303 non-null    float64
 6   restecg   303 non-null    float64
 7   thalach   303 non-null    float64
 8   exang     303 non-null    float64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    float64
 11  ca        303 non-null    object 
 12  thal      303 non-null    object 
 13  num       303 non-null    int64  
dtypes: float64(11), int64(1), object(2)
memory usage: 33.3+ KB
```

2. Pre-Processing

As we can notice that the fields : `ca` and `thal` are taken as objects, it means that there are some null values. Let us take steps to precess this.

2.1 Null Values in `ca`

The number of major vessels colored by flourosopy.

In [12]:

```
# To get all the unique values of the `cp` column

df['ca'].unique()
```

Out[12]:

```
array(['0.0', '3.0', '2.0', '1.0', '?'], dtype=object)
```

Notice that there is a question mark `?` indicating missing values.

In [9]:

```
# The rows where we have '?' as the data

df[df['ca'] == '?']
```

Out[9]:

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	th
166	52.0	1.0	3.0	138.0	223.0	0.0	0.0	169.0	0.0	0.0	1.0	?	3
192	43.0	1.0	4.0	132.0	247.0	1.0	2.0	143.0	1.0	0.1	2.0	?	7
287	58.0	1.0	2.0	125.0	220.0	0.0	0.0	144.0	0.0	0.4	2.0	?	7
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	?	3

As there are only a few records, we will remove these.

In [14]:

```
# Get the indices of these instances

indices = df[df['ca'] == '?'].index
```

In [15]:

```
# Remove these rows from the dataset

df.drop(indices, axis=0, inplace=True)
```

In [40]:

```
# Now change the column ca into float

df = df.astype({'ca' : 'int64'})
```

In [41]:

```
# Now we can notice that the 'ca' field is also of float datatype
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 297 entries, 0 to 301
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       297 non-null    float64
 1   gender    297 non-null    float64
 2   cp        297 non-null    float64
 3   trestbps  297 non-null    float64
 4   chol      297 non-null    float64
 5   fbs       297 non-null    float64
 6   restecg   297 non-null    float64
 7   thalach   297 non-null    float64
 8   exang     297 non-null    float64
 9   oldpeak   297 non-null    float64
 10  slope     297 non-null    float64
 11  ca        297 non-null    int64  
 12  thal      297 non-null    float64
 13  num       297 non-null    int64  
dtypes: float64(12), int64(2)
memory usage: 34.8 KB
```

2.2 Null Values in `thal`

In [28]:

```
# To get all the unique values of the `thal` column

df['thal'].unique()
```

Out[28]:

```
array(['6.0', '3.0', '7.0', '?'], dtype=object)
```

Notice that there is a question mark `?` indicating missing values.

In [29]:

```
# The rows where we have '?' as the data

df[df['thal'] == '?']
```

Out[29]:

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tl
87	53.0	0.0	3.0	128.0	216.0	0.0	2.0	115.0	0.0	0.0	1.0	0.0	
266	52.0	1.0	4.0	128.0	204.0	1.0	0.0	156.0	1.0	1.0	2.0	0.0	

As there are only a few records, we will remove these.

In [30]:

```
# Get the indices of these instances

indices = df[df['thal'] == '?'].index
```

In [31]:

```
# Remove these rows from the dataset

df.drop(indices, axis=0, inplace=True)
```

In [42]:

```
# Now change the column ca into float

df = df.astype({'thal' : 'int64'})
```

In [43]:

```
# Now we can notice that the 'ca' field is also of integer datatype
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 297 entries, 0 to 301
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       297 non-null    float64
 1   gender    297 non-null    float64
 2   cp        297 non-null    float64
 3   trestbps  297 non-null    float64
 4   chol      297 non-null    float64
 5   fbs       297 non-null    float64
 6   restecg   297 non-null    float64
 7   thalach   297 non-null    float64
 8   exang     297 non-null    float64
 9   oldpeak   297 non-null    float64
 10  slope     297 non-null    float64
 11  ca        297 non-null    int64  
 12  thal      297 non-null    int64  
 13  num       297 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 34.8 KB
```

2.3 Convert the remaining columns into appropriate data types

In [50]:

```
df = df.astype({
    'age' : 'int64',
    'gender' : 'int64',
    'cp' : 'int64',
    'trestbps' : 'int64',
    'chol' : 'int64',
    'fbs' : 'int64',
    'restecg' : 'int64',
    'thalach' : 'int64',
    'exang' : 'int64',
    'oldpeak' : 'float64',
    'slope' : 'int64',
    'ca' : 'int64',
    'thal' : 'int64',
    'num' : 'int64',
})
```

In [51]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 297 entries, 0 to 301
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       297 non-null    int64  
 1   gender    297 non-null    int64  
 2   cp        297 non-null    int64  
 3   trestbps  297 non-null    int64  
 4   chol      297 non-null    int64  
 5   fbs       297 non-null    int64  
 6   restecg   297 non-null    int64  
 7   thalach   297 non-null    int64  
 8   exang     297 non-null    int64  
 9   oldpeak   297 non-null    float64
 10  slope     297 non-null    int64  
 11  ca        297 non-null    int64  
 12  thal      297 non-null    int64  
 13  num       297 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 34.8 KB
```

3. Saving the Dataset

In [52]:

```
# We save this modified dataset into a CSV file for faster access

df.to_csv('cleveland_data.csv', index=False)
```

In [ ]:

```

```

Decision Tree

In this Notebook, we will implement various decision trees on the `Cleveland Heart Disease` dataset obtained from the `UCI` repository , and after the required pre-processing steps as covered before.

In [1]:

```
# Import Statements

# For accessing the datasets
import pandas as pd

# For splitting the dataset into train and test sets
from sklearn.model_selection import train_test_split

# For implementing the decision tree
from sklearn.tree import DecisionTreeClassifier

# For evaluvating the performance of the classifier
from sklearn import metrics

# For Normalization/Scaling of the features
from sklearn.preprocessing import StandardScaler
```

1. Reading the dataset

In [2]:

```
df = pd.read_csv('cleveland_data.csv')
df.head()
```

Out[2]:

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	1	145	233	1	2	150	0	2.3	3	0	6
1	67	1	4	160	286	0	2	108	1	1.5	2	3	3
2	67	1	4	120	229	0	2	129	1	2.6	2	2	7
3	37	1	3	130	250	0	0	187	0	3.5	3	0	3
4	41	0	2	130	204	0	2	172	0	1.4	1	0	3

2. Feature Selection

We will list out the columns which we will consider to be relevant to making accurate predictions, and also the target variable.

In [126]:

```
# The features are all the first 13 columns

# Taking all the 13 features gives about 60% test accuracy
features = list(df.columns[:13])
features
```

Out[126]:

```
['age',
 'gender',
 'cp',
 'trestbps',
 'chol',
 'fbs',
 'restecg',
 'thalach',
 'exang',
 'oldpeak',
 'slope',
 'ca',
 'thal']
```

In [127]:

```
# Target Variable is the last column
target = 'num'
```

In [128]:

```
# Load the X and the Y variables

X = df[features]
Y = df[target]
```

3. Test Train Split

We split the entire dataset into a testing and a training datasets.

- The `Train` dataset will be used during the training phase to optimize the algorithm
- The `Test` dataset will be used to validate the model on unseen data

In [129]:

```
# Here we split the rows of the dataset randomly between the train and test set,
and the split between the two will be a 80% - 20% between the train and test set
s respectively

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_
state=101)
```

4. Building and Training the Classifier Models

In this section, we build the Decision tree models, and also implement their training process.

4.1 ID3 Decision Tree

This uses the `information gain` or `entropy` for the attribute selection process on each level of the splitting process.

In [130]:

```
# Initialize the Classifier model with maximum depth of 5 layers to prevent over
fitting

id3_classifier = DecisionTreeClassifier(criterion="entropy", random_state=42, ma
x_depth=6, min_samples_leaf=5)
```

In [131]:

```
# Fit the model on the train set
# Training Phase

id3_classifier = id3_classifier.fit(X_train, Y_train)
```

4.2 CART Decision Tree

This uses the `Gini Index` for the attribute selection process on each level of the splitting process.

In [155]:

```
# Initialize the Classifier model
# criterion:'gini' is also the default value, so we can also leave this field wh
ile creating the model.
# This has been initialized with maximum depth of 6 layers to prevent overfittin
g

cart_classifier = DecisionTreeClassifier(criterion="gini", random_state=101, max
_depth=6, min_samples_leaf=5)
```

In [156]:

```
# Fit the model on the train set
# Training Phase

cart_classifier = cart_classifier.fit(X_train, Y_train)
```

5. Evaluation of the Models

Here we test the models that we have trained earlier, and then show their individual scores and metrics.

5.1 ID3 Decision Tree

In [134]:

```
# Find the training Accuracy of the model

id3_train__y_pred = id3_classifier.predict(X_train)
```

In [135]:

```
# Calculate the accuracy of this model

id3_train_accuracy = metrics.accuracy_score(Y_train, id3_train__y_pred)
print("Train Accuracy of the ID3 Model : ", id3_train_accuracy)
```

```
Train Accuracy of the ID3 Model :  0.7468354430379747
```

In [136]:

```
# Predict the outputs for the test set

id3_y_pred = id3_classifier.predict(X_test)
```

In [137]:

```
# Calculate the test accuracy of this model

id3_accuracy = metrics.accuracy_score(Y_test, id3_y_pred)
print("Test Accuracy of the ID3 Model : ", id3_accuracy)
```

```
Test Accuracy of the ID3 Model :  0.55
```

In [138]:

```
# Print the Confusion Matrix of the results from the test set

print("Confusion Matrix of the ID3 model is : \n", metrics.confusion_matrix(Y_te
st, id3_y_pred))
```

```
Confusion Matrix of the ID3 model is :
 [[30  4  1  5  0]
 [ 6  0  1  2  1]
 [ 2  1  1  1  2]
 [ 0  0  1  2  0]
 [ 0  0  0  0  0]]
```

In [158]:

```
# Print the Classification Report

print("Classification Report of the ID3 model is : \n", metrics.classification_r
eport(Y_test, id3_y_pred))
```

```
Classification Report of the ID3 model is :
              precision    recall  f1-score   support

           0       0.79      0.75      0.77        40
           1       0.00      0.00      0.00        10
           2       0.25      0.14      0.18         7
           3       0.20      0.67      0.31         3
           4       0.00      0.00      0.00         0

    accuracy                           0.55        60
   macro avg       0.25      0.31      0.25        60
weighted avg       0.57      0.55      0.55        60
```

5.2 CART Decision Tree

In [159]:

```
# Find the training Accuracy of the model

cart_train__y_pred = cart_classifier.predict(X_train)
```

In [160]:

```
# Calculate the accuracy of this model

cart_train_accuracy = metrics.accuracy_score(Y_train, cart_train__y_pred)
print("Train Accuracy of the CART Model : ", cart_train_accuracy)
```

```
Train Accuracy of the CART Model :  0.7510548523206751
```

In [161]:

```
# Predict the outputs for the test set

cart_y_pred = cart_classifier.predict(X_test)
```

In [162]:

```
# Calculate the test accuracy of this model

cart_accuracy = metrics.accuracy_score(Y_test, cart_y_pred)
print("Test Accuracy of the CART Model : ", cart_accuracy)
```

```
Test Accuracy of the CART Model :  0.6
```

In [163]:

```
# Print the Confusion Matrix of the results from the test set

print("Confusion Matrix of the CART model is : \n", metrics.confusion_matrix(Y_t
est, cart_y_pred))
```

```
Confusion Matrix of the CART model is :
 [[33  5  1  1  0]
 [ 6  1  2  1  0]
 [ 2  1  1  2  1]
 [ 0  0  2  1  0]
 [ 0  0  0  0  0]]
```

In [165]:

```
# Print the Classification Report

print("Classification Report of the CART model is : \n", metrics.classification_
report(Y_test, cart_y_pred))
```

```
Classification Report of the CART model is :
               precision    recall  f1-score   support

           0       0.80      0.82      0.81        40
           1       0.14      0.10      0.12        10
           2       0.17      0.14      0.15         7
           3       0.20      0.33      0.25         3
           4       0.00      0.00      0.00         0

    accuracy                           0.60        60
   macro avg       0.26      0.28      0.27        60
weighted avg       0.59      0.60      0.59        60
```

6. Results

In [147]:

```
# Print Training Accuracies

print("Train Accuracy of the ID3 Model : ", id3_train_accuracy)
print("Train Accuracy of the CART Model : ", cart_train_accuracy)
```

```
Train Accuracy of the ID3 Model :  0.7468354430379747
Train Accuracy of the CART Model :  0.7510548523206751
```

In [148]:

```
# Print Test set Accuracies

print("Test Accuracy of the ID3 Model : ", id3_accuracy)
print("Test Accuracy of the CART Model : ", cart_accuracy)
```

```
Test Accuracy of the ID3 Model :  0.55
Test Accuracy of the CART Model :  0.6
```

Conclusion : We can notice here that the `CART` model which uses the `GINI` index has achieved slightly greater accuracy than the `ID3` model which uses `Entropy` .

In [ ]: