Decision Tree

November 8, 2021

Classification with Decision Tree

Sample	Attributes				Buy new
	Age	Income	Student	Credit rating	Computer
S01	Youth	High	No	Good	No
S02	Youth	High	No	Excellent	No
S03	Middle-aged	High	Yes	Good	Yes
S04	Senior	Medium	Yes	Good	Yes
S05	Senior	Low	Yes	Good	Yes
S06	Senior	Low	Yes	Excellent	No
S07	Middle-aged	Low	Yes	Excellent	Yes
S08	Youth	Medium	No	Good	No
S09	Youth	Low	Yes	Good	Yes
S10	Senior	Medium	Yes	Good	Yes
S11	Youth	High	Yes	Excellent	Yes
S12	Middle-aged	Medium	No	Excellent	Yes
S13	Senior	Medium	No	Excellent	No
S14	Middle-aged	High	Yes	Good	Yes

Classification with Decision Tree

Decide if Edward, the senior student, would buy a new PC with his high income and good credit rating.

Sample	Attributes				Buy new
	Age	Income	Student	Credit rating	Computer
Edward	Senior	High	Yes	Good	?

Decision Tree using Information Theory

$$I(\frac{p}{n}, \frac{q}{n}) = -\frac{p}{n}\log_2\frac{p}{n} - \frac{q}{n}\log_2\frac{q}{n}$$

- Without looking at the attributes:
  $I(\frac{9}{14}, \frac{5}{14}) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$
- Age = Youth
  $I(\frac{2}{5}, \frac{3}{5}) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$
- Age = Middle-aged
  $I(\frac{4}{4}, \frac{0}{4}) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0.000$
- Age = Senior
  $I(\frac{3}{5}, \frac{2}{5}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$
- $\text{info(Age)} = (\frac{5}{14} * 0.971 + \frac{4}{14} * 0 + \frac{5}{14} * 0.971) = 0.694$
- IG = 0.940 - 0.694 = 0.247

Decision Tree using Information Theory

First we have to choose the root node: The one with the highest IG

Attributes		Y	N	Sum	I	Info	IG
Before	None	9	5	14	0.940	0.940	0
Age	Youth	2	3	5	0.971	0.694	0.247
	Middle-aged	4	0	4	0.000		
	Senior	3	2	5	0.971		
Income	High	3	2	5	0.971	0.925	0.015
	Medium	3	2	5	0.971		
	Low	3	1	4	0.811		
Student	Yes	8	1	9	0.503	0.581	**0.359**
	No	1	4	5	0.722		
Credit_rating	Excellent	3	3	6	1.000	0.892	0.048
	Good	6	2	8	0.811		

We choose the node **student as root**. We then choose the node for the branch student = yes:

Decision Tree using Information Theory

Attributes		Y	N	Sum	I	Info	IG
Before	None	8	1	9	0.503	0.503	0.000
Age	Youth	2	0	2	0.000	0.361	0.143
	Middle-aged	3	0	3	0.000		
	Senior	3	1	4	0.811		
Income	High	3	0	3	0.000	0.361	0.143
	Medium	2	0	2	0.000		
	Low	3	1	4	0.811		
Credit_rating	Excellent	2	1	3	0.918	0.306	**0.197**
	Good	6	0	6	0.000		

We then choose the **node Credit_Rating** and the following rule: **If student and credit_rating = Good, then Yes.**

Decision Tree using Information Theory

For the branch student = no, we choose the following node:

Attributes		Y	N	Sum	I	Info	IG
Before	None	1	4	5	0.722	0.722	0.000
Age	Youth	0	3	3	0.000	0.000	**0.722**
	Middle-aged	1	0	1	0.000		
	Senior	0	1	1	0.00		
Income	High	0	2	2	0.000	0.551	0.171
	Medium	1	2	3	0.918		
	Low	0	0	0	0.000		
Credit_rating	Excellent	1	2	3	0.918	0.551	0.171
	Good	0	2	2	0.000		

We then choose the **node age** with the following rule.

- If not student and youth, then No ;
- If not student and middle-aged, then Yes ;
- If not student and senior, then No ;

We then need to choose the node for the branch Credit_Rating = Excellent.

Decision Tree using Information Theory

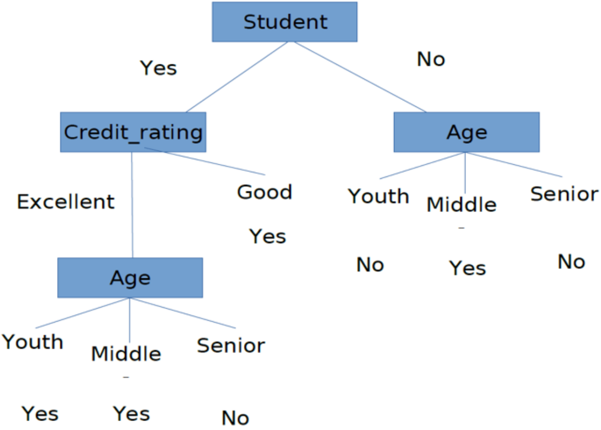
We then need to choose the node for the branch Credit_Rating = Excellent.

Attributes		Y	N	Sum	I	Info	IG
Before	None	2	1	3	0.918	0.918	0
Age	Youth	1	0	1	0	0.000	**0.918**
	Middle-aged	1	0	1	0		
	Senior	0	1	1	0		
Income	High	1	0	1	0	0.222	0.696
	Medium	0	0	0	0		
	Low	1	1	2	1.000		

We then choose the node age with the following rule.

- If student and credit_rating = excellent and Youth, then Yes ;
- If student and credit_rating = excellent and Middle-Aged, then Yes ;
- If student and credit_rating = excellent and senior, then Yes ;

Decision Tree using Information Theory



Using the rule: **If student and credit rating = Good then Yes**
We can conclude that Edward will buy a new computer

Decision Tree Classifier

```
import pandas as pd
pd.set_option('display.max_colwidth', None)
computer = pd.read_csv('/Users/catherine/Desktop/NLP/MachineLearning/MachineLearning202
print(computer.head(14), "\n")

Edward = pd.read_csv('/Users/catherine/Desktop/NLP/MachineLearning/MachineLearning2021/
print(Edward.head())
```

```
            Age  Income Student Credit rating Buy new Computer
0         Youth    High      No          Good               No
1         Youth    High      No     Excellent               No
2   Middle-aged    High     Yes          Good              Yes
3        Senior  Medium     Yes          Good              Yes
4        Senior     Low     Yes          Good              Yes
5        Senior     Low     Yes     Excellent               No
6   Middle-aged     Low     Yes     Excellent              Yes
7         Youth  Medium      No          Good               No
8         Youth     Low     Yes          Good              Yes
9        Senior  Medium     Yes          Good              Yes
10        Youth    High     Yes     Excellent              Yes
11  Middle-aged  Medium      No     Excellent              Yes
12       Senior  Medium      No     Excellent               No
13  Middle-aged    High     Yes          Good              Yes

      Age Income Student Credit rating
0  Senior   High     Yes          Good
```

Decision Tree Classifier: Prediction

```
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier

DT_classifier = DecisionTreeClassifier()

le = preprocessing.LabelEncoder()

x_train = computer[["Age", "Income", "Student", "Credit rating"]]
#converts to 0 and 1
x_train = pd.DataFrame(columns=x_train.columns, data=le.fit_transform(x_train.values.flatten()).reshape(x_train.shape))
print(x_train)
y_train = le.fit(computer["Buy new Computer"])
y_train = le.transform(computer["Buy new Computer"])#converts to 0 and 1
print("y_train =", y_train)
x_test = Edward[["Age", "Income", "Student", "Credit rating"]]
x_test = pd.DataFrame(columns=x_test.columns, data=le.fit_transform(x_test.values.flatten()).reshape(x_test.shape))

# we want to predict if Edward will buy a new computer
DT_classifier.fit(x_train, y_train)

y_pred = DT_classifier.predict(x_test)
print("")
print("Edward  =", y_pred)
```

```
    Age  Income  Student  Credit rating
0     9       2        6              1
1     9       2        6              0
2     5       2        8              1
3     7       4        8              1
4     7       3        8              1
5     7       3        8              0
6     5       3        8              0
7     9       4        6              1
8     9       3        8              1
9     7       4        8              1
10    9       2        8              0
11    5       4        6              0
12    7       4        6              0
13    5       2        8              1
y_train = [0 0 1 1 1 0 1 0 1 1 1 1 0 1]

Edward  = [1]
```

Decision Tree Classifier: Performance

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split

Xd_train_dt, Xd_test_dt, y_train_dt, y_test_dt = train_test_split(x_train, y_train, test_size=0.35)
print(Xd_train_dt)
print("y_train = ",y_train_dt, "\n")
print(Xd_test_dt)

DT_classifier.fit(Xd_train_dt, y_train_dt)

y_pred = DT_classifier.predict(Xd_test_dt)

DT_Accuracy = accuracy_score(y_test_dt, y_pred)

print("y_test = ",y_test_dt)
print("y_pred = ",y_pred,"\n")
print("DT_Accuracy = ", DT_Accuracy, "\n")
print("confusion_matrix \n", confusion_matrix(y_test_dt, y_pred))
```

```
    Age  Income  Student  Credit rating
7     9       4        6              1
9     7       4        8              1
5     7       3        8              0
6     5       3        8              0
11    5       4        6              0
1     9       2        6              0
3     7       4        8              1
10    9       2        8              0
2     5       2        8              1
y_train =  [0 1 0 1 1 0 1 1 1]

    Age  Income  Student  Credit rating
12    7       4        6              0
0     9       2        6              1
4     7       3        8              1
8     9       3        8              1
13    5       2        8              1
y_test =  [0 0 1 1 1]
y_pred =  [0 0 1 1 1]

DT_Accuracy =  1.0

confusion_matrix
 [[2 0]
 [0 3]]
```

Decision Tree Classifier: pre-pruning

Maximum depth of the tree can be used as a control variable for pre-pruning.

```
DT_classifier_en = DecisionTreeClassifier(criterion="entropy", max_depth=2)

DT_classifier_en.fit(Xd_train_dt, y_train_dt)

y_pred = DT_classifier_en.predict(Xd_test_dt)

DT_Accuracy_en = accuracy_score(y_test_dt, y_pred)

print("y_test = ",y_test_dt)
print("y_pred = ",y_pred,"\n")
print("DT_Accuracy_en = ", DT_Accuracy_en, "\n")
print("confusion_matrix \n", confusion_matrix(y_test_dt, y_pred))
```

```
y_test =  [0 0 1 1 1]
y_pred =  [0 0 1 1 1]

DT_Accuracy_en =  1.0

confusion_matrix
 [[2 0]
 [0 3]]
```

Decision Tree Classifier: Feature selection

```
best_features = SelectKBest(score_func=chi2, k=4)

fit = best_features.fit(Xd_train_dt,y_train_dt)
df_scores = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(Xd_train_dt.columns)

# concatenate dataframes
feature_scores = pd.concat([df_columns, df_scores],axis=1)
feature_scores.columns = ['Feature_Name','Score']  # name output columns
print(feature_scores.nlargest(4,'Score'))  # print all 4 features
```

```
    Feature_Name     Score
0            Age  1.142857
2        Student  0.272727
3  Credit rating  0.125000
1         Income  0.017857
```

```
best_features = SelectKBest(score_func=mutual_info_classif, k=4)

fit = best_features.fit(Xd_train_dt,y_train_dt)
df_scores = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(Xd_train_dt.columns)

# concatenate dataframes
feature_scores = pd.concat([df_columns, df_scores],axis=1)
feature_scores.columns = ['Feature_Name','Score']  # name output columns
print(feature_scores.nlargest(4,'Score'))  # print all 4 features
```

```
    Feature_Name     Score
2        Student  0.942725
0            Age  0.079762
1         Income  0.000000
3  Credit rating  0.000000
```

```
best_features = SelectKBest(score_func=f_classif, k=4)

fit = best_features.fit(Xd_train_dt,y_train_dt)
df_scores = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(Xd_train_dt.columns)

# concatenate dataframes
feature_scores = pd.concat([df_columns, df_scores],axis=1)
feature_scores.columns = ['Feature_Name','Score']  # name output columns
print(feature_scores.nlargest(4,'Score'))  # print all 4 features
```

```
    Feature_Name     Score
0            Age  3.500000
2        Student  2.333333
3  Credit rating  0.179487
1         Income  0.056911
```