Tubes1A_13517115

February 2, 2020

Tugas Besar Bagian A

Oleh: - Ignatius Timothy Manullang / 13517044 - Fatur Rahman / 13517056 - Fata Nugraha / 13517109 - Edward Alexander Jaya / 13517115

1. Pembacaan Dataset

[1]:
```
# Import all dependencies

from id3 import Id3Estimator
from id3 import export_text

from sklearn.preprocessing import LabelEncoder
from sklearn.datasets import load_iris
from sklearn import tree

import pandas as pd
import numpy as np

import pprint

import warnings
warnings.filterwarnings("ignore")
```

```
/usr/local/lib/python3.7/site-packages/sklearn/externals/six.py:31:
FutureWarning: The module is deprecated in version 0.21 and will be removed in
version 0.23 since we've dropped support for Python 2.7. Please rely on the
official version of six (https://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", FutureWarning)
```

Pembelajaran dataset Iris

[2]:
```
# Load iris
iris = load_iris()
```

Decision Tree Model for iris:

[3]:
```
decision_tree = tree.DecisionTreeClassifier()
decision_tree = decision_tree.fit(iris.data, iris.target)
```

```
DecisionTreeModel = tree.export_text(decision_tree,␣
 ↪feature_names=iris['feature_names'])
print(DecisionTreeModel)
```

```
|--- petal width (cm) <= 0.80
|   |--- class: 0
|--- petal width (cm) >  0.80
|   |--- petal width (cm) <= 1.75
|   |   |--- petal length (cm) <= 4.95
|   |   |   |--- petal width (cm) <= 1.65
|   |   |   |   |--- class: 1
|   |   |   |--- petal width (cm) >  1.65
|   |   |   |   |--- class: 2
|   |   |--- petal length (cm) >  4.95
|   |   |   |--- petal width (cm) <= 1.55
|   |   |   |   |--- class: 2
|   |   |   |--- petal width (cm) >  1.55
|   |   |   |   |--- petal length (cm) <= 5.45
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- petal length (cm) >  5.45
|   |   |   |   |   |--- class: 2
|   |--- petal width (cm) >  1.75
|   |   |--- petal length (cm) <= 4.85
|   |   |   |--- sepal width (cm) <= 3.10
|   |   |   |   |--- class: 2
|   |   |   |--- sepal width (cm) >  3.10
|   |   |   |   |--- class: 1
|   |   |--- petal length (cm) >  4.85
|   |   |   |--- class: 2
```

ID3 Model for iris:

[4]:
```
estimator = Id3Estimator()
estimator = estimator.fit(iris.data, iris.target)

ID3Model = export_text(estimator.tree_, feature_names=iris['feature_names'])
print(ID3Model)
```

```
petal length (cm) <=2.45: 0 (50)
petal length (cm) >2.45
|   petal width (cm) <=1.75
|   |   sepal length (cm) <=7.10
|   |   |   sepal width (cm) <=2.85: 1 (27/4)
|   |   |   sepal width (cm) >2.85: 1 (22)
|   |   sepal length (cm) >7.10: 2 (1)
|   petal width (cm) >1.75
```

```
|   |   sepal length (cm) <=5.95
|   |   |   sepal width (cm) <=3.10: 2 (6)
|   |   |   sepal width (cm) >3.10: 1 (1)
|   |   sepal length (cm) >5.95: 2 (39)
```

Pembelajaran dataset play-tennis

[5]:
```
# Read play-tennis dataset
df = pd.read_csv('datasets/play_tennis.csv')

# Then drop df['day']
df = df.drop('day', axis=1)
```

[6]:
```
# Define target variable, which is the variable of classes
target = df['play']

print("Target values: ")
print(target)

# Drop play attribute
df = df.drop('play', axis=1)
```

```
Target values:
0      No
1      No
2     Yes
3     Yes
4     Yes
5      No
6     Yes
7      No
8     Yes
9     Yes
10    Yes
11    Yes
12    Yes
13     No
Name: play, dtype: object
```

[7]:
```
# Variable to store encoded values
df_encoded = df

# Use label encoder to encode data
LE = LabelEncoder()

# Feature names for tree generation purposes
feature_names_var = ["outlook", "temp", "humidity", "wind"]
```

[8]:
```
# Store as a map: key -> encoded value, value -> real value
dictOfValues = {}

for key in feature_names_var:
    # Encode the data
    df_encoded[key] = LE.fit_transform(df[key])

    # Map encoded values with real values:
    dictOfValues[key] = {}

    for index in range(len(df_encoded[key])):
        encoded_value = df_encoded[key][index]
        real_value = LE.inverse_transform(df[key])[index]
        dictOfValues[key][encoded_value] = real_value

for key in feature_names_var:
    # Sort
    dictOfValues[key] = sorted(dictOfValues[key].items(), key=lambda x: x[0])
```

[9]:
```
# Transpose the encoded data
transposed_df_encoded = df_encoded.transpose()

# Define data variable
data = []
for index in range (0, 14):
    data.append(transposed_df_encoded[index])
```

Hasil pembelajaran:

Decision Tree Model for play-tennis:

[10]:
```
decision_tree = tree.DecisionTreeClassifier()
decision_tree = decision_tree.fit(data, target)
DecisionTreeModel = tree.export_text(decision_tree,␣
 ↪feature_names=feature_names_var)

print("Tree:")
print(DecisionTreeModel)
```

```
Tree:
|--- outlook <= 0.50
|   |--- class: Yes
|--- outlook >  0.50
|   |--- humidity <= 0.50
|   |   |--- outlook <= 1.50
|   |   |   |--- wind <= 0.50
|   |   |   |   |--- class: No
|   |   |   |--- wind >  0.50
```

```
|   |   |   |   |--- class: Yes
|   |   |--- outlook >  1.50
|   |   |   |--- class: No
|   |--- humidity >  0.50
|   |   |--- wind <= 0.50
|   |   |   |--- outlook <= 1.50
|   |   |   |   |--- class: No
|   |   |   |--- outlook >  1.50
|   |   |   |   |--- class: Yes
|   |   |--- wind >  0.50
|   |   |   |--- class: Yes
```

Outlook: - outlook <= 0.50, artinya outlook = Overcast - outlook <= 1.50, artinya outlook = Rain - selain itu, artinya outlook = Sunny

Temp: - temp <= 0.50, artinya temp = Cool - temp <= 1.50 atau <= 1.00, artinya temp = Hot - selain itu, artinya temp = Mild

Wind - wind <= 0.50, artinya wind = Weak - selain itu, artinya wind = Strong

Humidity - humidity <= 0.50, artinya humidity = High - selain itu, artinya humidity = High

ID3 Model for play-tennis:

[11]:
```
estimator = Id3Estimator()
fitEstimator = estimator.fit(data, target)

ID3Model = export_text(fitEstimator.tree_, feature_names=feature_names_var)

print("Tree:")
print(ID3Model)
```

```
Tree:

outlook <=0.50: Yes (4)
outlook >0.50
|   humidity <=0.50
|   |   temp <=1.50: No (2)
|   |   temp >1.50
|   |   |   wind <=0.50: No (1)
|   |   |   wind >0.50: No (1/1)
|   humidity >0.50
|   |   wind <=0.50
|   |   |   temp <=1.00: No (1)
|   |   |   temp >1.00: Yes (1)
|   |   wind >0.50: Yes (3)
```

Outlook: - outlook <= 0.50, artinya outlook = Overcast - outlook <= 1.50, artinya outlook = Rain - selain itu, artinya outlook = Sunny

Temp: - temp <= 0.50, artinya temp = Cool - temp <= 1.50 atau <= 1.00, artinya temp = Hot - selain itu, artinya temp = Mild

Wind - wind <= 0.50, artinya wind = Weak - selain itu, artinya wind = Strong

Humidity - humidity <= 0.50, artinya humidity = High - selain itu, artinya humidity = High

Berikut adalah arti setiap kode angka pada setiap atribut:

[12]:
```
pprint.pprint(sorted(dictOfValues.items(), key=lambda x: x[0]))
```

```
[('humidity', [(0, 'High'), (1, 'Normal')]),
 ('outlook', [(0, 'Overcast'), (1, 'Rain'), (2, 'Sunny')]),
 ('temp', [(0, 'Cool'), (1, 'Hot'), (2, 'Mild')]),
 ('wind', [(0, 'Strong'), (1, 'Weak')])]
```

2. Perbandingan algoritma pada hal 56 buku Machine Learning Tom Mitchell dengan kedua library DecisionTreeClassifier dan Id3Estimator
   2a. Penentuan atribut terbaik
   Algoritma ID3 pada buku Machine Learning Tom Mitchell vs library DecisionTreeClassifier
   Berbeda dengan algoritma ID3 pada buku Machine Learning Tom Mitchell, library DecisionTreeClassifier menggunakan Gini gain untuk menentukan atribut terbaik. Gini Gain merupakan penurunan Gini Impurity setelah kumpulan data dibagi pada suatu atribut. Gini Impurity merupakan kemungkinan klasifikasi salah dari elemen yang dipilih secara acak jika dilabel secara acak menurut distribusi kelas di suatu dataset. Gini gain didapatkan dengan mengkalkulasi Gini Impurity dari seluruh dataset, dikurangi dengan Gini Impurity dari setiap branch setelah dilakukan splitting. Atribut terbaik adalah atribut yang memiliki Gini gain tertinggi. Keterangan: G = Gini Impurity C = jumlah pembagi data / jumlah kelas data p(i) = kemungkinan secara acak memilih elemen dari kelas i
   Algoritma ID3 pada buku Machine Learning Tom Mitchell vs library Id3Estimator
   Kedua algoritma menentukan atribut terbaik dengan information gain, yang didapat dengan menghitung decrease dari entropy dan mencari atribut yang mengurangi entropy paling sedikit dari entropy sampel. Atribut terbaik adalah atribut yang memiliki information gain tertinggi.

2b. Penanganan label dari cabang setiap nilai atribut

Algoritma ID3 pada buku Machine Learning Tom Mitchell vs library DecisionTreeClassifier

Berbeda dengan Algoritma ID3 pada buku Machine Learning Tom Mitchell, label dari cabang setiap nilai atribut di DecisionTreeClassifier merupakan strategy yang digunakan untuk melakukan split pada setiap node, yang selain bergantung pada dataset, juga bergantung pada value dari splitter, yaitu 'best' berarti strategi terbaik yang digunakan, atau 'random' berarti strategi random terbaik yang digunakan.

Algoritma ID3 pada buku Machine Learning Tom Mitchell vs library Id3Estimator

Penanganan label dari cabang atribut di Algoritma ID3 pada buku Machine Learning Tom Mitchell sama dengan label dari cabang setiap nilai atribut di ID3Estimator, yaitu setiap value atau range dari value dari suatu atribut terbaik yang dapat mengklasifikasikan training example.

2c. Penentuan label jika examples kosong di cabang tersebut

Algoritma ID3 pada buku Machine Learning Tom Mitchell vs library DecisionTreeClassifier

Pada kedua algoritma, ketika examples kosong pada suatu cabang, di bawah cabang tersebut diberi daun dengan label berisi nilai yang paling umum dari target atribut di dalam examples. Sebagai contoh, pada data pasien rumah sakit, ketika tidak terdapat pasien pria yang bergolongan darah AB dan secara umum pasien bergolongan darah AB memiliki label 2, maka pasien pria AB akan diberi label 2.

Algoritma ID3 pada buku Machine Learning Tom Mitchell vs library Id3Estimator

Pada kedua algoritma, ketika examples kosong pada suatu cabang, di bawah cabang tersebut diberi daun dengan label berisi nilai yang paling umum dari target atribut di dalam examples. Sebagai contoh, pada data pasien rumah sakit, ketika tidak terdapat pasien pria yang bergolongan darah AB dan secara umum pasien bergolongan darah AB memiliki label 2, maka pasien pria AB akan diberi label 2.

2d. Penanganan atribut kontinu

Algoritma ID3 halaman 56 Machine Learning Tom Mitchell vs DecisionTreeClassifier

Pada algoritma ID3 di halaman 56 Machine Learning Tom Mitchell, atribut kontinu dianggap diskrit sehingga untuk setiap nilai yang mungkin dari examples pada atribut tersebut akan diproses satu persatu. Pada algoritma DecisionTreeClassifier, untuk atribut kontinu A, algoritma akan secara dinamik membentuk atribut boolean baru Ac yang bernilai true jika A < c dan bernilai false untuk sebaliknya. c dipilih dengan cara melakukan sorting terlebih dahulu examples berdasarkan atribut A kemudian dicari examples yang bertetangga yang klasifikasinya berbeda, kemudian dapat dihasilkan set kandidat batas c yang berada ditengah antara nilai-nilai atribut A tersebut. Dari set kandidat c tersebut dipilih c yang menghasilkan information gain terbaik.

Algoritma ID3 halaman 56 Machine Learning Tom Mitchell vs Id3Estimator

Pada algoritma ID3 di halaman 56 Machine Learning Tom Mitchell, atribut kontinu dianggap diskrit sehingga untuk setiap nilai yang mungkin dari examples pada atribut tersebut akan diproses satu persatu. Pada algoritma Id3Estimator, untuk atribut kontinu A, algoritma akan secara dinamik membentuk atribut boolean baru Ac yang bernilai true jika A < c dan bernilai false untuk sebaliknya. c dipilih dengan cara melakukan sorting terlebih dahulu examples berdasarkan atribut A kemudian dicari examples yang bertetangga yang klasifikasinya berbeda, kemudian dapat dihasilkan set kandidat batas c yang berada ditengah antara nilai-nilai atribut A tersebut. Dari set kandidat c tersebut dipilih c yang menghasilkan information gain terbaik.

2e. Penanganan atribut dengan missing values

Algoritma ID3 halaman 56 Machine Learning Tom Mitchell vs DecisionTreeClassifier

Kedua algoritma mengabaikan atau tidak mendukung missing value (atau dianggap sebagai value baru jika missing value tersebut dikategorikan (e.g. diberi nilai sebagai nan, None). Sebagai contoh, pada data pasien rumah sakit, beberapa pasien bisa jadi tidak memiliki data golongan darah. Sehingga pada pohon keputusan “golongan darah = tidak ada” adalah sebuah cabang yang berbeda.

Algoritma ID3 halaman 56 Machine Learning Tom MItchell vs Id3Estimator

Kedua algoritma mengabaikan atau tidak mendukung missing value (atau dianggap sebagai value baru jika missing value tersebut dikategorikan (e.g. diberi nilai sebagai nan, None). Sebagai

contoh, pada data pasien rumah sakit, beberapa pasien bisa jadi tidak memiliki data golongan darah. Sehingga pada pohon keputusan “golongan darah = tidak ada” adalah sebuah cabang yang berbeda.

2f. Pruning dan parameter confidence

Algoritma ID3 pada buku Machine Learning Tom Mitchell vs library DecisionTreeClassifier

Berbeda dengan algortima ID3 pada buku Machine Learning Tom Mitchell yang tidak melakukan pruning sama sekali, DecisionTreeClassifier melakukan pruning pada decision tree sampai memenuhi parameter yang mengatur ukuran tree, seperti max_depth, min_samples_leaf, dan max_leaf_nodes. Parameter confidence untuk algoritma ini adalah error pada validasi atribut untuk semua item pada dataset, dan mengacu pada Minimal Cost-Complexity Pruning. Minimal Cost-Complexity Pruning menghilangkan subtree yang ketika dihilangkan menghasilkan minimum dari error tersebut.

Algoritma ID3 pada buku Machine Learning Tom Mitchell vs library Id3Estimator

Algoritma ID3 pada buku Machine Learning Tom Mitchell tidak melakukan pruning untuk mengoptimasi decision tree, dan berjalan sampai decision tree dapat mengklasifikasi training example secara sempurna. Algoritma ini juga tidak menggunakan parameter confidence, yaitu level yang digunakan untuk melakukan pruning. Berbeda dengan algoritma ID3 pada buku Machine Learning Tom Mitchell yang tidak melakukan pruning sama sekali, Id3Estimator menggunakan parameter prune yang merupakan boolean yang jika diset sebagai true, maka akan melakukan pruning pada decision tree. Parameter confidence yang digunakan adalah error dari node yang akan diprune. Jika error dari node lebih kecil dibandingkan error dari childrennya, maka node tersebut akan diprune. Pruning dilakukan sampai memenuhi parameter yang mengatur ukuran tree, yaitu max_depth.