Praktikum 5
Algoritma Klasifikasi Decision Tree

Tujuan

Setelah mengikuti praktikum ini, mahasiswa diharapkan mampu:

1. Memahami konsep algoritma klasifikasi decision tree.
2. Menentukan kriteria percabangan yang tepat pada decision tree.
3. Mengimplementasikan algoritma decision tree dengan berbagai kriteria percabangan.

Dasar Teori

Decision tree merupakan sebuah algoritma klasifikasi yang memanfaatkan struktur data tree untuk menentukan kelas dari suatu data. Terdapat tiga jenis node pada decision tree:

1. Root node, merupakan node yang tidak memiliki edge masukan dan memiliki nol atau lebih edge keluaran.
2. Internal node, memiliki tepat satu edge masukan dan memiliki dua atau lebih edge keluaran. Pada decision tree, internal node berfungsi menampung variabel yang mengalami percabangan.
3. Leaf atau terminal node, mempunyai tepat satu edge masukan dan tidak mempunyai edge keluaran. Pada decision tree, leaf berfungsi untuk menyimpan kelas suatu data.

Tahapan pertama dalam pembentukan decision tree adalah menentukan variabel apa yang mengalami percabangan. Idealnya, setelah percabangan terbentuk distribusi kelas yang homogen. Terdapat beberapa kriteria yang dapat digunakan untuk pembentukan cabang:

1. Gini Index

$$GINI(t) = 1 - \sum_{j} [p(j|t)]^2$$

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_I}{n} GINI(i)$$

Keterangan:
t : sebuah node pada tree
p(j|t) : frekuensi kelas *j* pada node *t*
n_i : Banyaknya data pada *child* i
n : Banyaknya data pada node *p*

2. Information Gain

$$GAIN_{split} = Enttropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

$$Entropy(t) = -\sum_{j} p(j|t) log_2 p(j|t)$$

3. Gain Ratio

$$GainRATIO_{split} = \frac{GAIN_{split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}$$

Pembentukan tree pada decision tree dilakukan menggunakan alur umum sebagai berikut:

1. Hitung kriteria pembentukan cabang (pilih salah satu) untuk setiap fitur pada data.
2. Lakukan percabangan menggunakan nilai kriteria tertinggi (Information Gain & Gain Ratio) atau kriteria terendah (Gini Index).
3. Ulangi langkah 1-2 untuk membentuk cabang menggunakan fitur yang tersisa.

Praktikum

1. Import Data

Praktikum kali ini menggunakan dataset Car Evaluation Dataset dari UCI Machine Learning Repository. Dataset ini telah digunakan pada praktikum sebelumnya. Detail fitur dapat Anda pelajari pada link yang tersedia.

Unduh dataset yang akan digunakan pada praktikum kali ini. Anda dapat menggunakan aplikasi wget untuk mendownload dataset dan menyimpannya dalam Google Colab. Jalankan cell di bawah ini untuk mengunduh dataset.

```
! wget --no-check-certificate
"https://drive.google.com/uc?export=download&id=1RXcXrwkVoOI-
hPvfJM1-FlmO_bB7M3YJ&amp;quot;" -O car_sample.csv
```

Setelah dataset berhasil diunduh, langkah berikutnya adalah membaca dataset dengan memanfaatkan fungsi readcsv dari library pandas. Lakukan pembacaan berkas csv ke dalam dataframe dengan nama data menggunakan fungsi readcsv. Jangan lupa untuk melakukan import library pandas terlebih dahulu

```
import pandas as pd
import numpy as np
data = pd.read_csv('car_sample.csv')
```

Cek isi dataset Anda dengan menggunakan perintah **head()**

```
data.head()
```

2. Membagi Data Menjadi Data Latih dan Data Uji

Metode pembelajaran mesin memerlukan dua jenis data :

1. Data latih : Digunakan untuk proses training metode klasifikasi
2. Data uji : Digunakan untuk proses evaluasi metode klasifikasi

Data uji dan data latih perlu dibuat terpisah (mutualy exclusive) agar hasil evaluasi lebih akurat. Data uji dan data latih dapat dibuat dengan cara membagi dataset dengan rasio tertentu, misalnya 80% data latih dan 20% data uji.

Library Scikit-learn memiliki fungsi train_test_split pada modul model_selection untuk membagi dataset menjadi data latih dan data uji. Bagilah dataset anda menjadi dua, yaitu **data_latih** dan **data_uji**. Agar pengacakan data dilakukan secara konstan, parameter **random_state** diisi dengan nilai integer tertentu, pada praktikum ini diset 101. Kemudian, nilai indeks pada data latih dan data uji diatur ulang agar berurutan nilainya.

```
from sklearn.model_selection import train_test_split
data_latih,data_uji =
train_test_split(data,test_size=0.2,random_state=101)
data_latih.reset_index(drop=True)
data_uji.reset_index(drop=True)
```

Tampilkan banyaknya data pada **data_latih** dan **data_uji**. Seharusnya **data_latih** terdiri dari 208 data, dan **data_uji** terdiri dari 52 data.

```
print(data_uji.shape[0])
print(data_latih.shape[0])
```

3. Menghitung GINI

Nilai Gini merupakan salah satu kriteria penentu variabel apa yang akan digunakan untuk membentuk cabang pada decision tree. Variabel dengan nilai Gini terendah akan digunakan sebagai pembentukan cabang

Buatlah fungsi bernama **hitung_gini** yang berfungsi menghitung nilai Gini dari suatu nilai pada sebuah variabel.

```
def hitung_gini(kolom_kelas):
  elemen,banyak = np.unique(kolom_kelas,return_counts = True)
  nilai_gini = 1 - np.sum([(banyak[i]/np.sum(banyak))**2 for i in
range(len(elemen))])
  return nilai_gini
```

Fungsi **hitung_gini** menerima nilai kelas pada variabel tertentu dengan nilai tertentu. Nilai dan frekuensi dari masing-masing kelas dihitung menggunakan fungsi np.unique dari library numpy, kemudian dilakukan kalkulasi nilai GINI.

Nilai GINI total dari sebuah variabel dihitung menggunakan fungsi **gini_split**.

```
def gini_split(data,nama_fitur_split,nama_fitur_kelas):
    nilai,banyak=
    np.unique(data[nama_fitur_split],return_counts=True)
    gini_split =
np.sum([(banyak[i]/np.sum(banyak))*hitung_gini(data.where(data
[nama_fitur_split]==nilai[i]).dropna()[nama_fitur_kelas]) for
i in range(len(nilai))])
    return gini_split
```

Fungsi **gini_split** menerima input berupa data, nama fitur yang akan dilakukan percabangan, serta nama fitur yang mengandung kelas data. Tahapan pertama dari fungsi ini adalah mendapatkan setiap elemen dari variabel yang akan dilakukan percabangan beserta frekuensinya. Selanjutnya dilakukan perhitungan nilai GINIsplit sesuai persamaan yang diberikan, dengan menghitung terlebih dahulu nilai GINI pada masing-masing elemennya.

Ujilah fungsi **gini_split** menggunakan **data_latih** pada variabel **buying** dan variabel kelas bernama **class**.

```
gini_split(data_latih,"buying","class")
```

4. Pembentukan Pohon

Pembentukan pohon dilakukan secara rekursif. Seperti metode rekursif pada umumnya, perlu ditentukan kondisi berhenti terlebih dahulu. Kondisi berhenti pada pembentukan pohon adalah:

1. Jika kelas pada data hanya ada satu, kembalikan kelas tersebut
2. Jika fitur data = 0 (tidak ada fitur yang tersisa), kembalikan kelas dari parent
3. Jika data kosong (tidak ada data), kembalikan kelas dengan frekuensi terbanyak

Selain kondisi berhenti tersebut, dilakukan pembentukan pohon secara rekursif menggunakan fungsi **buat_tree**.

```
def
buat_tree(data,data_awal,daftar_fitur,nama_fitur_kelas,kelas_parent_node=None):
  #jika hanya ada satu kelas pada data
  if len(np.unique(data[nama_fitur_kelas])) <= 1:
    return np.unique(data[nama_fitur_kelas])[0]
  #jika data kosong
  elif len(data)==0:
    return
np.unique(data_awal[nama_fitur_kelas])[np.argmax(np.unique(data_awal
[nama_fitur_kelas],return_counts=True)[1])]
```

```
  #jika tidak ada fitur yang terisa
  elif len(daftar_fitur) ==0:
    return kelas_parent_node
  else:
    kelas_parent_node =
np.unique(data[nama_fitur_kelas])[np.argmax(np.unique(data[nama_fitur_kelas]
,return_counts=True)[1])]
    nilai_split = [gini_split(data,fitur,nama_fitur_kelas) for fitur in
daftar_fitur]
    index_fitur_terbaik = np.argmin(nilai_split)
    fitur_terbaik = daftar_fitur[index_fitur_terbaik]
    tree = {fitur_terbaik:{}}
    daftar_fitur = [i for i in daftar_fitur if i != fitur_terbaik]
    for nilai in np.unique(data[fitur_terbaik]):
      sub_data = data.where(data[fitur_terbaik] == nilai).dropna()
      subtree =
buat_tree(sub_data,data_awal,daftar_fitur,nama_fitur_kelas,kelas_parent_node)
      tree[fitur_terbaik][nilai]=subtree
    return tree
```

Fungsi **buat_tree** memiliki parameter input berupa :

1. data : merupakan data yang membentuk tree. Seiring dengan bertambahnya kedalaman tree, maka akan ada fitur yang dihapus pada data
2. data_awal : merupakan data latih awal yang tidak mengalami penghapusan fitur.
3. daftar_fitur : merupakan daftar fitur-fitur yang digunakan dalam pembentukan tree. Seiring dengan bertambahnya kedalaman tree, maka akan ada fitur yang dihapus dari daftar_fitur.
4. nama_fitur_kelas : merupakan nama fitur pada data yang mengandung kelas data.
5. Kelas_parent_node : menunjukkan parent node dari suatu node

Variabel **kelas_parent_node** berisi kelas dengan frekuensi terbesar pada suatu node. Variabel ini berguna ketika kondisi berhenti nomor 2 terpenuhi (tidak ada fitur yang tersisa). Variabel **nilai_split** merupakan sebuah list yang berisi nilai kriteria percabangan (pada kasus ini, nilai GINIsplit) pada masing-masing fitur. **index_fitur_terbaik** dan **fitur_terbaik** berisi indeks dan nama fitur dengan nilai kriteria percabangan terendah. Selanjutnya, variabel **tree** berisi decision tree yang menggunakan struktur data dictionary dengan key berupa nama fitur dengan nilai kriteria percabangan tertinggi (**fitur_terbaik**). Selanjutnya, fitur dengan nilai kriteria percabangan tertinggi dihapus dari daftar fitur yang akan digunakan untuk membuat decision tree (**daftar_fitur**). Setelah itu, dilakukan looping pada masing-masing nilai di fitur **fitur_terbaik** untuk membentuk subtree dengan cara memanggil fungsi **buat_tree** secara rekursif.

Ujilah fungsi **buat_tree** menggunakan data latih yang tersedia.

```
tree = buat_tree(data_latih,data_latih,data_latih.columns[:-1],'class')
```

Tampilkan tree yang terbentuk. Gunakan library **pprint** untuk menampilkan dictionary secara teratur.

```
from pprint import pprint
pprint(tree)
```

5. Proses Prediksi

Proses prediksi kelas pada data uji dilakukan dengan melakukan tree traversal sampai menemui leaf.

```
def prediksi(data_uji,tree):
  for key in list(data_uji.keys()):
    if key in list(tree.keys()):
      try:
        hasil = tree[key][data_uji[key]]
      except:
        return 1
      hasil = tree[key][data_uji[key]]
      if isinstance(hasil,dict):
        return prediksi(data_uji,hasil)
      else:
        return hasil
```

Fungsi **prediksi** menerima input berupa data uji dan tree hasil training, keduanya dalam struktur data dictionary. Tahapan pertama dalam prediksi adalah memperoleh semua key dari data uji. Kemudian, setiap key pada data uji akan dicek apakah menjadi root node pada tree. Jika benar, maka variabel **hasil** akan berisi *child* dari *root* pada tree dengan nilai yang sama dengan data uji. Jika variabel **hasil** merupakan sebuah dictionary, maka dilakukan pemanggilan fungsi **prediksi** secara rekursif. Jika tidak, maka variabel **hasil** merupakan leaf node yang berisi kelas dari data uji. Perhatikan bahwa terdapat penggunaan **try except** pada pencarian hasil. Hal ini dilakukan untuk mengantisipasi apabila ada nilai pada data uji yang tidak terdapat pada decision tree. Jika ini terjadi, maka kelas data uji diberi nilai 1.

6. Proses Pengujian

Lakukan pengujian menggunakan data uji. Kelas pada data uji perlu dihapus dan data uji perlu diubah menjadi dictionary.

```
data_uji_dict = data_uji.iloc[:,:-1].to_dict(orient = "records")
```

Lakukan pengujian terhadap keseluruhan data uji menggunakan looping.

```
hasil_prediksi_total = []
for i in range(len(data_uji_dict)):
  hasil_prediksi = prediksi(data_uji_dict[i],tree)
  hasil_prediksi_total.append(hasil_prediksi)
```

Bandingkan hasil prediksi dengan label sebenarnya. Hitunglah banyaknya data uji yang memiliki kelas prediksi sama dengan kelas sebenarnya.

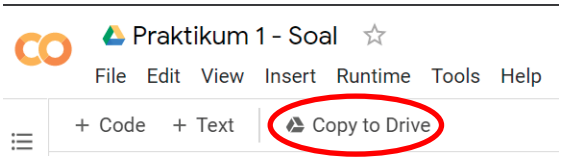
```
    print("Total prediksi benar: "
,sum(hasil_prediksi_total==data_uji['class']))
```

Tugas

Pada tugas kali ini Anda diminta memodifikasi metode pembentukan tree yang telah Anda agar metode tersebut menggunakan information gain sebagai dasar percabangan. Lengkapilah kerangka source code pada notebook yang tersedia dan jawablah pertanyaan yang ada.

Petunjuk pengerjaan soal:

1. Klik link berikut, pastikan Anda login menggunakan **akun student UB**.
   https://colab.research.google.com/drive/1himzMVkbDCjsw01ehFoE6K5QhTLiralv?usp=sharing
2. Klik tombol Copy to Drive



3. Beri nama file BAB 5-Nama-NIM.ipynb
4. Isilah cell yang kosong
5. Download file *.ipynb dengan cara klik **File -> Download .ipynb**
6. Kumpulkan file *.ipynb ke asisten (Google Classroom)