Dari tugas 06 sebelum ini, coba gunakan data yang sama, tapi gunakan pemprograman python, lalu buat tutorialnya.

Harap dapat dikumpulkan sebelum batas waktu yang ditentukan!

Tugas 07

Nama : Juminovario
NIM : 202420018
Kelas : MTI 23 Reg-A
MK : Advanced Database

Pada tutorial ini akan membahas bagaimana cara penggolahan data dengan algoritma apriori dengan tools rapid miner.

Sebelum masuk ke tahap penggolahan data, yang harus disiapkan adalah

• Data yang akan di olah

• Tools rapid miner

Untuk pengolahan data dengan algoritma apriori langkah pertama yang harus dilakukan adalah penetuan atribut, atribut yang di gunakan atau yang akan di pilih harus ada hubungan dan kaitan satu sama lain. Pada tutorial kali ini data yang akan digunakan adalah data penjualan pada sebuah toko furniture dan elektornik dengan 346 record dan 16 atribut tapi yang akan digunakan hanya 3 atribut saja.

Setelah penentuan atribut dilakukan, langkah selanjutnya yaitu tahap preprocessing data, pada tahap ini akan dilakukan beberapa hal, yaitu cleaning data dan transformas data.

Pada tahap cleaning dilakukan pembersihan data transaksi penjualan furniture dan elektronik. Tahap cleaning ini dilakukan dengan cara membuang data yang kosong atau data yang tidak sempurna kemudian membuang field-field yang tidak dibutuhkan agar tidak memperlambat dalam proses asosiasi dana mempercepat mendapatkan hasil pola gabungan.

Tahap selanjutnya adalah transformasi data, tahap ini akan dilakukan agar mempermudah saat memasukkan data ke dalam tools rapid miner dengan membuat inisial atau symbol symbol untuk data.

Seperti contoh berikut:

Tabel Inisial item pembelian data transaksi penjualan

No.	Items Pembelian
1	MC1 (mesin cuci 1 Tempat)
2	MC2 (mesin cuci 2 Tempat)
3	SP (speaker aktif)
4	RP (rak piring)
5	PFK (palung fadhil kaca)
6	LED24 (TV uk.24 dengan semua merk)
7	LED 32 (TV uk.32 dengan semua merk)
8	LED 43 (TV uk.43 dengan semua merk)
9	PR (parabola)
10	DG (digital)
11	LP2 (lemari pakaian 2 pintu)
12	LP3 (lemari pakaian 3 pintu)
13	LH (lemari hias)
14	LM (lemari mini)
15	MB (meja belajar)
16	KR (kursi)

Setelah dilakukan transformasi data, tahap selanjutnya adalah melakukan tranformasi data penjualan kedalam bentuk tabular. Berikut merupakan hasil transformasi data ke dalam bentuk tabular dapat dilihat pada Tabel:

Tabel Tabular atribut item pembelian data transaksi penjualan

cam	MC1	MC2	RP	PFK	LED24	LED32	……	JM
C1	0	0	0	0	0	0	……	0
C2	0	0	0	0	0	0	……	0
C3	1	0	0	0	0	0	……	0
C4	0	1	0	0	0	0	……	0
C5	0	0	0	0	0	0	……	0
C6	0	0	0	0	0	0	……	0
C7	0	0	0	0	0	0	……	0
…….	….	….	….	….	….	….	….	….
C346	0	0	0	0	1	0	……	0

Keterangan :

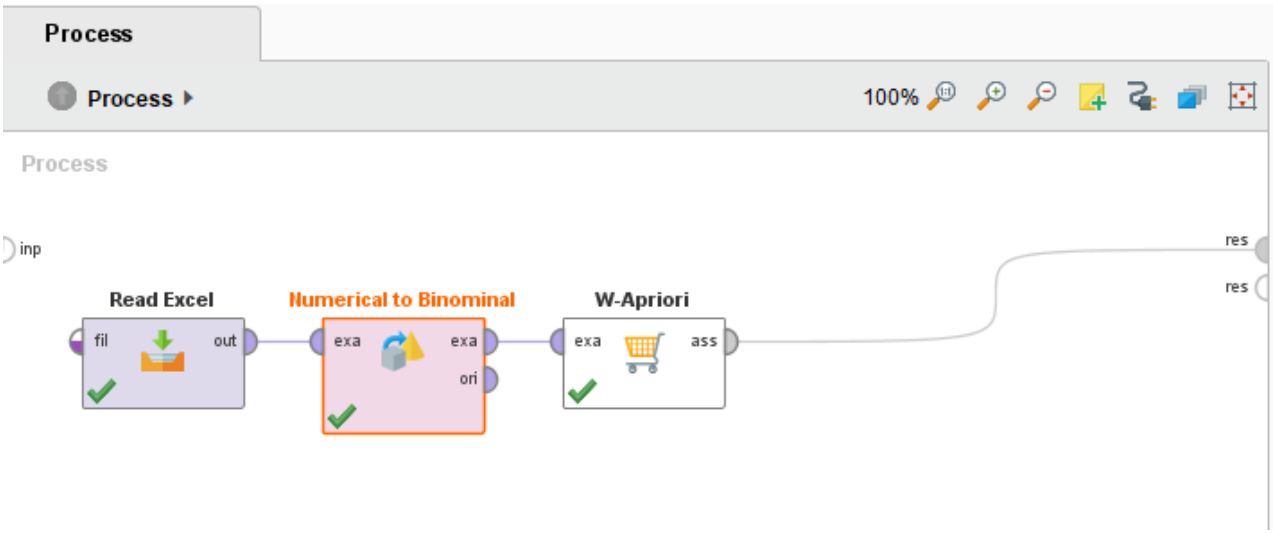
a) Customers merupakan nomor faktur setiap pembeli

b) MC1, MC2, RP, PFK, LED24 Dll merupakan barang-barang atau produk yang dijual oleh PT.Citra Mustika Pandawa cabang Kerinci.

c) 0 merupakan tanda bahwa barang tersebut tiak dibeli oleh pembeli

d) 1 merupakan tanda bahwa barang tersebut dibeli oleh pembeli

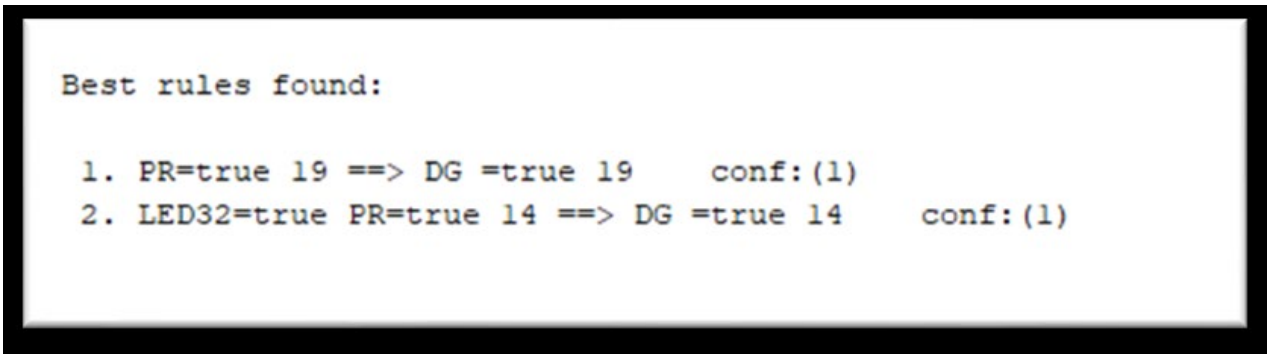
Setelah melalui tahap preprocessing dan transformasi data, selanjutnya adalah tahap asosiasi dengan menggunakan algoritma apriori untuk menentukan pola pembelian pelanggan. Hasil ini diukur dengan menggunakan nilai Support dan Confidence. Percobaan perhitungan ini menggunakan tools Rapid Miner dengan 346 record data. Dalam tahap imlementasi dengan rapidminer ini ada 3 operator yang digunakan dalam tools, yaitu: Read Exel, Numeric to Binominal dan W-Apriori. Operator pertama adalah Read Exel, operator ini berfungsi untuk tempat data dan akan diinputkan data yang sudah diolah. Karena data yang diolah berupa exel maka operator yang digunakan adalah read Exel. Operator kedua adalah Numeric to Binominal, operator ini berfungsi untuk merubah data yang tadinya numeric menjadi nominal dua nilai pada operator read Exel karena semua atribut dari masukan wajib merupakan bilangan binominal yaitu nilai true/false. Operator ketiga adalah W-Apriori, operator ini berfungsi untuk perhitungan algoritma apriori. Desain dari ketiga operator ini dapat dilihat pada gambar berikut:



Setelah ketiga operator terhubung dan dijalankan maka akan keluar hasil untuk item yang sering muncul dan saling berhubungan, pada data penjualan elektronik dan furniture ini hasil

yang didapat nilai minimal support 4% dan minimal confidance 90%. Hasil yang diperoleh dari rapidminer ada 2 rule yaitu:

1. Jika membeli parabola maka akan membeli digital
2. Jika membeli LED32 dan parabola maka akan memebeli digital



Demikian tutorial pengolahan data dengan algoritma apriori dengan tools rapidminer yang sangat sederhana ini, mohon maaf apabila banyak terdapat kesalahan, terima kasih.

Nama : Nanda Tri Haryati
NIM / Kelas : 202420016/MTI23-REG-A

Tugas 7

Dari tugas 06 sebelum ini, coba gunakan data yang sama, tapi gunakan pemprograman python, lalu buat tutorialnya.

Jawab :

Untuk pemrograman phyton nya saya menggunkan google colab karena laptop gak mendukung tuk install aplikasi python. Tiba-tiba latop error

Maaf ya Pak

Tutorialnya

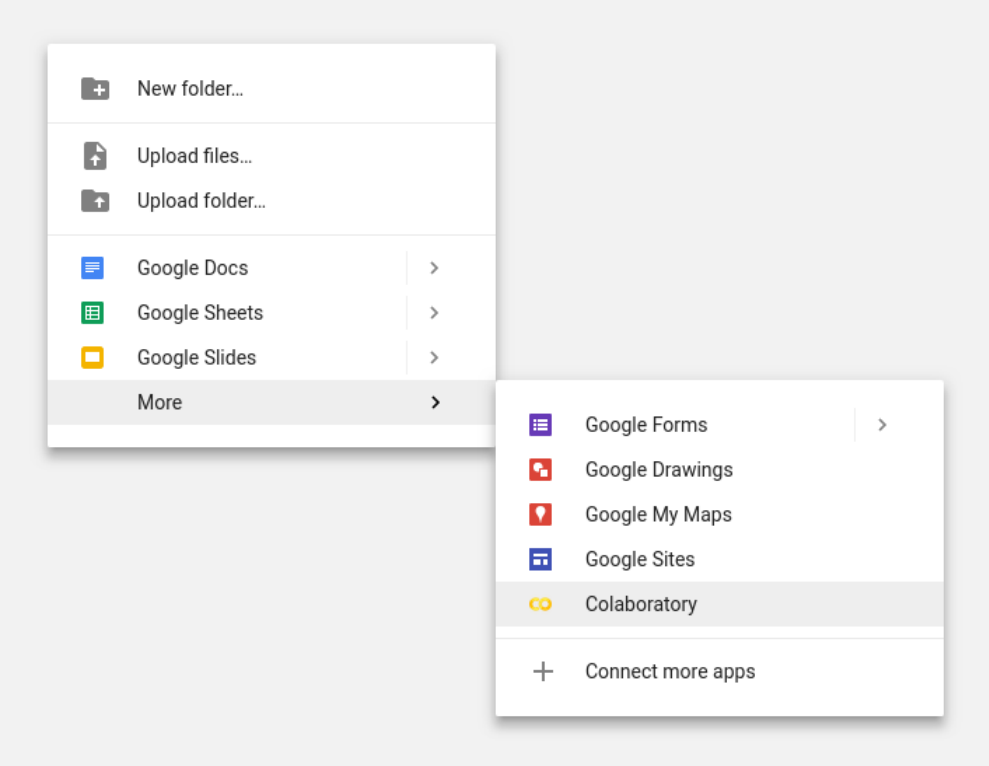
No	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	FALSE	No
2	Sunny	Hot	High	TRUE	No
3	Cloudy	Hot	High	FALSE	Yes
4	Rainy	Mild	High	FALSE	Yes
5	Rainy	Cool	Normal	FALSE	Yes
6	Rainy	Cool	Normal	TRUE	Yes
7	Cloudy	Cool	Normal	TRUE	Yes
8	Sunny	Mild	High	FALSE	No
9	Sunny	Cool	Normal	FALSE	Yes
10	Rainy	Mild	Normal	FALSE	Yes
11	Sunny	Mild	Normal	TRUE	Yes
12	Cloudy	Mild	High	TRUE	Yes
13	Cloudy	Hot	Normal	FALSE	Yes
14	Rainy	Mild	High	TRUE	No

Untuk dapat menggunakan Google Colab kita perlu menambahkan ekstensi baru ke Google Drive kita dengan cara klik tombol **New > More > Connect more apps** lalu tuliskan “colab” pada kolom search kemudian klik tombol **connect**.

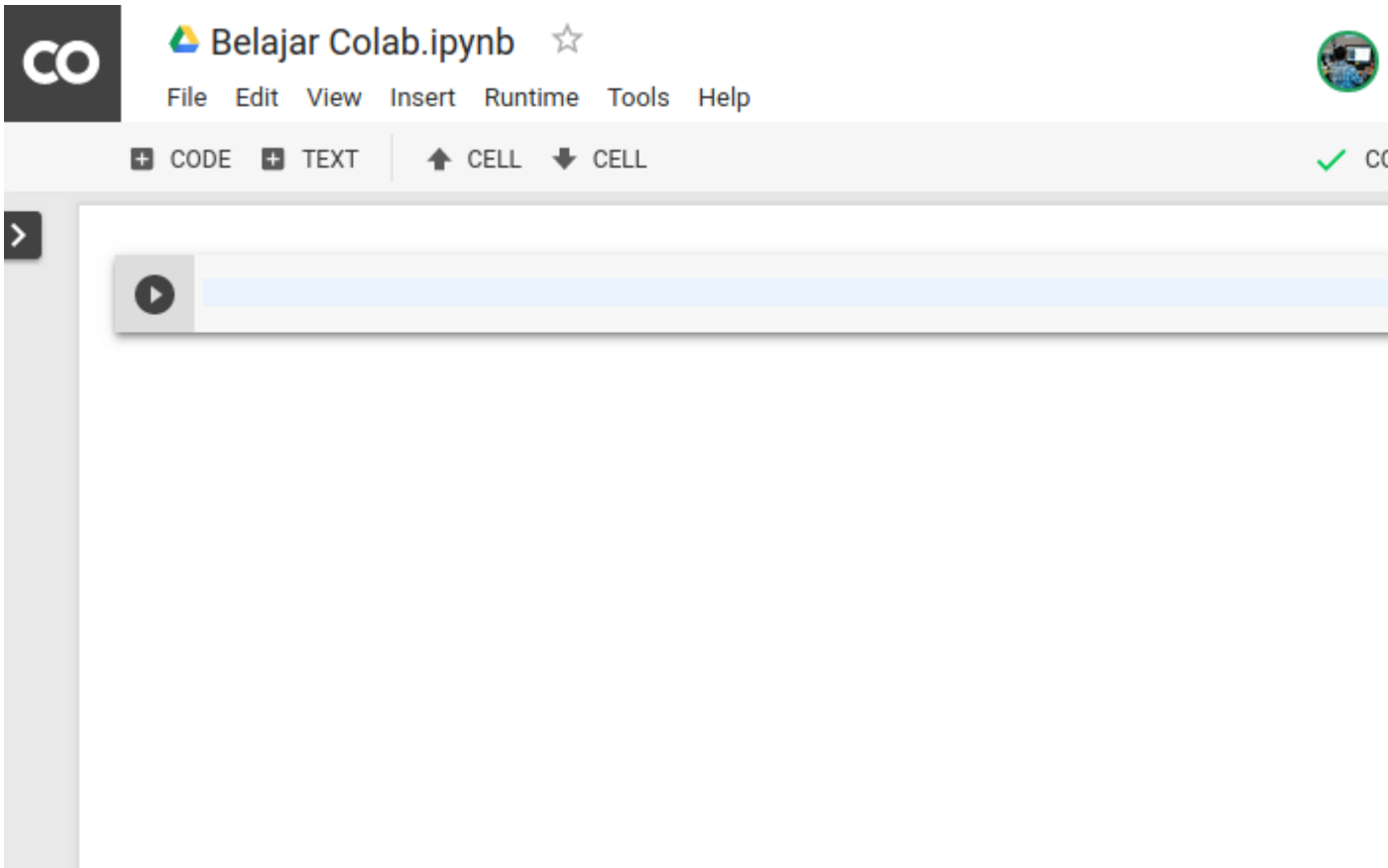
Membuat Notebook

Jika Colab sudah terintegrasi dengan Drive kita maka kita siap untuk menggunakannya, pertama kita buat direktori baru terlebih dahulu dengan cara klik tombol **New** kemudian pilih **Folder** lalu berikan nama direktori tersebut (misal: `Colab`) kemudian buat file Notebook baru dengan cara klik kanan pada area kosong didalam direktori yang baru saja kita buat **Klik kanan > More > Colaboratory**.

Nama : Nanda Tri Haryati
NIM / Kelas : 202420016/MTI23-REG-A

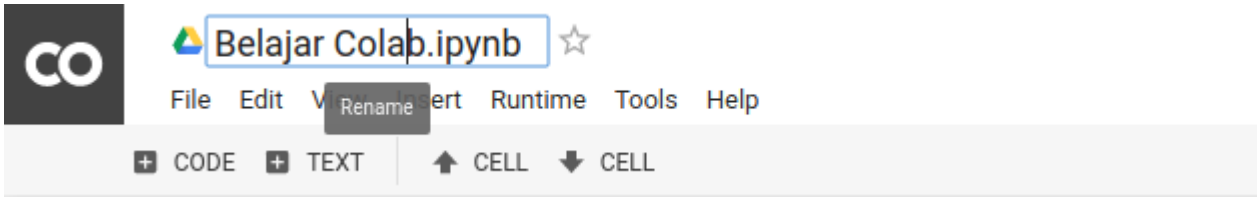


Maka tampilan awal dari Notebook kita adalah seperti berikut



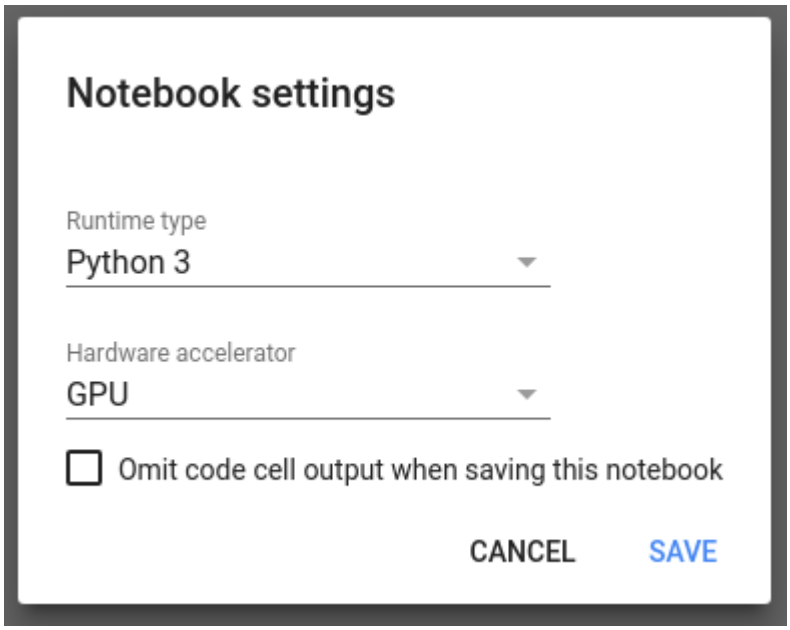
Nama : Nanda Tri Haryati
NIM / Kelas : 202420016/MTI23-REG-A

Ada baiknya kita berikan nama notebook kita dengan cara double klik pada area `Untitled0.ipynb` lalu tuliskan nama notebook seperti berikut



Setup GPU

Karena Colab menyediakan GPU gratis untuk penggunanya kita pun dapat memberikan pengaturan pada notebook untuk menggunakan layanan GPU gratis tersebut caranya klik menu **Edit > Notebook settings** kemudian ubah "Hardware accelerator" menjadi GPU dan kita juga dapat mengubah runtime versi Python pada notebook sedang aktif.



Menjalankan Kode Pertama

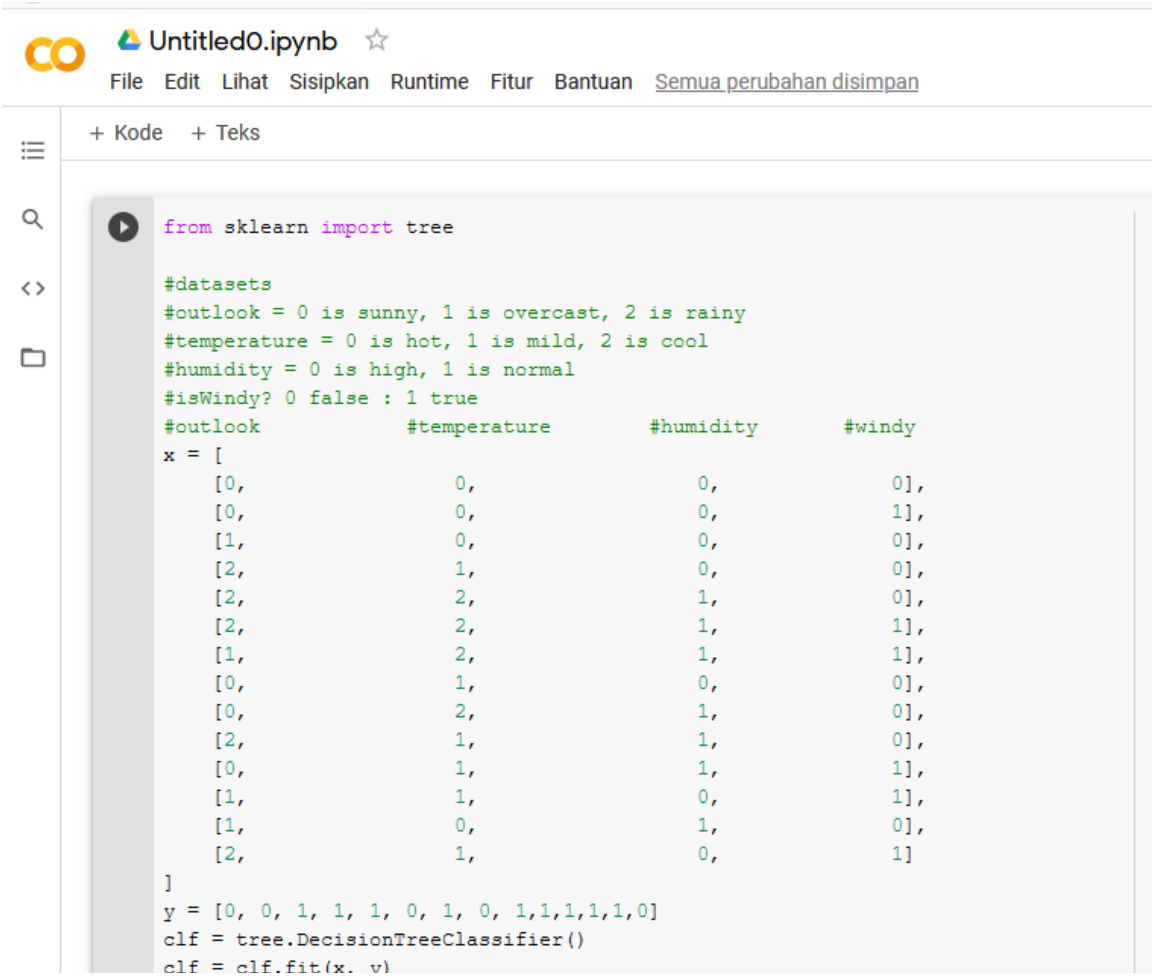
Proses setup sudah selesai dan mari kita mencoba beberapa operasi tipe data dasar, berikut potongan kodenya

```
from sklearn import tree

#datasets
#outlook = 0 is sunny, 1 is overcast, 2 is rainy
#temperature = 0 is hot, 1 is mild, 2 is cool
#humidity = 0 is high, 1 is normal
#isWindy? 0 false : 1 true
#outlook            #temperature        #humidity       #windy
x = [
    [0,                 0,                  0,              0],
    [0,                 0,                  0,              1],
```

Nama : Nanda Tri Haryati
NIM / Kelas : 202420016/MTI23-REG-A

```
    [1,                 0,                 0,                 0],
    [2,                 1,                 0,                 0],
    [2,                 2,                 1,                 0],
    [2,                 2,                 1,                 1],
    [1,                 2,                 1,                 1],
    [0,                 1,                 0,                 0],
    [0,                 2,                 1,                 0],
    [2,                 1,                 1,                 0],
    [0,                 1,                 1,                 1],
    [1,                 1,                 0,                 1],
    [1,                 0,                 1,                 0],
    [2,                 1,                 0,                 1]
]
y = [0, 0, 1, 1, 1, 0, 1, 0, 1,1,1,1,1,0]
clf = tree.DecisionTreeClassifier()
clf = clf.fit(x, y)


def prog() :
    outlook = int(input("Outlook? (0 is sunny, 1 overcast, 2 rainy) : "
))
    humidity = int(input("Temperature? (0 is hot, 1 is mild, 2 is cool)
 : "))
    temp = int(input("Humidty? (0 is high, 1 is normal) : "))
    isWindy = int(input("Is Windy? 0 false, 1 true : " ))

    if clf.predict([[outlook, humidity, temp, isWindy]]):
        print("\nYes,I Play golf")
    else:
        print("\nNo Play golf")


prog()
```

Nama : Nanda Tri Haryati
NIM / Kelas : 202420016/MTI23-REG-A



Untuk menjalankannya klik runtime / jalankan semuanya. Maka akan muncul seperti ini hasilnya

Nama : Nanda Tri Haryati
NIM / Kelas : 202420016/MTI23-REG-A

Selesai.

Nama : Oman Arrohman
Nim : 202420042
Kelas : MTI Reguler A
MK : Advanced Database

Tugas 7

Dari Tugas 06 sebelum ini, coba gunakan data yang sama, tapi gunakan pemprograman python, lalu buat tutorialnya.

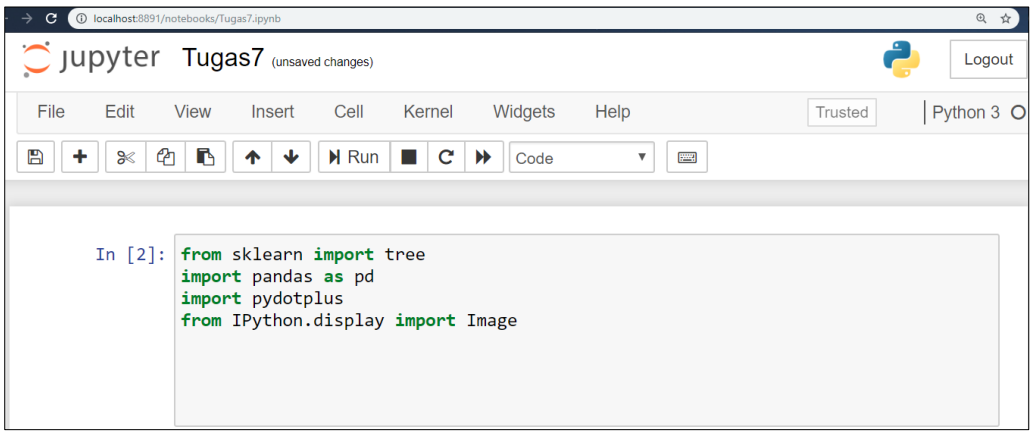
Dataset Berolahraga

Hari	Cuaca	Temperatur	Kecepatan Angin	Berolahraga
Hari ke 1	cerah	normal	pelan	ya
Hari ke 2	hujan	tinggi	pelan	tidak
Hari ke 3	cerah	normal	kencang	ya
Hari ke 4	cerah	normal	pelan	ya
Hari ke 5	hujan	tinggi	kencang	tidak
Hari ke 6	hujan	tinggi	pelan	ya
Hari ke 7	cerah	normal	kencang	tidak

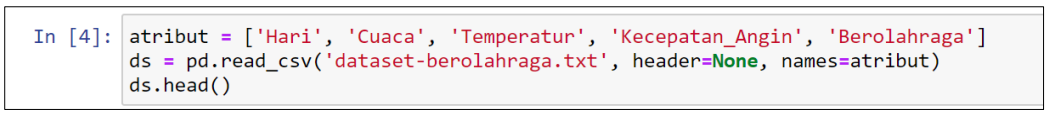
Tutorial atau tata cara Perhitungan dan Penerapan Algoritma C 4.5 Pada Phyton Dengan Jupyter Notebook

1. Pastikan Sudah menginstal **Phyton** dan **Jupyter Notebook**
2. Siapkan Dataset dengan format CSV
3. Pastikan Library pendukung seperti **Scikit-Learn**, **Pydotplus**, dan **Graphviz**, sedangkan untuk **pandas** dan **IPhyton.display** sudah tersedia sendiri pada saat menginstal phyton.
   Link dibawah untuk download :
   Scikit-learn > https://scikit-learn.org/stable/install.html
   Graphviz > https://www2.graphviz.org/Packages/stable/windows/10/cmake/Release/x64/
   Pydotplus > https://pypi.org/project/pydotplus/
4. Untuk **graphviz** silahkan atur path dan masukan ke **system environtment variables** sesuai dengan directory penginstalan pada komputer, misal : **C:\Program Files\Graphviz 2.44.1\bin;**
5. Setelah semua selesai lalu masuk ke halaman **Jupyter Notebook** > pilih **File > New** > **Phyton 3**

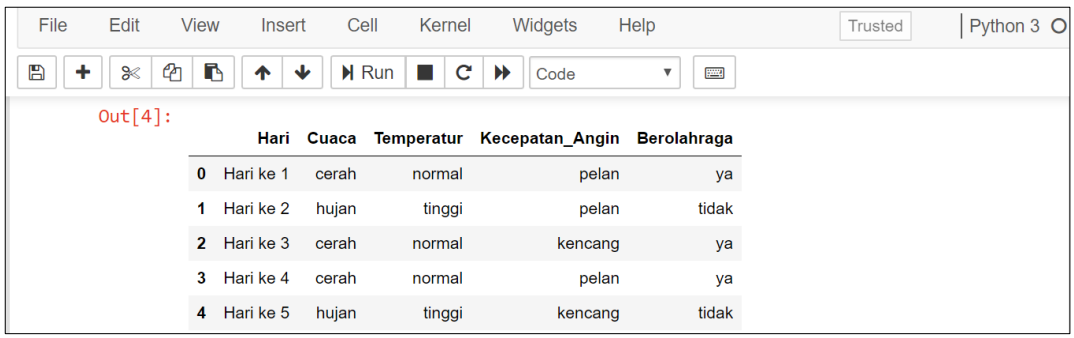
6. Masukan semua variabel library yang telah di instal seperti gambar berikut :



7. Lalu setelah selesai lalu masukan atribut sesuai dataset yang telah dibaut seperti gambar dibawah dan panggil sesuai nama dataset seperti gambar dibawah dengan format csv atau txt.



8. Maka data akan muncul seperti pada gambar dibawah ini.



	Hari	Cuaca	Temperatur	Kecepatan_Angin	Berolahraga
0	Hari ke 1	cerah	normal	pelan	ya
1	Hari ke 2	hujan	tinggi	pelan	tidak
2	Hari ke 3	cerah	normal	kencang	ya
3	Hari ke 4	cerah	normal	pelan	ya
4	Hari ke 5	hujan	tinggi	kencang	tidak

9. Lalu cara yang selanjutnya yaitu mengubah data menjadi data binner dengan cara pada gambar dibawah kecuali atribut ‘berolahraga’ karena merupakan atribut target.

10. Dan maka dataset yang kita masukan sudah berubah menjadi data binner yang hanya berupa angka 1 dan angka 0 seperti gambar dibawah :

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

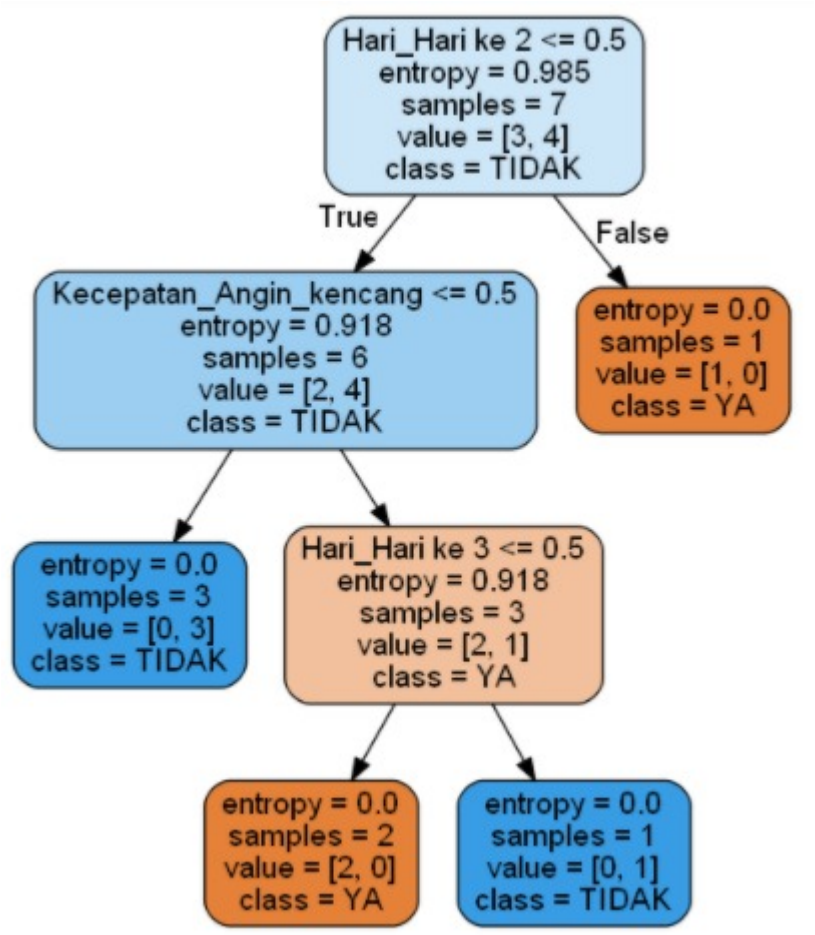
Out[6]:

	Hari_Hari ke 1	Hari_Hari ke 2	Hari_Hari ke 3	Hari_Hari ke 4	Hari_Hari ke 5	Hari_Hari ke 6	Hari_Hari ke 7	Cuaca_cerah	Cuaca_l
0	1	0	0	0	0	0	0	1	
1	0	1	0	0	0	0	0	0	
2	0	0	1	0	0	0	0	1	
3	0	0	0	1	0	0	0	1	
4	0	0	0	0	1	0	0	0	

11. Pada Cara terakhir ini yaitu untuk menampilkan decision tree :

```
In [12]: hasil = tree.DecisionTreeClassifier(criterion='entropy')
         hasil_train = hasil.fit(olahraga, ds['Berolahraga'])

In [9]: dot_data = tree.export_graphviz(hasil_train, out_file=None, feature_names=list(o
                                        class_names=['TIDAK', 'YA'], rounded=True, filled
        graph = pydotplus.graph_from_dot_data(dot_data)
        Image(graph.create_png())
```

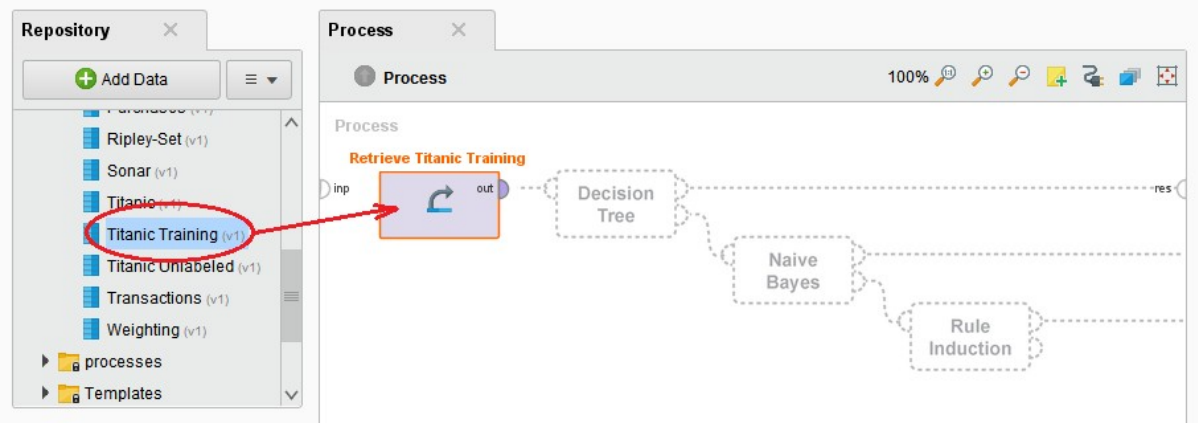
12. Maka Tampilan Decision Tree akan muncul seperti gambar dibawah ini :



Nama : Puspita Dewi Setyadi
Nim : 202420011
Kelas : MTI 23 Reguler A

Langkah 1. Menyiapkan data pelatihan

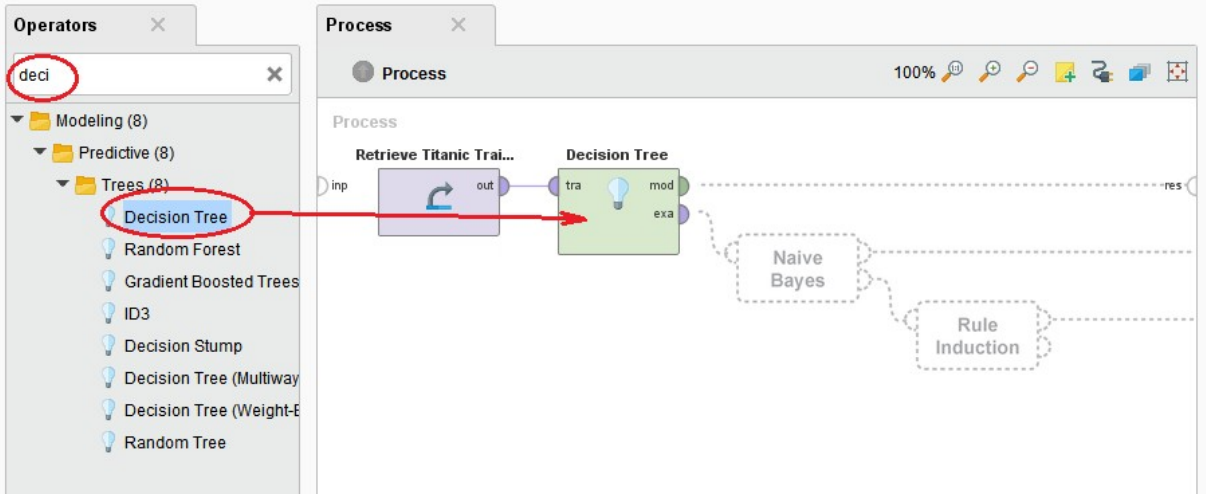
Seret data **Titanic Training** dari Sampel repositori ke dalam proses Anda.



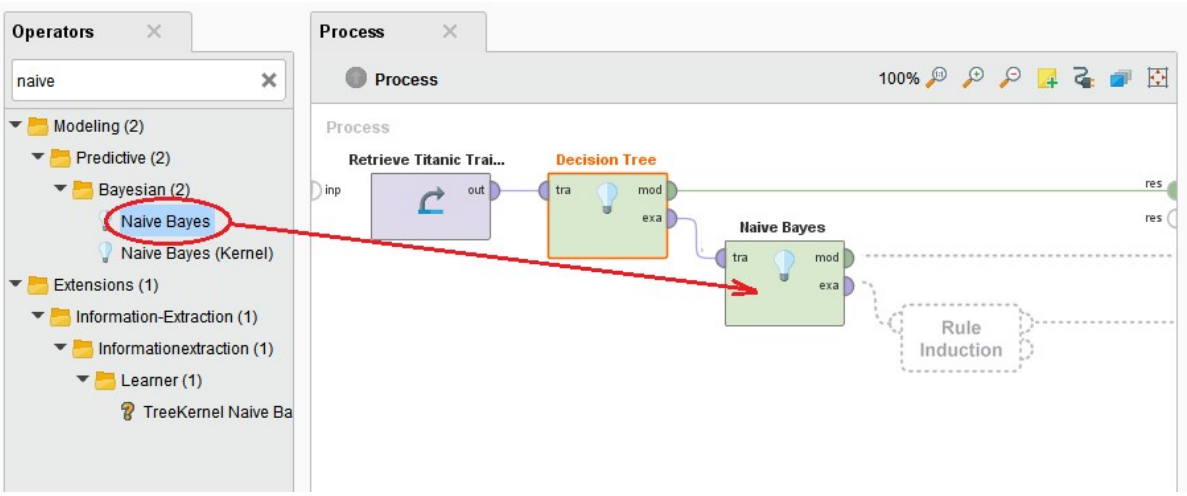
Kita telah menyiapkan data Pelatihan Titanic untuk model pelatihan: dalam artian tidak ada nilai yang hilang serta labelnya telah didefinisikan. Harap diingat bahwa label adalah atribut yang ingin kita prediksi , dalam kasus ini : selamat(survived). kita memerlukan data pelatihan dengan label yang dikenal sebagai masukan untuk metode pembelajaran mesin semacam ini. Inilah sebabnya mengapa kita menyebut metode pembelajaran yang diawasi (*supervised learning*)

Langkah 2. Membangun tiga model yang berbeda.

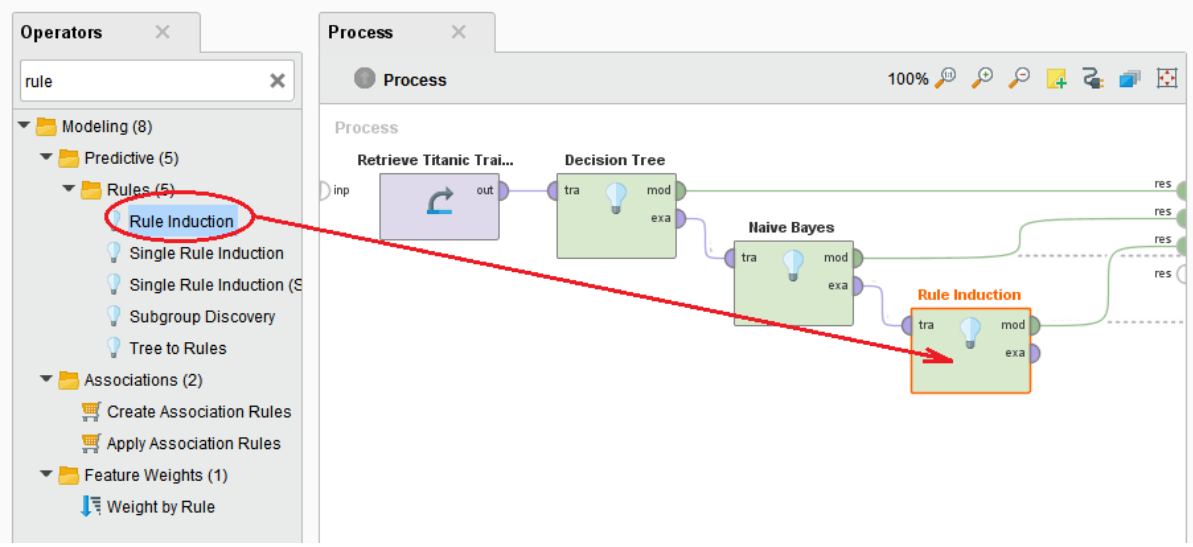
1. Seret pada operator **Decision Tree** dan hubungkan ke port "**out**" dari **Retrieve Titanic Training**.



2. Seret di operator **Naive Bayes** dan hubungkan port input sampel set nya dengan output "exa" dari **Decision Tree**.



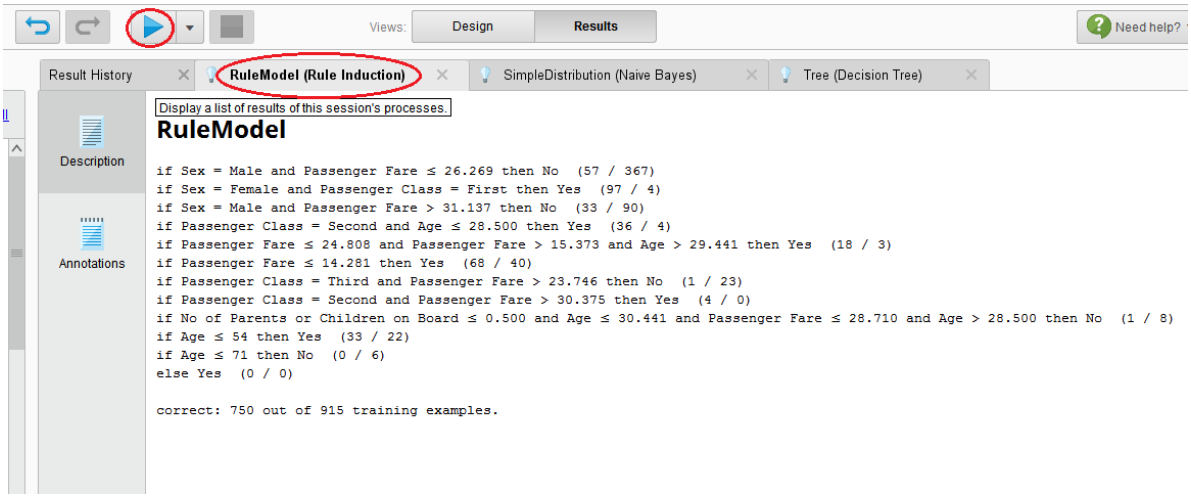
3. Seret ke operator **Rule Induction** dan hubungkan contohnya dengan set port input dengan output "exa" dari **Naive Bayes**.



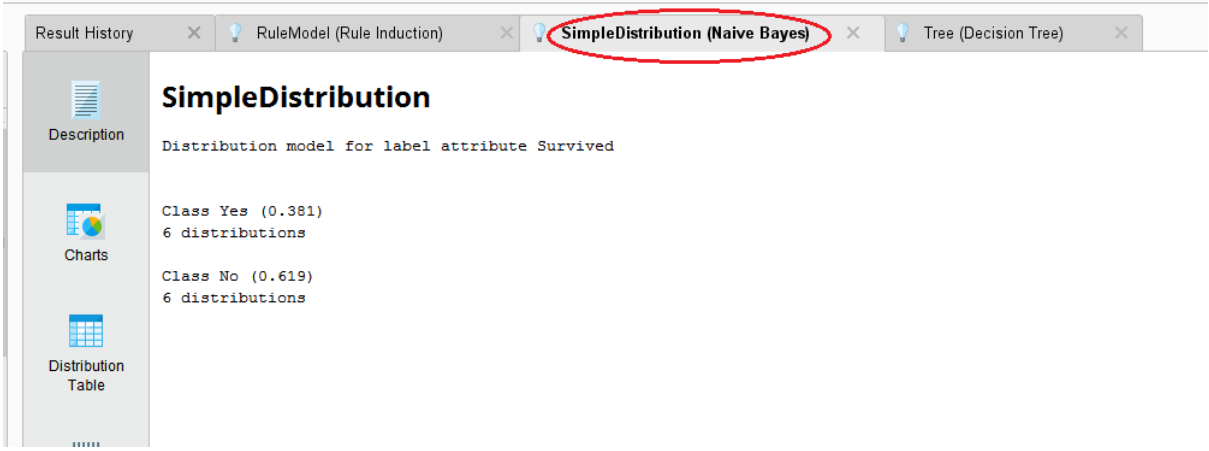
4. Hubungkan port "mod" dari operator pemodelan ke port hasil "res" di kanan, lalu jalankan prosesnya.

5. Periksa tiga model yang berbeda.

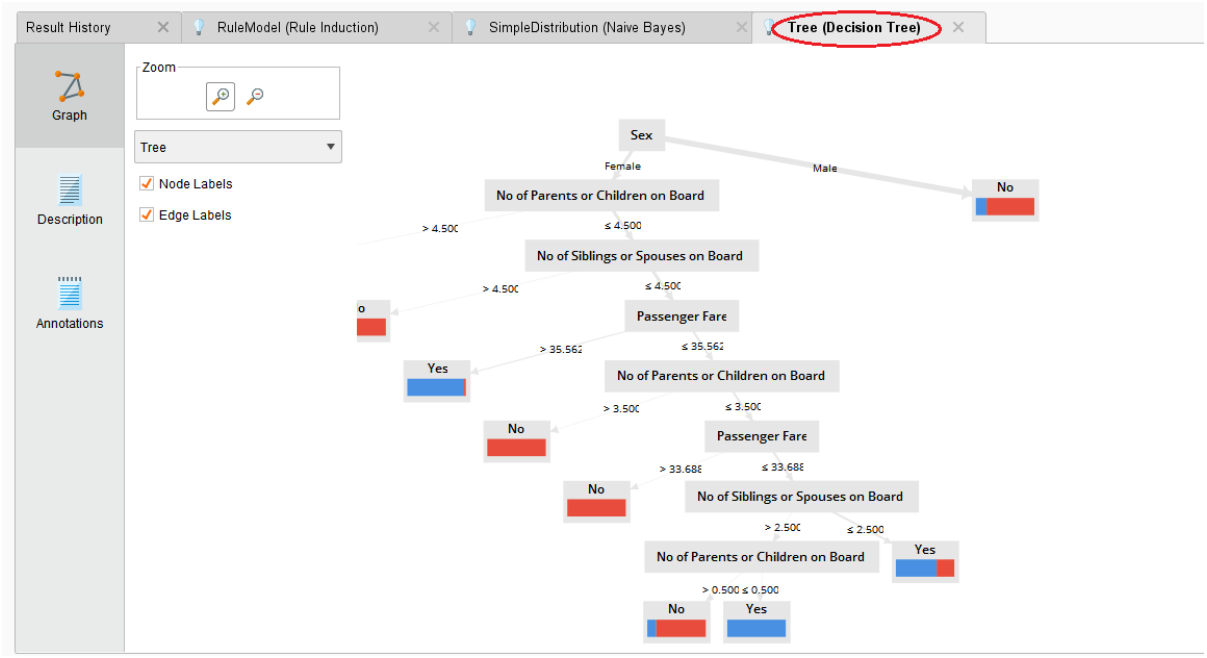
Hasil prediksi dari **Rule Model**



Hasil prediksi dari **Naive Bayes**



Hasil prediksi dari **Decision Tree** (Pohon Keputusan)



Pohon keputusan dengan jelas menunjukkan bahwa ukuran keluarga lebih menentukan daripada kelas penumpang untuk wanita. Pola perilaku ini tidak bisa dideteksi untuk pria. Secara umum, pria memiliki kemungkinan lebih rendah untuk bertahan hidup ("dahulukan wanita dan anak "). Cara termudah untuk melihat hal ini adalah pada visualisasi Chart model **Naive Bayes**. Meski biasanya Naive Bayes bukan tipe model yang paling akurat, secara umum aturan yang ditetapkan adalah format yang mudah dibaca, di mana bisa berguna saat kita ingin menafsirkan model.

Tugas 07

Advanced Database

Dari tugas 06 sebelum ini, coba gunakan data yang sama, tapi gunakan pemprograman python, lalu buat tutorialnya.

Harap dapat dikumpulkan sebelum batas waktu yang ditentukan!

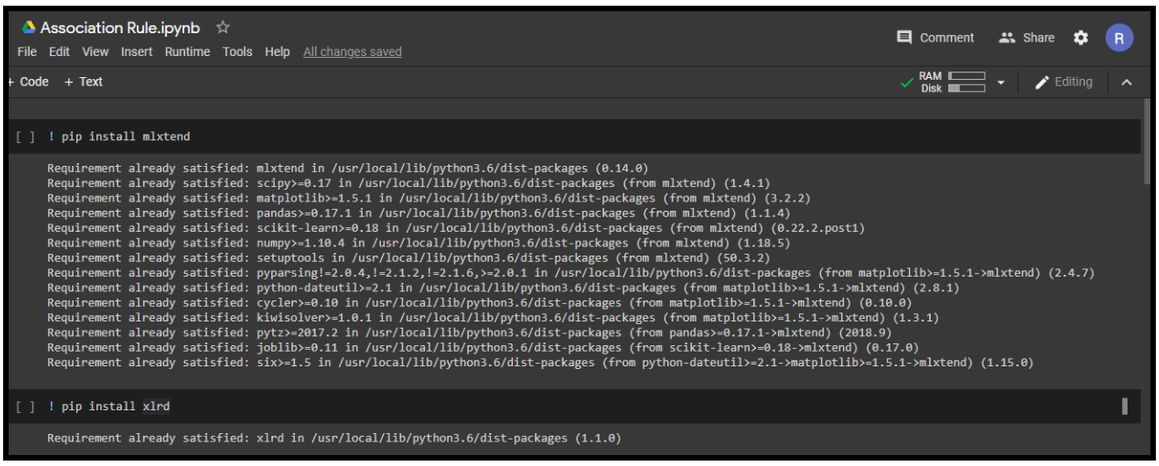
Data Perangkat Praktik Jaringan

No	Tester	Tang Crimping	RJ45	Kabel UTP
1	1	1	1	1
2	0	1	1	1
3	1	0	0	1
4	1	0	1	0
5	1	1	0	1
6	0	1	1	1
7	1	1	1	0

Berdasarkan data perangkat disamping untuk praktik jaringan maka langsung saja dilakukan proses *Association Rule* dengan menggunakan Bahasa pemograman *Python*, dimana dalam proses tersebut saya menggunakan platform yang disediakan google yaitu *Google Collab Research*.

Step 1. Install plugin Mlxtend

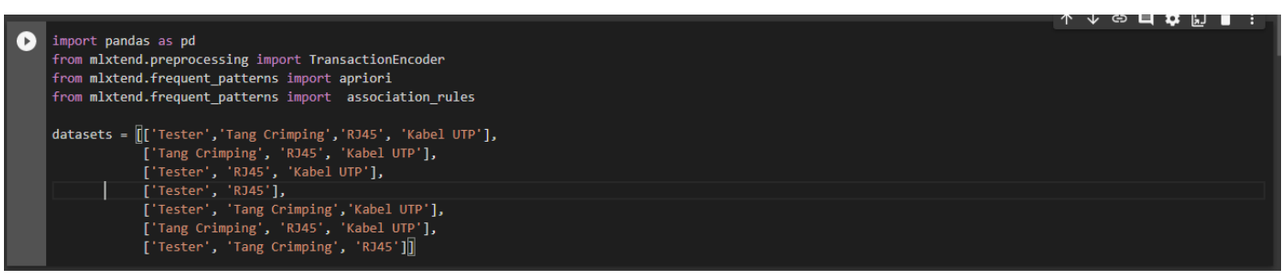
Proses ini dilakukan untuk load library yang dibutuhkan dalam proses Association Rule.



Proses Instalasi Plugin Mlxtend

Step 2. Import Library & Datasets

Proses selanjut nya yaitu mengimport library Transaction Encoder, Apriori, Association Rule dan datasets yang akan digunakan.



```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import  association_rules

datasets = [['Tester','Tang Crimping','RJ45', 'Kabel UTP'],
            ['Tang Crimping', 'RJ45', 'Kabel UTP'],
            ['Tester', 'RJ45', 'Kabel UTP'],
            ['Tester', 'RJ45'],
            ['Tester', 'Tang Crimping','Kabel UTP'],
            ['Tang Crimping', 'RJ45', 'Kabel UTP'],
            ['Tester', 'Tang Crimping', 'RJ45']]
```

Proses Import Library & Datasets

https://colab.research.google.com/drive/1HDctIu8DCxPVsydjEH28Sd_pkmdRP52H?usp=sharing

Step.3 Transaction Encoder

Proses ini digunakan untuk menemukan semua itemset yang memenuhi minimum support.

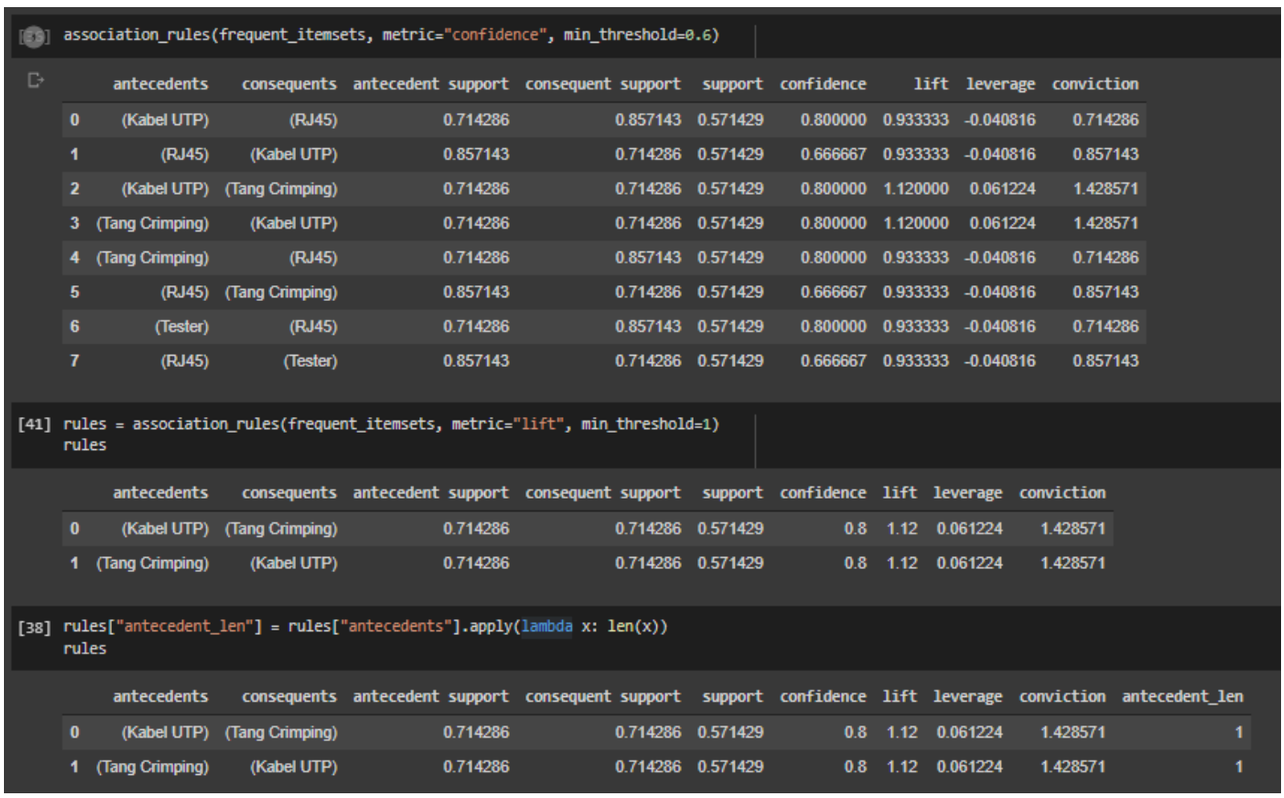
```
te = TransactionEncoder()
te_array = te.fit(datasets).transform(datasets)
df = pd.DataFrame(te_array, columns=te.columns_)
frequent_itemsets = apriori(df, min_support= 0.5, use_colnames=True)

frequent_itemsets
```

	support	itemsets
0	0.714286	(Kabel UTP)
1	0.857143	(RJ45)
2	0.714286	(Tang Crimping)
3	0.714286	(Tester)
4	0.571429	(Kabel UTP, RJ45)
5	0.571429	(Kabel UTP, Tang Crimping)
6	0.571429	(Tang Crimping, RJ45)
7	0.571429	(Tester, RJ45)

Proses Transaction Encoder

Step.4 Rule Generation & Selection Criteria

Proses ini digunakan untuk mengekstrak semua aturan yang memiliki high-confidence dari itemsets yang ditemukan dari langkah sebelumnya.



```
association_rules(frequent_itemsets, metric="confidence", min_threshold=0.6)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Kabel UTP)	(RJ45)	0.714286	0.857143	0.571429	0.800000	0.933333	-0.040816	0.714286
1	(RJ45)	(Kabel UTP)	0.857143	0.714286	0.571429	0.666667	0.933333	-0.040816	0.857143
2	(Kabel UTP)	(Tang Crimping)	0.714286	0.714286	0.571429	0.800000	1.120000	0.061224	1.428571
3	(Tang Crimping)	(Kabel UTP)	0.714286	0.714286	0.571429	0.800000	1.120000	0.061224	1.428571
4	(Tang Crimping)	(RJ45)	0.714286	0.857143	0.571429	0.800000	0.933333	-0.040816	0.714286
5	(RJ45)	(Tang Crimping)	0.857143	0.714286	0.571429	0.666667	0.933333	-0.040816	0.857143
6	(Tester)	(RJ45)	0.714286	0.857143	0.571429	0.800000	0.933333	-0.040816	0.714286
7	(RJ45)	(Tester)	0.857143	0.714286	0.571429	0.666667	0.933333	-0.040816	0.857143

```
[41] rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
     rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Kabel UTP)	(Tang Crimping)	0.714286	0.714286	0.571429	0.8	1.12	0.061224	1.428571
1	(Tang Crimping)	(Kabel UTP)	0.714286	0.714286	0.571429	0.8	1.12	0.061224	1.428571

```
[38] rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
     rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
0	(Kabel UTP)	(Tang Crimping)	0.714286	0.714286	0.571429	0.8	1.12	0.061224	1.428571	1
1	(Tang Crimping)	(Kabel UTP)	0.714286	0.714286	0.571429	0.8	1.12	0.061224	1.428571	1

Proses Rule generation $ Selection Criteria

TUGAS 07

APRIORI - PHYTON

Dibuat Oleh

Robby Prabowo

202420001

Dosen Pengampu

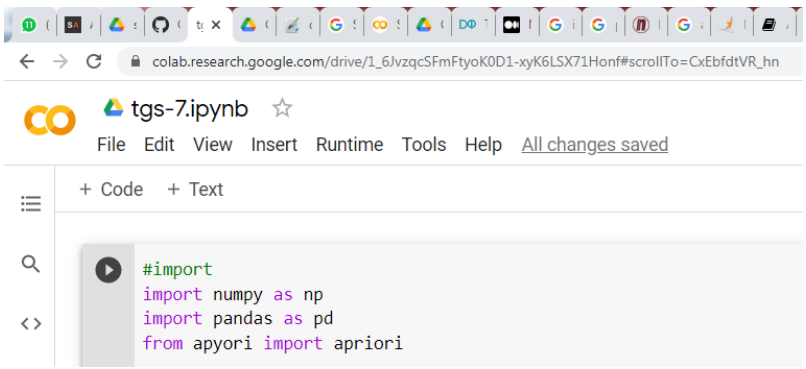
TRI BASUKI KURNIAWAN, S.Kom, M.Eng, Ph.D

Program Pasca Sarjana

Universitas Binadarma Palembang

2020/2021

Step 1: Import the libraries



Catatan penulis : pada Collabs modul apyori harus diinstall dulu menggunakan perintah

```
!pip install apyori
```

Step 2: Load the dataset

```
#loading data sheet
#import datasheet
from google.colab import drive
drive.mount('/content/drive')
store_data = pd.read_csv('drive/My Drive/Colab Notebooks/data-sheet-tugas-7-ok.csv', header=None)
```

Step 3: Have a glance at the records

```
#having glance of record
store_data

#look shape
store_data.shape
```

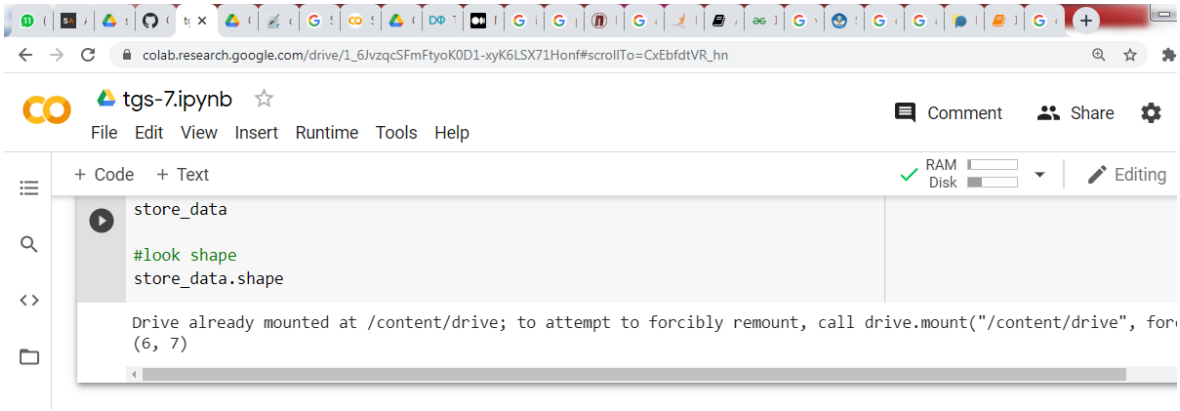
Output

```
store_data
```

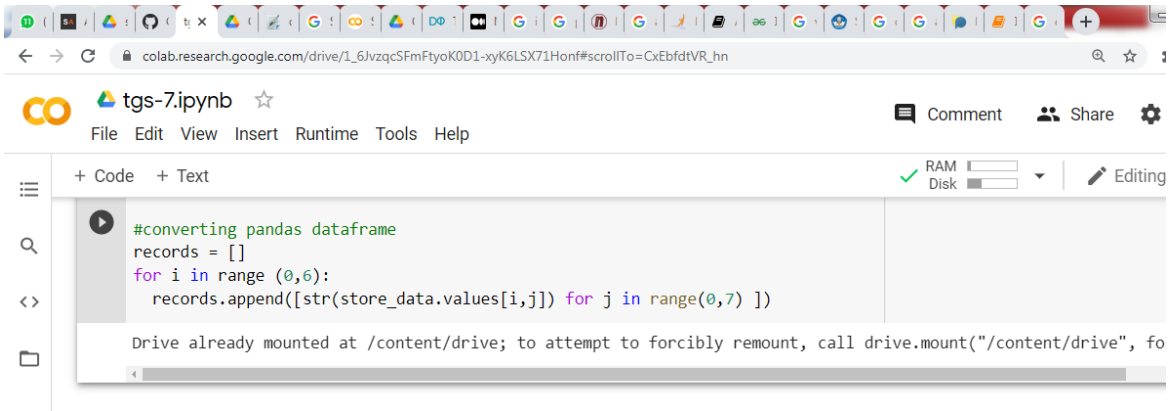
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive",
```

	0	1	2	3	4	5	6
0	PENA	ROTI	MENTEGA	TELUR	BUNCIS	SUSU	KECAP
1	PENA	ROTI	MENTEGA	NaN	NaN	NaN	NaN
2	NaN	ROTI	MENTEGA	TELUR	NaN	NaN	NaN
3	NaN	NaN	NaN	TELUR	BUNCIS	SUSU	NaN
4	NaN	ROTI	MENTEGA	TELUR	NaN	NaN	NaN
5	NaN	ROTI	MENTEGA	TELUR	NaN	SUSU	KECAP

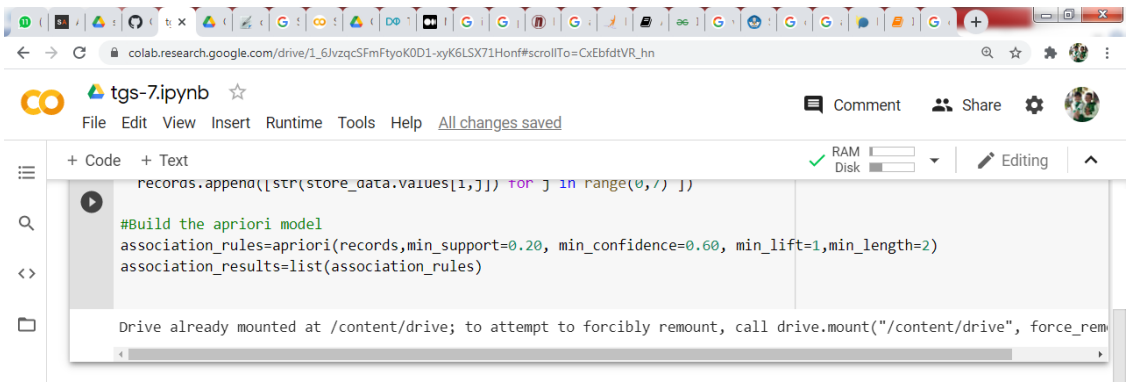
Step 4: Look at the shape



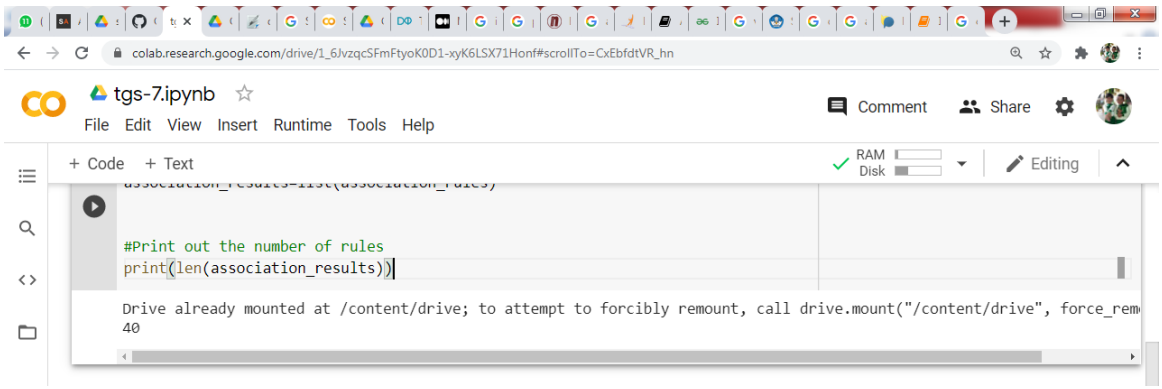
Step 5: Convert Pandas DataFrame into a list of lists



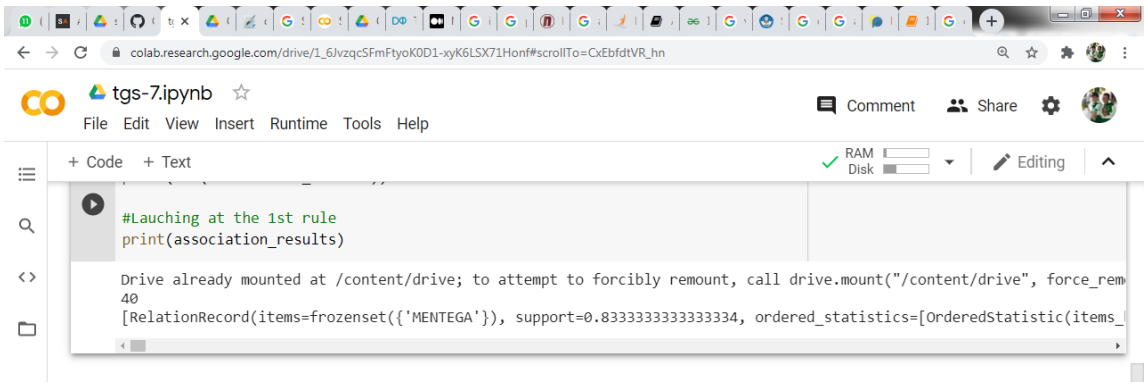
Step 6: Build the Apriori model



Step 7: Print out the number of rules



Step 8: Have a glance at the rule



Script Phyton Lengkap

```
#import
import numpy as np
import pandas as pd
from apyori import apriori

#loading data sheet
#import datasheet
from google.colab import drive
drive.mount('/content/drive')
store_data = pd.read_csv('drive/My Drive/Colab Notebooks/data-sheet-
tugas-7-ok.csv', header=None)

#having glance of record
store_data

#look shape
store_data.shape

#converting pandas dataframe
records = []
for i in range (0,6):
  records.append([str(store_data.values[i,j]) for j in range(0,7) ])

#Build the apriori model
association_rules=apriori(records,min_support=0.20, min_confidence=0.60
, min_lift=1,min_length=2)
association_results=list(association_rules)


#Print out the number of rules
print(len(association_results))

#Lauching at the 1st rule
print(association_results)
```

Nama : Shabila Fitri Aulia
Nim : 202420024
Kelas : MTI A 23
MK : Advanced Databased

Tugas 07

Dari tugas 06 sebelum ini, coba gunakan data yang sama, tapi gunakan pemprograman python, lalu buat tutorialnya.

Harap dapat dikumpulkan sebelum batas waktu yang ditentukan!

Jawaban :

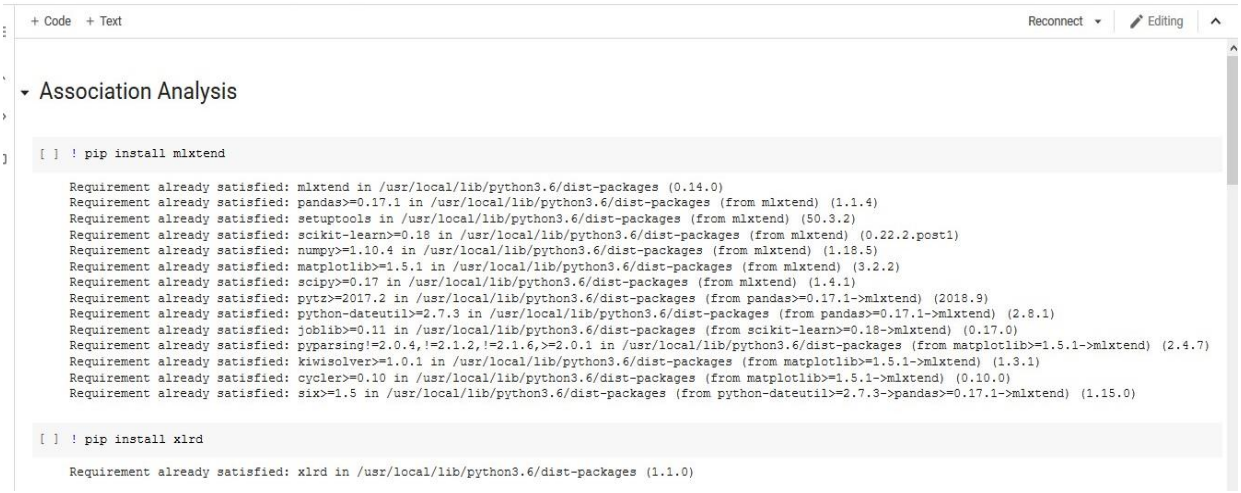
Dari Tugas 6 didapatkan data sebagai berikut :

TID	PENA	ROTI	MENTEGA	TELUR	BUNCIS	SUSU	KECAP
001	1	1	1	0	0	0	0
002	0	1	1	1	0	0	0
003	0	0	0	1	1	1	0
004	0	1	1	0	0	0	0
005	0	1	1	1	0	1	1

Dari data diatas akan saya coba untuk membuat association rule meggunakan bahasa pyhton dengan colab.research.google.com, adapun langkahnya sebagai berikut :

Langkah 1,

Download Libabry Association Rule



Langkah 2,

Masukkan data mengenai assosiation rule yang akan diproses (data terlampir diatas), Tentukan minimum supportnya pada kali ini saya menggunakan 0.6

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] ! pip install xlrd
```

```
Requirement already satisfied: xlrd in /usr/local/lib/python3.6/dist-packages (1.1.0)
```

Association Rules Generation from Frequent Itemsets

Source: https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/

```
[ ] # ! pip install mlxtend
    import pandas as pd
    from mlxtend.preprocessing import TransactionEncoder
    from mlxtend.frequent_patterns import apriori
    from mlxtend.frequent_patterns import association_rules

    dataset = [['Pena', 'Roti', 'Mentega'],
               ['Roti', 'Mentega', 'Telur'],
               ['Telur', 'Buncis', 'Susu'],
               ['Roti', 'Mentega'],
               ['Roti', 'Mentega', 'Telur', 'Susu', 'Kecap']]

    te = TransactionEncoder()
    te_ary = te.fit(dataset).transform(dataset)
    df = pd.DataFrame(te_ary, columns=te.columns_)
    frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)

    frequent_itemsets
```

Langkah 3,

Tentukan nilai rules sesuai yang diinginkan :

```
rules[ (rules['antecedent_len'] >= 2) &
       (rules['confidence'] > 0.75) &
       (rules['lift'] > 1.2) ]
```

Langkah 4,

Kemudian RUN All dari seluruh config yang sudah ada, adapun hasilnya sebagai berikut :

	support	itemsets
0	0.8	(Mentega)
1	0.8	(Roti)
2	0.6	(Telur)
3	0.8	(Roti, Mentega)

```
[ ] association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
    rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
    rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf	1
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf	1

Didaptkan kesimpulan bahwa Assosiation rule sebagaimana diatas untuk hasilny.

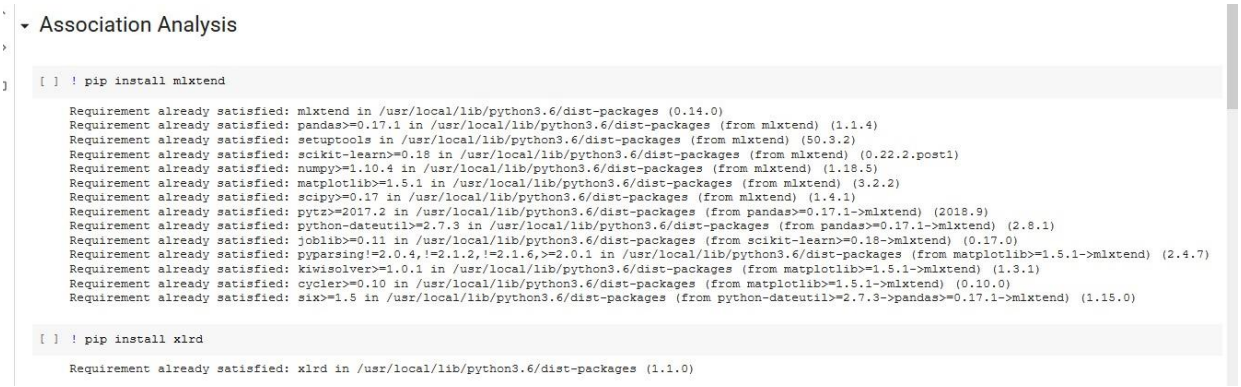
Nama : Siti Ratu Delima
Nim :202420025
Kelas :MTI 23

TID	PENA	ROTI	MENTEGA	TELUR	BUNCIS	SUSU	KECAP
001	1	1	1	0	0	0	0
002	0	1	1	1	0	0	0
003	0	0	0	1	1	1	0
004	0	1	1	0	0	0	0
005	0	1	1	1	0	1	1

Dari data diatas akan saya coba untuk membuat association rule meggunakan bahasa pyhton dengan colab.research.google.com, adapun langkahnya sebagai berikut :

Langkah 1,

Download Libabry Association Rule



Langkah 2,

Masukkan data mengenai assosiation rule yang akan diproses (data terlampir diatas), Tentukan minimum supportnya pada kali ini saya menggunakan 0.6

```
[ ] ! pip install xlrd
```

```
Requirement already satisfied: xlrd in /usr/local/lib/python3.6/dist-packages (1.1.0)
```

Association Rules Generation from Frequent Itemsets

Source: https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/

```
[ ] # ! pip install mlxtend
    import pandas as pd
    from mlxtend.preprocessing import TransactionEncoder
    from mlxtend.frequent_patterns import apriori
    from mlxtend.frequent_patterns import association_rules

    dataset = [['Pena', 'Roti', 'Mentega'],
               ['Roti', 'Mentega', 'Telur'],
               ['Telur', 'Buncis', 'Susu'],
               ['Roti', 'Mentega'],
               ['Roti', 'Mentega', 'Telur', 'Susu', 'Kecap']]

    te = TransactionEncoder()
    te_ary = te.fit(dataset).transform(dataset)
    df = pd.DataFrame(te_ary, columns=te.columns_)
    frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)

    frequent_itemsets
```

Langkah 3,

Tentukan nilai rules sesuai yang diinginkan :

```
rules[ (rules['antecedent_len'] >= 2) &
       (rules['confidence'] > 0.75) &
       (rules['lift'] > 1.2) ]
```

Langkah 4,

Kemudian RUN All dari seluruh config yang sudah ada, adapun hasilnya sebagai berikut :

	support	itemsets
0	0.8	(Mentega)
1	0.8	(Roti)
2	0.6	(Telur)
3	0.8	(Roti, Mentega)

```
[ ] association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
    rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
    rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf	1
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf	1

Didaptkan kesimpulan bahwa Assosiation rule sebagaimana diatas untuk hasilny.

TUTORIAL DECISION TREE MENGGUNAKAN PYTHON

Dataset yang akan kami gunakan aadalah dataset untuk menentukan status kelayakan penerima beasiswa yang terdiri dari 2 kelas yaitu kelas Ya Layak dan Tidak Layak menerima beasiswa serta terdiri dari 6 atribut/feature antara lain:

1. Jenis Tinggal (JT)
2. Alat Transportasi (AT)
3. Pekerjaan Ayah (PKA)
4. Penghasilan Ayah (PHA)
5. Pekerjaan Ibu (PKI)
6. Penghasilan Ibu (PHI)

Berikut merupakan dataset yang telah melalui proses preprocessing dan siap untuk diproses kedalam teknik klasifikasi. Dataset yang digunakan sebanyak 240 record yang akan dibagi menjadi data training dan data testing dengan menggunakan teknik hold-out 70%-30%, dimana masing-masing data training berjumlah sebesar 70% dari total data yaitu 168 record dan data testing berjumlah 72 record dengan persentase 30% dari keseluruhan data.

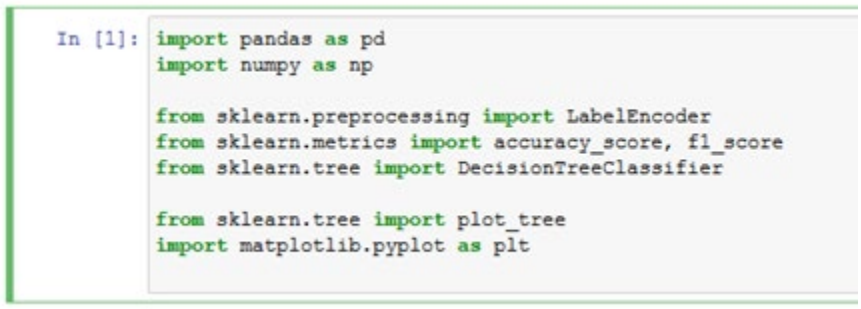
Data Training yang digunakan

No	Nama	JK	JT	AT	PKA	PHA	PKI	PHI	Status Kelayakan
1	S-1	L	0,500	1,000	0,083	0,625	0,833	1,000	Tidak
2	S-2	L	0,250	0,143	0,667	0,375	0,833	1,000	Tidak
3	S-3	L	0,250	1,000	0,667	0,500	0,833	1,000	Tidak
4	S-4	L	0,250	0,286	0,500	0,250	0,833	1,000	Tidak
5	S-5	L	0,250	0,000	0,000	0,500	0,917	0,875	Tidak
6	S-6	P	0,250	1,000	0,333	0,500	0,083	0,875	Tidak
7	S-7	L	0,250	0,286	0,667	0,250	0,833	1,000	Tidak
8	S-8	L	0,250	0,000	0,083	0,500	0,833	1,000	Tidak
9	S-9	L	0,250	0,286	0,917	0,375	0,833	1,000	Tidak
10	S-10	P	0,250	1,000	0,083	0,625	0,667	0,625	Tidak
…	…	…	…	…	…	…	…	…	…
168	S-168	P	0,250	0,571	0,083	0,625	0,667	0,625	Tidak

Data Testing yang digunakan

No	Nama	JK	JT	AT	PKA	PHA	PKI	PHI	Status Kelayakan
1	S-169	L	0,250	0,143	0,917	0,500	0,500	1,000	Tidak
2	S-170	P	0,250	1,000	0,667	0,750	0,667	0,625	Tidak
3	S-171	P	0,250	1,000	0,917	0,500	0,833	1,000	Tidak
4	S-172	L	0,250	1,000	0,083	0,625	0,833	1,000	Tidak
5	S-173	L	0,250	1,000	1,000	0,625	1,000	0,500	Tidak
6	S-174	L	0,250	0,000	0,917	0,500	0,833	1,000	Tidak
7	S-175	L	0,250	1,000	0,333	0,500	0,833	1,000	Ya
8	S-176	L	0,250	1,000	0,917	0,500	0,833	1,000	Tidak
9	S-177	P	0,250	1,000	0,083	0,875	0,833	1,000	Ya
10	S-178	L	0,250	1,000	0,917	0,500	0,833	1,000	Tidak
…	…	…	…	…	…	…	…	…	…
72	S-240	P	0,250	1,000	0,917	0,625	0,833	1,000	Tidak

- Jika data training dan data testing telah tersedia, kemudian buka lembaran kerja pada jupyter notebook lalu import beberapa library yang diperlukan, sebagaimana dibawah ini.




```
import pandas as pd
import numpy as np

from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, f1_score
from sklearn.tree import DecisionTreeClassifier

from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
```

3. Kemudian, import data yang akan digunakan sebagaimana dibawah ini

```
def read_file(filename):
    df = pd.read_excel(filename)
    print(df.shape)
    df['Status Kelayakan'] = df['Status Kelayakan'].apply(lambda x: x.strip().lower())
    return df

FILE_HO_TRAIN = 'A_Train_HO.xlsx'
FILE_HO_TEST = 'A_Test_HO.xlsx'

df_ho_train = read_file(FILE_HO_TRAIN)
df_ho_test = read_file(FILE_HO_TEST)

(168, 10)
(72, 10)
```

Pada output yang dihasilkan dapat dilihat bahwa terdapat 2 dataset dengan masing-masing berjumlah 168 record & 10 row yang merupakan data training serta data dengan 72 record dengan 10 row yang merupakan data testing.

- Kemudian, tentukan label/kelas yang digunakan sebagaimana dibawah ini,

```
target_encoder = LabelEncoder()
df_ho_train['Target'] = target_encoder.fit_transform(df_ho_train['Status Kelayakan'])
df_ho_test['Target'] = target_encoder.transform(df_ho_test['Status Kelayakan'])
```

- Dari 10 row yang ada, tidak semua row/kolom dijadikan sebagai atribut sebagaimana penjelasan sebelumnya, agar hanya kolom atribut yang diproses maka kolom yang berfungsi sebagai label/kelas atau kolom Nomor, id, serta row pelengkap perlu didrop terlebih dahulu sebagaimana dibawah ini.

```
x_ho_train = df_ho_train.drop(['Status Kelayakan', 'No', 'Nama', 'Target', 'JK'], axis=1)
y_ho_train = df_ho_train['Target']

x_ho_test = df_ho_test.drop(['Status Kelayakan', 'No', 'Nama', 'Target', 'JK'], axis=1)
y_ho_test = df_ho_test['Target']
```

```
x_ho_test = df_ho_test.drop(['Status Kelayakan', 'No', 'Nama', 'Target', 'JK'
], axis=1)

y_ho_test = df_ho_test['Target']
```

- Kemudian, input model algoritma decision tree untuk memproses data sekaligus melihat akurasi yang dihasilkan dengan menggunakan algoritma tersebut


```
model = DecisionTreeClassifier(criterion='gini')

model.fit(x_ho_train, y_ho_train)

model.score(x_ho_train, y_ho_train)

predicted= model.predict(x_ho_test)


accuracy_ho = accuracy_score(predicted, y_ho_test) * 100

print('Akurasi pada Algoritma Decision Tree menggunakan Hold-out dengan perse
ntase 70:30 = %s' % accuracy_ho)
```

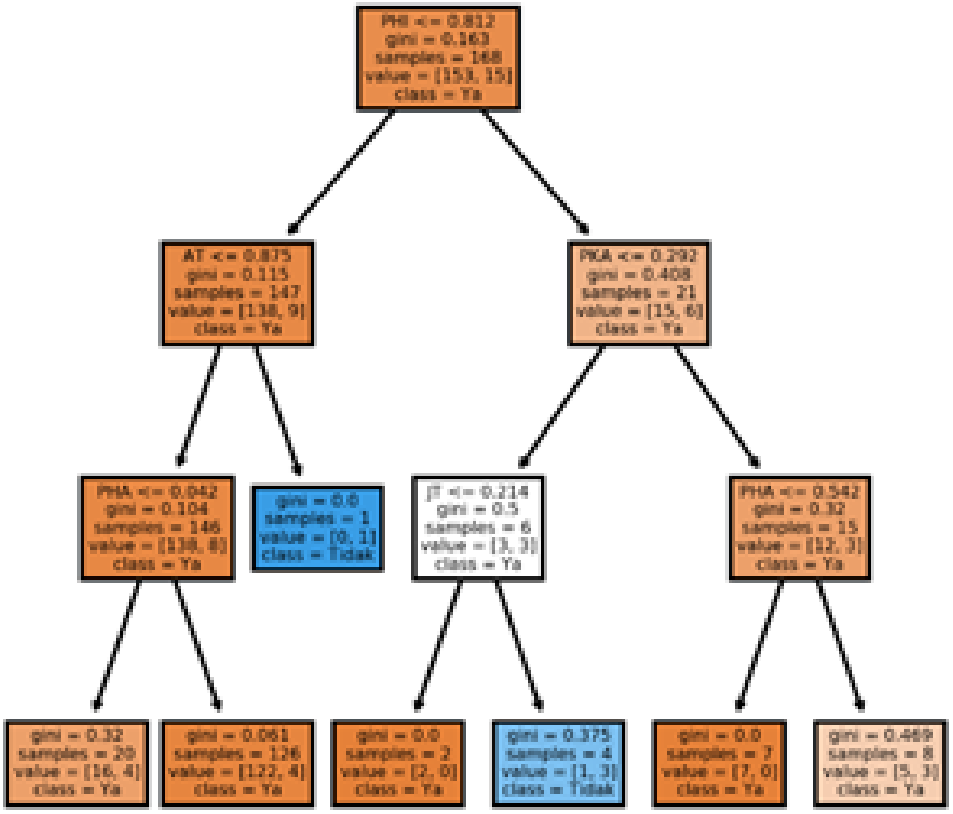
Dapat dilihat pada gambar diatas, bahwa Algoritma Decision Tree menggunakan Hold-out dengan persentase 70:30 menghasilkan akurasi sebesar 84.72%

- Untuk melihat visualisasi dari hasil algoritma decision tree dengan menggunakan plot tree dapat dilihat pada gambar dibawah ini,


```
clf = DecisionTreeClassifier(max_depth = 3)


clf.fit(x_ho_train, y_ho_train)

tree.plot_tree(clf));

fn=['AT','JT','PHA','PHI','PKA','PKI']
```

```
cn=['Ya', 'Tidak']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
plot_tree(clf,
                feature_names = fn,
                class_names=cn,
                filled = True);fig.savefig('imagename.png')
```

Proses diatas akan menghasilkan plot tree dari algoritma Decision Tree untuk kasus status kelayakan beasiswa sebagaimana gambar berikut.



Tugas 07

Dari tugas 06 sebelum ini, coba gunakan data yang sama, tapi gunakan pemprograman python, lalu buat tutorialnya.

Harap dapat dikumpulkan sebelum batas waktu yang ditentukan !

Nama : Vero Faloris
Nim : 202420032
Kelas : MTI 23 Reguler A
Mk : Advanced Database

LINK RUJUKAN :

https://www.academia.edu/7712860/Belajar_Data_Mining_dengan_RapidMiner

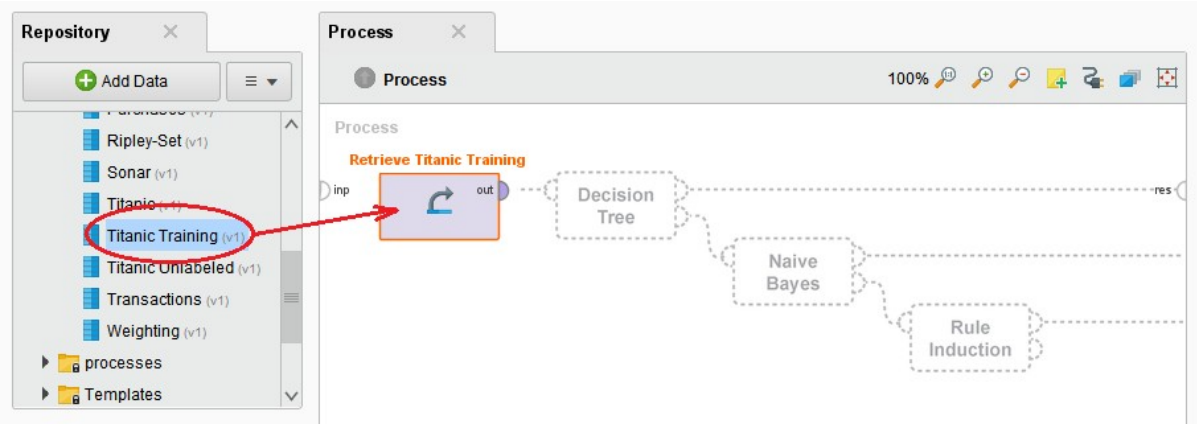
https://medium.com/@ksnugroho/menerapkan-model-machine-learning-pada-rapidminer-142259846e13

Cari beberapa tutorial yang membahas pengolahan data menggunakan metode prediksi pada tool rapidminer. Silahkan buat ringkasan tutorialnya kembali dengan menggunakan dataset yang kamu buat sendiri. Tuliskan dalam format ms word dan sertakan semua sumber rujukan tutorial yang anda gunakan.

Dalam tutorial ini, kita akan membuat tiga model klasifikasi yang berbeda untuk data Titanic kita: pohon keputusan (decision tree),seperangkat aturan (a set of rules), dan model Bayes. Kita akan menjelajahi model-model tersebut dan melihat apakah kita bisa mengetahui lebih banyak tentang peristiwa kecelakaan itu dan lebih memahami siapa yang memiliki kesempatan terbaik untuk bertahan hidup.

Langkah 1. Menyiapkan data pelatihan

Seret data **Titanic Training** dari Sampel repositori ke dalam proses Anda.

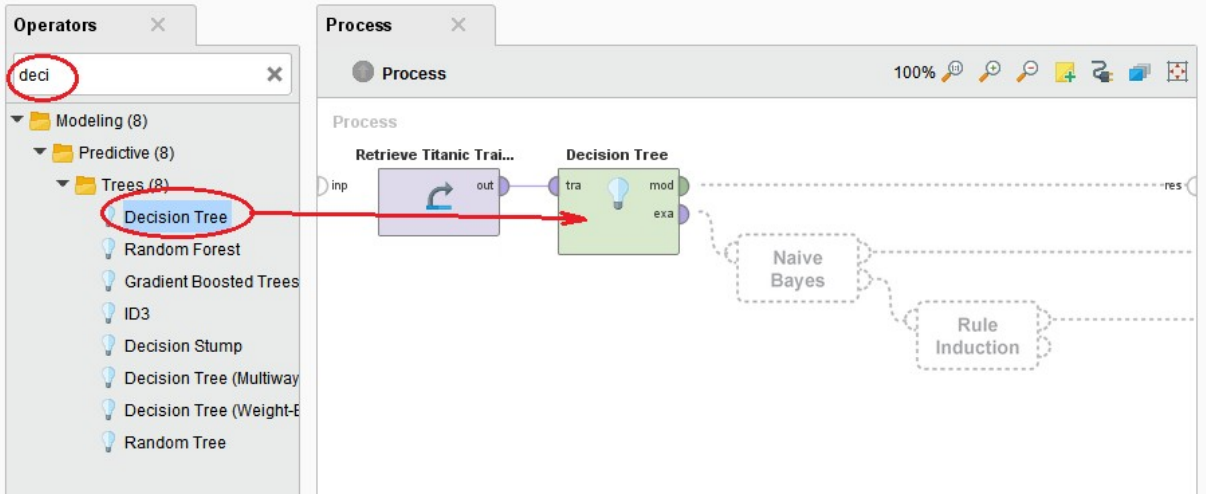


Kita telah menyiapkan data Pelatihan Titanic untuk model pelatihan: dalam artian tidak ada nilai

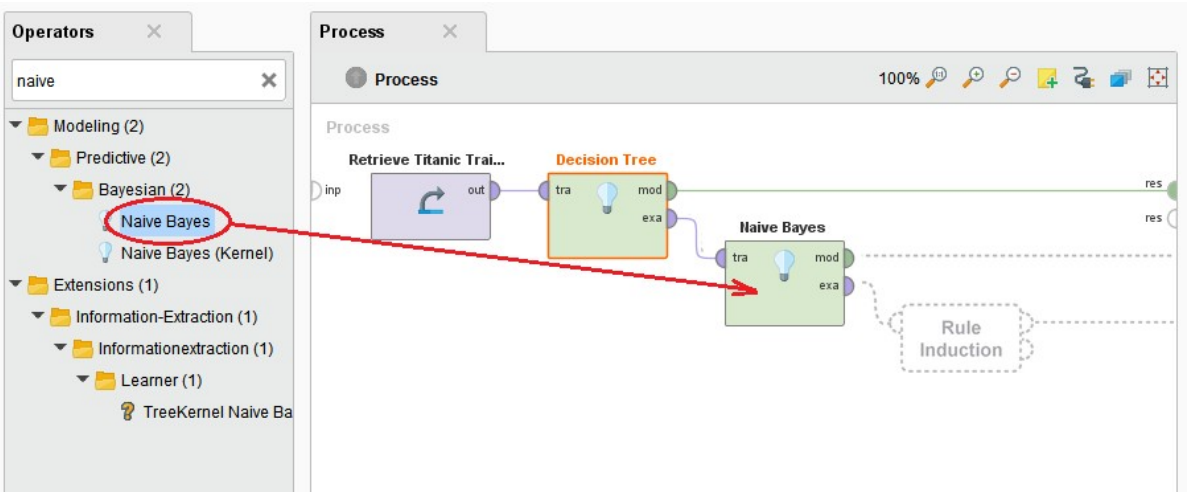
yang hilang serta labelnya telah didefinisikan. Harap diingat bahwa label adalah atribut yang ingin kita prediksi , dalam kasus ini : selamat(survived). kita memerlukan data pelatihan dengan label yang dikenal sebagai masukan untuk metode pembelajaran mesin semacam ini. Inilah sebabnya mengapa kita menyebut metode pembelajaran yang diawasi (*supervised learning*)

Langkah 2. Membangun tiga model yang berbeda.

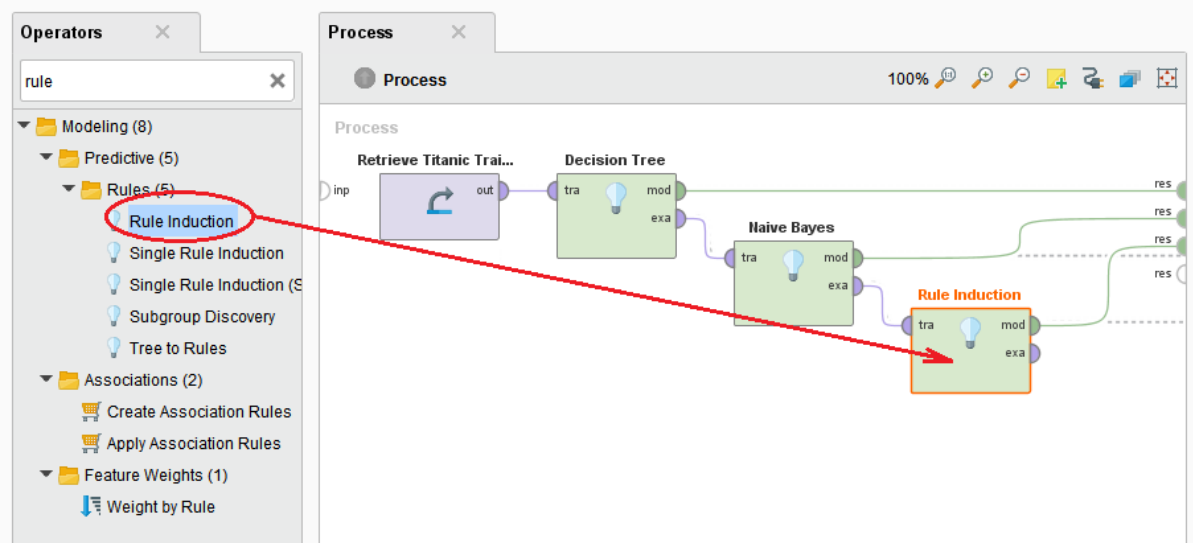
1. Seret pada operator **Decision Tree** dan hubungkan ke port "**out**" dari **Retrieve Titanic Training**.



2. Seret di operator **Naive Bayes** dan hubungkan port input sampel set nya dengan output "exa" dari **Decision Tree**.



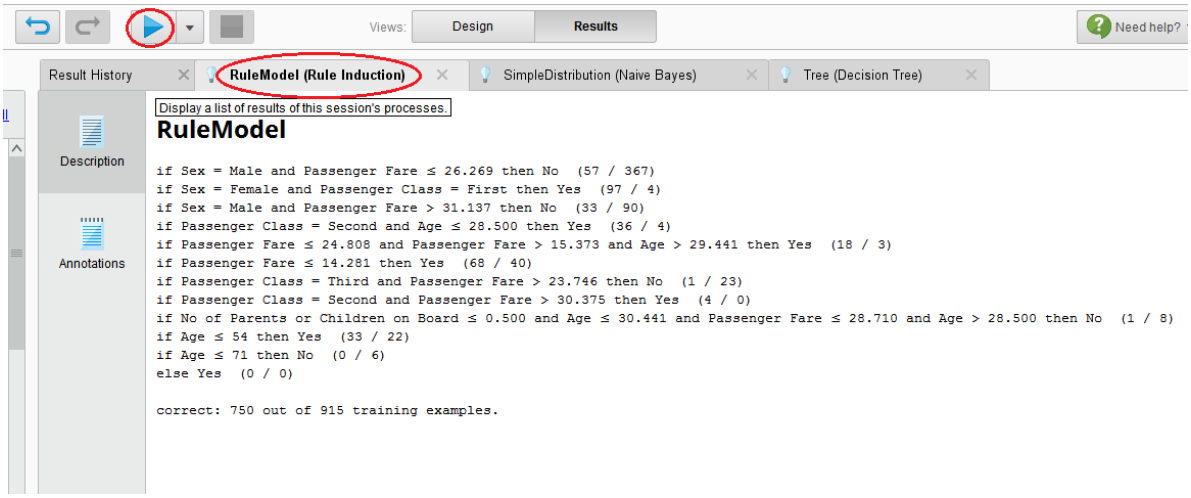
3. Seret ke operator **Rule Induction** dan hubungkan contohnya dengan set port input dengan output "exa" dari **Naive Bayes**.



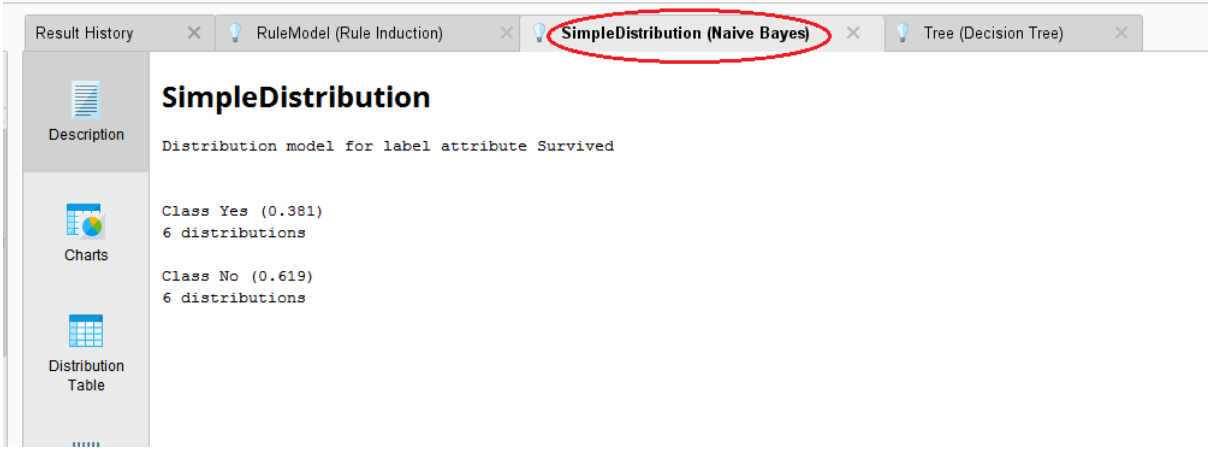
4. Hubungkan port "mod" dari operator pemodelan ke port hasil "res" di kanan, lalu jalankan prosesnya.

5. Periksa tiga model yang berbeda.

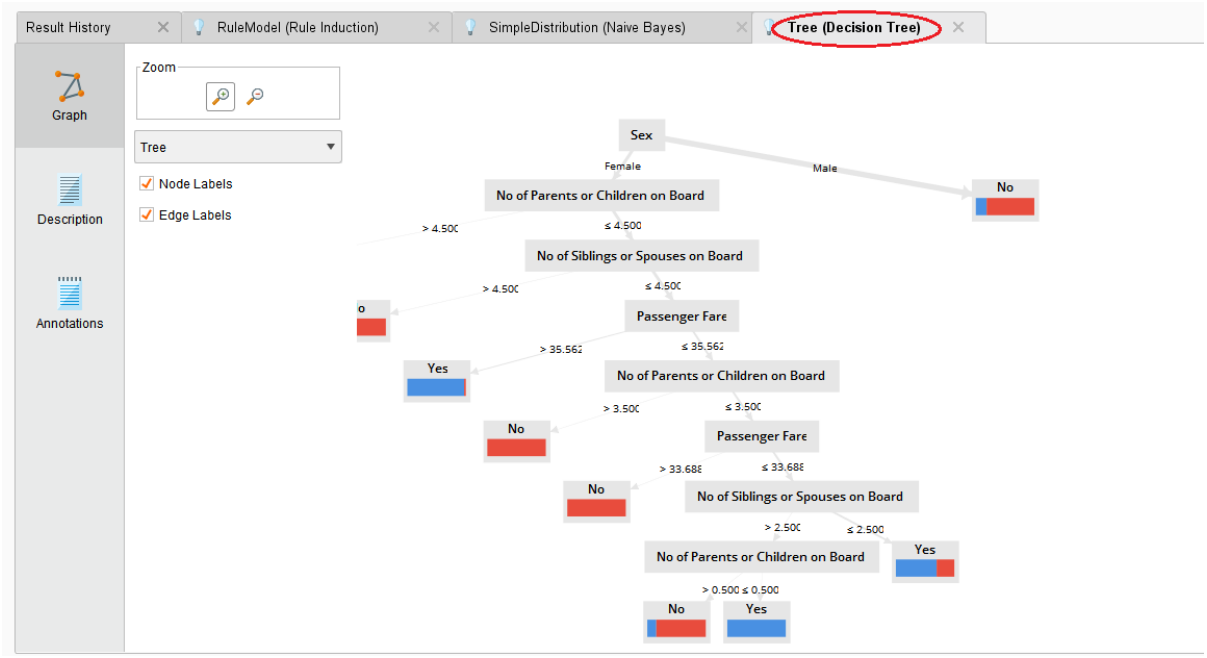
Hasil prediksi dari **Rule Model**



Hasil prediksi dari **Naive Bayes**



Hasil prediksi dari **Decision Tree** (Pohon Keputusan)



Pohon keputusan dengan jelas menunjukkan bahwa ukuran keluarga lebih menentukan daripada kelas penumpang untuk wanita. Pola perilaku ini tidak bisa dideteksi untuk pria. Secara umum, pria memiliki kemungkinan lebih rendah untuk bertahan hidup ("dahulukan wanita dan anak "). Cara termudah untuk melihat hal ini adalah pada visualisasi Chart model **Naive Bayes**. Meski biasanya Naive Bayes bukan tipe model yang paling akurat, secara umum aturan yang ditetapkan adalah format yang mudah dibaca, di mana bisa berguna saat kita ingin menafsirkan model.

Nama : Wahyu Putra Adi Wibowo

Nim : 202420041

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
```

```
col_names = ['Age','Gender','Polyuria','Polydispia','Sudden Weight loss','Weakness',
             'Polyphagia','Genital Thrush','Visual blurring','Itching','Irritability',
             'Delayed Healing','Partial Paresis','Muscle stiffness','Alopecia','Obesity','Label']
# load dataset
diag = pd.read_csv("diabetes_data1.csv")
```

Pada kolom name diperlukan pada saat loading dataset tidak mempunyai header/ bisa menggunakan set **header** = None, pada kasus ini untuk data setyang berformat csv sudah memiliki header

```
diag.head()
```

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity	
0	40	1	0	1	0	1	0	0	0	1	0	1	0	1	1	1	Po
1	58	1	0	0	0	1	0	0	1	0	0	0	1	0	1	0	Po
2	41	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0	Po
3	45	1	0	0	1	1	1	1	0	1	0	1	0	0	0	0	Po
4	60	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	Po

Catatan Output :

- Gender 1 = Male ,
- 0 = Female ,
- Other attribute 1 = Yes,
- 0 = No

```
#split dataset in features and target variable
feature_cols = ['Age','Polyuria','Polydispia','Sudden Weight loss','Weakness',
                'Polyphagia','Genital Thrush','Visual blurring','Itching',
                'Irritability','Delayed Healing','Partial Paresis','Muscle stiffness','Alopecia','Obesity','Gender']
#X = diag[feature_cols] # Features
#y = diag.Label # Target variable
X=diag.iloc[:, 0:16].values
y=diag.iloc[:, 16].values
```

Note : Terdapat dua cara dalam pemilihan attribute, yaitu bisa menggunakan feature_cols & label (comment)/index.

Variable feature_cols akn digunakan saat menggambar sebuah grafis

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30% test
```

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
clf_tree = DecisionTreeClassifier(criterion='Entropy', random_state=1)
```

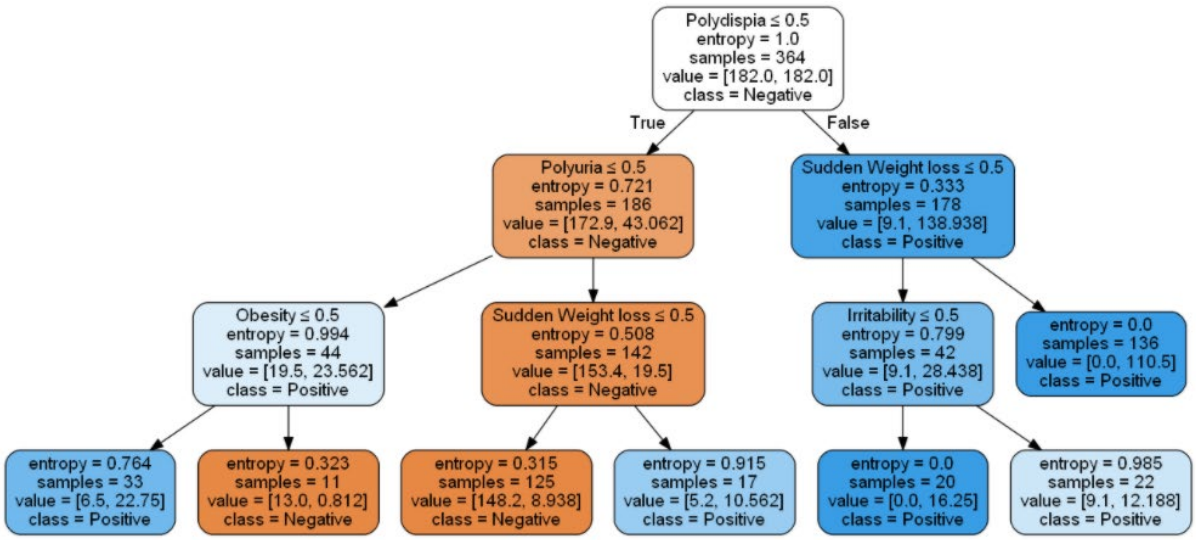
```
 # Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Accuracy : 0.9423076923076923

```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

clf = DecisionTreeClassifier(criterion = 'entropy', max_depth = 3, min_samples_split = 20, class_weight = "balanced")
clftree = clf.fit(X_train,y_train)


dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,
                feature_names = feature_cols,
                class_names=['Negative','Positive'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Mengunakan modul graphviz utk menampilkan hasil dari penganalisa data**Output :**



Catatan:

- Label = indikasi Diabetes
- Age : **Umur**
- Gender : **Jenis Kelamin**
- Polyuria : **sering buang air kecil**
- Polydispia : **sering merasa haus**
- Sudden Weight loss : **Berat Badan turun tiba-tiba**

- Weakness : **Lemah**
- Polyphagia : **banyak makan (sering lapar)**
- Genital Thrush : **gatal di kemaluan**
- Visual blurring : **Penglihatan Buram**
- Itching : **Gatal**
- Irritability : **gampang marah**
- Delayed Healing : **Sulit sembuh (luka)**
- Partial Paresis : **lumpuh sebagian**
- Muscle stiffness : **otot kaku**
- Alopecia : **kebotakan, rambut rontok**
- Obesity : **Kegemukan**

NAMA : WIDIA ASTUTI

NIM : 202420021

MATA KULIAH : ADVANCED DATABASE

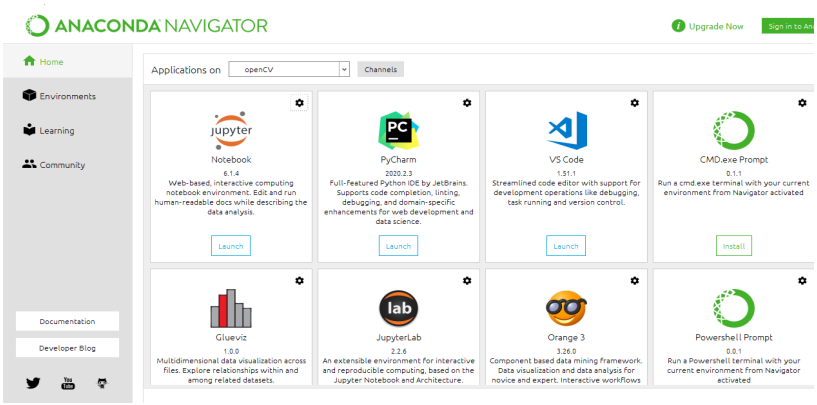
TUGAS 7

Dari tugas 06 sebelum ini, coba gunakan data yang sama, tapi gunakan pemprograman python, lalu buat tutorialnya.

Jawab :

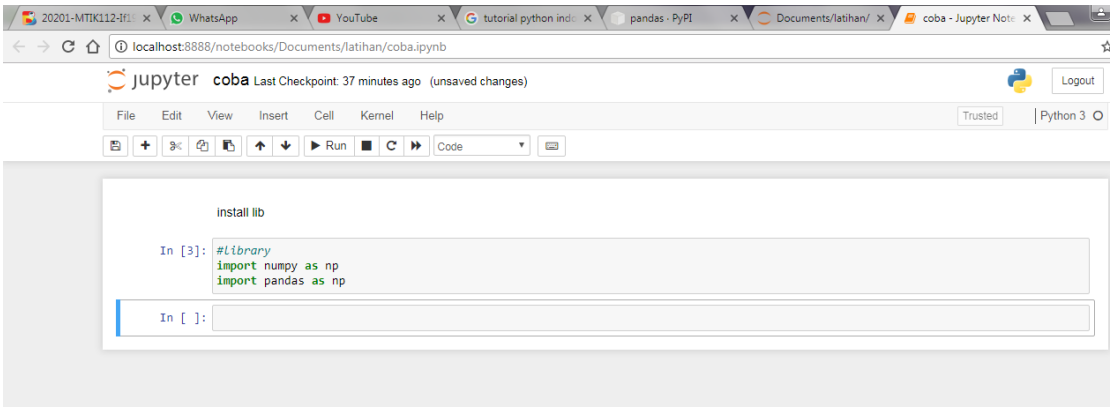
angkah-langkah

1. Install Python 3.8
2. Install Anaconda (www.anaconda.com)



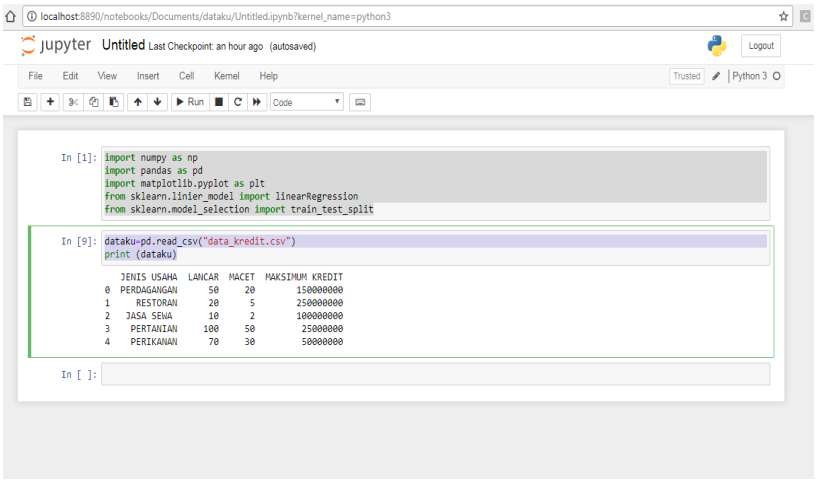
- Buat environment baru “OpenCV→pilih python 3.8
- Klik “openCV”→open library
- Buka google →pip.py→numpy→copy to clipboad→kembali ke environmet→paste→enter
- Buka google →pip.py→pandas→copy to clipboad→kembali ke environmet→paste→enter

3. Install jupyter
   - Klik launch→pilih documents→new→folder →rename”latihan”
   - Pilih “latihan”→new→python 3
   - Kembali ke” documents/latihan”→klik “untitled.ipynb”→shutdown
   - Pilih untitled.ipynb→rename→coba
   - Start (note:run →shift enter



4. Kembali ke Jupyter

   Ketik :
   - import numpy as np
   - import pandas as pd
   - import matplotlib.pyplot as plt
   - from sklearn.linier_model import linearRegression
   - from sklearn.model_selection import train_test_split
     → run
   - dataku=pd.read_csv("data_kredit.csv")
   - print (dataku)
     →run



Sumber :

https://www.youtube.com/watch?v=AnIU-QHyUXE

https://www.youtube.com/watch?v=HQaCF2F_MRo

https://www.youtube.com/watch?v=Xr7_zL0jyQY

TUGAS 07

APRIORI (MARKET BASCET ANALYSIS) - PHYTON

Dibuat Oleh

Aan Novrianto

Dosen Pengampu

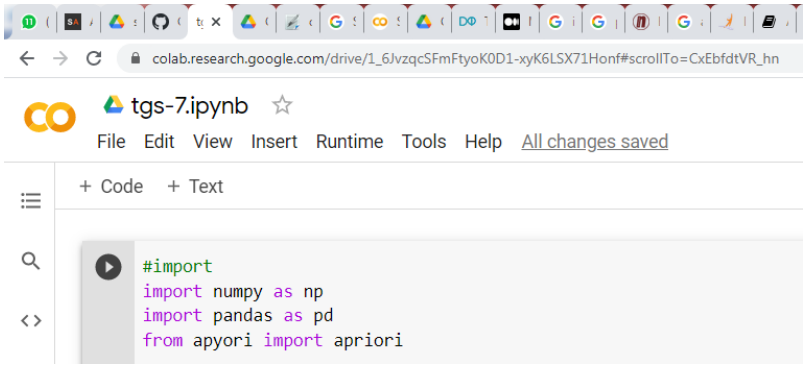
TRI BASUKI KURNIAWAN, S.Kom, M.Eng, Ph.D

Program Pasca Sarjana

Universitas Binadarma Palembang

2020/2021

Step 1: Import the libraries



Catatan penulis : pada Collabs modul apyori harus diinstall dulu menggunakan perintah

```
!pip install apyori
```

Step 2: Load the dataset

```
#loading data sheet
#import datasheet
from google.colab import drive
drive.mount('/content/drive')
store_data = pd.read_csv('drive/My Drive/Colab Notebooks/data-sheet-tugas-7-ok.csv', header=None)
```

Step 3: Have a glance at the records

```
#having glance of record
store_data

#look shape
store_data.shape
```

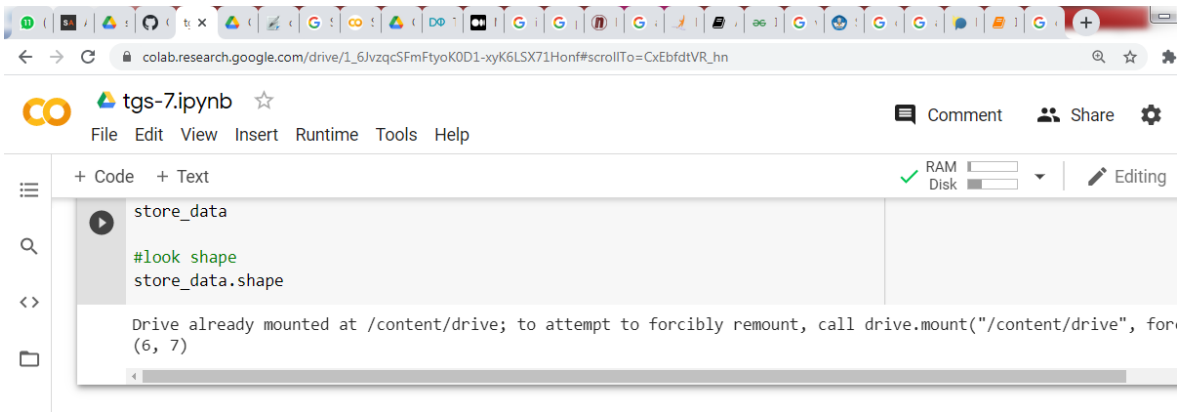
Output

```
store_data
```

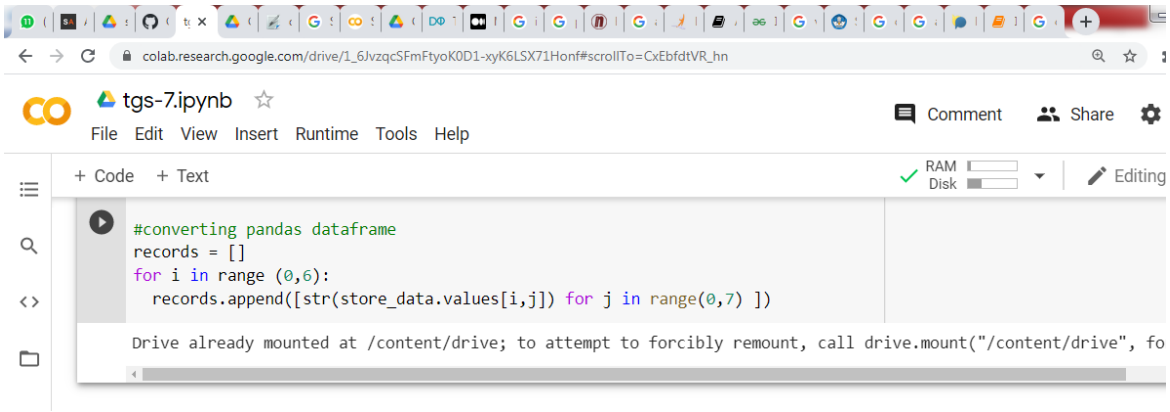
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive",
```

	0	1	2	3	4	5	6
0	PENA	ROTI	MENTEGA	TELUR	BUNCIS	SUSU	KECAP
1	PENA	ROTI	MENTEGA	NaN	NaN	NaN	NaN
2	NaN	ROTI	MENTEGA	TELUR	NaN	NaN	NaN
3	NaN	NaN	NaN	TELUR	BUNCIS	SUSU	NaN
4	NaN	ROTI	MENTEGA	TELUR	NaN	NaN	NaN
5	NaN	ROTI	MENTEGA	TELUR	NaN	SUSU	KECAP

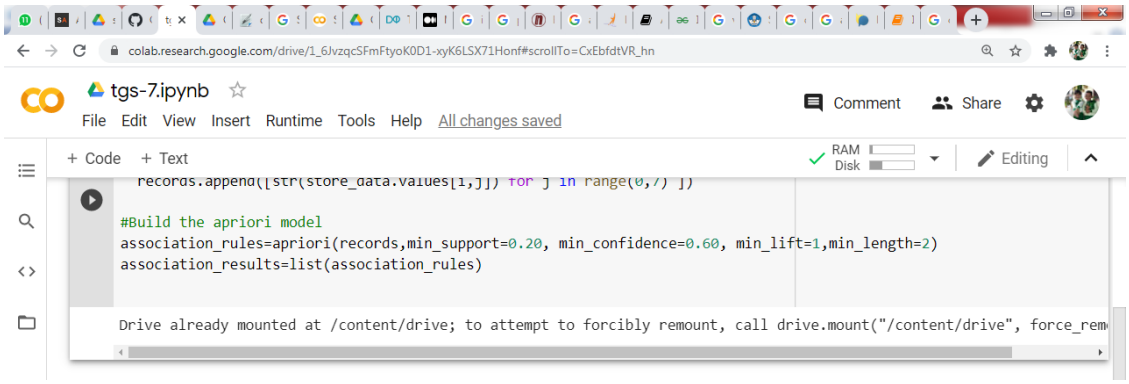
Step 4: Look at the shape



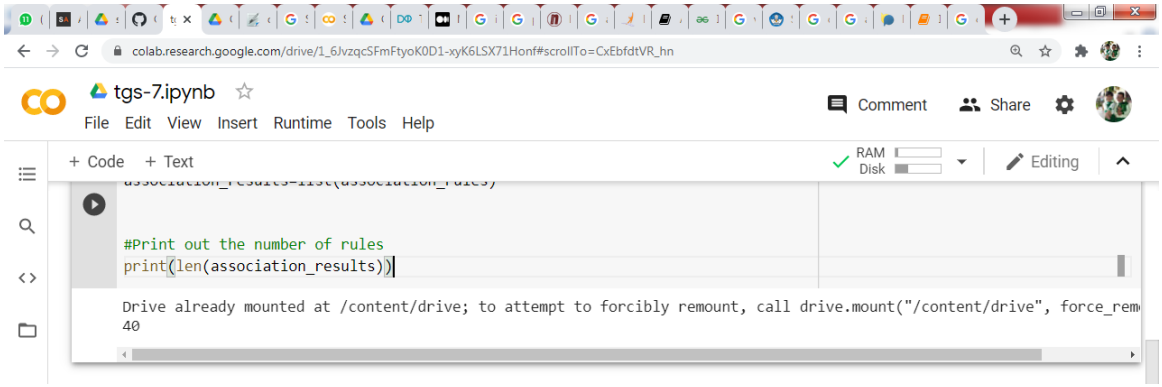
Step 5: Convert Pandas DataFrame into a list of lists



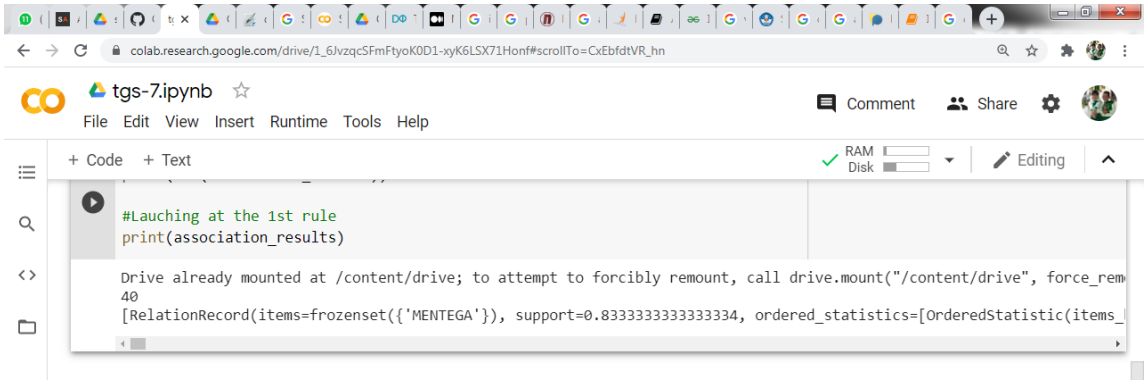
Step 6: Build the Apriori model



Step 7: Print out the number of rules



Step 8: Have a glance at the rule



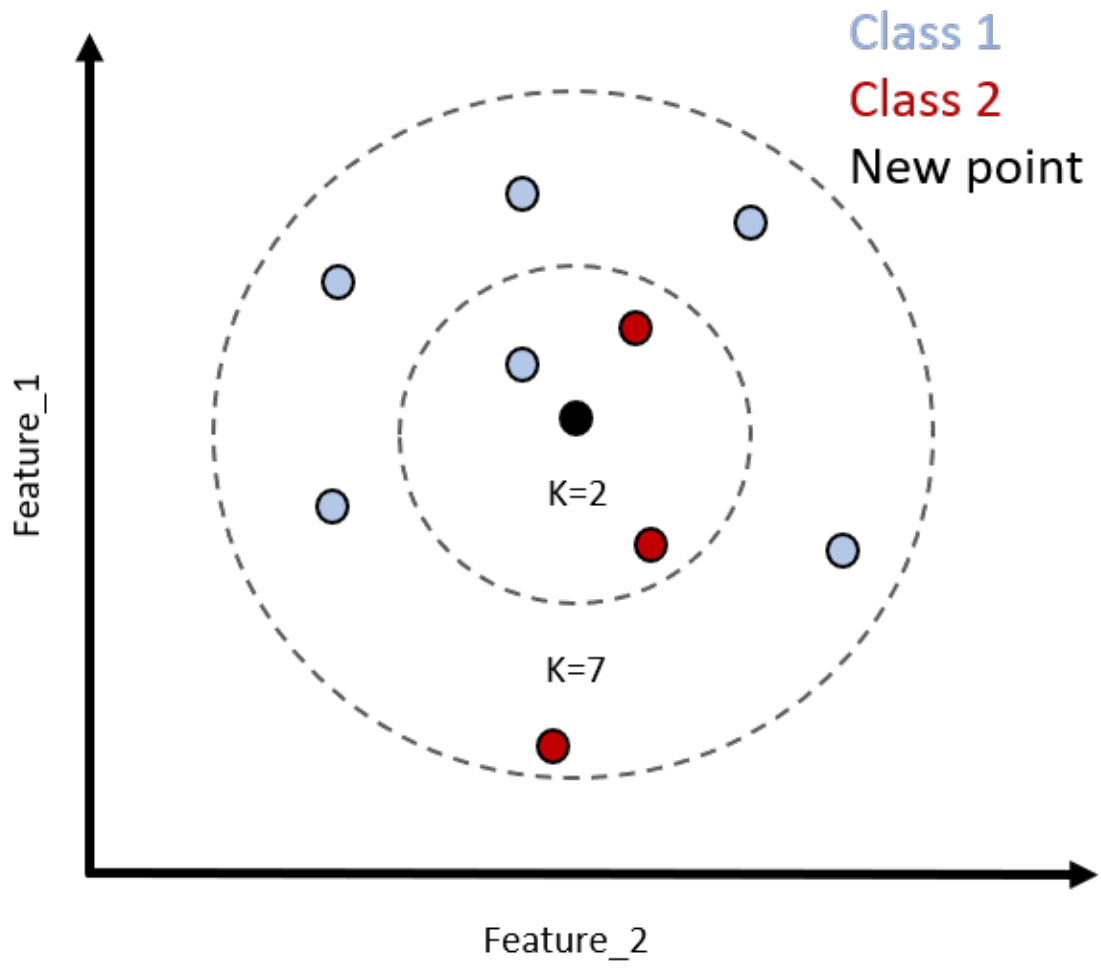
Script Phyton Lengkap

```
#import
import numpy as np
import pandas as pd
from apyori import apriori

#loading data sheet
#import datasheet
from google.colab import drive
drive.mount('/content/drive')
store_data = pd.read_csv('drive/My Drive/Colab Notebooks/data-sheet-
tugas-7-ok.csv', header=None)

#having glance of record
store_data

#look shape
store_data.shape

#converting pandas dataframe
records = []
for i in range (0,6):
  records.append([str(store_data.values[i,j]) for j in range(0,7) ])

#Build the apriori model
association_rules=apriori(records,min_support=0.20, min_confidence=0.60
, min_lift=1,min_length=2)
association_results=list(association_rules)


#Print out the number of rules
print(len(association_results))

#Lauching at the 1st rule
print(association_results)
```

NAMA : AHMAD ALI MA'MUN
NIM : 202420037

Tutorial Algoritma K-Nearest Neighbours Dengan Python

K-Nearest Neighbours (KNN)
KNN adalah algoritma pembelajaran mesin yang diawasi yang dapat digunakan untuk menyelesaikan masalah klasifikasi dan regresi. Prinsip KNN adalah nilai atau kelas suatu titik data yang ditentukan oleh titik data di sekitar nilai tersebut.
Untuk memahami algoritma klasifikasi KNN seringkali paling baik ditunjukkan melalui contoh. Tutorial ini akan mendemonstrasikan bagaimana Anda dapat menggunakan KNN dengan Python dengan masalah klasifikasi Anda sendiri. Notebook Jupyter yang sesuai dengan contoh ini dapat ditemukan di sini, jika Anda ingin mengikutinya.
Algoritme prediksi menghitung jarak dari titik x yang tidak diketahui, ke semua titik dalam data Anda. Titik-titik dalam data Anda kemudian diurutkan dengan menambah jarak dari x. Prediksi dilakukan dengan memprediksi label mayoritas dari titik terdekat 'K'.
Memilih K akan memengaruhi kelas tempat poin baru akan ditetapkan.
Dalam contoh di bawah ini, memilih nilai K 2 akan menetapkan titik yang tidak diketahui (lingkaran hitam) ke kelas 2. Namun, jika nilai K adalah 7, titik yang tidak diketahui akan ditetapkan ke kelas 1.



Membuat dataset palsu

Pertama, kita mengimpor pustaka yang kita butuhkan, dan kemudian membuat dataset palsu menggunakan fungsi makeblobs dari sklearn. Kita dapat mengirimkan jumlah sampel, fitur dalam kumpulan data kita, berapa banyak pusat atau kelas yang akan memasukkan data, dan terakhir deviasi standar dari cluster tersebut. Untuk konsistensi antara beberapa jalannya notebook Jupyter ini, saya telah menetapkan integer 101 ke parameter random_state.

Catatan, untuk memulai, kita akan memiliki standar deviasi cluster yang besar. Ini akan memperkenalkan varians ke dalam klasifikasi, yang dapat kita perbaiki nanti, dengan secara khusus memilih nilai K yang optimal. Ini dapat dicapai dengan menggunakan metode siku.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.datasets import make_blobs
```

```
data = make_blobs(n_samples=300, n_features=5, centers=2, cluster_std=6.0, random_state=101)
```

```
data
```

```
(array([[ -0.95757537,   3.36332609, -15.54675979, -14.02967497,
           1.50545246],
        [-11.12008037,  -0.86726927, -19.42687054, -22.99153445,
          12.8409123 ],
        [  5.02786886,  -2.84037069,  -5.9094317 , -16.29765383,
           7.77075032],
        ...,
        [ -8.02114181,   2.29827056, -13.80731349, -10.89022536,
           1.99399904],
        [ 10.87670302,   3.25562702,  -6.25095388,  -0.92884525,
           8.18286695],
        [  7.86530195, -11.18764669,   6.36417619,  -2.87676038,
           1.31626729]]),
 array([0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
        1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
        1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
        0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0,
        1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
        0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
```

Fungsi makeblobs dari sklearn mengembalikan tupel 2 elemen. Kita dapat membuat kerangka data fitur kita menggunakan pd.DataFrame, dan meneruskan indeks tupel pertama yang sesuai dengan data fitur. Elemen kedua dari tupel data sesuai dengan label fitur.

```
df_feat = pd.DataFrame(data[0], columns=['feature_' + str(i) for i in range(1, 6)])
```

```
df_feat.head(2)
```

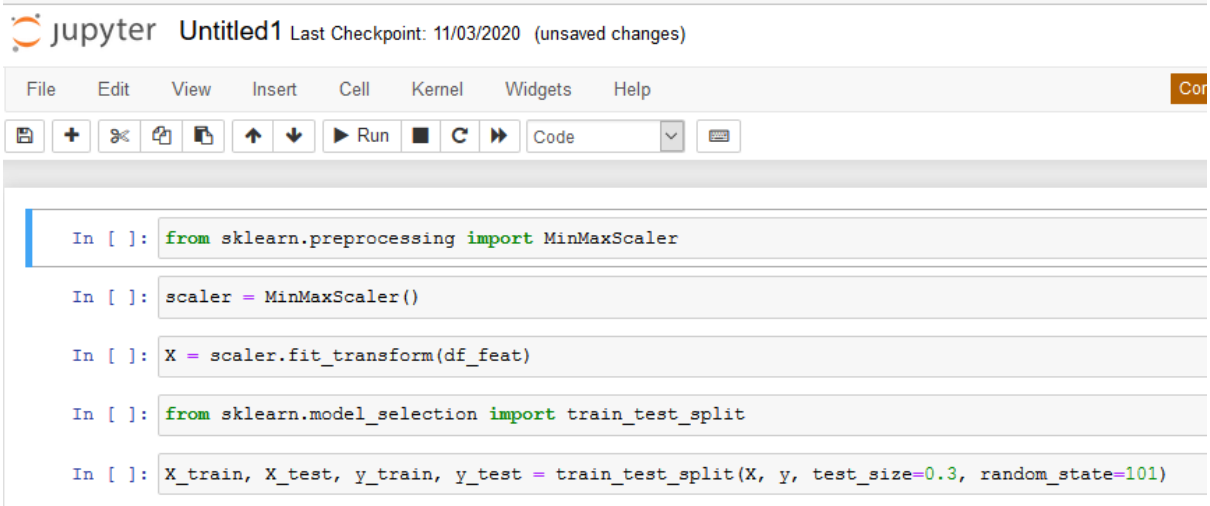
	feature_1	feature_2	feature_3	feature_4	feature_5
0	-0.957575	3.363326	-15.546760	-14.029675	1.505452
1	-11.120080	-0.867269	-19.426871	-22.991534	12.840912

```
y = data[1]
```

```
y
```

```
array([0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1,
```

Sekarang kita dapat menskalakan data dengan mengimpor MinMaxScaler dari Sklearn.preprocessing. Tidak seperti algoritme pembelajaran mesin lainnya, kami menyesuaikan dan mengubah semua data pelatihan, sebelum melakukan pemisahan pengujian kereta kami.



Algoritme prediksi dan pengoptimalan

Untuk mengimplementasikan prediksi dalam kode, kita mulai dengan mengimpor KNeighboursClassifier dari sklearn.neighbours. Kami kemudian membuat instance dari KNeighboursClassifier, dengan meneruskan argumen 1 ke n_neighbours, dan menetapkan ini ke variabel knn. Nilai yang diteruskan ke n_neighbours mewakili nilai K.

Kami kemudian menyesuaikan dengan data pelatihan, sebelum membuat prediksi, dengan memanggil metode prediksi pada objek KNeighboursClassifier kami.

Sekarang kita dapat menilai keakuratan prediksi menggunakan klasifikasi_report dan konfusi_matriks.

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier(n_neighbors=1)
```

```
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

```
predictions = knn.predict(X_test)
```

```
predictions
```

```
array([0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 1])
```

Metrik tersebut menunjukkan bahwa akurasinya sudah sangat baik. Ini mungkin karena fakta bahwa kami membuat kumpulan data dengan makeblobs dan secara khusus meminta 2 pusat. Namun, kami sengaja menempatkan nilai yang besar untuk standar deviasi cluster untuk memperkenalkan varians. Hal ini mengakibatkan kesalahan klasifikasi pada 4 poin dalam kumpulan data kami.

```
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
[[42  1]
 [ 3 44]]
              precision    recall  f1-score   support

           0       0.93      0.98      0.95        43
           1       0.98      0.94      0.96        47

    accuracy                           0.96        90
   macro avg       0.96      0.96      0.96        90
weighted avg       0.96      0.96      0.96        90
```

Meningkatkan Akurasi Data

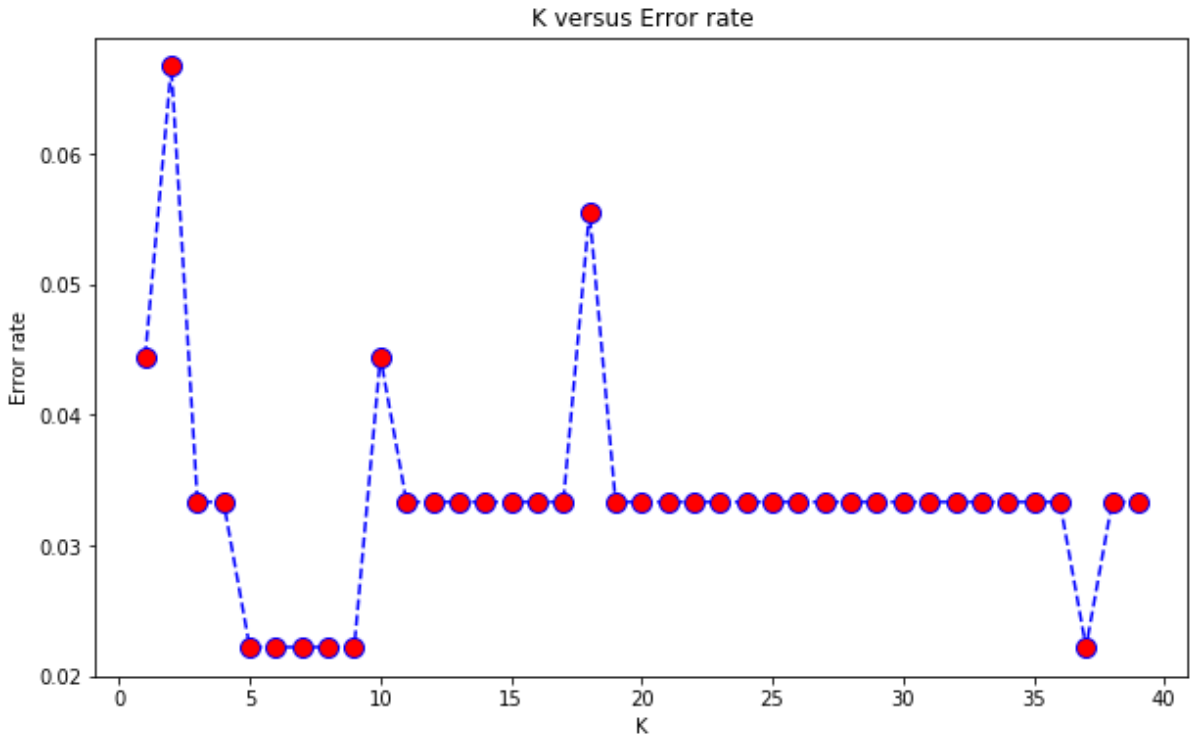
Kami dapat mencoba untuk meningkatkan akurasi hasil kami dengan memodifikasi jumlah tetangga. Ini dapat dicapai dengan menggunakan metode siku.

Pertama-tama kita melakukan iterasi melalui 40 nilai tetangga, membuat instance objek KNeighboursClassifier dengan jumlah tetangga tersebut. Kami kemudian dapat menyesuaikan data pelatihan dengan model KNN ini, mendapatkan prediksi, dan menambahkan nilai rata-rata antara prediksi, pred_i dan nilai yang benar, y_test.

Jika pred_i dan y_test tidak cocok dalam array, nilai true dikembalikan yang memiliki nilai 1. Semakin tinggi angkanya, semakin tidak akurat klasifikasi tersebut.
Nilai yang lebih rendah untuk tingkat kesalahan akan sesuai dengan model yang berkinerja lebih baik.
Hasil ini dapat diplot menggunakan rentang nilai i pada sumbu x, versus tingkat kesalahan pada sumbu y.

```
error_rate = []

for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)

    error_rate.append(np.mean(pred_i != y_test))


plt.figure(figsize=(10, 6))

plt.plot(range(1, 40), error_rate, color='blue', linestyle='--',
         markersize=10, markerfacecolor='red', marker='o')
plt.title('K versus Error rate')
plt.xlabel('K')
plt.ylabel('Error rate')
```

Sekarang kita dapat memilih nilai K terendah yang akan menghasilkan, tingkat kesalahan terendah. Di sini, kita bisa memilih 5.



Sekarang kita dapat, menjalankan kembali penilaian akurasi dengan matriks kebingungan dan laporan klasifikasi sekali lagi, untuk melihat apakah kita mengklasifikasikan 4 poin yang tidak selaras dengan lebih akurat. Kami telah meningkatkan, dari 4 poin yang salah diklasifikasikan menjadi 2.

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
predictions = knn.predict(X_test)


print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
[[42  1]
 [ 1 46]]
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        43
           1       0.98      0.98      0.98        47

    accuracy                           0.98        90
   macro avg       0.98      0.98      0.98        90
weighted avg       0.98      0.98      0.98        90
```

Pelatihan tentang titik data baru

Sekarang kita dapat membuat titik data menggunakan data asli. Pertama kita membuat dua dataframe; satu dengan fitur dan satu lagi dengan label, menggabungkannya menjadi satu kerangka data, dan memilih baris pertama, sebagai titik data untuk memprediksi labelnya. Kita harus ingat untuk menskalakan titik data karena model dilatih pada data yang diskalakan.

Prediksi menunjukkan bahwa data titik 1, akan memberikan label 0, yang sesuai dengan titik dataset asli, diverifikasi dengan memanggil df.head (1).

```
features = pd.DataFrame(data=data[0], columns=['feature_' + str(i) for i in range(1, 6)])
lables = pd.DataFrame(data[1], columns=['labels'])
dataset = pd.concat([features, lables], axis=1)
data_point_1 = scaler.transform(np.array(dataset.iloc[0][:-1]).reshape(-1, 5))
knn.predict(data_point_1)[0]

# Output
# 0
```

	feature_1	feature_2	feature_3	feature_4	feature_5	labels
0	-0.957575	3.363326	-15.54676	-14.029675	1.505452	0

```
data_point_1 = scaler.transform(np.array(dataset.iloc[0][:-1]).reshape(-1, 5))
```

```
data_point_1
```

```
array([[0.3943128 , 0.58849094, 0.1949979 , 0.21287012, 0.54980842]])
```

```
knn.predict(data_point_1)[0]
```

```
0
```

TUGAS 7
ADVANCED DATABASE

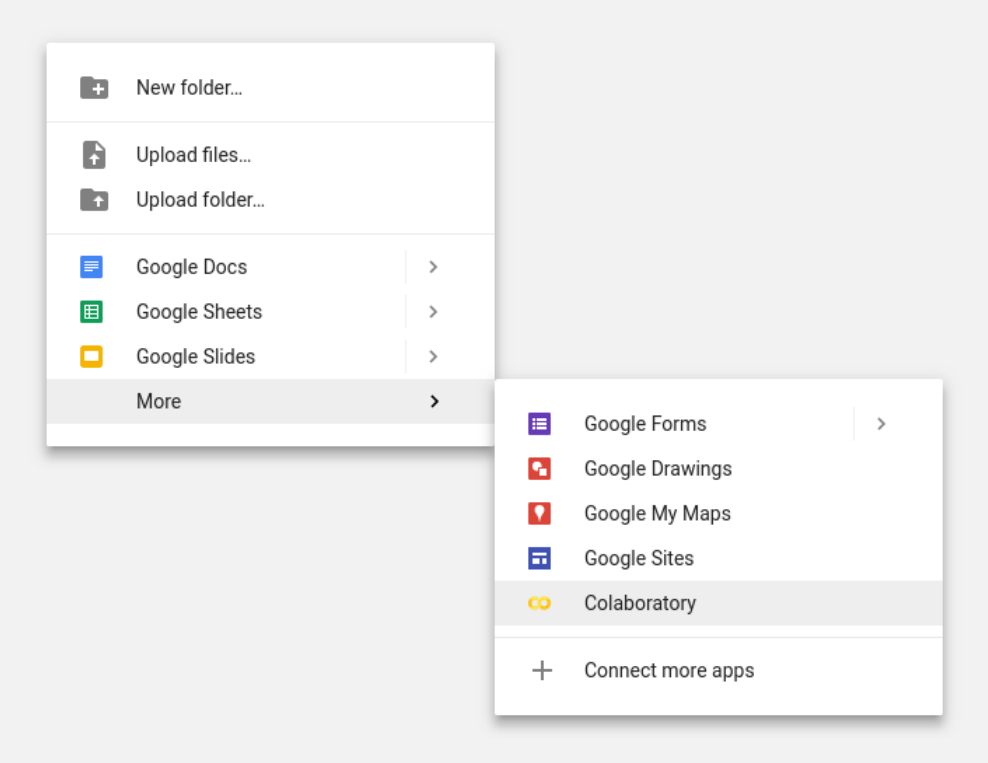
Dari tugas 06 sebelum ini, coba gunakan data yang sama, tapi gunakan pemprograman python, lalu buat tutorialnya.

Jawab :

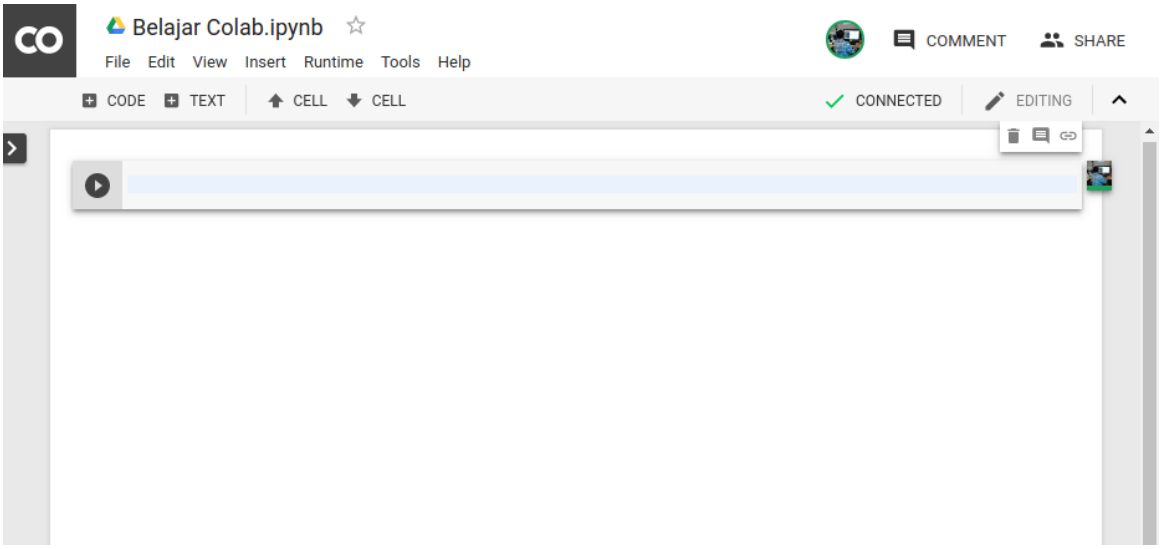
Untuk dapat menggunakan Google Colab kita perlu menambahkan ekstensi baru ke Google Drive kita dengan cara klik tombol **New > More > Connect more apps** lalu tuliskan "colab" pada kolom search kemudian klik tombol **connect**.

- **Membuat Notebook**

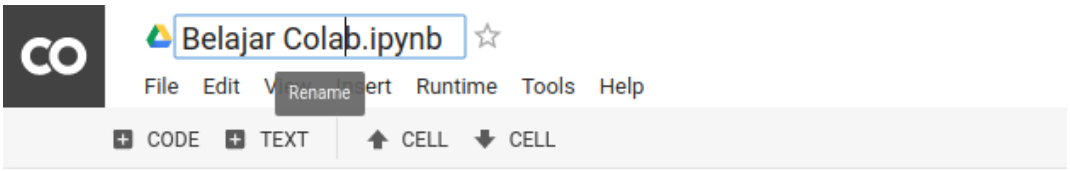
  Jika Colab sudah terintegrasi dengan Drive kita maka kita siap untuk menggunakannya, pertama kita buat direktori baru terlebih dahulu dengan cara klik tombol **New** kemudian pilih **Folder** lalu berikan nama direktori tersebut (misal: Colab) kemudian buat file Notebook baru dengan cara klik kanan pada area kosong didalam direktori yang baru saja kita buat **Klik kanan > More > Colaboratory**.



Maka tampilan awal dari Notebook kita adalah seperti berikut

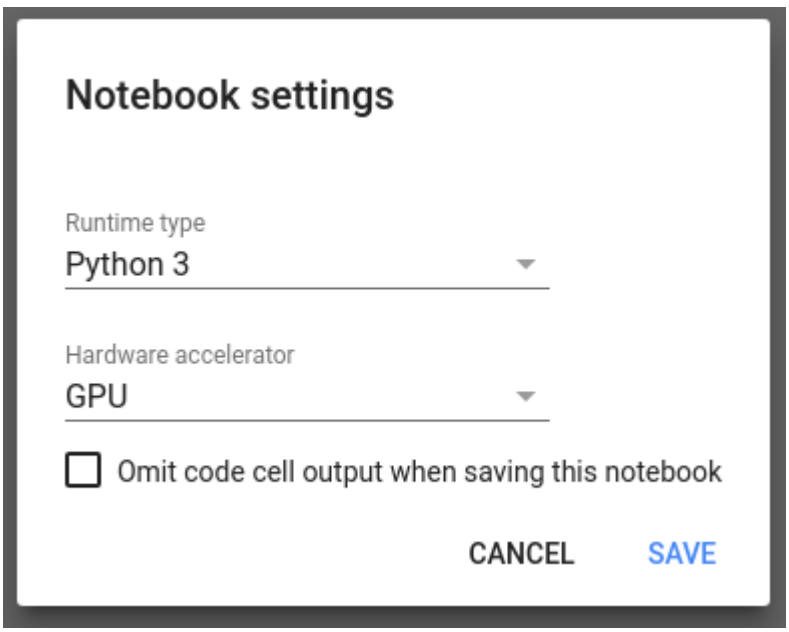


Ada baiknya kita berikan nama notebook kita dengan cara double klik pada area **Untitled0.ipnyb** lalu tuliskan nama notebook seperti berikut.



- **Setup GPU**

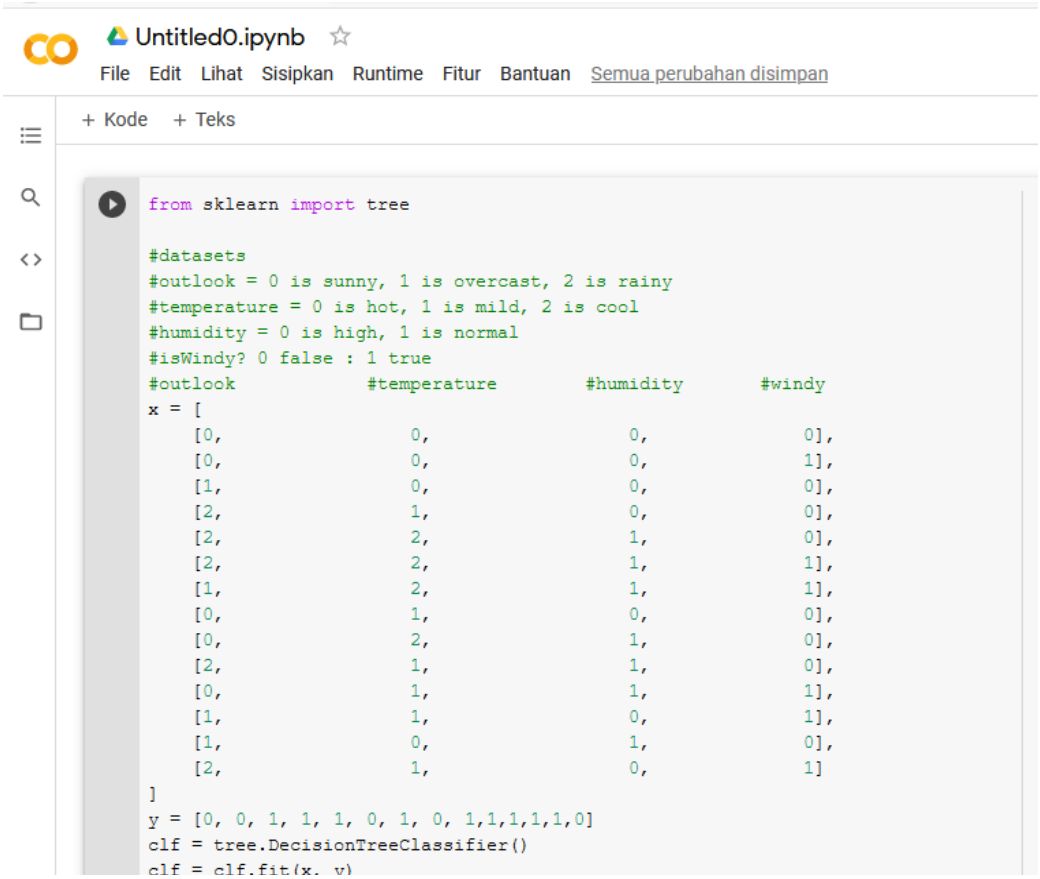
  Karena Colab menyediakan GPU gratis untuk penggunanya kita pun dapat memberikan pengaturan pada notebook untuk menggunakan layanan GPU gratis tersebut caranya klik menu **Edit > Notebook settings** kemudian ubah "Hardware accelerator" menjadi GPU dan kita juga dapat mengubah runtime versi Python pada notebook sedang aktif.



- **Menjalankan Kode Pertama**

  Proses setup sudah selesai dan mari kita mencoba beberapa operasi tipe data dasar, berikut potongan kodenya

```
from sklearn import tree

#datasets
#outlook = 0 is sunny, 1 is overcast, 2 is rainy
#temperature = 0 is hot, 1 is mild, 2 is cool
#humidity = 0 is high, 1 is normal
#isWindy? 0 false : 1 true
#outlook        #temperature      #humidity      #windy
x = [
    [0,             0,              0,          0],
    [0,             0,              0,          1],
    [1,             0,              0,          0],
    [2,             1,              0,          0],
    [2,             2,              1,          0],
    [2,             2,              1,          1],
    [1,             2,              1,          1],
    [0,             1,              0,          0],
    [0,             2,              1,          0],
    [2,             1,              1,          0],
    [0,             1,              1,          1],
    [1,             1,              0,          1],
    [1,             0,              1,          0],
    [2,             1,              0,          1]
]
y = [0, 0, 1, 1, 1, 0, 1, 0, 1,1,1,1,1,0]
clf = tree.DecisionTreeClassifier()
clf = clf.fit(x, y)


def prog() :
    outlook = int(input("Outlook? (0 is sunny, 1 overcast, 2 rainy) : "))
    humidity = int(input("Temperature? (0 is hot, 1 is mild, 2 is cool) : "))
    temp = int(input("Humidty? (0 is high, 1 is normal) : "))
    isWindy = int(input("Is Windy? 0 false, 1 true : " ))

    if clf.predict([[outlook, humidity, temp, isWindy]]):
        print("\nYes,I Play golf")
    else:
        print("\nNo Play golf")


prog()
```



Untuk menjalankannya klik runtime / jalankan semuanya. Maka akan muncul seperti ini hasilnya

Nama : Andry Meylani
NIM : 202420009
TUGAS 07

Dari Tugas 6 didapatkan data sebagai berikut :

TID	PENA	ROTI	MENTEGA	TELUR	BUNCIS	SUSU	KECAP
001	1	1	1	0	0	0	0
002	0	1	1	1	0	0	0
003	0	0	0	1	1	1	0
004	0	1	1	0	0	0	0
005	0	1	1	1	0	1	1

Dari data diatas akan saya coba untuk membuat association rule meggunakan bahasa pyhton dengan colab.research.google.com, berikut langkah-langkahnya :

Langkah 1
Download Libabry Association Rule

Langkah 2,
Masukkan data mengenai assosiation rule yang akan diproses (data terlampir diatas), Tentukan minimum supportnya pada kali ini saya menggunakan 0.6

```
[ ] # ! pip install mlxtend
    import pandas as pd
    from mlxtend.preprocessing import TransactionEncoder
    from mlxtend.frequent_patterns import apriori
    from mlxtend.frequent_patterns import association_rules

    dataset = [['Pena', 'Roti', 'Mentega'],
               ['Roti', 'Mentega', 'Telur'],
               ['Telur', 'Buncis', 'Susu'],
               ['Roti', 'Mentega'],
               ['Roti', 'Mentega', 'Telur', 'Susu', 'Kecap']]

    te = TransactionEncoder()
    te_ary = te.fit(dataset).transform(dataset)
    df = pd.DataFrame(te_ary, columns=te.columns_)
    frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)

    frequent_itemsets
```

Langkah 3,

Tentukan nilai rules sesuai yang diinginkan :

```
rules[ (rules['antecedent_len'] >= 2) &
       (rules['confidence'] > 0.75) &
       (rules['lift'] > 1.2) ]
```

Langkah 4

Kemudian RUN All dari seluruh config yang sudah ada, adapun hasilnya sebagai berikut :

	support	itemsets
0	0.8	(Mentega)
1	0.8	(Roti)
2	0.6	(Telur)
3	0.8	(Roti, Mentega)

```
[ ] association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
    rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
    rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf	1
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf	1

TUGAS 07

APRIORI (MARKET BASCET ANALYSIS) - PHYTON

Dibuat Oleh

Ari Hardiyantoro Susanto

Dosen Pengampu

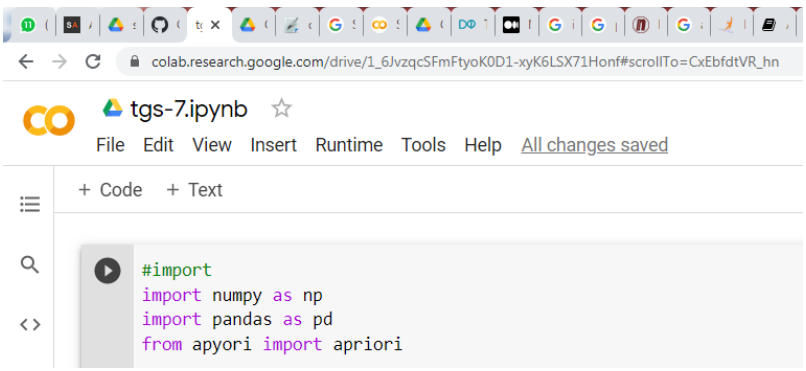
TRI BASUKI KURNIAWAN, S.Kom, M.Eng, Ph.D

Program Pasca Sarjana

Universitas Binadarma Palembang

2020/2021

Step 1: Import the libraries



Pada Collabs modul apriori harus diinstall dulu menggunakan perintah

!pip install apyori

Step 2: Load the dataset

```
#loading data sheet
#import datasheet
from google.colab import drive
drive.mount('/content/drive')
store_data = pd.read_csv('drive/My Drive/Colab Notebooks/data-sheet-tugas-7-ok.csv', header=None)
```

Step 3: Have a glance at the records

```
#having glance of record
store_data

#look shape
store_data.shape
```

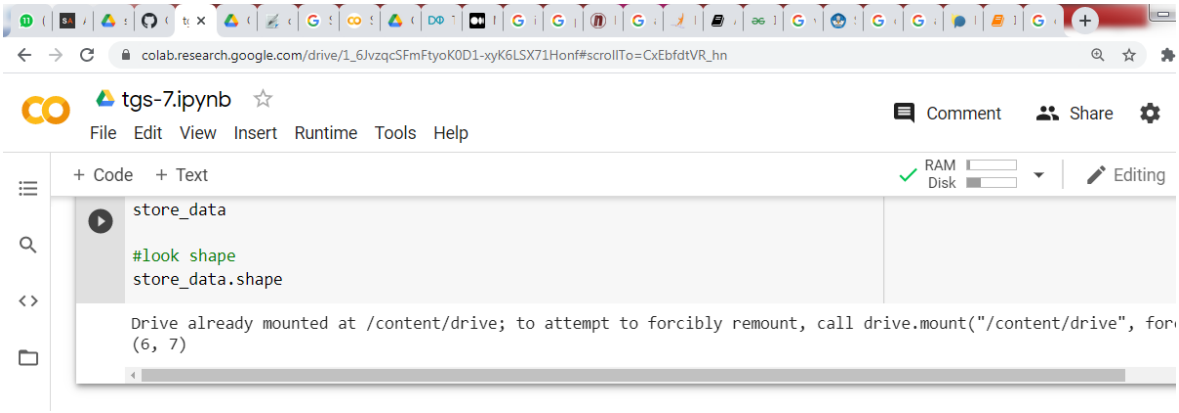
Output

```
store_data
```

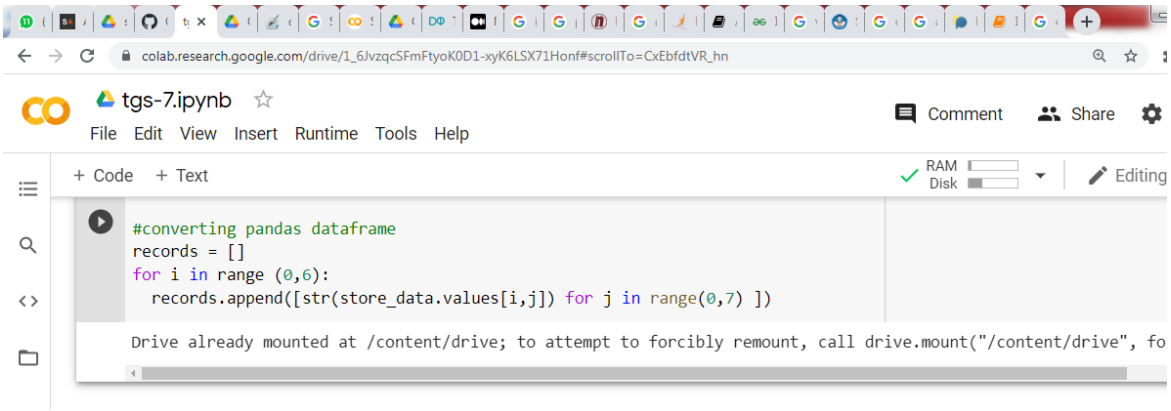
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive",
```

	0	1	2	3	4	5	6
0	PENA	ROTI	MENTEGA	TELUR	BUNCIS	SUSU	KECAP
1	PENA	ROTI	MENTEGA	NaN	NaN	NaN	NaN
2	NaN	ROTI	MENTEGA	TELUR	NaN	NaN	NaN
3	NaN	NaN	NaN	TELUR	BUNCIS	SUSU	NaN
4	NaN	ROTI	MENTEGA	TELUR	NaN	NaN	NaN
5	NaN	ROTI	MENTEGA	TELUR	NaN	SUSU	KECAP

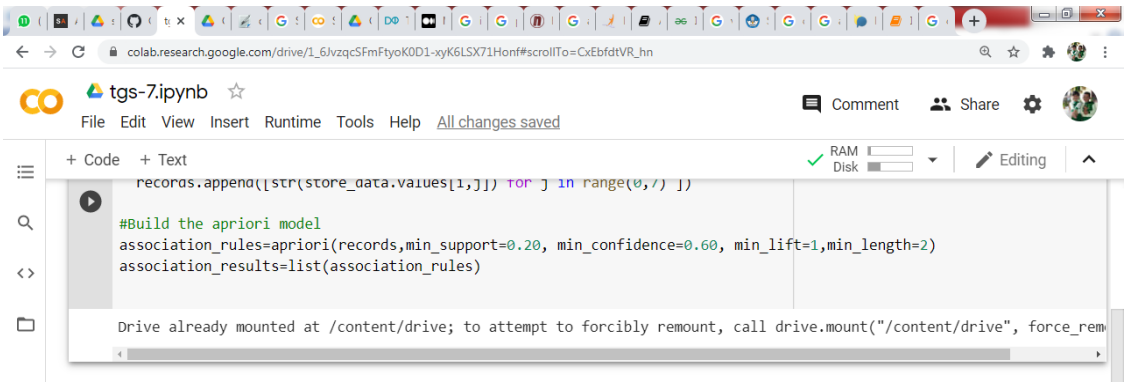
Step 4: Look at the shape



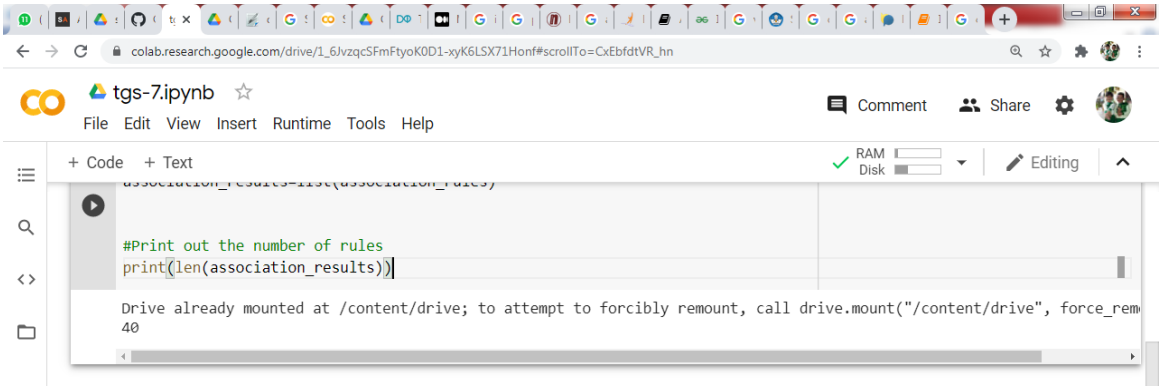
Step 5: Convert Pandas Data Frame into a list of lists



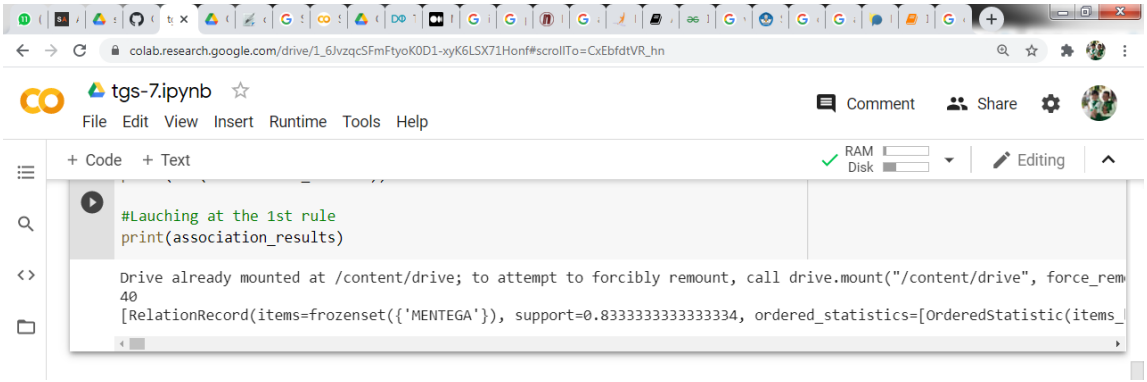
Step 6: Build the Apriori model



Step 7: Print out the number of rules



Step 8: Have a glance at the rule



Script Phyton Lengkap

```
#import
import numpy as np
import pandas as pd
from apyori import apriori

#loading data sheet
#import datasheet
from google.colab import drive
drive.mount('/content/drive')
store_data = pd.read_csv('drive/My Drive/Colab Notebooks/data-sheet-tugas-7-
ok.csv', header=None)

#having glance of record
store_data

#look shape
store_data.shape

#converting pandas dataframe
records = []
for i in range (0,6):
  records.append([str(store_data.values[i,j]) for j in range(0,7) ])

#Build the apriori model
association_rules=apriori(records,min_support=0.20, min_confidence=0.60, min_lift=1,min_leng
th=2)
association_results=list(association_rules)


#Print out the number of rules
print(len(association_results))

#Lauching at the 1st rule
print(association_results)
```

NIM : 202420008
NAMA : BERI PERIMA

TUTORIAL PENGOLAHAN DATA ALGORITMA DECISION TREE C4.5 DENGAN MENGGUNAKAN PYTHON

MODEL PREDIKSI ELEKTABILITAS CALON LEGISLATIF MENGGUNAKAN DATA PEMILU SEBAGAI DATA TRAINING

A. Decision Tree

(Andriani, 2012) Pada decision tree terdapat 3 jenis node, yaitu: a. Root Node, merupakan node paling atas, pada node ini tidak ada input dan bisa tidak mempunyai output atau mempunyai output lebih dari satu. b. Internal Node, merupakan node percabangan, pada node ini hanya terdapat satu input dan mempunyai output minimal dua. Leaf Node atau Terminal N

B. Algoritma C4.5

Algoritma C4.5 dan pohon keputusan merupakan dua model yang tidak dapat terpisahkan, karena untuk membangun sebuah pohon keputusan dibutuhkan algoritma C4.5.

Menurut (Nasari, 2014) Secara umum alur proses algoritma C4.5 untuk membangun pohon keputusan dalam data mining adalah:

1. Pilih atribut sebagai simpul akar.
2. Buat cabang untuk tiap-tiap nilai.
3. Bagi kasus dalam cabang.
4. Ulangi proses untuk setiap cabang sampai semua kasus pada cabang memiliki kelas yang sama.

Pemilihan atribut sebagai simpul, baik akar (root) atau simpul internal didasarkan pada nilai Gain tertinggi dari atribut-atribut yang ada.

Untuk Menghitung nilai entropy digunakan rumus:

$$Entropy\,(S) = \sum_{i=1}^{n} -pi \log_2 pi$$

Keterangan :
S : himpunan kasus.
n : jumlah partisi S
Pi : proporsi Si terhadap S

Kemudian hitung nilai informationgain menggunakan rumus:

$$Gain\,(S,A) = entropy(S) - \sum_{i=1}^{n} \frac{|Si|}{S} * entropy(Si)$$

NIM : 202420008
NAMA : BERI PERIMA

Keterangan:
S : himpunan kasus
A : fitur
n : jumlah partisi atribut A
|S1| : proporsi Si terhadap S
|S2| : jumlah kasus dalam S

Pada penelitian ini Algoritma C4.5 digunakan untuk perhitungan nilai information gain pada setiap paramater penentu keputusan pemilihan batik. Untuk melakukan perhitungan nilai entropy dan information gain tertinggi dari setiap parameter, sistem akan menghitung berapa banyak jumlah “Ya” dan “Tidak” dari setiap parameter.

Berikut ini merupakan contoh kasus Elektebilitas Calon Legislatif untuk memudahkan penjelasan mengenai algoritma C4.5.

Dataset yang digunakan menggunakan dataset “datapemilukpu.csv”

Tujuan dari tutorial ini adalah untuk memprediksi elektabilitas calon legislative dengan parameter “Ya” atau “Tidak”. Dataset berisi data sekitar 425 dengan 11 atribut,

Tahap awal , melakukan pemanggilan dataset student-por.csv dan menseting modul pandas, sklearn Decision Tree dan memilihi data training maupun data testing, seperti kode dibawah ini

```
# Load libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
```

```
col_names = ['NAMA PARTAI POLITIK','NAMA CALON LEGESLATIF','JENIS KELAMIN','KECAMATAN',
             'NO.URUT PARPOL', 'SUARA SAH PARTAI', 'JUML.PEROLEHAN KURSI', 'DAERAH PEMILIHAN',
             'NO.URUT CALEG','SUARA SAH CALEG','TERPILIH ATAU TIDAK']
# load dataset
diag = pd.read_csv("datapemilukpu.csv")
diag.head()
```

NIM : 202420008
NAMA : BERI PERIMA

Tampilan dataset dalam bentuk table

	NAMA PARTAI POLITIK	NAMA CALON LEGESLATIF	JENIS KELAMIN	KECAMATAN	NO.URUT PARPOL	SUARA SAH PARTAI	JUML.PEROLEHAN KURSI	DAERAH PEMILIHAN	NO.URUT CALEG	SUARA SAH CALEG	TERPILIH ATAU TIDAK
0	HATI NURANI RAKYAT	TOTO SUKISNO,BSc	L	LEBAKSIU	1	18578	1	1	1	594	TIDAK
1	HATI NURANI RAKYAT	EDI PURYANTO,SH	L	SLAWI	1	18578	1	1	2	943	TIDAK
2	HATI NURANI RAKYAT	ELI RETNOWATI,SH	P	SLAWI	1	18578	1	1	3	1730	TIDAK
3	HATI NURANI RAKYAT	SAHYUDIN	L	DUKUHWARU	1	18578	1	1	4	2508	YA
4	HATI NURANI RAKYAT	H.FAJAR SIGIT KUSUMAJAYA,SH	L	SLAWI	1	18578	1	2	1	923	TIDAK

Menyesuaikan attribute yang dibutuhkan

```
M↓
diag  = diag.drop(["NAMA PARTAI POLITIK", "NAMA CALON LEGESLATIF", "JENIS KELAMIN", "KECAMATAN",
                   "NO.URUT PARPOL", "JUML.PEROLEHAN KURSI", "NO.URUT CALEG"], axis = 1)
diag.head()
```

	SUARA SAH PARTAI	DAERAH PEMILIHAN	SUARA SAH CALEG	TERPILIH ATAU TIDAK
0	18578	1	594	TIDAK
1	18578	1	943	TIDAK
2	18578	1	1730	TIDAK
3	18578	1	2508	YA
4	18578	2	923	TIDAK

Pisahkan dataset feature column dan target variable

```
M↓
#split dataset in features and target variable
feature_cols = ['SUARA SAH CALEG','SUARA SAH PARTAI','DAERAH PEMILIHAN']

#X = diag[feature_cols] # Features
#y = diag.Label # Target variable
X=diag.iloc[:, 0:3].values
y=diag.iloc[:, 3].values
```

Pisahkan dataset data Training set dan test set

```
M↓
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30%
test
```

NIM : 202420008
NAMA : BERI PERIMA

Membuat objek klasifikasi

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
clf_tree = DecisionTreeClassifier(criterion='Entropy', random_state=1)
```

```
# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

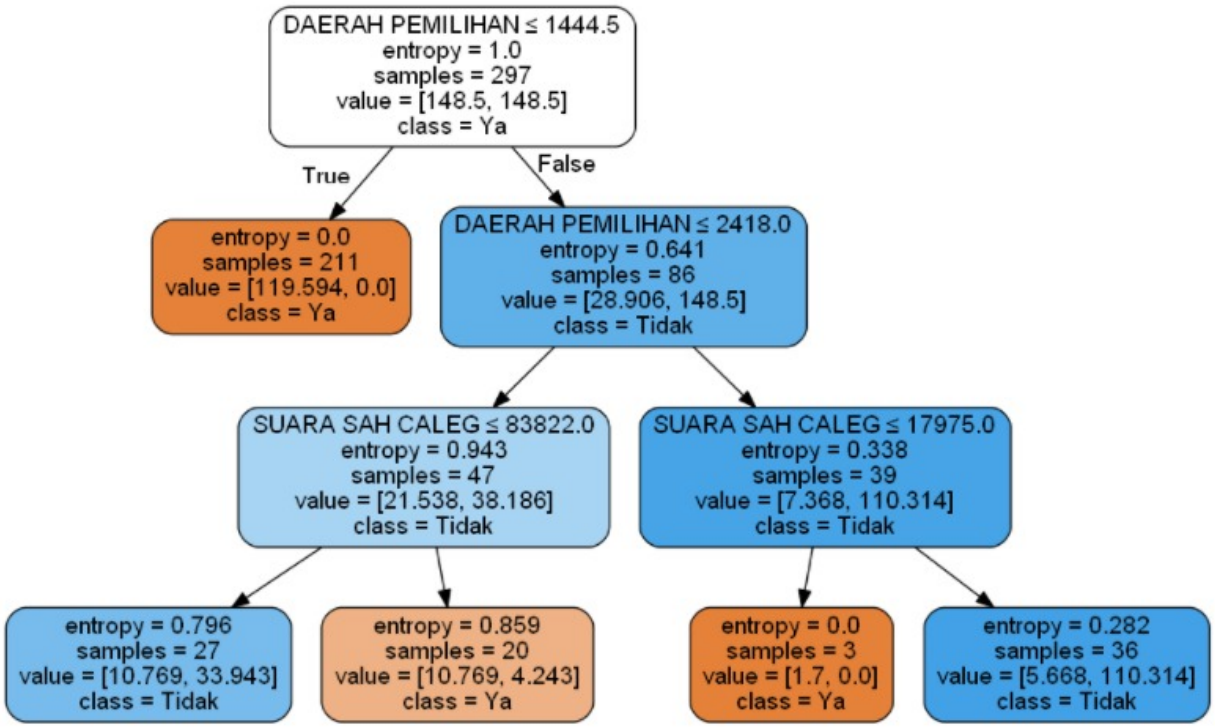
```
Accuracy: 0.859375
```

Hasil Accuracy = 0.859375

```
from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus

clf = DecisionTreeClassifier(criterion = 'entropy', max_depth = 3, min_samples_split = 20,
class_weight = "balanced")
clftree = clf.fit(X_train,y_train)


dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,
                feature_names = feature_cols,
                class_names=['Ya','Tidak'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('datapemilukpu.png')
Image(graph.create_png())
```

NIM : 202420008
NAMA : BERI PERIMA

Bentuk Pohon Keputusan



Nama : Bhijanta Wyasa WM
NIM : 202420019
Kelas : MTI 23

Tugas 07

Dari tugas 06 sebelum ini, coba gunakan data yang sama, tapi gunakan pemprograman python, lalu buat tutorialnya.
Harap dapat dikumpulkan sebelum batas waktu yang ditentukan!

Jawaban :

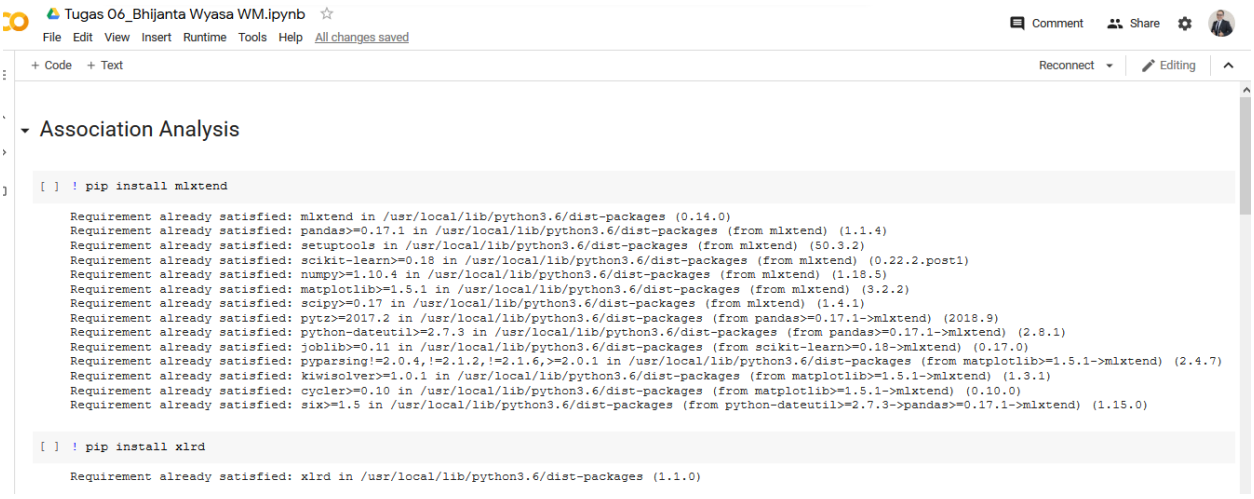
Dari Tugas 6 didapatkan data sebagai berikut :

TID	PENA	ROTI	MENTEGA	TELUR	BUNCIS	SUSU	KECAP
001	1	1	1	0	0	0	0
002	0	1	1	1	0	0	0
003	0	0	0	1	1	1	0
004	0	1	1	0	0	0	0
005	0	1	1	1	0	1	1

Dari data diatas akan saya coba untuk membuat association rule meggunakan bahasa pyhton dengan colab.research.google.com, adapun langkahnya sebagai berikut :

Langkah 1,

Download Libabry Association Rule



Nama : Bhijanta Wyasa WM
NIM : 202420019
Kelas : MTI 23

Langkah 2,

Masukkan data mengenai assosiation rule yang akan diproses (data terlampir diatas),

Tentukan minimum supportnya pada kali ini saya menggunakan 0.6

Tugas 06_Bhijanta Wyasa WM.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] ! pip install xlrd
```

```
Requirement already satisfied: xlrd in /usr/local/lib/python3.6/dist-packages (1.1.0)
```

Association Rules Generation from Frequent Itemsets

Source: https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/

```
[ ] # ! pip install mlxtend
    import pandas as pd
    from mlxtend.preprocessing import TransactionEncoder
    from mlxtend.frequent_patterns import apriori
    from mlxtend.frequent_patterns import association_rules

    dataset = [['Pena', 'Roti', 'Mentega'],
               ['Roti', 'Mentega', 'Telur'],
               ['Telur', 'Buncis', 'Susu'],
               ['Roti', 'Mentega'],
               ['Roti', 'Mentega', 'Telur', 'Susu', 'Kecap']]

    te = TransactionEncoder()
    te_ary = te.fit(dataset).transform(dataset)
    df = pd.DataFrame(te_ary, columns=te.columns_)
    frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)

    frequent_itemsets
```

Langkah 3,

Tentukan nilai rules sesuai yang diinginkan :

```
rules[ (rules['antecedent_len'] >= 2) &
       (rules['confidence'] > 0.75) &
       (rules['lift'] > 1.2) ]
```

Nama : Bhijanta Wyasa WM
NIM : 202420019
Kelas : MTI 23

Langkah 4,

Kemudian RUN All dari seluruh config yang sudah ada, adapun hasilnya sebagai berikut :

	support	itemsets
0	0.8	(Mentega)
1	0.8	(Roti)
2	0.6	(Telur)
3	0.8	(Roti, Mentega)

```
[ ] association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
    rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf

```
[ ] rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
    rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedent_len
0	(Roti)	(Mentega)	0.8	0.8	0.8	1.0	1.25	0.16	inf	1
1	(Mentega)	(Roti)	0.8	0.8	0.8	1.0	1.25	0.16	inf	1

Didaptkan kesimpulan bahwa Assosiation rule sebagaimana diatas untuk hasilny.

Nama : Cornelia tri Wahyuni
Nim : 202420044
Kelas : MTI Reguler A
MK : Advanced Database

Tugas 7

Dari Tugas 06 sebelum ini, coba gunakan data yang sama, tapi gunakan pemprograman python, lalu buat tutorialnya.

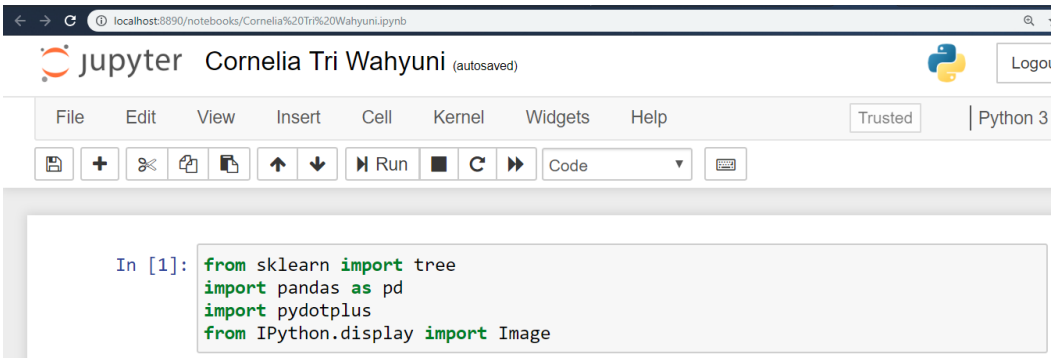
Dataset PSB (Penerimaan Siswa Baru)

A	B	C	D	E
Peserta	**Nilai**	**Ujian Kopetensi**	**Wawancara**	**Diterima**
P1	tinggi	bagus	baik	ya
P2	tinggi	cukup	baik	ya
P3	tinggi	kurang	buruk	tidak
P4	sedang	cukup	baik	ya
P5	sedang	bagus	baik	ya
P6	sedang	cukup	baik	ya
P7	sedang	kurang	buruk	tidak
P8	rendah	bagus	baik	ya
P9	rendah	cukup	buruk	tidak

Tutorial atau tata cara Perhitungan dan Penerapan Algoritma C 4.5 Pada Phyton Dengan Jupyter Notebook Pada Dataset Penerimaan Siswa Baru

1. Pastikan Sudah menginstal **Phyton** dan **Jupyter Notebook**
2. Siapkan Dataset dengan format CSV
3. Pastikan Library pendukung seperti **Scikit-Learn**, **Pydotplus**, dan **Graphviz**, sedangkan untuk **pandas** dan **IPhyton.display** sudah tersedia sendiri pada saat menginstal phyton.
   Link dibawah untuk download :
   Scikit-learn > https://scikit-learn.org/stable/install.html
   Graphviz > https://www2.graphviz.org/Packages/stable/windows/10/cmake/Release/x64/
   Pydotplus > https://pypi.org/project/pydotplus/
4. Untuk **graphviz** silahkan atur path dan masukan ke **system environtment variables** sesuai dengan directory penginstalan pada komputer, misal : **C:\Program Files\Graphviz 2.44.1\bin;**
5. Setelah semua selesai lalu masuk ke halaman **Jupyter Notebook** > pilih **File > New** > **Phyton 3**

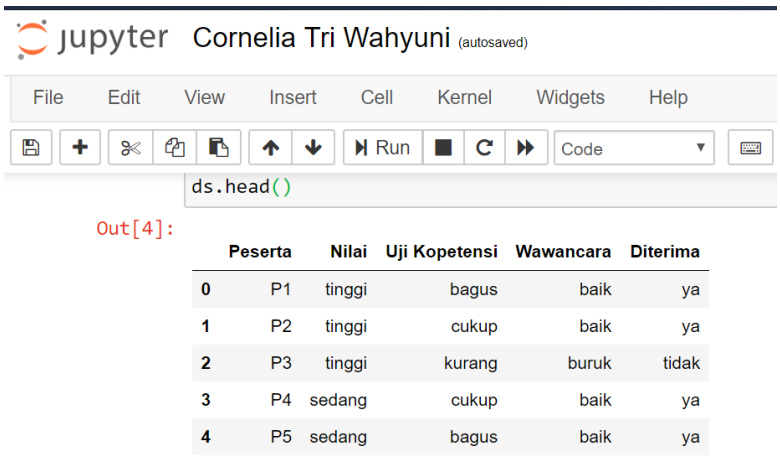
6. Masukan semua variabel library yang telah di instal seperti gambar berikut :



7. Lalu setelah selesai lalu masukan atribut sesuai dataset yang telah dibaut seperti gambar dibawah dan panggil sesuai nama dataset seperti gambar dibawah dengan format csv atau txt.

```
In [4]: atribut = ['Peserta', 'Nilai', 'Uji Kopetensi', 'Wawancara', 'Diterima']
        ds = pd.read_csv('dataset-psb.csv', header=None, names=atribut)
        ds.head()
```

8. Maka data akan muncul seperti pada gambar dibawah ini.



	Peserta	Nilai	Uji Kopetensi	Wawancara	Diterima
0	P1	tinggi	bagus	baik	ya
1	P2	tinggi	cukup	baik	ya
2	P3	tinggi	kurang	buruk	tidak
3	P4	sedang	cukup	baik	ya
4	P5	sedang	bagus	baik	ya

9. Lalu cara yang selanjutnya yaitu mengubah data menjadi data binner dengan cara pada gambar dibawah kecuali atribut **‘wawancara’** karena merupakan atribut target.

```
In [6]: psb = pd.get_dummies(ds[['Peserta', 'Nilai', 'Uji Kopetensi', 'Wawancara']])
        psb.head()
```

10. Dan maka dataset yang kita masukan sudah berubah menjadi data binner yang hanya berupa angka 1 dan angka 0 seperti gambar dibawah :

Out[6]:

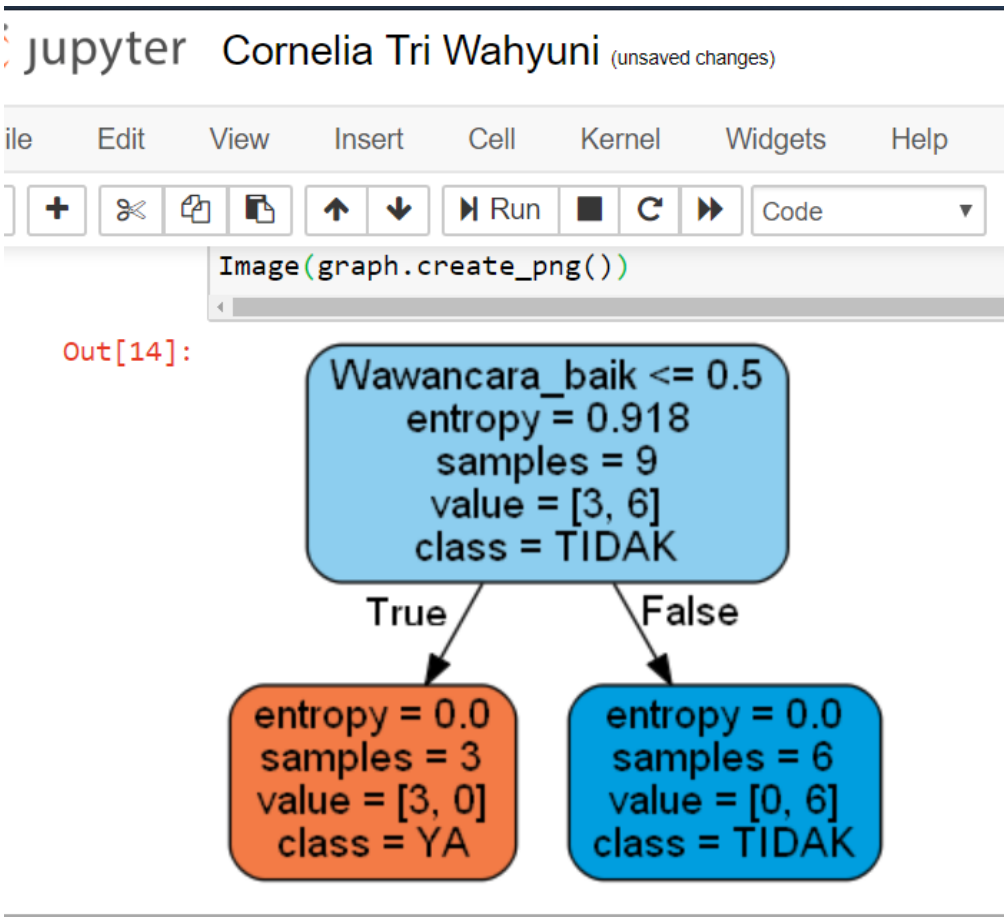
	Peserta_P1	Peserta_P2	Peserta_P3	Peserta_P4	Peserta_P5	Peserta_P6	Peserta_P7	Peserta_P8	Peserta_P9	Nilai_rendah	Nilai_sedang	Nilai_tinggi	Ko
0	1	0	0	0	0	0	0	0	0	0	0	1	
1	0	1	0	0	0	0	0	0	0	0	0	1	
2	0	0	1	0	0	0	0	0	0	0	0	1	
3	0	0	0	1	0	0	0	0	0	0	1	0	
4	0	0	0	0	1	0	0	0	0	0	1	0	

11. Pada Cara terakhir ini yaitu untuk menampilkan decision tree :

In [8]:
```
hasil = tree.DecisionTreeClassifier(criterion='entropy')
hasil_train = hasil.fit(psb, ds['Diterima'])
```

In [1]:
```
dot_data = tree.export_graphviz(hasil_train, out_file=None, feature_names=list(psb.columns.values),
                                class_names=['YA', 'TIDAK'], rounded=True, filled=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

12. Maka Tampilan Decision Tree akan muncul seperti gambar dibawah ini :



Nama : Cynthia Anisa Agatha

NIM : 202420022

```
import pandas as pd #Import Library Pandas
import numpy as np #Import Library Numpy
from sklearn.preprocessing import Imputer #Import Library Preprocesing dan Imputer
df = pd.read_csv("Data.csv") #Impot csv data
print(df['Income'].head(10))
#Data Cleansing NaN
imputer =Imputer(missing_values= 'NaN',strategy='drop',axis=0)
# Function to split the dataset
def splitdataset(balance_data):
    # Separating the target variable
    X = balance_data.values[:, 1:5]
    Y = balance_data.values[:, 0]

    # Splitting the dataset into train and test
    X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size = 0.3, random_state = 100)

    return X, Y, X_train, X_test, y_train, y_test

# Function to perform training with giniIndex.
def train_using_gini(X_train, X_test, y_train):

    # Creating the classifier object
    clf_gini = DecisionTreeClassifier(criterion = "gini",
            random_state = 100,max_depth=3, min_samples_leaf=5)
```

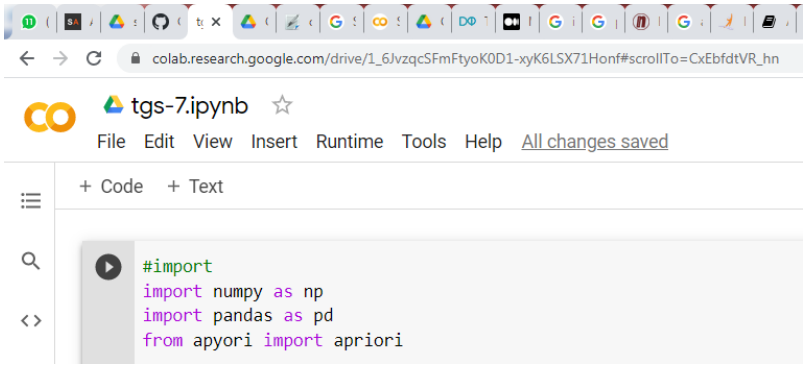
```
    # Performing training
    clf_gini.fit(X_train, y_train)
    return clf_gini

# Function to perform training with entropy.
def tarin_using_entropy(X_train, X_test, y_train):

    # Decision tree with entropy
    clf_entropy = DecisionTreeClassifier(
            criterion = "entropy", random_state = 100,
            max_depth = 3, min_samples_leaf = 5)

    # Performing training
    clf_entropy.fit(X_train, y_train)
    return clf_entropy

# Function to make predictions
def prediction(X_test, clf_object):

    # Predicton on test with giniIndex
    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)
    return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):
```

```
    print("Confusion Matrix: ",
        confusion_matrix(y_test, y_pred))

    print ("Accuracy : ",
    accuracy_score(y_test,y_pred)*100)

    print("Report : ",
    classification_report(y_test, y_pred))

# Driver code
def main():

    # Building Phase
    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = tarin_using_entropy(X_train, X_test, y_train)

    # Operational Phase
    print("Results Using Gini Index:")

    # Prediction using gini
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)

    print("Results Using Entropy:")
    # Prediction using entropy
    y_pred_entropy = prediction(X_test, clf_entropy)
    cal_accuracy(y_test, y_pred_entropy)
```

```
# Calling main function
if __name__=="__main__":
    main()
```

Nama : Efrik Kartono Ahsa
MK : Advanced Database
NIM : 202420030

Step 1: Import the libraries



Catatan penulis : pada Collabs modul apyori harus diinstall dulu menggunakan perintah

```
!pip install apyori
```

Step 2: Load the dataset

```
#loading data sheet
#import datasheet
from google.colab import drive
drive.mount('/content/drive')
store_data = pd.read_csv('drive/My Drive/Colab Notebooks/data-sheet-tugas-7-ok.csv', header=None)
```

Step 3: Have a glance at the records

```
#having glance of record
store_data

#look shape
store_data.shape
```

Output

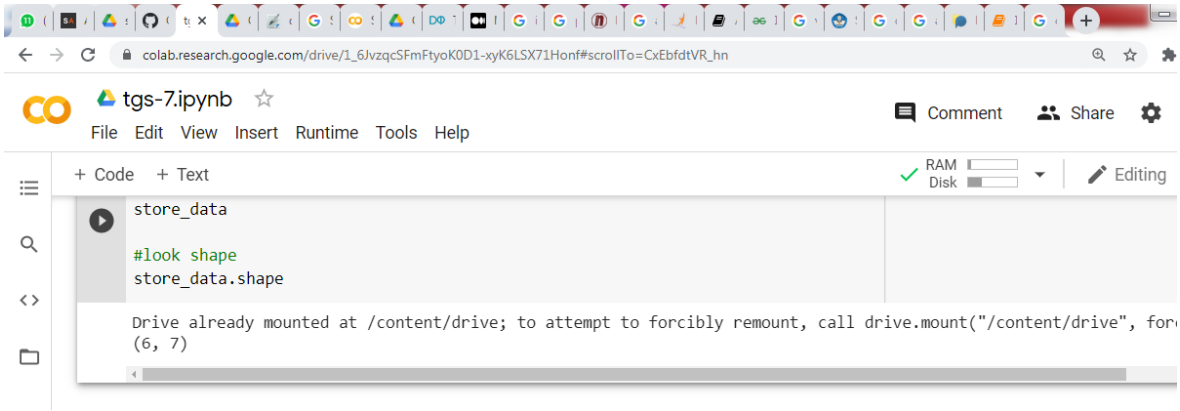
```
store_data
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive",
```

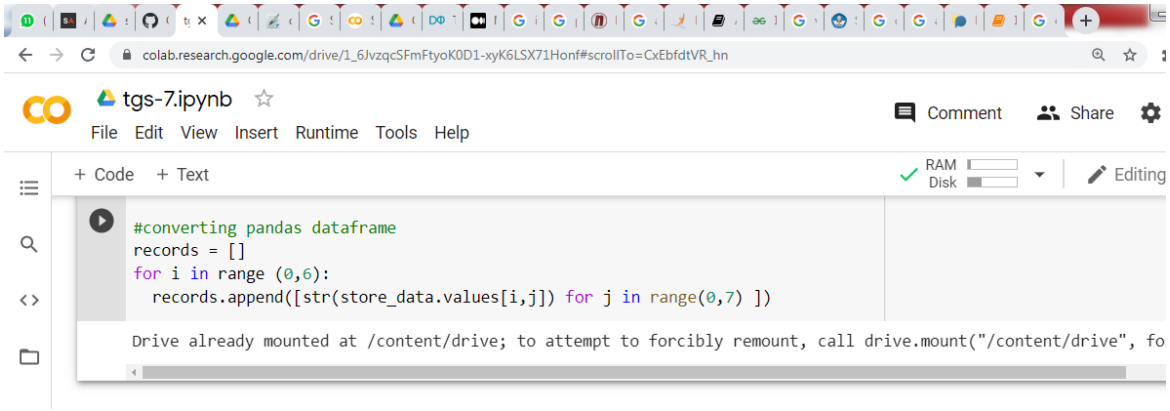
	0	1	2	3	4	5	6
0	PENA	ROTI	MENTEGA	TELUR	BUNCIS	SUSU	KECAP
1	PENA	ROTI	MENTEGA	NaN	NaN	NaN	NaN
2	NaN	ROTI	MENTEGA	TELUR	NaN	NaN	NaN
3	NaN	NaN	NaN	TELUR	BUNCIS	SUSU	NaN
4	NaN	ROTI	MENTEGA	TELUR	NaN	NaN	NaN
5	NaN	ROTI	MENTEGA	TELUR	NaN	SUSU	KECAP

Nama : Efrik Kartono Ahsa
MK : Advanced Database
NIM : 202420030

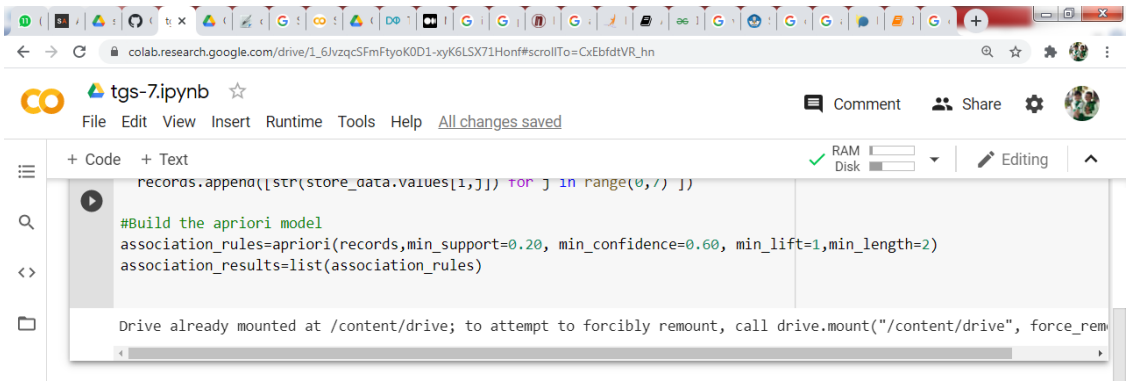
Step 4: Look at the shape



Step 5: Convert Pandas DataFrame into a list of lists

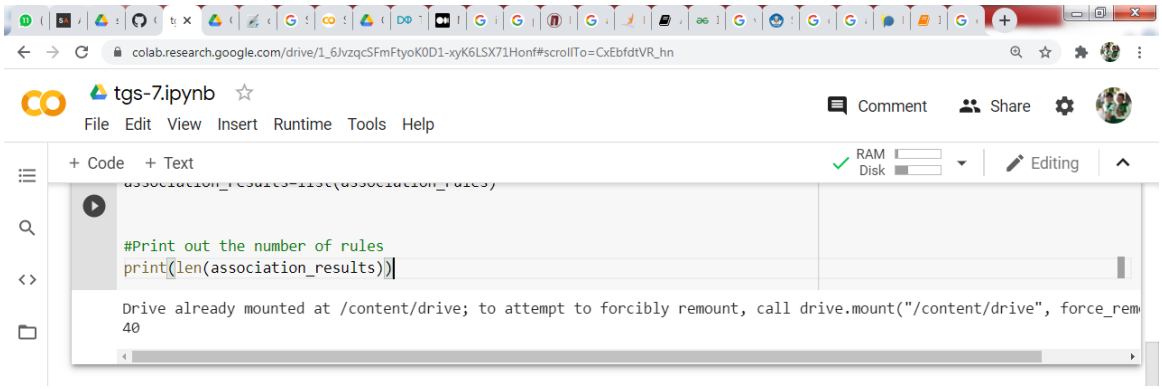


Step 6: Build the Apriori model

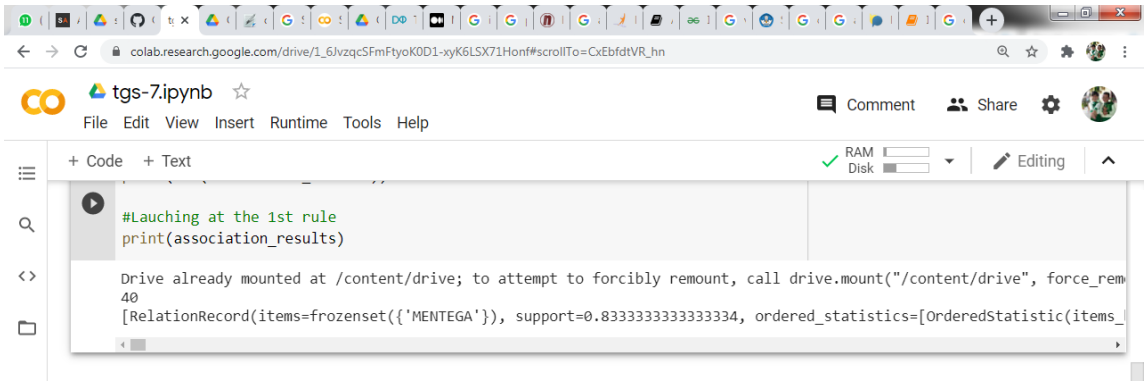


Nama : Efrik Kartono Ahsa
MK : Advanced Database
NIM : 202420030

Step 7: Print out the number of rules



Step 8: Have a glance at the rule



Nama : Efrik Kartono Ahsa
MK : Advanced Database
NIM : 202420030

Script Phyton Lengkap

```
#import
import numpy as np
import pandas as pd
from apyori import apriori

#loading data sheet
#import datasheet
from google.colab import drive
drive.mount('/content/drive')
store_data = pd.read_csv('drive/My Drive/Colab Notebooks/data-sheet-
tugas-7-ok.csv', header=None)

#having glance of record
store_data

#look shape
store_data.shape

#converting pandas dataframe
records = []
for i in range (0,6):
  records.append([str(store_data.values[i,j]) for j in range(0,7) ])

#Build the apriori model
association_rules=apriori(records,min_support=0.20, min_confidence=0.60
, min_lift=1,min_length=2)
association_results=list(association_rules)


#Print out the number of rules
print(len(association_results))

#Lauching at the 1st rule
print(association_results)
```

TUGAS 07
ADVANCED DATABASE

Dari Tugas 06 sebelum ini, coba gunakan data yang sama, tapi gunakan pemrograman python, lalu buat tutorialnya.

Langkah tutorial :

Langkah Awal Yang Disiapkan

A. Python

Package yang harus diinstal :

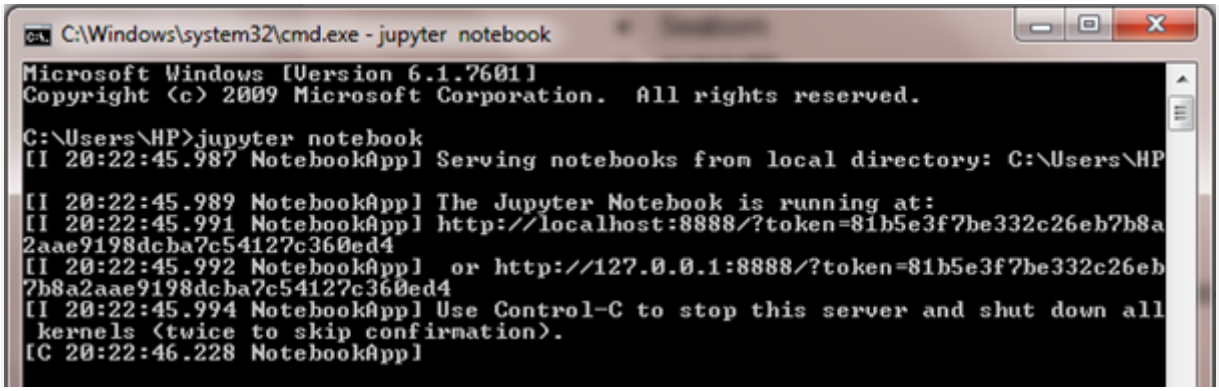
- Pandas
- Numpy
- Sklearn
- Seaborn
- matplotlib

B. Jupyter notebook

C. Dataset

Langkah Kerja

1. Install semua aplikasi Python dan jupyter notebook, lalu run jupyter notebook via command prompt.



Gambar 1.1 Cara menjalankan jupyter notebook

2. Saya akan menggunakan "Prediksi-Kelulusan" sebagai dataset yang digunakan pada kali ini. Dataset terdiri dari 12 sampel dari Prediksi kelulusan apakah akan :

- Tepat waktu
- Terlambat

S1	S2	S3	S4	S5	Keterangan
3.71	3.89	3.89	3.89	3.99	Tepat Waktu
3.67	3.81	3.87	3.87	3.78	Tepat Waktu
3.66	3.78	3.87	3.78	3.82	Tepat Waktu
3.59	3.78	3.81	3.78	3.82	Tepat Waktu
3.54	3.77	3.81	3.77	3.66	Tepat Waktu
3.49	3.75	3.79	3.71	3.81	Tepat Waktu
1.78	1.99	2.22	2.31	2.12	Terlambat
1.67	1.99	2.35	2.55	2.11	Terlambat
1.44	1.56	2.23	2.55	2.18	Terlambat
1.78	1.99	2.12	2.11	2.22	Terlambat
1.78	1.89	2.11	2.43	2.32	Terlambat
1.76	1.84	1.99	2.32	2.43	Terlambat

Dataset yang digunakan(Prediksi-Kelulusan.csv)

Dua fitur diukur dari sampel meliputi nilai IP pada : Semester 1, Semester 2, Semester 3, Semester 4, dan Semester 5.

```
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
import math
```

Gambar 1.2 Import package

```
from pandas.plotting import scatter_matrix
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
```

Gambar 1.3 Sklearn module

```
data = pd.read_csv("Prediksi-Kelulusan.csv")
data.head()
```

	S1	S2	S3	S4	S5	Keterangan
0	3.71	3.89	3.89	3.89	3.99	Tepat Waktu
1	3.67	3.81	3.87	3.87	3.78	Tepat Waktu
2	3.66	3.78	3.87	3.78	3.82	Tepat Waktu
3	3.59	3.78	3.81	3.78	3.82	Tepat Waktu
4	3.54	3.77	3.81	3.77	3.66	Tepat Waktu

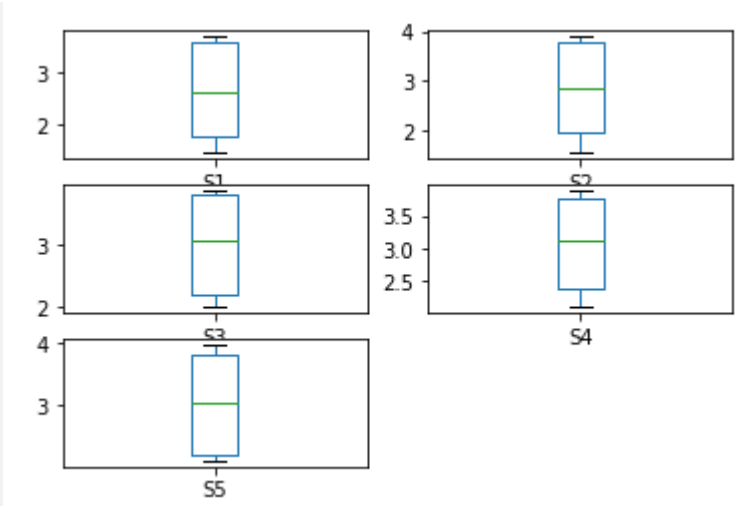
Gambar 1.4 Import dataset dan Output yang dihasilkan

3. Selanjutnya saya akan membuat suatu boxplot dengan syntax di bawah ini

```
#box and whisker plots
data.plot(kind='box', subplots=True, layout=(3,2), sharex=False, sharey=False)
plt.show()
```

Gambar 1.5 Syntax boxplot

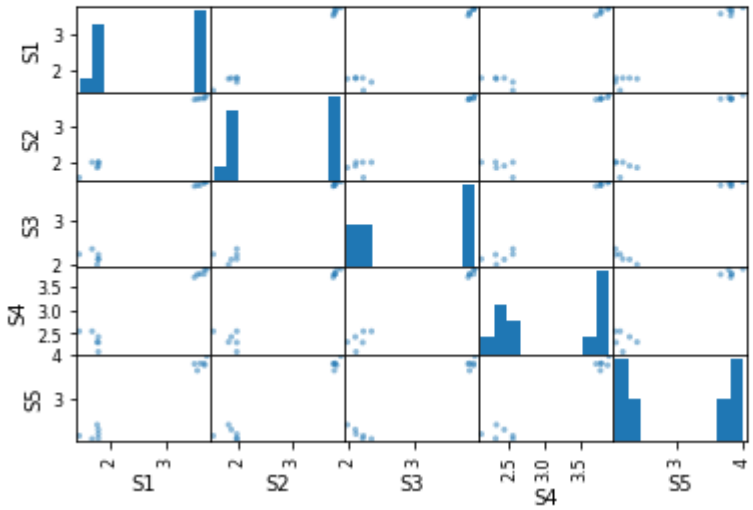
Dan otputnya akan seperti dibawah ini :



Gambar 1.6 Output dari boxplot

4. Untuk menampilkan scatter plot dari semua variable gunakan perintah dibawah ini.

```
#scatter plot matrix
scatter_matrix(data)
plt.show()
```



Gambar 4.1 Syntax Scatter Plot dan Output dari Scatter Plot

5. Sebelum masuk ke analisis KNN, import sklearn pada program, kemudian ketikkan script program seperti dibawah. Script dibagi menjadi 2 bagian data 80% untuk data training, dan 20% untuk data test, sehingga pada test_size=0.2

```
x = data.iloc[:, :-5].values
y = data.iloc[:, 5].values

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=0.2)
```

6. Setelah data di-split menjadi 80% training set dan 20% test set, selanjutnya masuk ke analisis KNN, ketikan script seperti dibawah. Goals dari penggunaan metode analisis ini adalah menemukan faktor-k yang sesuai, dilihat dari accuracy yang dihasilkan. Semakin tinggi akurasi, maka factor-k tersebut yang akan dijadikan referensi untuk semua data set.

```
#Import knearest neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier

#Create KNN classifier
knn = KNeighborsClassifier(n_neighbors=8)

#Train the model using the training sets
knn.fit(x_train, y_train)

#Predict the response for test dataset
y_pred = knn.predict(x_test)
```

```
#Import scikit-Learn metrics module for accurancy
from sklearn import metrics

#Model Accuracy, how often is the classifier carrect?
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.3333333333333333
```

Gambar 6.1 menghitung akurasi dari k=8

```
#Import knearest neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier

#Create KNN classifier
knn = KNeighborsClassifier(n_neighbors=3)

#Train the model using the training sets
knn.fit(x_train, y_train)

#Predict the response for test dataset
y_pred = knn.predict(x_test)
```

```
#Import scikit-Learn metrics module for accurancy
from sklearn import metrics

#Model Accuracy, how often is the classifier carrect?
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 1.0
```

Gambar 6.2 menghitung akurasi dari k=3

SELESAI

NIM : 202420040 ISTIARSO

Pengolahan data dengan Python dan menggunakan dataset yang sama yg di gunakan di RapidMiner.

Penjelasan :

Dataset terdiri dari 520 records dan 17 attribute/Feature/column, attribute ke-17 di jadikan label/class yg akan di analisa menggunakan Teknik data mining.

Sebagian besar datanya adalah nominal yg mempunyai dua nilai yang di sebut binominal, contoh : male/female, Yes/No, Positive/Negative.

Mengingat data label/class berupa data nominal, maka pilihan modeling data miningnya adalah classification, dalam hal ini di pilih algoritma Decision Tree

```
# Load Libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
```

```
col_names = ['Age','Gender','Polyuria','Polydispia','Sudden Weight loss','Weakness',
             'Polyphagia','Genital Thrush','Visual blurring','Itching','Irritability',
             'Delayed Healing','Partial Paresis','Muscle stiffness','Alopecia','Obesity','Label']
# load dataset
diag = pd.read_csv("diabetes_data1.csv")
```

Penyebutan kolom name diperlukan jika loading dataset tidak mempunyai heade atau menggunakan opsi set **header** = None, namun dalam case ini, data set csv sudah memiliki header

```
diag.head()
```

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity	
0	40	1	0	1	0	1	0	0	0	1	0	1	0	1	1	1	Po
1	58	1	0	0	0	1	0	0	1	0	0	0	1	0	1	0	Po
2	41	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0	Po
3	45	1	0	0	1	1	1	1	0	1	0	1	0	0	0	0	Po
4	60	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	Po

Catatan Output : Gender 1 = Male , 0 = Female , Other attribute 1 = Yes, 0 = No

```
#split dataset in features and target variable
feature_cols = ['Age','Polyuria','Polydispia','Sudden Weight loss','Weakness',
                'Polyphagia','Genital Thrush','Visual blurring','Itching',
                'Irritability','Delayed Healing','Partial Paresis','Muscle stiffness','Alopecia','Obesity','Gender']
#X = diag[feature_cols] # Features
#y = diag.Label # Target variable
X=diag.iloc[:, 0:16].values
y=diag.iloc[:, 16].values
```

NIM : 202420040 ISTIARSO

Note : Ada dua cara pemilihan attribute, menggunakan feature_cols dan label (di comment), atau mengunakan index.

Variable feature_cols akan di gunakan nanti saat akan menggambar graphic -nya

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30% test
```

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
clf_tree = DecisionTreeClassifier(criterion='Entropy', random_state=1)
```

```
 # Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

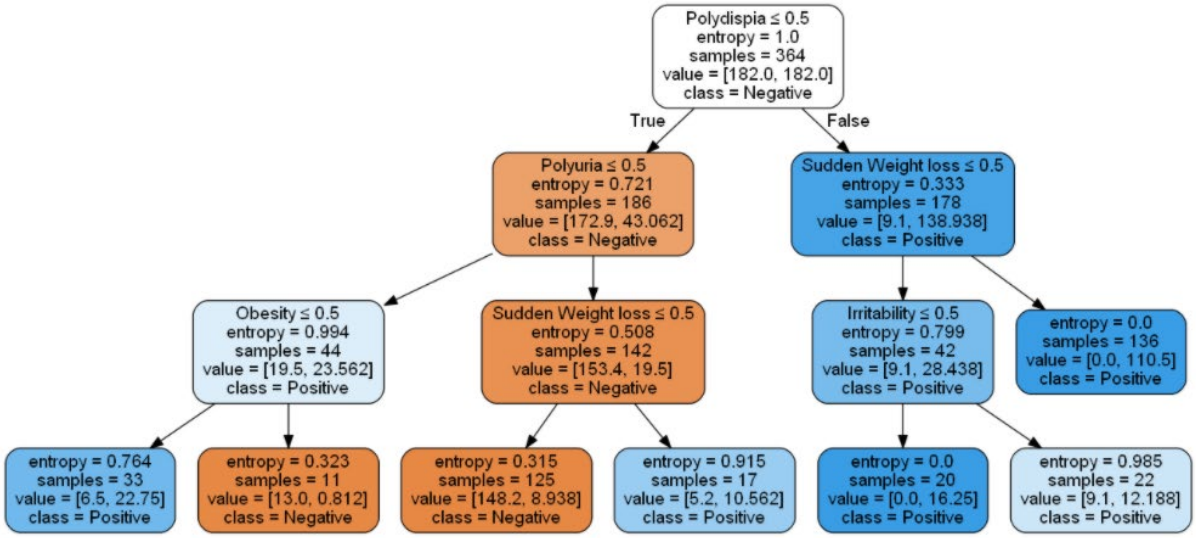
```
Accuracy: 0.9423076923076923
```

```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

clf = DecisionTreeClassifier(criterion = 'entropy', max_depth = 3, min_samples_split = 20, class_weight = "balanced")
clftree = clf.fit(X_train,y_train)


dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,
                feature_names = feature_cols,
                class_names=['Negative','Positive'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Mengunakan module graphviz untuk menampilkan hasil analisanya(dibawah)

NIM : 202420040 ISTIARSO

Output :



Catatan:
Age : Umur
Gender : Jenis Kelamin
Polyuria : sering buang air kecil
Polydispia : sering merasa haus
Sudden Weight loss : Berat Badan turun tiba-tiba
Weakness : Lemah
Polyphagia : banyak makan (sering lapar)
Genital Thrush : gatal di kemaluan
Visual blurring : Penglihatan Buram
Itching : Gatal
Irritability : gampang marah
Delayed Healing : Sulit sembuh (luka)
Partial Paresis : lumpuh sebagian
Muscle stiffness : otot kaku
Alopecia : kebotakan, rambut rontok
Obesity : Kegemukan
Label > indikasi Diabetes

Tutorial Algoritma K-Nearest Neighbours Dengan Python

K-Nearest Neighbours (KNN)

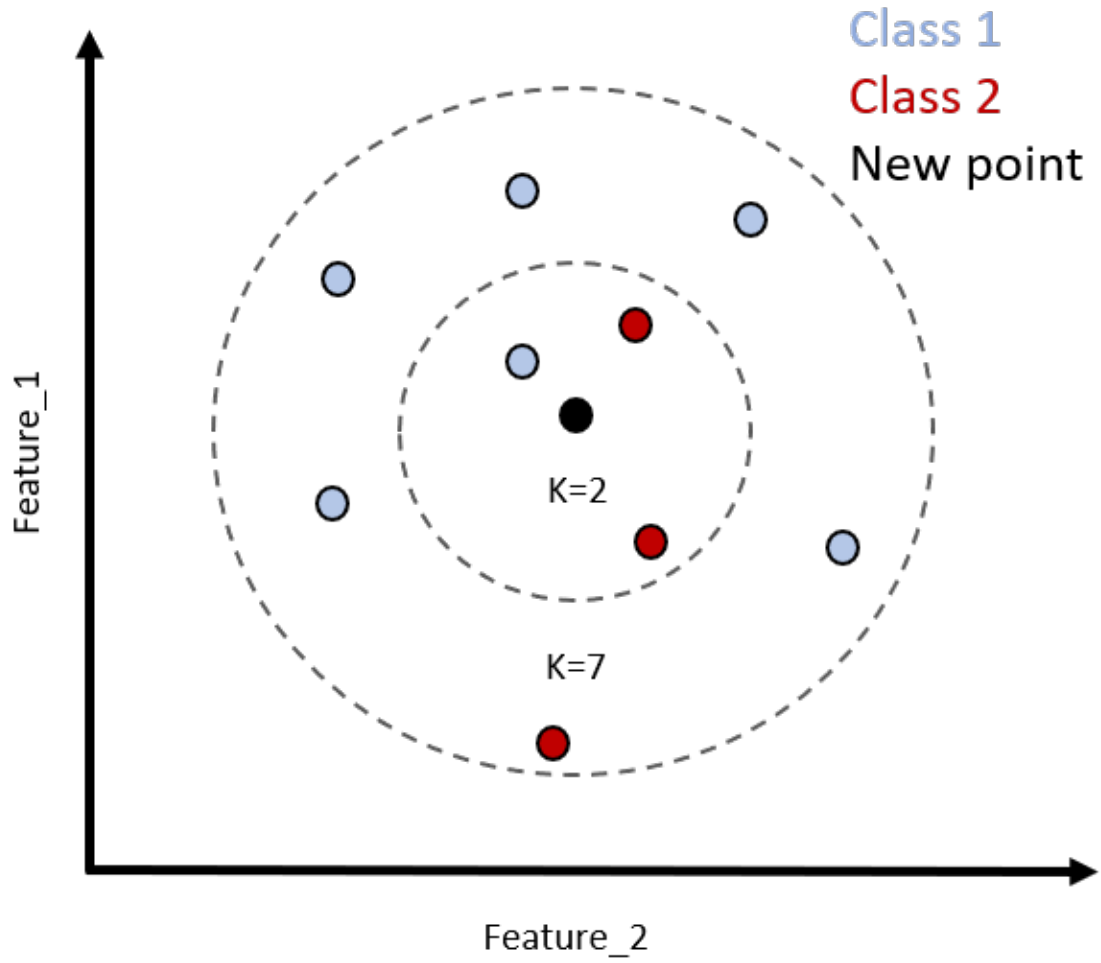
KNN adalah algoritma pembelajaran mesin yang diawasi yang dapat digunakan untuk menyelesaikan masalah klasifikasi dan regresi. Prinsip KNN adalah nilai atau kelas suatu titik data yang ditentukan oleh titik data di sekitar nilai tersebut.

Untuk memahami algoritma klasifikasi KNN seringkali paling baik ditunjukkan melalui contoh. Tutorial ini akan mendemonstrasikan bagaimana Anda dapat menggunakan KNN dengan Python dengan masalah klasifikasi Anda sendiri. Notebook Jupyter yang sesuai dengan contoh ini dapat ditemukan di sini, jika Anda ingin mengikutinya.

Algoritme prediksi menghitung jarak dari titik x yang tidak diketahui, ke semua titik dalam data Anda. Titik-titik dalam data Anda kemudian diurutkan dengan menambah jarak dari x. Prediksi dilakukan dengan memprediksi label mayoritas dari titik terdekat 'K'.

Memilih K akan memengaruhi kelas tempat poin baru akan ditetapkan.

Dalam contoh di bawah ini, memilih nilai K 2 akan menetapkan titik yang tidak diketahui (lingkaran hitam) ke kelas 2. Namun, jika nilai K adalah 7, titik yang tidak diketahui akan ditetapkan ke kelas 1.



Membuat dataset palsu

Pertama, kita mengimpor pustaka yang kita butuhkan, dan kemudian membuat dataset palsu menggunakan fungsi makeblobs dari sklearn. Kita dapat mengirimkan jumlah sampel, fitur dalam kumpulan data kita, berapa banyak pusat atau kelas yang akan memasukkan data, dan terakhir deviasi standar dari cluster tersebut. Untuk konsistensi antara beberapa jalannya notebook Jupyter ini, saya telah menetapkan integer 101 ke parameter random_state.
Catatan, untuk memulai, kita akan memiliki standar deviasi cluster yang besar. Ini akan memperkenalkan varians ke dalam klasifikasi, yang dapat kita perbaiki nanti, dengan secara khusus memilih nilai K yang optimal. Ini dapat dicapai dengan menggunakan metode siku.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.datasets import make_blobs
```

```
data = make_blobs(n_samples=300, n_features=5, centers=2, cluster_std=6.0, random_state=101)
```

```
data
```

```
(array([[ -0.95757537,   3.36332609, -15.54675979, -14.02967497,
           1.50545246],
        [-11.12008037,  -0.86726927, -19.42687054, -22.99153445,
          12.8409123 ],
        [  5.02786886,  -2.84037069,  -5.9094317 , -16.29765383,
           7.77075032],
        ...,
        [ -8.02114181,   2.29827056, -13.80731349, -10.89022536,
           1.99399904],
        [ 10.87670302,   3.25562702,  -6.25095388,  -0.92884525,
           8.18286695],
        [  7.86530195, -11.18764669,   6.36417619,  -2.87676038,
           1.31626729]]),
 array([0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
        1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
        1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
        0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0,
        1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
        0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
```

Fungsi makeblobs dari sklearn mengembalikan tupel 2 elemen. Kita dapat membuat kerangka data fitur kita menggunakan pd.DataFrame, dan meneruskan indeks tupel pertama yang sesuai dengan data fitur. Elemen kedua dari tupel data sesuai dengan label fitur.

```
df_feat = pd.DataFrame(data[0], columns=['feature_' + str(i) for i in range(1, 6)])
```

```
df_feat.head(2)
```

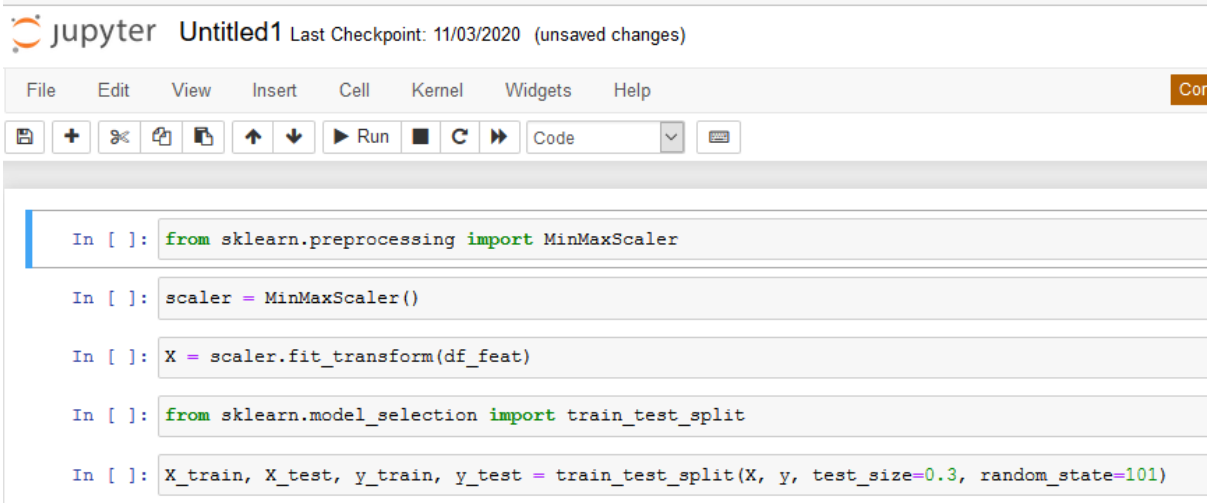
	feature_1	feature_2	feature_3	feature_4	feature_5
0	-0.957575	3.363326	-15.546760	-14.029675	1.505452
1	-11.120080	-0.867269	-19.426871	-22.991534	12.840912

```
y = data[1]
```

```
y
```

```
array([0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1,
```

Sekarang kita dapat menskalakan data dengan mengimpor MinMaxScaler dari Sklearn.preprocessing. Tidak seperti algoritme pembelajaran mesin lainnya, kami menyesuaikan dan mengubah semua data pelatihan, sebelum melakukan pemisahan pengujian kereta kami.



Algoritme prediksi dan pengoptimalan

Untuk mengimplementasikan prediksi dalam kode, kita mulai dengan mengimpor KNeighboursClassifier dari sklearn.neighbours. Kami kemudian membuat instance dari KNeighboursClassifier, dengan meneruskan argumen 1 ke n_neighbours, dan menetapkan ini ke variabel knn. Nilai yang diteruskan ke n_neighbours mewakili nilai K.

Kami kemudian menyesuaikan dengan data pelatihan, sebelum membuat prediksi, dengan memanggil metode prediksi pada objek KNeighboursClassifier kami.

Sekarang kita dapat menilai keakuratan prediksi menggunakan klasifikasi_report dan konfusi_matriks.

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier(n_neighbors=1)
```

```
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

```
predictions = knn.predict(X_test)
```

```
predictions
```

```
array([0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 1])
```

Metrik tersebut menunjukkan bahwa akurasinya sudah sangat baik. Ini mungkin karena fakta bahwa kami membuat kumpulan data dengan makeblobs dan secara khusus meminta 2 pusat. Namun, kami sengaja menempatkan nilai yang besar untuk standar deviasi cluster untuk memperkenalkan varians. Hal ini mengakibatkan kesalahan klasifikasi pada 4 poin dalam kumpulan data kami.

```
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
[[42  1]
 [ 3 44]]
              precision    recall  f1-score   support

           0       0.93      0.98      0.95        43
           1       0.98      0.94      0.96        47

    accuracy                           0.96        90
   macro avg       0.96      0.96      0.96        90
weighted avg       0.96      0.96      0.96        90
```

Meningkatkan Akurasi Data

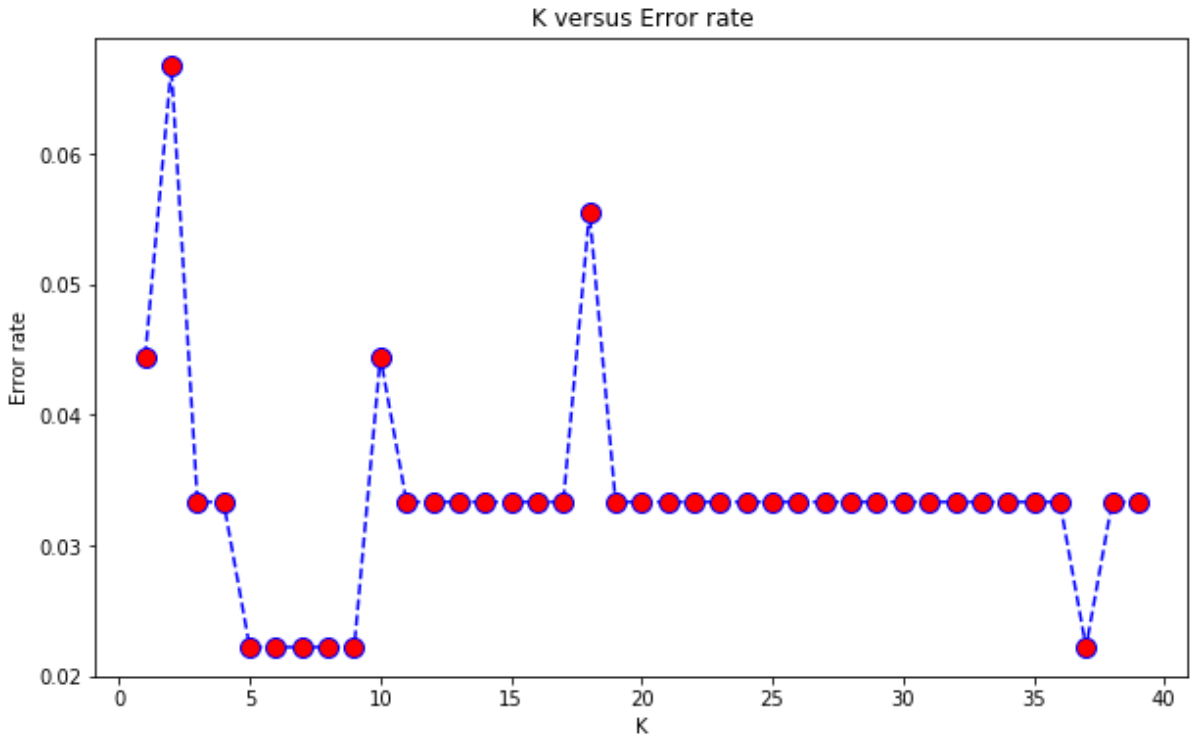
Kami dapat mencoba untuk meningkatkan akurasi hasil kami dengan memodifikasi jumlah tetangga. Ini dapat dicapai dengan menggunakan metode siku.

Pertama-tama kita melakukan iterasi melalui 40 nilai tetangga, membuat instance objek KNeighboursClassifier dengan jumlah tetangga tersebut. Kami kemudian dapat

menyesuaikan data pelatihan dengan model KNN ini, mendapatkan prediksi, dan menambahkan nilai rata-rata antara prediksi, pred_i dan nilai yang benar, y_test.
Jika pred_i dan y_test tidak cocok dalam array, nilai true dikembalikan yang memiliki nilai 1. Semakin tinggi angkanya, semakin tidak akurat klasifikasi tersebut.
Nilai yang lebih rendah untuk tingkat kesalahan akan sesuai dengan model yang berkinerja lebih baik.
Hasil ini dapat diplot menggunakan rentang nilai i pada sumbu x, versus tingkat kesalahan pada sumbu y.

```
error_rate = []

for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)

    error_rate.append(np.mean(pred_i != y_test))


plt.figure(figsize=(10, 6))

plt.plot(range(1, 40), error_rate, color='blue', linestyle='--',
         markersize=10, markerfacecolor='red', marker='o')
plt.title('K versus Error rate')
plt.xlabel('K')
plt.ylabel('Error rate')
```

Sekarang kita dapat memilih nilai K terendah yang akan menghasilkan, tingkat kesalahan terendah. Di sini, kita bisa memilih 5.



Sekarang kita dapat, menjalankan kembali penilaian akurasi dengan matriks kebingungan dan laporan klasifikasi sekali lagi, untuk melihat apakah kita mengklasifikasikan 4 poin yang tidak selaras dengan lebih akurat. Kami telah meningkatkan, dari 4 poin yang salah diklasifikasikan menjadi 2.

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
predictions = knn.predict(X_test)


print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
[[42  1]
 [ 1 46]]
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        43
           1       0.98      0.98      0.98        47

    accuracy                           0.98        90
   macro avg       0.98      0.98      0.98        90
weighted avg       0.98      0.98      0.98        90
```

Pelatihan tentang titik data baru

Sekarang kita dapat membuat titik data menggunakan data asli. Pertama kita membuat dua dataframe; satu dengan fitur dan satu lagi dengan label, menggabungkannya menjadi satu kerangka data, dan memilih baris pertama, sebagai titik data untuk memprediksi labelnya. Kita harus ingat untuk menskalakan titik data karena model dilatih pada data yang diskalakan.

Prediksi menunjukkan bahwa data titik 1, akan memberikan label 0, yang sesuai dengan titik dataset asli, diverifikasi dengan memanggil df.head (1).

```
features = pd.DataFrame(data=data[0], columns=['feature_' + str(i) for i in range(1, 6)])
lables = pd.DataFrame(data[1], columns=['labels'])
dataset = pd.concat([features, lables], axis=1)
data_point_1 = scaler.transform(np.array(dataset.iloc[0][:-1]).reshape(-1, 5))
knn.predict(data_point_1)[0]

# Output
# 0
```

	feature_1	feature_2	feature_3	feature_4	feature_5	labels
0	-0.957575	3.363326	-15.54676	-14.029675	1.505452	0

```
data_point_1 = scaler.transform(np.array(dataset.iloc[0][:-1]).reshape(-1, 5))
```

```
data_point_1
```

```
array([[0.3943128 , 0.58849094, 0.1949979 , 0.21287012, 0.54980842]])
```

```
knn.predict(data_point_1)[0]
```

```
0
```