PRAKTIKUM DATA MINING

Nama : Ahmad Izza Zain Firdaus

NIM : 19051214063

Kelas/Angkatan : SIB/2019

Algoritma : Naïve Bayes

Jenis Analisis : Association

Dataset : academic.csv

Keterangan Dataset : dataset berisikan data pelajar-mahasiswa meliputi dari informasi diri hingga perilaku dalam pembelajaran dan dinilai terhadap keaktifan pelajar

Metode Preprocessing : metode preprocessing dibagi menjadi beberapa tahapan setelah melakukan import data dan memanggil library yang dibutuhkan, dilakukan pengecekan isian dari masing-masing kolom untuk dianalisa

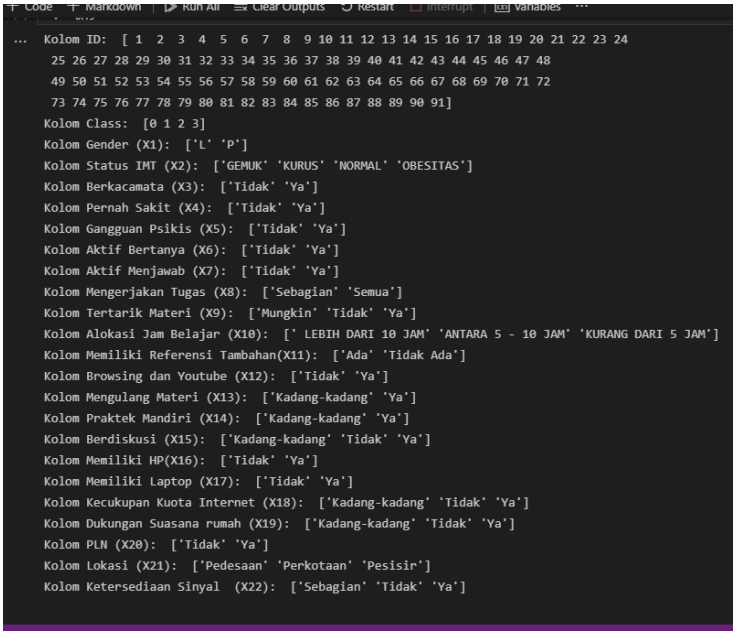
1. Dilakukan Import data untuk digunakan

```
[2]
import pandas as pd
import numpy as np
#memanggil data yang dibutuhkan
df=pd.read_csv('academic.csv')
```

2. Mengecek jenis isian dari masing-masing kolom karena masih dalam bentuk data kategorik

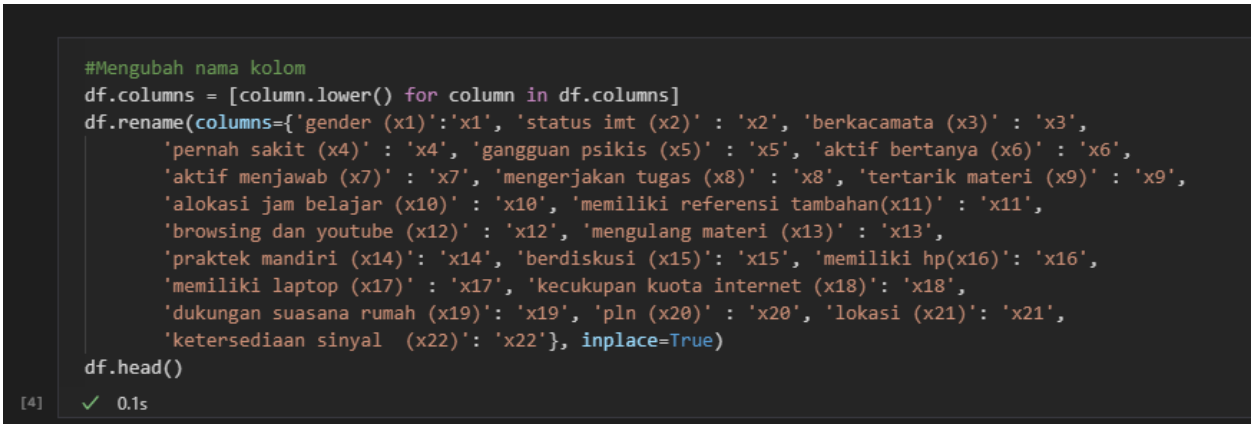
```
[3]
for column in df.columns:
    print(f"Kolom {column}: ", np.sort(df[column].unique()))
✓ 0.1s
... Kolom ID:  [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

Didapati hasil isian kolom sebagai berikut:



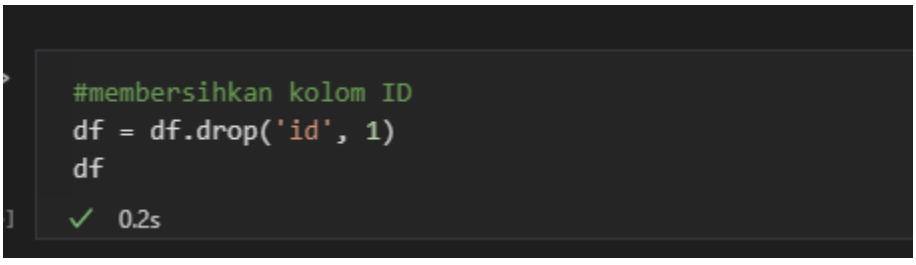
```
Kolom ID:  [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91]
Kolom Class:  [0 1 2 3]
Kolom Gender (X1):  ['L' 'P']
Kolom Status IMT (X2):  ['GEMUK' 'KURUS' 'NORMAL' 'OBESITAS']
Kolom Berkacamata (X3):  ['Tidak' 'Ya']
Kolom Pernah Sakit (X4):  ['Tidak' 'Ya']
Kolom Gangguan Psikis (X5):  ['Tidak' 'Ya']
Kolom Aktif Bertanya (X6):  ['Tidak' 'Ya']
Kolom Aktif Menjawab (X7):  ['Tidak' 'Ya']
Kolom Mengerjakan Tugas (X8):  ['Sebagian' 'Semua']
Kolom Tertarik Materi (X9):  ['Mungkin' 'Tidak' 'Ya']
Kolom Alokasi Jam Belajar (X10):  [' LEBIH DARI 10 JAM' 'ANTARA 5 - 10 JAM' 'KURANG DARI 5 JAM']
Kolom Memiliki Referensi Tambahan(X11):  ['Ada' 'Tidak Ada']
Kolom Browsing dan Youtube (X12):  ['Tidak' 'Ya']
Kolom Mengulang Materi (X13):  ['Kadang-kadang' 'Ya']
Kolom Praktek Mandiri (X14):  ['Kadang-kadang' 'Ya']
Kolom Berdiskusi (X15):  ['Kadang-kadang' 'Tidak' 'Ya']
Kolom Memiliki HP(X16):  ['Tidak' 'Ya']
Kolom Memiliki Laptop (X17):  ['Tidak' 'Ya']
Kolom Kecukupan Kuota Internet (X18):  ['Kadang-kadang' 'Tidak' 'Ya']
Kolom Dukungan Suasana rumah (X19):  ['Kadang-kadang' 'Tidak' 'Ya']
Kolom PLN (X20):  ['Tidak' 'Ya']
Kolom Lokasi (X21):  ['Pedesaan' 'Perkotaan' 'Pesisir']
Kolom Ketersediaan Sinyal  (X22):  ['Sebagian' 'Tidak' 'Ya']
```

3. Setelah mendapatkan isian dari masing-masing kolom, karena penamaan kolom terbilang rumit, maka dilakukan pengubahan nama kolom untuk memudahkan proses berikutnya



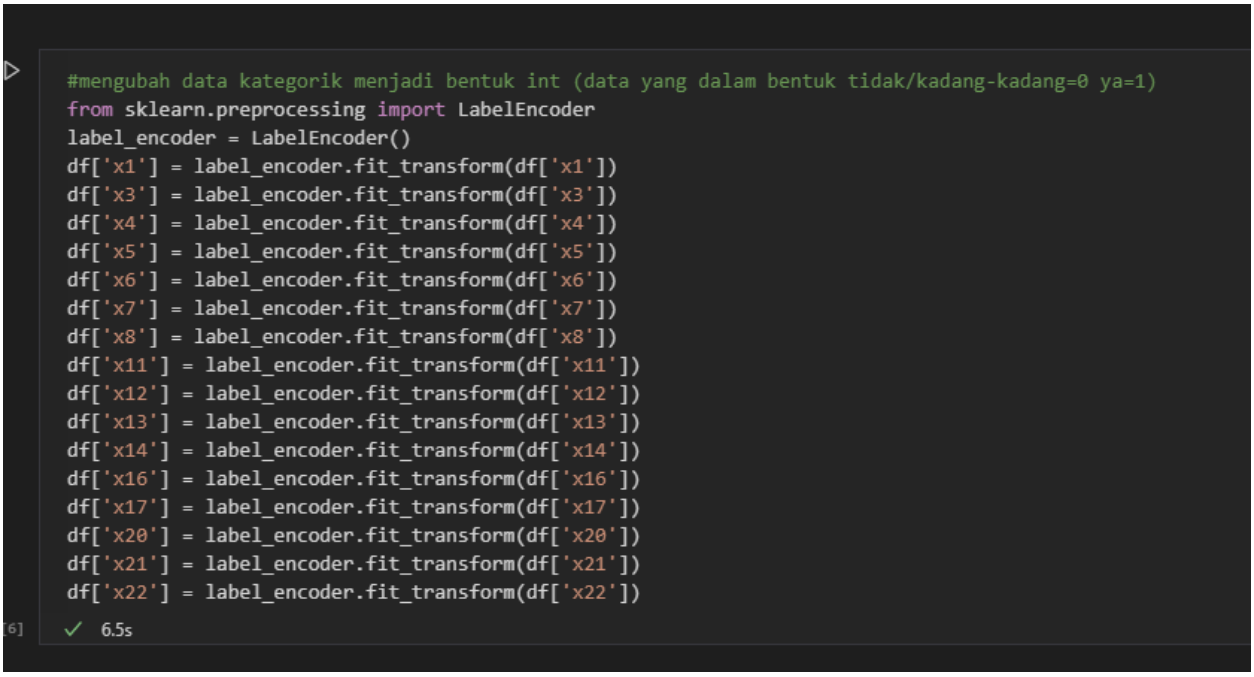
```
#Mengubah nama kolom
df.columns = [column.lower() for column in df.columns]
df.rename(columns={'gender (x1)':'x1', 'status imt (x2)' : 'x2', 'berkacamata (x3)' : 'x3',
       'pernah sakit (x4)' : 'x4', 'gangguan psikis (x5)' : 'x5', 'aktif bertanya (x6)' : 'x6',
       'aktif menjawab (x7)' : 'x7', 'mengerjakan tugas (x8)' : 'x8', 'tertarik materi (x9)' : 'x9',
       'alokasi jam belajar (x10)' : 'x10', 'memiliki referensi tambahan(x11)' : 'x11',
       'browsing dan youtube (x12)' : 'x12', 'mengulang materi (x13)' : 'x13',
       'praktek mandiri (x14)': 'x14', 'berdiskusi (x15)': 'x15', 'memiliki hp(x16)': 'x16',
       'memiliki laptop (x17)' : 'x17', 'kecukupan kuota internet (x18)': 'x18',
       'dukungan suasana rumah (x19)': 'x19', 'pln (x20)' : 'x20', 'lokasi (x21)': 'x21',
       'ketersediaan sinyal  (x22)': 'x22'}, inplace=True)
df.head()
[4]  ✓ 0.1s
```

4. Berdasarkan point nomor 2, kolom id tidak diperlukan dalam proses sebagai variabel independen maupun dependen, oleh karena itu bisa dilakukan pengeluaran variabel



```
#membersihkan kolom ID
df = df.drop('id', 1)
df
]  ✓ 0.2s
```

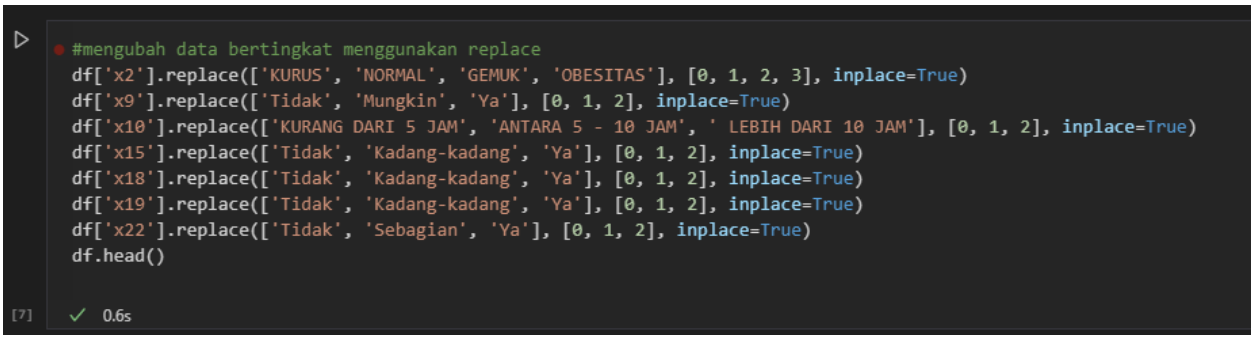
5. Dilakukan pengubahan tipe data dari masing masing kolom, dimulai dari data kategorik yang bersifat nominal, diubah dengan bantuan fungsi preprocessing di sklearn yakni LabelEncoder(), semua data termasuk data nominal selain x2, x9, x10, x15, x18, x19, x22



```
#mengubah data kategorik menjadi bentuk int (data yang dalam bentuk tidak/kadang-kadang=0 ya=1)
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['x1'] = label_encoder.fit_transform(df['x1'])
df['x3'] = label_encoder.fit_transform(df['x3'])
df['x4'] = label_encoder.fit_transform(df['x4'])
df['x5'] = label_encoder.fit_transform(df['x5'])
df['x6'] = label_encoder.fit_transform(df['x6'])
df['x7'] = label_encoder.fit_transform(df['x7'])
df['x8'] = label_encoder.fit_transform(df['x8'])
df['x11'] = label_encoder.fit_transform(df['x11'])
df['x12'] = label_encoder.fit_transform(df['x12'])
df['x13'] = label_encoder.fit_transform(df['x13'])
df['x14'] = label_encoder.fit_transform(df['x14'])
df['x16'] = label_encoder.fit_transform(df['x16'])
df['x17'] = label_encoder.fit_transform(df['x17'])
df['x20'] = label_encoder.fit_transform(df['x20'])
df['x21'] = label_encoder.fit_transform(df['x21'])
df['x22'] = label_encoder.fit_transform(df['x22'])
```

[6] ✓ 6.5s

6. Sisa data uang bersifat ordinal diubah secara manual dengan menggunakan perintah replace



```
#mengubah data bertingkat menggunakan replace
df['x2'].replace(['KURUS', 'NORMAL', 'GEMUK', 'OBESITAS'], [0, 1, 2, 3], inplace=True)
df['x9'].replace(['Tidak', 'Mungkin', 'Ya'], [0, 1, 2], inplace=True)
df['x10'].replace(['KURANG DARI 5 JAM', 'ANTARA 5 - 10 JAM', ' LEBIH DARI 10 JAM'], [0, 1, 2], inplace=True)
df['x15'].replace(['Tidak', 'Kadang-kadang', 'Ya'], [0, 1, 2], inplace=True)
df['x18'].replace(['Tidak', 'Kadang-kadang', 'Ya'], [0, 1, 2], inplace=True)
df['x19'].replace(['Tidak', 'Kadang-kadang', 'Ya'], [0, 1, 2], inplace=True)
df['x22'].replace(['Tidak', 'Sebagian', 'Ya'], [0, 1, 2], inplace=True)
df.head()
```

[7] ✓ 0.6s

7. Setelah semua data diubah kedalam bentuk integer maka selanjutnya dilakukan penetuan data yang menjadi variabel dependen dan independen. Kolom class akan menjadi variabel dependen diwakili y dan selain itu menjadi variabel independen diwakili x

```
x=df.drop('class', axis=1)
y=df['class']
```
9] ✓ 0.9s

8. Setelah dimasukkan ke dalam variabel, selanjutnya dibagi menjadi nilai yang akan dijadikan train dan test dengan fungsi sklearn, untuk ukuran menggunakan pembagian 7:3

```
#melakukan splitting data menggunakan perbandingan 3:7
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
```
[70] ✓ 0.7s

Pembahasan :

1. Gaussian Naïve Bayes

Tahapan penggunaan Gaussian Naïve Bayes dengan library sklearn adalah dengan import naïve_bayes dan memanggil GaussianNB, dari proses tersebut didapati hasil sebagai berikut, dimana nilai akurasi hanya sebesar 45% dengan menggunakan cross validate

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_validate,cross_val_score
model=GaussianNB()
model.fit(x_train, y_train)
cv_score1=cross_validate(model,x,y,cv=10, return_train_score=True)
cv_score2=cross_val_score(model,x,y,cv=10)
print (cv_score1['train_score'].mean(), cv_score1['test_score'].mean())
print(cv_score2.mean()) #test_score
```
[55] ✓ 2.3s

```
... 0.5823998795543511 0.45111111111111113
    0.45111111111111113
```

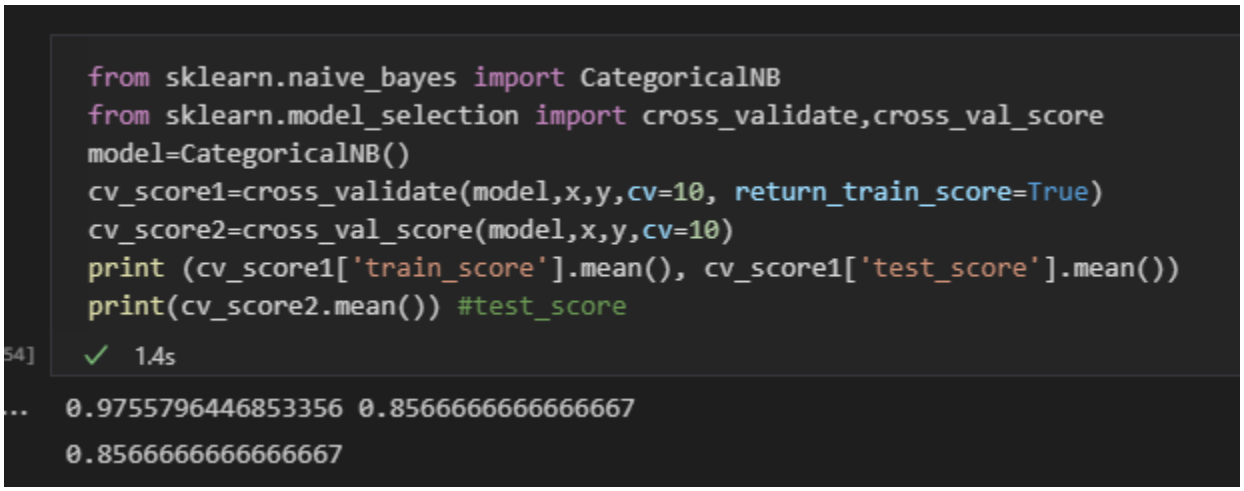
Lalu jika dicoba melakukan tuning hyperparameter didapati hasil akurasi sebesar 84,9%

```
#Tuning Hyperparameter
from sklearn.model_selection import RepeatedStratifiedKFold
cv_method = RepeatedStratifiedKFold(n_splits=10,  n_repeats=3, random_state=10)
param_grid_nb = {'var_smoothing': np.logspace(0,-9, num=100)}

from sklearn.model_selection import GridSearchCV
gs_NB = GridSearchCV(estimator=model, param_grid=param_grid_nb, cv=cv_method,verbose=1,scoring='accuracy')
gs_NB.fit(x,y)
print(gs_NB.best_params_,gs_NB.best_score_)
```

```
[53] ✓ 110.1s
... Fitting 30 folds for each of 100 candidates, totalling 3000 fits
{'var_smoothing': 0.533669923120631} 0.8496296296296297
```

2. Categorical Naïve Bayes



```
from sklearn.naive_bayes import CategoricalNB
from sklearn.model_selection import cross_validate,cross_val_score
model=CategoricalNB()
cv_score1=cross_validate(model,x,y,cv=10, return_train_score=True)
cv_score2=cross_val_score(model,x,y,cv=10)
print (cv_score1['train_score'].mean(), cv_score1['test_score'].mean())
print(cv_score2.mean()) #test_score
```

```
54] ✓ 1.4s
... 0.9755796446853356 0.8566666666666667
    0.8566666666666667
```

Kesimpulan :

Dibandingkan dengan metode Algoritma sebelumnya yakni decision tree dan knn naïve bayes memiliki nilai yang terbaik, karena nilai akurasi berada pada persentase diatas 80% terlebih lagi dalam fungsi naïve bayes tidak memiliki banyak parameter yang dapat diubah jika dibandingkan dengan algoritma yang lain

https://drive.google.com/file/d/13jRlG0IOGPCvaqAqzoHPNB_s2CJv5WoX/view?usp=sharing