PRAKTIKUM DATA MINING

Nama : Ahmad Izza Zain Firdaus

NIM : 19051214063

Kelas/Angkatan : SIB/2019

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 data dipisah dilakukan reduksi dimensi, untuk reduksi dimensi sendiir menggunakan modul dari sklearn yakni decomposition.
   Untuk n-componen diisi menggunakan jumlah kelas X

```
REDUKSI DIMENSI DENGAN MENGGUNAKAN PCA

import numpy as np
from sklearn.decomposition import PCA
pca = PCA(n_components=22)
pca.fit(x)
x= pca.fit_transform(x)

[188] ✓ 0.2s
```

9. Selanjutnya diperiksa apakah persebaran kelas sudah cukup merata. Jika belum maka dilakukan mengatasi *imbalancing*

```
#menghitung persebaran class
y.value_counts()

[189] ✓ 0.3s

... 1    32
    0    29
    2    28
    3     2
    Name: class, dtype: int64
```

10. Untuk praktikum kali ini saya menggunakan oversampling SMOTE, dimana nilai akan disamaakan menjadi jumlah data tertinggi

```
OVERSAMPLING DENGAN SMOTE

from imblearn.over_sampling import SMOTE
smote=SMOTE(k_neighbors=1)
x_resampled, y_resampled = smote.fit_resample(x, y)
y_resampled.value_counts()

90] ✓ 0.2s

.. 0    32
   1    32
   2    32
   3    32
   Name: class, dtype: int64
```

1. Dimulai dengan mendeklarasikan metode validasi kfold

```
PEMILIHAN VALIDASI DENGAN MENGGUNAKAN STARIFIED K-FOLD


# Model Validasi Dengan StratifiedKFold
from sklearn.model_selection import StratifiedKFold
kfold = StratifiedKFold(n_splits=3, shuffle=True ,random_state=0)
✓ 0.3s
```

Pembahasan:

Semua model berikut memiliki dua jenis hasil berdasarkan pada asal data, dimana pertama data yang masih imbalance dan yang kedua data yang sudah dibalance-kan

1. DECISSION TREE

   Untuk hasil perhitungan awal menggunakan data asli didapatkan hasil dimana kriteria menggunaakn entropy, dan kedalaman cabang=9 dengan nilai tes 75

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
dt = DecisionTreeClassifier()
param_dt = {'criterion': ['gini', 'entropy'], 'splitter': ['best', 'random'], 'max_depth': np.arange(0, 20)}
dt = GridSearchCV(dt, param_grid=param_dt, scoring='accuracy', cv=kfold)
dt.fit(x, y)
print("Parameter Terbaik Adalah %s Dengan Nilai Test: %0.0f" % (dt.best_params_, dt.best_score_ * 100 ))
[192] ✓ 2.5s
... Parameter Terbaik Adalah {'criterion': 'entropy', 'max_depth': 9, 'splitter': 'random'} Dengan Nilai Test: 75
```

Lalu jika menggunakan data yang sudah dibalance kan hasil meningkat menjadi 83 persen

```
# Decission Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
dt = DecisionTreeClassifier()
param_dt = {'criterion': ['gini', 'entropy'], 'splitter': ['best', 'random'], 'max_depth': np.arange(0, 15)}
dt = GridSearchCV(dt, param_grid=param_dt, scoring='accuracy', cv=kfold)
dt.fit(x_resampled, y_resampled)
print("Parameter Terbaik Adalah %s Dengan Nilai Test: %0.0f" % (dt.best_params_, dt.best_score_ * 100 ))
[118] ✓ 2.3s
... Parameter Terbaik Adalah {'criterion': 'gini', 'max_depth': 13, 'splitter': 'best'} Dengan Nilai Test: 83
```

2. KNN

Selanjutnya jika menggunakan KNN, didapati hasil akurasi 85, dengan menggunakan algoritma ball tree dan neighbor=4

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
knn = KNeighborsClassifier()
param_knn = {'n_neighbors': np.arange(2, 61), 'weights': ['uniform', 'distance'], 'algorithm': ['auto', 'ball_tree'
knn = GridSearchCV(knn, param_grid=param_knn, scoring='accuracy', cv=kfold)
knn.fit(x, y)
print("Parameter Terbaik Adalah %s Dengan Nilai Test: %0.0f" % (knn.best_params_, knn.best_score_ * 100 ))
```

✓ 16.1s

```
Parameter Terbaik Adalah {'algorithm': 'ball_tree', 'n_neighbors': 4, 'weights': 'uniform'} Dengan Nilai Test: 85
```

Jika dilakukan menggunakan data yang sudah diimangkan maka didapati nilai test juga meningkat

```
# K-Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
knn = KNeighborsClassifier()
param_knn = {'n_neighbors': np.arange(2, 61), 'weights': ['uniform', 'distance'], 'algorithm': ['auto', 'ball_tree',
knn = GridSearchCV(knn, param_grid=param_knn, scoring='accuracy', cv=kfold)
knn.fit(x_resampled, y_resampled)
print("Parameter Terbaik Adalah %s Dengan Nilai Test: %0.0f" % (knn.best_params_, knn.best_score_ * 100 ))
```

[220] ✓ 15.3s

```
... Parameter Terbaik Adalah {'algorithm': 'auto', 'n_neighbors': 3, 'weights': 'distance'} Dengan Nilai Test: 91
```

3. Jika menggunakan data awal, maka didapati nilai test sebesar 79

GAUSSIAN NB

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import GridSearchCV
nb = GaussianNB()
param_nb = {'var_smoothing': np.logspace(0, -15, 100)}
nb = GridSearchCV(nb, param_grid=param_nb, scoring='accuracy', cv=kfold)
nb.fit(x, y)
print("Parameter Terbaik Adalah %s Dengan Nilai Test: %0.0f" % (nb.best_params_, nb.best_score_ * 100 ))
```

[221] ✓ 2.1s

```
... Parameter Terbaik Adalah {'var_smoothing': 1.0} Dengan Nilai Test: 79
```

Sedangkan setelah data dibalancing maka nilai test meningkat

```
# Naive Bayes
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import GridSearchCV
nb = GaussianNB()
param_gnb = {'var_smoothing': np.logspace(0, -15, 100)}
nb = GridSearchCV(nb, param_grid=param_nb, scoring='accuracy', cv=kfold)
nb.fit(x_resampled, y_resampled)
print("Parameter Terbaik Adalah %s Dengan Nilai Test: %0.0f" % (nb.best_params_, nb.best_score_ * 100 ))
```

✓ 2.8s

```
Parameter Terbaik Adalah {'var_smoothing': 0.17475284000076838} Dengan Nilai Test: 90
```

4. SVC

Dengan menggunakan svc didapati nilai akurasi sebesar 86

SVC

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
svc = SVC()
param_grid = {'C':np.logspace(start = -15, stop = 100, base = 1.05), 'kernel':['linear', 'poly', 'rbf', 'si
svc = GridSearchCV(svc, param_grid = param_grid, scoring='accuracy', cv=3)
svc.fit(x, y)
print("Parameter Terbaik Adalah %s Dengan Nilai: %0.0f" % (svc.best_params_, svc.best_score_ * 100 ))
```

✓ 8.4s

```
Parameter Terbaik Adalah {'C': 0.85272785940597, 'gamma': 'scale', 'kernel': 'sigmoid'} Dengan Nilai: 86
```

Lalu saat dilakukan balancing meningkat menjadi 94

```
# Support Vector Machine
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
svc = SVC()
param_grid = {'C':np.logspace(start = -15, stop = 100, base = 1.05), 'kernel':['linear', 'poly', 'rbf', '
svc = GridSearchCV(svc, param_grid = param_grid, scoring='accuracy', cv=3)
svc.fit(x_resampled, y_resampled)
print("Parameter Terbaik Adalah %s Dengan Nilai: %0.0f" % (svc.best_params_, svc.best_score_ * 100 ))
```

[224] ✓ 9.4s

```
... Parameter Terbaik Adalah {'C': 5.973381346823118, 'gamma': 'auto', 'kernel': 'rbf'} Dengan Nilai: 94
```

5. RANDOM FOREST

Dengan menggunakan random forest didapati hasil sebesar 78

RANDOM FOREST CLASSIFIER

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
rf = RandomForestClassifier()
param_rf = {'n_estimators': np.arange(10, 100), 'criterion': ['gini', 'entropy']}
rf = GridSearchCV(rf, param_grid=param_rf, scoring='accuracy', cv=kfold)
rf.fit(x, y)
print("Parameter Terbaik Adalah %s Dengan Nilai Test: %0.0f" % (rf.best_params_, rf.best_sc
```

[225] ✓ 187.1s

```
... Parameter Terbaik Adalah {'criterion': 'gini', 'n_estimators': 70} Dengan Nilai Test: 78
```

Lalu saat dilakukan balancing meningkat menjadi 90

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
rf = RandomForestClassifier()
param_rf = {'n_estimators': np.arange(10, 100), 'criterion': ['gini', 'entropy']}
rf = GridSearchCV(rf, param_grid=param_rf, scoring='accuracy', cv=kfold)
rf.fit(x_resampled, y_resampled)
print("Parameter Terbaik Adalah %s Dengan Nilai Test: %0.0f" % (rf.best_params_, rf.best_score_ * 100 ))
```

[226] ✓ 193.5s

```
... Parameter Terbaik Adalah {'criterion': 'gini', 'n_estimators': 84} Dengan Nilai Test: 90
```

6. REGRESI LOGISTIK

Hasil awal regresi logistic sebesar 85

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
lr = LogisticRegression()
param_lr =  {'penalty': ['l1', 'l2', 'elasticnet', 'none'], 'C': np.logspace(0, -15, 100), 'dual': [True, False]}
lr = GridSearchCV(lr, param_grid=param_lr, scoring='accuracy', cv=kfold)
lr.fit(x, y)
print("Parameter Terbaik Adalah %s Dengan Nilai Test: %0.0f" % (lr.best_params_, lr.best_score_ * 100 ))
```

[227] ✓ 37.3s

```
... Parameter Terbaik Adalah {'C': 0.7054802310718643, 'dual': False, 'penalty': 'l2'} Dengan Nilai Test: 85
```

Saat balance-kan menjadi 91

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
lr = LogisticRegression()
param_lr =  {'penalty': ['l1', 'l2', 'elasticnet', 'none'], 'C': np.logspace(0, -15, 100), 'dual'
lr = GridSearchCV(lr, param_grid=param_lr, scoring='accuracy', cv=kfold)
lr.fit(x_resampled, y_resampled)
print("Parameter Terbaik Adalah %s Dengan Nilai Test: %0.0f" % (lr.best_params_, lr.best_score_ *
```

228] ✓ 44.2s

```
... Parameter Terbaik Adalah {'C': 1.0, 'dual': False, 'penalty': 'l2'} Dengan Nilai Test: 91
```

KESIMPULAN:

METODE	SEBELUM	SESUDAH	SELISIH
DECISSION TREE	68	83	15
KNN	85	91	6
NAÏVE BAYES	79	90	11
SVC	86	94	8
RAND. FOREST	78	90	12
REGRESI LINEAR	85	91	6

Dari data diatas dapat disimpulkan bahwa menggunaan balancing meningkatkan akurasi metode, terutama pada decision tree