Laporan praktikum Data Mining

Agung Dwi Nugroho
3122600006

dataset titanic.csv, dan tampilkan

```
import pandas as pd

dataset = pd.read_csv('titanic.csv')
dataset
```

Analisis :
Menampilkan data csv menjadi table

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

test_dataset <- titanic_test.csv, dan tampilkan

```
import pandas as pd

test_dataset = pd.read_csv('titanic_test.csv')

test_dataset
```

Untuk menampilkan data dari file titanic_test.csv, kita dapat menggunakan library pandas dengan memanggil method read_csv

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...

Hilangkan baris data yang terdapat missing values (catat posisi data yang hilang → pos_missing_train

```
import pandas as pd

pos_missing_test = test_dataset[['Age', 'Fare']][test_dataset[['Age', 'Fare']].isnull().any(axis=1)].index.tolist()
test_data = test_dataset[['Age', 'Fare']].dropna()
print(test_data)
print(pos_missing_test)
```

Kode tersebut bertujuan untuk mengidentifikasi dan menghapus missing values pada kolom Age dan Fare dari DataFrame test_dataset. isnull(): Mengecek apakah terdapat missing value (nilai kosong) di setiap elemen dari kolom Age dan Fare. any(axis=1): Memeriksa jika ada setidaknya satu nilai kosong di baris tersebut (baik di kolom Age maupun Fare). dropna(): Menghapus semua baris yang memiliki missing value di kolom Age atau Fare.

```
      Age      Fare
0    34.5    7.8292
1    47.0    7.0000
2    62.0    9.6875
3    27.0    8.6625
4    22.0   12.2875
..    ...       ...
409   3.0   13.7750
411  37.0   90.0000
412  28.0    7.7750
414  39.0  108.9000
415  38.5    7.2500

[331 rows x 2 columns]
[10  22  29  33  36  39  41  47  5
```

train_label <- ambil dataset kolom kelas (Survived), yang bukan pos_missing_train

```
import pandas as pd

train_label = dataset['Survived'][~dataset.index.isin(pos_missing_train)]
train_label
```

	Survived
0	0
1	1
2	1
3	1
4	0

Kode tersebut melakukan filtering terhadap kolom Survived dalam DataFrame dataset untuk menghapus baris yang memiliki missing value berdasarkan daftar pos_missing_train. dataset.index.isin(pos_missing_train): Mengecek apakah index dari setiap baris di dataset termasuk dalam daftar pos_missing_train, yang berisi index dari baris-baris dengan missing value.

test_label <- titanic_testlabel.csv, yang bukan pos_missing_test

```
import pandas as pd

train_data_min = train_data.min()
train_data_max = train_data.max()
train_data = (train_data - train_data_min) / (train_data_max - train_data_min)

print(train_data)
```

Kode tersebut melakukan filtering pada dataset titanic_testlabel.csv untuk menghapus baris yang memiliki missing values sesuai dengan daftar pos_missing_test, dan hasil akhirnya adalah subset dari kolom Survived yang berisi data valid (tanpa missing values pada kolom lain).

	Age	Fare
0	0.271174	0.014151
1	0.472229	0.139136
2	0.321438	0.015469
3	0.434531	0.103644
4	0.434531	0.015713
..	...	...

train_data <- lakukan normalisasi pada train_data dengan Min-Max 0-1 (catat nilai min dan max setiap atribut

```
import pandas as pd

train_data_min = train_data.min()
train_data_max = train_data.max()
test_data = (test_data - train_data_min) / (train_data_max - train_data_min)

print(test_data)
```

Analisis :
Membuat normalisasi data dengan metode min max dengan rumus (data - datakecil) / (databesar - datakecil)

	Age	Fare
0	34.5	7.8292
1	47.0	7.0000
2	62.0	9.6875
3	27.0	8.6625
4	22.0	12.2875

Z-Score

```
data_z_score_manual = (data - data.mean()) / data.std()

data_normalisais_z_score_manual = pd.DataFrame(data_z_score_manual, columns=['Age', 'Fare'])
data_normalisais_z_score_manual
```

Kode ini melakukan normalisasi data menggunakan teknik Min-Max Scaling pada dataset train_data, di mana setiap nilai diubah menjadi rentang antara 0 dan 1.

	Age	Fare
0	-0.595670	-0.502163
1	0.634089	0.786404
2	-0.288230	-0.488580
3	0.403509	0.420494
4	0.403509	-0.486064

test_data <- lakukan normalisasi pada test_data dengan Min-Max 0-1 (dengan nilai min dan max setiap atribut pada Langkah 7

```
k=1 -> class result = [1 1 1 1 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 1 1 0 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1
 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1
 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1
 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0
 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1
 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1 1 1 1 1]
k=1 -> precision ratio = 0.4350453172205438, error ratio = 0.5649546827794563
k=2 -> class result = [0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 1 0 0 0
 1 0 0 1 0 1 0 1 1 1 0 0 1 0 0 0 0 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 0 0 0 0 1
 0 1 1 0 1 0 0 0 1 1 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0
 1 1 1 1 1 0 0 0 1 1 0 1 0 1 0 0 0 1 0 0 1 1 1 0 0 0 1 1 1 1 1 1 1 0 1 1 1 0
 1 0 0 1 1 0 1 0 1 0 0 1 1 1 0 0 0 1 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0
 1 0 1 0 0 1 0 0 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 1 1
 1 0 1 0 1 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 1 1 0 0 1 0 0 1
 1 0 0 1 1 0 1 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 0 1 0]
k=2 -> precision ratio = 0.6163141993957704, error ratio = 0.38368580060422963
k=3 -> class result = [0 0 0 0 1 1 0 1 0 1 1 1 0 1 1 0 0 0 0 1 0 1 1 1 0 1 0 1 1
```

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Misal 'train_data', 'train_label', 'test_data', dan 'test_label' sudah ada
class_result = {}  # Dictionary untuk menyimpan hasil klasifikasi setiap k

# Lakukan klasifikasi dengan k dari 1 hingga 10
for k in range(1, 11):
    knn = KNeighborsClassifier(n_neighbors=k)  # Inisialisasi model k-NN dengan k tertentu
    knn.fit(train_data, train_label)  # Melatih model dengan data latih

    # Prediksi data uji
    class_result[k] = knn.predict(test_data)

    # Menghitung akurasi untuk k tertentu
    precision_ratio = accuracy_score(test_label, class_result[k])
    error_ratio = 1 - precision_ratio

    # Output hasil prediksi, akurasi, dan error ratio
    print(f"k={k} -> class result = {class_result[k]}")
    print(f"k={k} -> precision ratio = {precision_ratio}, error ratio = {error_ratio}")
```

Analisis :
Membuat normalisasi data dengan metode sigmodial dengan

Link google collab :
https://colab.research.google.com/drive/1wXTxT-btZnXX4EHS7_gyhajddzYnd6U5#scrollTo=S8KyxhrnAjfv