



Praktikum Data Mining

Minggu Ke-4



Faris Saifullah
3124640034

Input

```
#no.1
import pandas as pd
dataset = pd.read_csv('titanic.csv')
print(dataset.head())
```

Analisa:
Menampilkan dataset dari file data titanic.csv

Output

```
PassengerId  Survived  Pclass  \
0             1         0       3
1             2         1       1
2             3         1       3
3             4         1       1
4             5         0       3

      Name  Sex  Age  SibSp  \
0  Braund, Mr. Owen Harris  male  22.0  1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0  1
2  Heikkinen, Miss. Laina  female  26.0  0
3  Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0  1
4  Allen, Mr. William Henry  male  35.0  0

   Parch  Ticket   Fare Cabin Embarked
0      0   A/5 21171   7.2500   NaN      S
1      0    PC 17599  71.2833   C85      C
2      0  STON/O2. 3101282   7.9250   NaN      S
3      0   113803   53.1000  C123      S
4      0   373450   8.0500   NaN      S
```

Input

```
#no.2
dataset = pd.read_csv('titanic_test.csv')
print(dataset.head())
```

Analisa:
Menampilkan dataset dari file data titanic_test.csv

Output

	PassengerId	Pclass	Name	Sex	\
0	892	3	Kelly, Mr. James	male	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	
2	894	2	Myles, Mr. Thomas Francis	male	
3	895	3	Wirz, Mr. Albert	male	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	

	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	34.5	0	0	330911	7.8292	NaN	Q
1	47.0	1	0	363272	7.0000	NaN	S
2	62.0	0	0	240276	9.6875	NaN	Q
3	27.0	0	0	315154	8.6625	NaN	S
4	22.0	1	1	3101298	12.2875	NaN	S

Input

```
!no.3
train_data = pd.read_csv('titanic.csv')
train_data_features = train_data[['Age', 'Fare']]
pos_missing_train = train_data_features[train_data_features.isna().any(axis=1)].index.tolist()
train_data_cleaned = train_data_features.dropna()
print("Data setelah missing values dihilangkan:")
print(train_data_cleaned.head())
print("\nPosisi missing values yang dihilangkan:")
print(pos_missing_train[:5])
```

Analisa:

Menghilangkan baris data yang terdapat missing values pada file titanic.csv

Output

```
Data setelah missing values dihilangkan:
   Age  Fare
0  22.0   7.2500
1  38.0  71.2833
2  26.0   7.9250
3  35.0  53.1000
4  35.0   8.0500

Posisi missing values yang dihilangkan:
[5, 17, 19, 26, 28]
```

Input

```
lno.4
test_data = pd.read_csv('titanic_test.csv')
test_data_features = test_data[['Age', 'Fare']]
pos_missing_test = test_data_features[test_data_features.isna().any(axis=1)].index.tolist()
test_data_cleaned = test_data_features.dropna()
print("Data setelah missing values dihilangkan:")
print(test_data_cleaned.head())
print("\nPosisi missing values yang dihilangkan:")
print(pos_missing_test[:5])
```

Analisa:

Menghilangkan baris data yang terdapat missing values
pada file titanic_test.csv

Output

```
Data setelah missing values dihilangkan:
   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

Posisi missing values yang dihilangkan:
[10, 22, 29, 33, 36]
```

Input

```
#no.5
df = pd.read_csv('titanic.csv')
train_label = df['Survived'].dropna()
print("Data pada kolom 'Survived' yang tidak memiliki nilai kosong:")
print(train_label.head())
```

Output

```
Data pada kolom 'Survived' yang tidak memiliki nilai kosong:
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

Analisa:

Menampilkan data kolom 'Survived' yang tidak memiliki nilai kosong



Input

```
#no.6
df_test = pd.read_csv('titanic_testlabel.csv')
test_label = df_test.dropna()
print("Data pada titanic_testlabel.csv yang tidak memiliki nilai kosong:")
print(test_label.head())
```

Output

```
Data pada titanic_testlabel.csv yang tidak memiliki nilai kosong:
  PassengerId  Survived
0          892         0
1          893         1
2          894         0
3          895         0
4          896         1
```

Analisa:

Menampilkan data yang tidak memiliki nilai kosong pada file
titanic_testlabel.csv



Input

```
lno.7
df = pd.read_csv('titanic.csv')
df_numeric = df.select_dtypes(include=['float64', 'int64']).copy()
min_values = df_numeric.min()
max_values = df_numeric.max()
new_min = 0
new_max = 1
train_data = (df_numeric - min_values) * (new_max - new_min) / (max_values - min_values) + new_min
print("Data setelah normalisasi (0-1):")
print(train_data.head())
print("\nNilai minimum setiap atribut sebelum normalisasi:")
print(min_values)
print("\nNilai maksimum setiap atribut sebelum normalisasi:")
print(max_values)
```

Analisa:
Menormalisasi kolom yang berada di file titanic.csv

Output

```
Data setelah normalisasi (0-1):
  PassengerId  Survived  Pclass     Age  SibSp  Parch    Fare
0   0.000000     0.0      1.0  0.271174  0.125   0.0  0.014151
1   0.001124     1.0      0.0  0.472229  0.125   0.0  0.139136
2   0.002247     1.0      1.0  0.321438  0.000   0.0  0.015469
3   0.003371     1.0      0.0  0.434531  0.125   0.0  0.103644
4   0.004494     0.0      1.0  0.434531  0.000   0.0  0.015713

Nilai minimum setiap atribut sebelum normalisasi:
PassengerId    1.00
Survived       0.00
Pclass         1.00
Age            0.42
SibSp          0.00
Parch          0.00
Fare           0.00
dtype: float64

Nilai maksimum setiap atribut sebelum normalisasi:
PassengerId    891.0000
Survived       1.0000
Pclass         3.0000
Age            80.0000
SibSp          8.0000
Parch          6.0000
Fare          512.3292
dtype: float64
```


Input

```
#!/no.8
df = pd.read_csv('titanic_test.csv')
df_numeric = df.select_dtypes(include=['float64', 'int64']).copy()
min_values = df_numeric.min()
max_values = df_numeric.max()
new_min = 0
new_max = 1
train_data = (df_numeric - min_values) * (new_max - new_min) / (max_values - min_values) + new_min
print("Data setelah normalisasi (0-1):")
print(train_data.head())
print("\nNilai minimum setiap atribut sebelum normalisasi:")
print(min_values)
print("\nNilai maksimum setiap atribut sebelum normalisasi:")
print(max_values)
```

Analisa:

Menormalisasi kolom yang berada di file titanic_test.csv

Output

```
Data setelah normalisasi (0-1):
  PassengerId  Pclass    Age  SibSp  Parch    Fare
0    0.000000    1.0  0.452723  0.000  0.000000  0.015282
1    0.002398    1.0  0.617566  0.125  0.000000  0.013663
2    0.004796    0.5  0.815377  0.000  0.000000  0.018909
3    0.007194    1.0  0.353818  0.000  0.000000  0.016908
4    0.009592    1.0  0.287881  0.125  0.111111  0.023984

Nilai minimum setiap atribut sebelum normalisasi:
PassengerId    892.00
Pclass          1.00
Age             0.17
SibSp           0.00
Parch           0.00
Fare            0.00
dtype: float64

Nilai maksimum setiap atribut sebelum normalisasi:
PassengerId    1309.0000
Pclass          3.0000
Age            76.0000
SibSp           8.0000
Parch           9.0000
Fare           512.3292
dtype: float64
```

Input

```
##No.9
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report

df_train = pd.read_csv('titanic.csv')
df_test = pd.read_csv('titanic_test.csv')
df_train = df_train.select_dtypes(include=['float64', 'int64']).dropna()
df_test = df_test.select_dtypes(include=['float64', 'int64']).dropna()
X_train = df_train.drop('Survived', axis=1)
y_train = df_train['Survived']

X_test = df_test.drop('Survived', axis=1, errors='ignore')

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

for k in range(1, 11):
    print(f"\nKlasifikasi dengan k = {k}")

    knn = KNeighborsClassifier(n_neighbors=k)

    knn.fit(X_train_scaled, y_train)

    y_pred = knn.predict(X_test_scaled)

    print("Hasil klasifikasi:")
    print(classification_report(y_train, knn.predict(X_train_scaled)))

    if 'Survived' in df_test.columns:
        print("Classification report untuk test_data:")
        print(classification_report(df_test['Survived'], y_pred))
```

Klasifikasi dengan k = 1					
Hasil klasifikasi:					
	precision	recall	f1-score	support	
0	1.00	1.00	1.00	424	
1	1.00	1.00	1.00	290	
accuracy			1.00	714	
macro avg	1.00	1.00	1.00	714	
weighted avg	1.00	1.00	1.00	714	

Klasifikasi dengan k = 2					
Hasil klasifikasi:					
	precision	recall	f1-score	support	
0	0.78	1.00	0.87	424	
1	1.00	0.58	0.73	290	
accuracy			0.83	714	
macro avg	0.89	0.79	0.80	714	
weighted avg	0.87	0.83	0.82	714	

Klasifikasi dengan k = 3					
Hasil klasifikasi:					
	precision	recall	f1-score	support	
0	0.83	0.84	0.84	424	
1	0.77	0.75	0.76	290	
accuracy			0.81	714	
macro avg	0.80	0.80	0.80	714	
weighted avg	0.81	0.81	0.81	714	

Klasifikasi dengan k = 4					
Hasil klasifikasi:					
	precision	recall	f1-score	support	
0	0.75	0.92	0.83	424	
1	0.83	0.56	0.66	290	
accuracy			0.77	714	
macro avg	0.79	0.74	0.75	714	
weighted avg	0.78	0.77	0.76	714	

Klasifikasi dengan k = 5					
Hasil klasifikasi:					
	precision	recall	f1-score	support	
0	0.79	0.86	0.82	424	
1	0.76	0.67	0.72	290	
accuracy			0.78	714	
macro avg	0.78	0.77	0.77	714	
weighted avg	0.78	0.78	0.78	714	

Klasifikasi dengan k = 6					
Hasil klasifikasi:					
	precision	recall	f1-score	support	
0	0.75	0.91	0.82	424	
1	0.81	0.57	0.67	290	
accuracy			0.77	714	
macro avg	0.78	0.74	0.75	714	
weighted avg	0.78	0.77	0.76	714	

Klasifikasi dengan k = 7					
Hasil klasifikasi:					
	precision	recall	f1-score	support	
0	0.77	0.85	0.81	424	
1	0.74	0.64	0.69	290	
accuracy			0.76	714	
macro avg	0.76	0.74	0.75	714	
weighted avg	0.76	0.76	0.76	714	

Klasifikasi dengan k = 8					
Hasil klasifikasi:					
	precision	recall	f1-score	support	
0	0.74	0.89	0.81	424	
1	0.78	0.55	0.64	290	
accuracy			0.75	714	
macro avg	0.76	0.72	0.73	714	
weighted avg	0.76	0.75	0.74	714	

Klasifikasi dengan k = 9					
Hasil klasifikasi:					
	precision	recall	f1-score	support	
0	0.76	0.84	0.80	424	
1	0.73	0.61	0.67	290	
accuracy			0.75	714	
macro avg	0.75	0.73	0.73	714	
weighted avg	0.75	0.75	0.75	714	

Klasifikasi dengan k = 10					
Hasil klasifikasi:					
	precision	recall	f1-score	support	
0	0.74	0.89	0.81	424	
1	0.78	0.54	0.64	290	
accuracy			0.75	714	
macro avg	0.76	0.72	0.73	714	
weighted avg	0.76	0.75	0.74	714	

Output

Input

```
lno.10
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score

df_train = pd.read_csv('titanic.csv')
df_test = pd.read_csv('titanic_test.csv')
df_train = df_train.select_dtypes(include=['float64', 'int64']).dropna()
df_test = df_test.select_dtypes(include=['float64', 'int64']).dropna()
X_train = df_train.drop('Survived', axis=1)
y_train = df_train['Survived']
X_test = df_test.drop('Survived', axis=1, errors='ignore')
y_test = df_test['Survived'] if 'Survived' in df_test.columns else nonzero
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
results = {}

for k in range(1, 11):
    print(f"\nKlasifikasi dengan k = {k}")

    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_scaled, y_train)
    y_pred = knn.predict(X_test_scaled)
    results[k] = y_pred
    if y_test is not None:
        accuracy = accuracy_score(y_test, y_pred)
        error_ratio = 1 - accuracy
        print(f"Rasio error untuk k = {k}: {error_ratio:.4f}")
    else:
        print("Kolom 'Survived' tidak ada di df_test, jadi rasio error tidak dapat dihitung."),
```

Output

```
Klasifikasi dengan k = 1
Kolom 'Survived' tidak ada di df_test, jadi rasio error tidak dapat dihitung.

Klasifikasi dengan k = 2
Kolom 'Survived' tidak ada di df_test, jadi rasio error tidak dapat dihitung.

Klasifikasi dengan k = 3
Kolom 'Survived' tidak ada di df_test, jadi rasio error tidak dapat dihitung.

Klasifikasi dengan k = 4
Kolom 'Survived' tidak ada di df_test, jadi rasio error tidak dapat dihitung.

Klasifikasi dengan k = 5
Kolom 'Survived' tidak ada di df_test, jadi rasio error tidak dapat dihitung.

Klasifikasi dengan k = 6
Kolom 'Survived' tidak ada di df_test, jadi rasio error tidak dapat dihitung.

Klasifikasi dengan k = 7
Kolom 'Survived' tidak ada di df_test, jadi rasio error tidak dapat dihitung.

Klasifikasi dengan k = 8
Kolom 'Survived' tidak ada di df_test, jadi rasio error tidak dapat dihitung.

Klasifikasi dengan k = 9
Kolom 'Survived' tidak ada di df_test, jadi rasio error tidak dapat dihitung.

Klasifikasi dengan k = 10
Kolom 'Survived' tidak ada di df_test, jadi rasio error tidak dapat dihitung.
```