Praktikum Data Mining Minggu Ke-5

Faris Saifullah
3124640034

Input

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

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
```

Analisa:
Menampilkan dataset dari file data titanic.csv

Input

```python
#no.2a
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy_holdout = accuracy_score(y_test, y_pred)
print(f"Akurasi Hold-out Method (70%-30%): {accuracy_holdout:.4f}")
```

```python
#no.2b
kf = KFold(n_splits=10, shuffle=True, random_state=42)
accuracies_kfold = []
for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracies_kfold.append(accuracy_score(y_test, y_pred))
print(f"Akurasi rata-rata K-Fold (k=10): {sum(accuracies_kfold) / len(accuracies_kfold):.4f}")
```

```python
#no.2c
loo = LeaveOneOut()
accuracies_loo = []
for train_index, test_index in loo.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracies_loo.append(accuracy_score(y_test, y_pred))
print(f"Akurasi rata-rata Leave-One-Out (LOO): {sum(accuracies_loo) / len(accuracies_loo):.4f}")
```

Output

```
Akurasi Hold-out Method (70%-30%): 0.6828
```

```
Akurasi rata-rata K-Fold (k=10): 0.6936
```

```
Akurasi rata-rata Leave-One-Out (LOO): 0.7138
```

Analisa:
Menghitung rata-rata metode yang digunakan dalam perhitungan data

Input

```python
#no.3
train_data = data[['Sex', 'Age', 'Pclass', 'Fare']].copy()
train_data.loc[:, 'Age'] = train_data.groupby('Pclass')['Age'].transform(lambda x: x.fillna(x.mean()))
print(train_data.head())
print("\nCek missing values:\n", train_data.isnull().sum())
```

Output

```
   Sex   Age  Pclass     Fare
0    0  22.0       3   7.2500
1    1  38.0       1  71.2833
2    1  26.0       3   7.9250
3    1  35.0       1  53.1000
4    0  35.0       3   8.0500

Cek missing values:
 Sex       0
Age       0
Pclass    0
Fare      0
```

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

Input

```
#no.4
label = data['Survived'].copy()
print(label.head())
```

Output

```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

Analisa:
Menampilkan data survived pada file titanic.csv

Input

```python
#no.5
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
train_data_normalized = pd.DataFrame(scaler.fit_transform(train_data), columns=train_data.columns)
min_values = train_data.min()
max_values = train_data.max()
print("Data setelah normalisasi:\n", train_data_normalized.head())
print("\nNilai minimum setiap atribut sebelum normalisasi:\n", min_values)
print("\nNilai maksimum setiap atribut sebelum normalisasi:\n", max_values)
```

Output

```
Data setelah normalisasi:
    Sex       Age  Pclass      Fare
0  0.0  0.271174     1.0  0.014151
1  1.0  0.472229     0.0  0.139136
2  1.0  0.321438     1.0  0.015469
3  1.0  0.434531     0.0  0.103644
4  0.0  0.434531     1.0  0.015713

Nilai minimum setiap atribut sebelum normalisasi:
 Sex       0.00
Age        0.42
Pclass     1.00
Fare       0.00
dtype: float64

Nilai maksimum setiap atribut sebelum normalisasi:
 Sex         1.0000
Age         80.0000
Pclass       3.0000
Fare       512.3292
dtype: float64
```

Analisa:
menormalisasi pada train_data dengan Min-Max 0-1

Input

```python
#no.6
from sklearn.model_selection import train_test_split

train_data, test_data = train_test_split(data[['Sex', 'Age', 'Pclass', 'Fare']], test_size=0.3, random_state=42)
scaler = MinMaxScaler()
train_data_normalized = pd.DataFrame(scaler.fit_transform(train_data), columns=train_data.columns)
test_data_normalized = pd.DataFrame(scaler.transform(test_data), columns=test_data.columns)
min_values_test = test_data.min()
max_values_test = test_data.max()
print("Data test setelah normalisasi:\n", test_data_normalized.head())
print("\nNilai minimum setiap atribut test sebelum normalisasi:\n", min_values_test)
print("\nNilai maksimum setiap atribut test sebelum normalisasi:\n", max_values_test)
```

Analisa:
menormalisasi pada test_data dengan Min-Max 0-1

Output

```
Data test setelah normalisasi:
    Sex       Age  Pclass      Fare
0  0.0  0.346569     1.0  0.029758
1  0.0  0.384267     0.5  0.020495
2  0.0  0.246042     1.0  0.015469
3  1.0  0.070118     0.5  0.064412
4  1.0  0.170646     1.0  0.021942

Nilai minimum setiap atribut test sebelum normalisasi:
 Sex       0.00
Age        0.83
Pclass     1.00
Fare       0.00
dtype: float64

Nilai maksimum setiap atribut test sebelum normalisasi:
 Sex         1.000
Age         71.000
Pclass       3.000
Fare       262.375
dtype: float64
```

Input

```python
#no.7
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
error_ratio_holdout = 1 - accuracy_score(y_test, y_pred)
print(f"Error Ratio Hold-out Method (70%-30%): {error_ratio_holdout:.4f}")
kf = KFold(n_splits=10, shuffle=True, random_state=42)
error_ratios_kfold = []
for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    error_ratios_kfold.append(1 - accuracy_score(y_test, y_pred))
print(f"Error Ratio rata-rata K-Fold (k=10): {sum(error_ratios_kfold) / len(error_ratios_kfold):.4f}")
loo = LeaveOneOut()
error_ratios_loo = []
for train_index, test_index in loo.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    error_ratios_loo.append(1 - accuracy_score(y_test, y_pred))
print(f"Error Ratio rata-rata Leave-One-Out (LOO): {sum(error_ratios_loo) / len(error_ratios_loo):.4f}")
```

Output

```
Error Ratio Hold-out Method (70%-30%): 0.3246
Error Ratio rata-rata K-Fold (k=10): 0.2963
Error Ratio rata-rata Leave-One-Out (LOO): 0.2851
```

Analisa:
Menampilkan ratio error setiap metode yang digunakan ketika melakukan perhitungan data