Laporan praktikum Data Mining

Agung Dwi Nugroho
3122600006

dataset titanic.csv

```
import pandas as pd

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

print(dataset)
```

	PassengerId	Survived	Pclass	\
0	1	0	3	
1	2	1	1	
2	3	1	3	
3	4	1	1	
4	5	0	3	
..	...	...	...	
886	887	0	2	
887	888	1	1	
888	889	0	3	
889	890	1	1	
890	891	0	3	

Kode ini mengimpor pustaka pandas dan membaca dataset Titanic dari file CSV ke dalam dataframe. Kemudian, dataset tersebut dicetak ke konsol untuk inspeksi awal data.

Hold-out Method (70%-30%)

```
# Import libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# 1. Load dataset titanic.csv
dataset = pd.read_csv('titanic.csv')

# 2. Lakukan validation model dengan metode Hold-out (70%-30%)
train_set, test_set = train_test_split(dataset, test_size=0.3, random_state=42)

# 3. Ambil fitur (Sex, Age, Pclass, Fare) dan lakukan imputasi missing values untuk Age
# Gunakan .loc untuk menghindari SettingWithCopyWarning
train_data = train_set.loc[:, ['Sex', 'Age', 'Pclass', 'Fare']]
test_data = test_set.loc[:, ['Sex', 'Age', 'Pclass', 'Fare']]
```

Hold-out Method (70%-30%)

```
# Konversi kolom 'Sex' menjadi numerik (0 untuk male, 1 untuk female)
train_data.loc[:, 'Sex'] = train_data['Sex'].map({'male': 0, 'female': 1})
test_data.loc[:, 'Sex'] = test_data['Sex'].map({'male': 0, 'female': 1})

# Imputasi missing values pada kolom 'Age' dengan mean berdasarkan 'Pclass'
train_data.loc[:, 'Age'] = train_data.groupby('Pclass')['Age'].transform(lambda x: x.fillna(x.mean()))
test_data.loc[:, 'Age'] = test_data.groupby('Pclass')['Age'].transform(lambda x: x.fillna(x.mean()))

# 4. Ambil kolom kelas (Survived) sebagai label
train_label = train_set['Survived']
test_label = test_set['Survived']

# 5. Normalisasi train_data menggunakan Min-Max (0-1)
scaler = MinMaxScaler()
train_data_scaled = scaler.fit_transform(train_data)

# 6. Normalisasi test_data menggunakan skala yang sama dari train_data
test_data_scaled = scaler.transform(test_data)
```

Hold-out Method (70%-30%)

```
# 7. Lakukan klasifikasi k-NN (k=3)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(train_data_scaled, train_label)

# 8. Prediksi data uji
predictions = knn.predict(test_data_scaled)

# 9. Hitung akurasi dan error ratio
accuracy = accuracy_score(test_label, predictions)
error_ratio = 1 - accuracy

# Tampilkan hasil
print(f"Accuracy: {accuracy:.2f}")
print(f"Error Ratio: {error_ratio:.2f}")
```

```
Accuracy: 0.81
Error Ratio: 0.19
```

Kode ini memuat dataset Titanic, membaginya menjadi data pelatihan dan pengujian (70%-30%), serta memilih fitur `Sex`, `Age`, `Pclass`, dan `Fare`. Fitur `Sex` dikonversi menjadi numerik, dan missing values pada `Age` diimputasi dengan mean berdasarkan `Pclass`. Data kemudian dinormalisasi menggunakan `MinMaxScaler`. Model k-NN (k=3) dilatih pada data pelatihan dan digunakan untuk memprediksi data pengujian, dengan akurasi dan error ratio dihitung untuk evaluasi.

K-Fold (k=10)

```python
# Import libraries
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# 1. Load dataset titanic.csv
dataset = pd.read_csv('titanic.csv')

# 2. Ambil fitur (Sex, Age, Pclass, Fare) dan lakukan imputasi missing values untuk Age
# Gunakan .loc untuk menghindari SettingWithCopyWarning
data = dataset.loc[:, ['Sex', 'Age', 'Pclass', 'Fare']]

# Konversi kolom 'Sex' menjadi numerik (0 untuk male, 1 untuk female)
data.loc[:, 'Sex'] = data['Sex'].map({'male': 0, 'female': 1})

# Imputasi missing values pada kolom 'Age' dengan mean berdasarkan 'Pclass'
data.loc[:, 'Age'] = data.groupby('Pclass')['Age'].transform(lambda x: x.fillna(x.mean()))

# 3. Ambil kolom kelas (Survived) sebagai label
label = dataset['Survived']
```

K-Fold (k=10)

```
# 4. Inisialisasi KFold (k=10)
kf = KFold(n_splits=10, shuffle=True, random_state=42)

# Variabel untuk menyimpan total akurasi
total_accuracy = 0

# 5. Iterasi melalui setiap fold
for train_index, test_index in kf.split(data):
    # Split data train dan test berdasarkan index dari KFold
    train_data, test_data = data.iloc[train_index], data.iloc[test_index]
    train_label, test_label = label.iloc[train_index], label.iloc[test_index]

    # Normalisasi train_data menggunakan Min-Max (0-1)
    scaler = MinMaxScaler()
    train_data_scaled = scaler.fit_transform(train_data)

    # Normalisasi test_data menggunakan skala yang sama dari train_data
    test_data_scaled = scaler.transform(test_data)

    # 6. Lakukan klasifikasi k-NN (k=3)
    knn = KNeighborsClassifier(n_neighbors=3)
    knn.fit(train_data_scaled, train_label)
```

K-Fold (k=10)

```
        # 7. Prediksi data uji
        predictions = knn.predict(test_data_scaled)

        # 8. Hitung akurasi untuk fold ini
        accuracy = accuracy_score(test_label, predictions)
        total_accuracy += accuracy

        # Cetak hasil untuk setiap fold
        print(f"Fold accuracy: {accuracy:.2f}")

# 9. Hitung akurasi rata-rata dan error ratio
average_accuracy = total_accuracy / 10
error_ratio = 1 - average_accuracy

# Tampilkan hasil akhir
print(f"\nAverage Accuracy: {average_accuracy:.2f}")
print(f"Error Ratio: {error_ratio:.2f}")
```

K-Fold (k=10)

```
Fold accuracy: 0.82
Fold accuracy: 0.83
Fold accuracy: 0.82
Fold accuracy: 0.80
Fold accuracy: 0.89
Fold accuracy: 0.88
Fold accuracy: 0.76
Fold accuracy: 0.82
Fold accuracy: 0.76
Fold accuracy: 0.87

Average Accuracy: 0.82
Error Ratio: 0.18
```

Kode ini memuat dataset Titanic, melakukan imputasi missing values, dan mengaplikasikan K-Fold Cross-Validation (k=10) dengan model k-NN (k=3).

LOO

```
# Import libraries
import pandas as pd
from sklearn.model_selection import LeaveOneOut
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# 1. Load dataset titanic.csv
dataset = pd.read_csv('titanic.csv')

# 2. Ambil fitur (Sex, Age, Pclass, Fare) dan lakukan imputasi missing values untuk Age
# Gunakan .loc untuk menghindari SettingWithCopyWarning
data = dataset.loc[:, ['Sex', 'Age', 'Pclass', 'Fare']]

# Konversi kolom 'Sex' menjadi numerik (0 untuk male, 1 untuk female)
data.loc[:, 'Sex'] = data['Sex'].map({'male': 0, 'female': 1})

# Imputasi missing values pada kolom 'Age' dengan mean berdasarkan 'Pclass'
data.loc[:, 'Age'] = data.groupby('Pclass')['Age'].transform(lambda x: x.fillna(x.mean()))

# 3. Ambil kolom kelas (Survived) sebagai label
label = dataset['Survived']
```

LOO

```
loo = LeaveOneOut()

# Variabel untuk menyimpan total akurasi
total_accuracy = 0
n_splits = loo.get_n_splits(data)  # Jumlah iterasi = jumlah baris dalam data

# 5. Iterasi melalui setiap fold LOO
for train_index, test_index in loo.split(data):
    # Split data train dan test berdasarkan index dari LOO
    train_data, test_data = data.iloc[train_index], data.iloc[test_index]
    train_label, test_label = label.iloc[train_index], label.iloc[test_index]

    # 6. Normalisasi train_data menggunakan Min-Max (0-1)
    scaler = MinMaxScaler()
    train_data_scaled = scaler.fit_transform(train_data)

    # Normalisasi test_data menggunakan skala yang sama dari train_data
    test_data_scaled = scaler.transform(test_data)

    # 7. Lakukan klasifikasi k-NN (k=3)
    knn = KNeighborsClassifier(n_neighbors=3)
    knn.fit(train_data_scaled, train_label)
```

LOO

```
    # 8. Prediksi data uji
    predictions = knn.predict(test_data_scaled)

    # 9. Hitung akurasi untuk fold ini
    accuracy = accuracy_score(test_label, predictions)
    total_accuracy += accuracy

# 10. Hitung akurasi rata-rata dan error ratio
average_accuracy = total_accuracy / n_splits
error_ratio = 1 - average_accuracy

# Tampilkan hasil
print(f"Average Accuracy: {average_accuracy:.2f}")
print(f"Error Ratio: {error_ratio:.2f}")
```

Kode ini memuat dataset Titanic, melakukan imputasi missing values, dan menerapkan Leave-One-Out Cross-Validation (LOO-CV) dengan model k-NN (k=3).

```
Average Accuracy: 0.83
Error Ratio: 0.17
```

Link google collab :
https://colab.research.google.com/drive/1VN59Qo6eONRaUuTgTx3kFhGfixv84CQc?usp=sharing