klasyfikacja

May 22, 2025

[1]:
```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
```

0.1 Wczytywanie danych

[2]:
```
column_names = [
    'Class',
    'Alcohol',
    'Malic acid',
    'Ash',
    'Alcalinity of ash',
    'Magnesium',
    'Total phenols',
    'Flavanoids',
    'Nonflavanoid phenols',
    'Proanthocyanins',
    'Color intensity',
    'Hue',
    'OD280/OD315 of diluted wines',
    'Proline'
]

data = pd.read_csv('Dane/wine.data')
data.columns = column_names
display(data.head(10))
```

	Class	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	\
0	1	13.20	1.78	2.14	11.2	100	
1	1	13.16	2.36	2.67	18.6	101	
2	1	14.37	1.95	2.50	16.8	113	
3	1	13.24	2.59	2.87	21.0	118	
4	1	14.20	1.76	2.45	15.2	112	
5	1	14.39	1.87	2.45	14.6	96	
6	1	14.06	2.15	2.61	17.6	121	
7	1	14.83	1.64	2.17	14.0	97	
8	1	13.86	1.35	2.27	16.0	98	

```
9      1    14.10        2.16  2.30               18.0        105

   Total phenols  Flavanoids  Nonflavanoid phenols  Proanthocyanins  \
0           2.65        2.76                  0.26             1.28
1           2.80        3.24                  0.30             2.81
2           3.85        3.49                  0.24             2.18
3           2.80        2.69                  0.39             1.82
4           3.27        3.39                  0.34             1.97
5           2.50        2.52                  0.30             1.98
6           2.60        2.51                  0.31             1.25
7           2.80        2.98                  0.29             1.98
8           2.98        3.15                  0.22             1.85
9           2.95        3.32                  0.22             2.38

   Color intensity   Hue  OD280/OD315 of diluted wines  Proline
0             4.38  1.05                          3.40     1050
1             5.68  1.03                          3.17     1185
2             7.80  0.86                          3.45     1480
3             4.32  1.04                          2.93      735
4             6.75  1.05                          2.85     1450
5             5.25  1.02                          3.58     1290
6             5.05  1.06                          3.58     1295
7             5.20  1.08                          2.85     1045
8             7.22  1.01                          3.55     1045
9             5.75  1.25                          3.17     1510
```

0.2 Podział na zbiór treningowy i testowy

[3]:
```
train_data, test_data = train_test_split(data, test_size=0.3, random_state=42)

X_train = train_data.drop('Class', axis=1)
Y_train = train_data['Class']

X_test = test_data.drop('Class', axis=1)
Y_test = test_data['Class']
```

0.3 Normalizacja danych

[4]:
```
scaler = StandardScaler()

scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

0.3.1 Normalizuje się po to aby:

- dane miały porównywalną skalę
- niektóre algorytmy są wrażliwe na skalę (np. K-NN) ### Wpływ na algorytmy

- **k-NN** - musi być normalizacja, gdyż algorytm działa na podstawie odległości między punktami
- **Random Forest** - nie jest wymagana normalizacja bo drzewa decyzyjne porównują wartość do liczby niezależnie od skali

0.4 Trening dla algorytmów KNeighborsClassifier oraz RandomForestClassifier

[5]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, Y_train)

rfc = RandomForestClassifier()
rfc.fit(X_train, Y_train)
```

[5]:
```
RandomForestClassifier()
```

0.5 Predykcja

[6]:
```
y_pred_knn = knn.predict(X_test)
y_pred_rfc = rfc.predict(X_test)
```

0.6 Metryki

- `Accuracy` - stosunek liczby poprawnych klasyfikacji do wszystkich przewidzianych, najlepiej działa, gdy klasy są w miarę zrównoważone liczebnie
- `Precision` - ile z przewidzianych jako dana klasa rzeczywiście do niej należy
  - przydatne, gdy ważne jest ograniczenie liczby fałszywych pozytywów
- `Recall` - ile z rzeczywistych przypadków danej klasy model wykrył
  - przydatne aby nie przeoczyć fałszywych negatywów
- `F-measure` - średnia harmoniczna `precision` i `recall`
  - dobry wskaźnik ogólnej jakości modelu, szczególnie przy niezrównoważonych klasach
- `Macierz konfuzji` - pokazuje, które klasy są najczęściej ze sobą mylone
  - Dobra do szybkiego wglądu w błędy modelu
- `Classification report` - zestawienie wszystkich powyższych metryk w tabeli, osobno dla każdej klasy

0.7 Analiza predykcji poszczególnych modeli

[7]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, classification_report, confusion_matrix

print("KNeighborsClassifier")
print("Accuracy = ", accuracy_score(Y_test, y_pred_knn))
print("Classification Report:\n", classification_report(Y_test, y_pred_knn))
print("Confusion Matrix:\n", confusion_matrix(Y_test, y_pred_knn))
```

```
KNeighborsClassifier
Accuracy =  0.9444444444444444
Classification Report:
               precision    recall  f1-score   support

           1       1.00      1.00      1.00        19
           2       1.00      0.86      0.92        21
           3       0.82      1.00      0.90        14

    accuracy                           0.94        54
   macro avg       0.94      0.95      0.94        54
weighted avg       0.95      0.94      0.94        54

Confusion Matrix:
 [[19  0  0]
 [ 0 18  3]
 [ 0  0 14]]
```

[8]:
```
print("\nRandomForestClassifier")
print("Accuracy = ", accuracy_score(Y_test, y_pred_rfc))
print("Classification Report:\n", classification_report(Y_test, y_pred_rfc))
print("Confusion Matrix:\n", confusion_matrix(Y_test, y_pred_rfc))
```

```
RandomForestClassifier
Accuracy =  0.9629629629629629
Classification Report:
               precision    recall  f1-score   support

           1       1.00      1.00      1.00        19
           2       1.00      0.90      0.95        21
           3       0.88      1.00      0.93        14

    accuracy                           0.96        54
   macro avg       0.96      0.97      0.96        54
weighted avg       0.97      0.96      0.96        54

Confusion Matrix:
 [[19  0  0]
 [ 0 19  2]
 [ 0  0 14]]
```

0.8 Interpretacja analizy

Oba modele osiągnęły bardzo zbliżone wyniki. Analiza macierzy pomyłek pokazuje, że oba klasyfikatory miały trudności jedynie z poprawnym rozpoznaniem drugiego rodzaju wina. Klasyfikator oparty na drzewie decyzyjnym poradził sobie z tym nieco lepiej. Możliwe, że różnica ta wynika z różnych rozmiarów modeli, tzn. po zwiększeniu liczby

sąsiadów w KNeighbors oba modele osiągnęły taką samą skuteczność.