lab10

May 20, 2024

1 Uczenie nadzorowane - klasyfikacja

[1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, confusion_matrix, classification_report
```

[2]:
```
colnames = ['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("wine.data", names=colnames, index_col=False)
display(data)
```

```
<ipython-input-2-24600e2d093f>:2: ParserWarning: Length of header or names does
not match length of data. This leads to a loss of data with index_col=False.
  data = pd.read_csv("wine.data", names=colnames, index_col=False)
```

	Alcohol	Malic_acid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	\
0	1	14.23	1.71	2.43	15.6	127	
1	1	13.20	1.78	2.14	11.2	100	
2	1	13.16	2.36	2.67	18.6	101	
3	1	14.37	1.95	2.50	16.8	113	
4	1	13.24	2.59	2.87	21.0	118	
..	…	…	…	…	…	…	
173	3	13.71	5.65	2.45	20.5	95	
174	3	13.40	3.91	2.48	23.0	102	
175	3	13.27	4.28	2.26	20.0	120	
176	3	13.17	2.59	2.37	20.0	120	
177	3	14.13	4.10	2.74	24.5	96	

	Flavanoids	Nonflavanoid_phenols	Proanthocyanins	Color_intensity	\
0	2.80	3.06	0.28	2.29	
1	2.65	2.76	0.26	1.28	
2	2.80	3.24	0.30	2.81	

```
3          3.85                  3.49             0.24             2.18
4          2.80                  2.69             0.39             1.82
..          …                     …                …                …
173        1.68                  0.61             0.52             1.06
174        1.80                  0.75             0.43             1.41
175        1.59                  0.69             0.43             1.35
176        1.65                  0.68             0.53             1.46
177        2.05                  0.76             0.56             1.35

       Hue  OD280/OD315_of_diluted wines  Proline
0     5.64                          1.04     3.92
1     4.38                          1.05     3.40
2     5.68                          1.03     3.17
3     7.80                          0.86     3.45
4     4.32                          1.04     2.93
..     …                            …        …
173   7.70                          0.64     1.74
174   7.30                          0.70     1.56
175  10.20                          0.59     1.56
176   9.30                          0.60     1.62
177   9.20                          0.61     1.60

[178 rows x 13 columns]
```

[3]:
```
display(data.info())
display(data.describe().T)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 13 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Alcohol                       178 non-null    int64
 1   Malic_acid                    178 non-null    float64
 2   Ash                           178 non-null    float64
 3   Alcalinity_of_ash             178 non-null    float64
 4   Magnesium                     178 non-null    float64
 5   Total_phenols                 178 non-null    int64
 6   Flavanoids                    178 non-null    float64
 7   Nonflavanoid_phenols          178 non-null    float64
 8   Proanthocyanins               178 non-null    float64
 9   Color_intensity               178 non-null    float64
 10  Hue                           178 non-null    float64
 11  OD280/OD315_of_diluted wines  178 non-null    float64
 12  Proline                       178 non-null    float64
dtypes: float64(11), int64(2)
memory usage: 18.2 KB
```

```
None
```

	count	mean	std	min	25%	\
Alcohol	178.0	1.938202	0.775035	1.00	1.0000	
Malic_acid	178.0	13.000618	0.811827	11.03	12.3625	
Ash	178.0	2.336348	1.117146	0.74	1.6025	
Alcalinity_of_ash	178.0	2.366517	0.274344	1.36	2.2100	
Magnesium	178.0	19.494944	3.339564	10.60	17.2000	
Total_phenols	178.0	99.741573	14.282484	70.00	88.0000	
Flavanoids	178.0	2.295112	0.625851	0.98	1.7425	
Nonflavanoid_phenols	178.0	2.029270	0.998859	0.34	1.2050	
Proanthocyanins	178.0	0.361854	0.124453	0.13	0.2700	
Color_intensity	178.0	1.590899	0.572359	0.41	1.2500	
Hue	178.0	5.058090	2.318286	1.28	3.2200	
OD280/OD315_of_diluted wines	178.0	0.957449	0.228572	0.48	0.7825	
Proline	178.0	2.611685	0.709990	1.27	1.9375	

	50%	75%	max
Alcohol	2.000	3.0000	3.00
Malic_acid	13.050	13.6775	14.83
Ash	1.865	3.0825	5.80
Alcalinity_of_ash	2.360	2.5575	3.23
Magnesium	19.500	21.5000	30.00
Total_phenols	98.000	107.0000	162.00
Flavanoids	2.355	2.8000	3.88
Nonflavanoid_phenols	2.135	2.8750	5.08
Proanthocyanins	0.340	0.4375	0.66
Color_intensity	1.555	1.9500	3.58
Hue	4.690	6.2000	13.00
OD280/OD315_of_diluted wines	0.965	1.1200	1.71
Proline	2.780	3.1700	4.00

1.0.1 Podział danych - funkcja train_test_split

[4]:
```
X = data.drop("Alcohol", axis=1)
y = data["Alcohol"]

# Podział na zestawy treningowy i testowy
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

1.0.2 Normalizacja danych

[5]:
```
scaler = StandardScaler()
X_train_norm = scaler.fit_transform(X_train)
X_test_norm = scaler.transform(X_test)
```

[6]:
```
print(np.std(X_train_norm, axis=0))
print(np.std(X_test_norm, axis=0))
print(np.mean(X_train_norm, axis=0))
print(np.mean(X_test_norm, axis=0))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
[0.94549584 0.87015188 0.90975138 0.82554542 0.84626227 0.90709006
 0.96978666 0.81939439 0.8946222  0.97650956 0.8806474  0.91019964]
[-3.44012768e-16  1.93898106e-16 -4.06560544e-17  1.93898106e-16
 -3.94050989e-16  1.65751606e-16  1.02421983e-16  3.00229325e-16
 -1.87643328e-17 -7.50573312e-17  2.97101936e-16  2.06407661e-16]
[ 0.13028396 -0.16124142  0.10079272  0.03117293 -0.23778426  0.04690163
  0.13419196 -0.23884446 -0.14562457  0.00103084  0.02266159  0.12964851]
```

Wykonana normalizacja zapewniła przeskalowanie danych. Był to etap ważny dla wykorzystywanych algorytmów, gdyż algorytmy te działają na odległościach (dystansach). Zapewniło to, że wszystkie cechy zostaną równo wykorzystane do modelu. Niweluje to dominacje i pozwala uzyskać lepsze klastry.

1.0.3 Trening

[7]:
```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_norm, y_train)
```

[7]:
```
KNeighborsClassifier(n_neighbors=3)
```

[8]:
```
rf = RandomForestClassifier(random_state=0)
rf.fit(X_train_norm, y_train)
```

[8]:
```
RandomForestClassifier(random_state=0)
```

1.0.4 Predykcja

[9]:
```
y_pred_knn = knn.predict(X_test_norm)
y_pred_rf = rf.predict(X_test_norm)
```

1.0.5 Analiza metryk klasyfikacji

1) ACCURACY SCORE Określa stosunek dobrze zakwalifikowanych przypadków do wszystkich przypadków. Przewidziane etykiety muszą dokładnie pasować do prawidłowych etykiet. Nie jest to opytmalna metryka, ale początkowo najlepiej pozwalająca stwierdzić skuteczność
2) PRECISION Określa stosunek tp / (tp + fp), gdzie tp oznacza liczbę dobrze przypisanych obserwacji do klasy, a fp to fałszywie pozytywne wyniki, czyli te źle przypisane.
3) RECALL Określa stosunek tp / (tp + fn), gdzie tp oznacza liczbę dobrze przypisanych obserwacji do klasy, a fn to false negative, czyli niepoprawnie zaklasyfikowanych wartości.

4) F-measures, inaczej F1-score lub F-score Jest to średnia harmoniczna z dwóch powyższych wartości określona wzorem **precision*recall/(precision+recall)*, ***czyli*** *2*tp / 2*tp+fp+fn**. Jest to wartość która łączy w sobie określenie precyzyjności modelu oraz czułości - uwzględnia obie te wartości i uzyskujemy jedną liczbę.

5) CONFUSION_MATRIX W niej dostaniemy informację o wszystkich wcześniej używanych określeniach, czyli:

- C[0,0] to fn (true negatives)
- C[1,0] to tp (false negatives)
- C[0,1] to fp (false positives)
- C[1,1] to tn (true positives) Pokazuje ona wszystkie możliwe opcje, czy klasyfikator dobrze przypisał wartości. W przypadku wielu klas (jak w tym przypadku) wartości te liczone są dla każdej klasy osobno i rozmiar macierzy się zmienia.

6) CLASSIFICATION_REPORT Pozwala zobaczyć głowne metryki dotyczące wykonanej klasyfikacji, w tym precision, recall, F1 score i support (liczba wystąpień każdej klasy w zbiorze testowym)

1.0.6 Obliczenie metryk

[10]:
```
# Metryki dla KNeighborsClassifier
print("KNeighborsClassifier Metrics:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_knn):.2f}")
print(f"Precision: {precision_score(y_test, y_pred_knn, average='weighted'):.
 ↪2f}")
print(f"Recall: {recall_score(y_test, y_pred_knn, average='weighted'):.2f}")
print(f"F1 Score: {f1_score(y_test, y_pred_knn, average='weighted'):.2f}")
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred_knn))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_knn))

# Metryki dla RandomForestClassifier
print("\nRandomForestClassifier Metrics:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf):.2f}")
print(f"Precision: {precision_score(y_test, y_pred_rf, average='weighted'):.
 ↪2f}")
print(f"Recall: {recall_score(y_test, y_pred_rf, average='weighted'):.2f}")
print(f"F1 Score: {f1_score(y_test, y_pred_rf, average='weighted'):.2f}")
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred_rf))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_rf))
```

```
KNeighborsClassifier Metrics:
Accuracy: 0.94
Precision: 0.95
Recall: 0.94
```

```
F1 Score: 0.94

Confusion Matrix:
[[14  0  0]
 [ 1 12  1]
 [ 0  0  8]]

Classification Report:
              precision    recall  f1-score   support

           1       0.93      1.00      0.97        14
           2       1.00      0.86      0.92        14
           3       0.89      1.00      0.94         8

    accuracy                           0.94        36
   macro avg       0.94      0.95      0.94        36
weighted avg       0.95      0.94      0.94        36


RandomForestClassifier Metrics:
Accuracy: 0.97
Precision: 0.97
Recall: 0.97
F1 Score: 0.97

Confusion Matrix:
[[14  0  0]
 [ 1 13  0]
 [ 0  0  8]]

Classification Report:
              precision    recall  f1-score   support

           1       0.93      1.00      0.97        14
           2       1.00      0.93      0.96        14
           3       1.00      1.00      1.00         8

    accuracy                           0.97        36
   macro avg       0.98      0.98      0.98        36
weighted avg       0.97      0.97      0.97        36
```

1.1 Interpretacja wyników

Model KNeighborsClassifier Wszystkie metryki mają wysokie wartości, są widoczne pomiędzy nimi pewne różnice. Model poprawnie klasyfikuje 94% próbek, natomiast pozytywnie identyfikuje aż 95% pozytywnych przypadków. Na podstawie macierzy można stwierdzić że model najczęściej myli się z próbkami 2 klasy. #### Model RandomForestClassifier RandomForestClassifier os-

iąga niewiele, ale jednak lepsze wyniki. Model w przypadku 97% próbek dokonuje prawidłowej klasyfikacji. W confusion matrix można zauważyć, że w przypadku RandomForestClassifier, jedna próbka, która w KNN została określona jako 3 klasy, w tym przypadku została poprawnie zaklasyfikowana jako klasa 2.

W przypadku tego zbioru danych lepiej sprawdził się algorytm RandomForest - okazał się on efektywniejszy w klasyfikacji typu wina na podstawie reszty parametrów. Różnice te mogą wynikać przykładowo z faktu, że drugi algorytm lepiej radzi sobie z nieliniowymi zależnosciami i lepiej się sprawdza ze skalowaniem danych.

[ ]: