Wykład Specjalizujący

Zadanie 3
KNN

Autor:

Jerzy Dębowski 151266

Import modułów z biblioteki sklearn

Do niezbędnego działania programu potrzebne są następujące moduły z biblioreki sklearn:

- datasets
- train_test_split
- KNeighborsClassifier
- classification_report
- accuracy_score

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

Załadowanie zbiorów danych

Następnie definiujemy zbiory win oraz irysów przy pomocy modułu datasets:

```
iris = datasets.load_iris()
wine = datasets.load_wine()
```

Fukcja testująca

Funkcja testująca knn oblicza odległości (euklidesowe, manhattan oraz chebyshev) dla paramterów liczby sąsiadów (w tym przypadku 1, 2 oraz 3).

Dane dzielone są na zbiory treningowe oraz testowe przy pomocy funkcji train_test_split. Następnie tworzony jest obiekt klasy KNeighboursClassifier z atrybutami k (ilość sąsiadów) oraz distance (typ odległości)

Następnie następuje dopasowanie obiektu do zbiorów treningowych oraz obliczana jest dokładność oraz generowany jest raport klasyfikacji. Wynik dla danej liczby sąsiadów oraz metryki odkległości zapisywany jest do słownika results

```
2 usages  jerzy
def test_knn(data, target, k_values, distances):
    results = {}
    for k in k_values:
        for distance in distances:
            X_train, X_test, y_train, y_test = train_test_split( *arrays: data, target, test_size=0.3, random_state=42)

            knn = KNeighborsClassifier(n_neighbors=k, metric=distance)
            knn.fit(X_train, y_train)

            predictions = knn.predict(X_test)
            accuracy = accuracy_score(y_test, predictions)
            report = classification_report(y_test, predictions)

            results[(k, distance)] = accuracy, report

    return results
```

Działanie oraz wynik programu

1. Definiujemy listę ilości najbliższych sąsiadów oraz listę metryk odległości
2. Uruchamiamy funkcję testową dla win oraz irysów
3. Szukamy najwyższych wartości dla wyników

```
k_values = [1, 2, 3]
distances = ['euclidean', 'manhattan', 'chebyshev']

iris_results = test_knn(iris.data, iris.target, k_values, distances)
wine_results = test_knn(wine.data, wine.target, k_values, distances)


best_iris = max(iris_results, key=iris_results.get)
best_wine = max(wine_results, key=wine_results.get)

print(best_iris)
print(iris_results[best_iris])


print(best_wine)
print(wine_results[best_wine])
```

```
(1, 'euclidean')
(1.0, '              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00        19\n           1       1.00      1.00      1.00        13\n           2       1.00      1.00      1.
(1, 'manhattan')
(0.8518518518518519, '              precision    recall  f1-score   support\n\n           0       0.85      0.89      0.87        19\n           1       0.89      0.81      0.85        21\n           2       0.80
```