

Kiểm tra học máy

Họ và tên: Nguyễn Viết Phương
MSV: 2251172460

Bài tập 1: Phân loại hoa Iris Mô tả:


Sử dụng tập dữ liệu Iris (có sẵn trong thư viện scikit-learn) để xây dựng một mô hình phân loại nhằm dự đoán loại hoa Iris dựa trên các đặc trưng của nó.

Yêu cầu:

1. Tải tập dữ liệu Iris và chia thành hai phần: tập huấn luyện (80%) và tập kiểm tra (20%).

a) Tải tập dữ liệu Iris

- Tải tập dữ liệu hoa Iris có sẵn trong thư viện sklearn
- Đặt biến X là dữ liệu không bao gồm nhãn
- Đặt y cho dữ liệu nhãn là target



```
1 iris = load_iris()
2 X = iris.data
3 y = iris.target
```

```
5.1, 3.5, 1.4, 0.2
4.9, 3. , 1.4, 0.2
4.7, 3.2, 1.3, 0.2
4.6, 3.1, 1.5, 0.2
5. , 3.6, 1.4, 0.2
5.4, 3.9, 1.7, 0.4
4.6, 3.4, 1.4, 0.3
5. , 3.4, 1.5, 0.2
4.4, 2.9, 1.4, 0.2
4.9, 3.1, 1.5, 0.1
5.4, 3.7, 1.5, 0.2
4.8, 3.4, 1.6, 0.2
4.8, 3. , 1.4, 0.1
4.3, 3. , 1.1, 0.1
5.8, 4. , 1.2, 0.2
5.7, 4.4, 1.5, 0.4
5.4, 3.9, 1.3, 0.4
5.1, 3.5, 1.4, 0.3
5.7, 3.8, 1.7, 0.3
5.1, 3.8, 1.5, 0.3
5.4, 3.4, 1.7, 0.2
5.1, 3.7, 1.5, 0.4
4.6, 3.6, 1. , 0.2
5.1, 3.3, 1.7, 0.5
4.8, 3.4, 1.9, 0.2
5. , 3. , 1.6, 0.2
5. , 3.4, 1.6, 0.4
5.2, 3.5, 1.5, 0.2
```

b) Chia thành hai phần: tập huấn luyện (80%) và tập kiểm tra (20%).

```
1 Xtrain, xtest, ytrain, ytest = train_test_split(
2     X, y, test_size=0.2, random_state=35)
3
```

- Chia 80% train , 20% cho test, đặt random_state = 35 để tránh overfitting

2. Tiền xử lý dữ liệu nếu cần thiết

- Vì tập dữ liệu trong load_iris trong thư viện sklearn đã sạch nên không cần phải xử lý

3. Áp dụng thuật toán K-Nearest Neighbors (KNN) để xây dựng mô hình. Thử nghiệm với các giá trị khác nhau của tham số k.

a) Tạo lớp KNstNeis


```
1 import numpy as np
2
3
4 class KNstNeis:
5     def __init__(self, n_neighbors=5, weights='uniform', algorithm='auto', metric='euclidean'):
6         self.n_neighbors = n_neighbors
7         self.weights = weights
8         self.algorithm = algorithm
9         self.metric = metric
10
11     def fit(self, X, y):
12         self.Xtrain = X
13         self.ytrain = y
14
15     def khCach(self, x):
16         if self.metric == 'euclidean':
17             kc = np.linalg.norm(self.Xtrain - x, axis=1)
18         else:
19             raise ValueError(
20                 "Chưa có chỉ số!")
21         return kc
22
23     def dudoan(self, X):
24         dd = []
25         for x in X:
26             kc = self.khCach(x)
27             csoGan = np.argsort(kc)[:self.n_neighbors]
28             nhanGan = self.ytrain[csoGan]
29
30             if self.weights == 'uniform':
31                 most = np.bincount(nhanGan).argmax()
32             elif self.weights == 'kc':
33                 # Avoid division by zero
34                 weights = 1 / (kc[csoGan] + 1e-5)
35                 ts = np.zeros(len(np.unique(self.ytrain)))
36                 for i, label in enumerate(nhanGan):
37                     ts[label] += weights[i]
38                 most = np.argmax(ts)
39             else:
40                 raise ValueError(
41                     "Weights phải 'uniform' hoặc 'kc'.")
42
43             dd.append(most)
44         return np.array(dd)
```

b) Áp dụng thuật toán K-Nearest Neighbors (KNN) để xây dựng mô hình



```
1 def kn_model(k):
2     kn = KNstNeis(n_neighbors=k)
3     kn.fit(Xtrain, ytrain)
4     ydd = kn.dudoan(xtest)
5     return accuracy_score(ytest, ydd)
```

c) Thử nghiệm với các giá trị khác nhau của tham số k
- Cho k lặp từ 1 đến 50 để xem các kết quả khả quan



```
1 k_val = range(1, 55)
2 accs = [kn_model(k) for k in k_val]
```

- Lặp qua các tham số k và in ra kết quả để biết xem từng tham số k thì kết quả dự đoán sẽ ra sao



```
1 for k, acc in zip(k_val, accs):  
2     print(f'k = {k}: Độ chính xác = {acc:.3f}')
```

4. Đánh giá mô hình trên tập kiểm tra và báo cáo độ chính xác

- **k = 1:** Độ chính xác = 0.967

→ Kết quả dự đoán khá tốt, tuy nhiên vẫn còn sai số, có thể overfitting nhẹ khi mô hình chỉ sử dụng điểm gần nhất để phân loại.

- **k = 2:** Độ chính xác = 1.000

→ Kết quả dự đoán rất tốt, với độ chính xác hoàn toàn. Tuy nhiên, việc đạt được độ chính xác 1.0 có thể là dấu hiệu của overfitting, khi mô hình khớp quá mức với dữ liệu huấn luyện.

- **k = 3:** Độ chính xác = 0.967

→ Kết quả dự đoán khá tốt, giống như với $k=1$, nhưng có sự giảm nhẹ trong độ chính xác, do việc sử dụng nhiều điểm gần nhất có thể làm giảm tính đặc trưng của dữ liệu.

- **k = 4, 5:** Độ chính xác = 0.967

→ Tương tự như $k=3$, độ chính xác duy trì ở mức khá, không có sự cải thiện lớn từ việc tăng số lượng lân cận.

- **k = 6:** Độ chính xác = 1.000

→ Độ chính xác hoàn toàn, tương tự như với $k=2$, có thể mô hình vẫn overfitting nhưng mức độ tốt hơn.

- **k = 7-16:** Độ chính xác = 1.000

→ Mô hình duy trì độ chính xác rất tốt với các giá trị k từ 7 đến 16, cho thấy mô hình hoạt động ổn định trong khoảng giá trị này và đạt độ chính xác hoàn hảo. Tuy nhiên, cần kiểm tra lại với dữ liệu mới để đảm bảo không có overfitting.

- **k = 17-19:** Độ chính xác = 0.967

→ Hiệu suất của mô hình giảm nhẹ, tuy nhiên vẫn ở mức tốt.

- **k = 20-36:** Độ chính xác = 1.000

→ Hiệu suất lại đạt độ chính xác tối đa, cho thấy đây là khoảng giá trị k tối ưu cho mô hình.

- **k = 37-43:** Độ chính xác = 1.000

→ Độ chính xác vẫn duy trì ở mức tốt, tuy nhiên có thể hiệu suất sẽ giảm với các giá trị lớn hơn.

- **k = 44:** Độ chính xác = 0.967

→ Hiệu suất bắt đầu giảm, độ chính xác không còn ở mức tối đa, cho thấy k quá lớn có thể làm giảm khả năng phân loại chính xác của mô hình.

- **k = 45-46:** Độ chính xác = 0.933

→ Hiệu suất giảm đáng kể, chỉ còn 0.933, cho thấy mô hình không còn hoạt động tốt khi k quá lớn.

- **k = 47-49**: Độ chính xác = 0.967

→ Hiệu suất cải thiện nhẹ, nhưng không đạt mức tốt nhất.

- **k = 50-51**: Độ chính xác = 0.933

→ Mô hình lại gặp khó khăn trong việc phân loại, độ chính xác giảm xuống 0.933.

- **k = 52**: Độ chính xác = 0.967

→ Hiệu suất tăng nhẹ nhưng vẫn không bằng các giá trị kkk tối ưu.

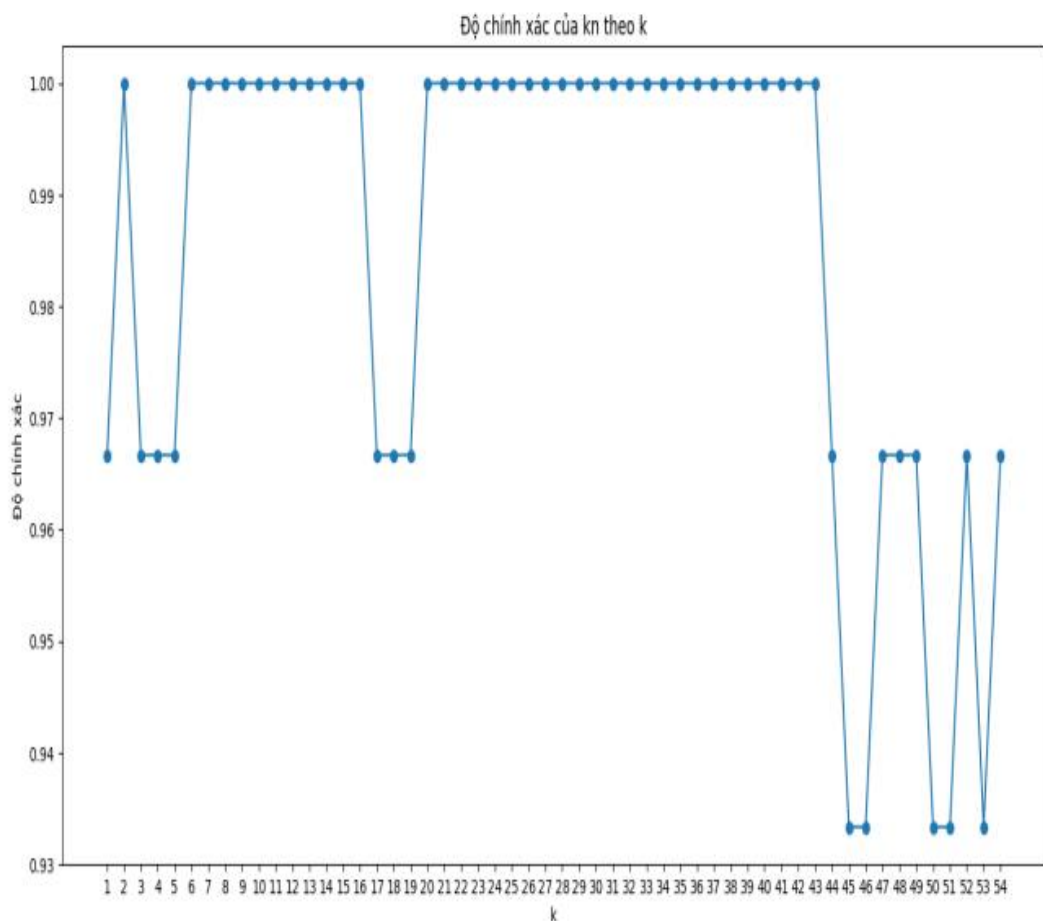
- **k = 53**: Độ chính xác = 0.933

→ Hiệu suất giảm đáng kể với kkk quá lớn, cho thấy kkk này không phù hợp.

- **k = 54**: Độ chính xác = 0.967

→ Mô hình cải thiện một chút, nhưng không phải là kết quả tối ưu.

5. Vẽ biểu đồ so sánh độ chính xác của mô hình với các giá trị khác nhau của k



6. Giải thích về sự lựa chọn giá trị k tốt nhất.

- Với kết quả dự đoán trên kết hợp với sơ đồ , tham số của k cho kết quả tốt nhất và tránh overfitting là $k = 1, 4, 3, 5, 17, 18, 19, 44, 47, 48, 49$
- Với các kết quả k cho kết quả bằng 1 thì không nên vì mô hình dự đoán quá tốt thành ra overfitting
- Với các kết quả k gần 50 cho kết quả không tốt dưới 0.933 với 1 số tham số k