AI VIET NAM – COURSE 2023

Project: K-Nearest Neighbors and Decision Tree

Ngày 31 tháng 8 năm 2023

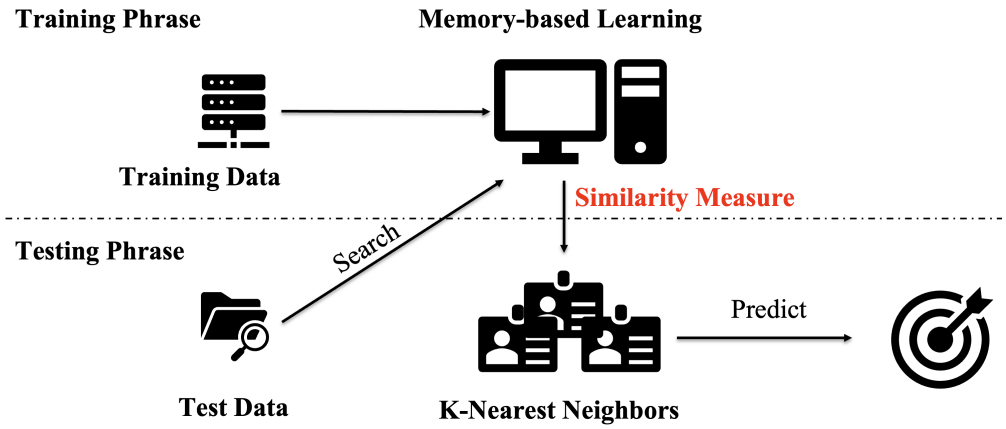
1. K-Nearest Neighbors

K-Nearest Neighbors (KNN) là một trong những thuật toán học máy có giám sát cơ bản. KNN còn được gọi là Lazy Learning, Memory-Based Learning,... Với bước huấn luyện mô hình, chỉ đơn giản là lưu trữ lại giá trị của dữ liệu huấn luyện, vì vậy KNN là phương pháp học máy không tham số (Non-Parametric). Ở bước dự đoán, mô hình sẽ dử dụng các độ đo khoảng cách để tìm các hàng xóm lân cận.

Một số độ đo thường được sử dụng trong KNN như:

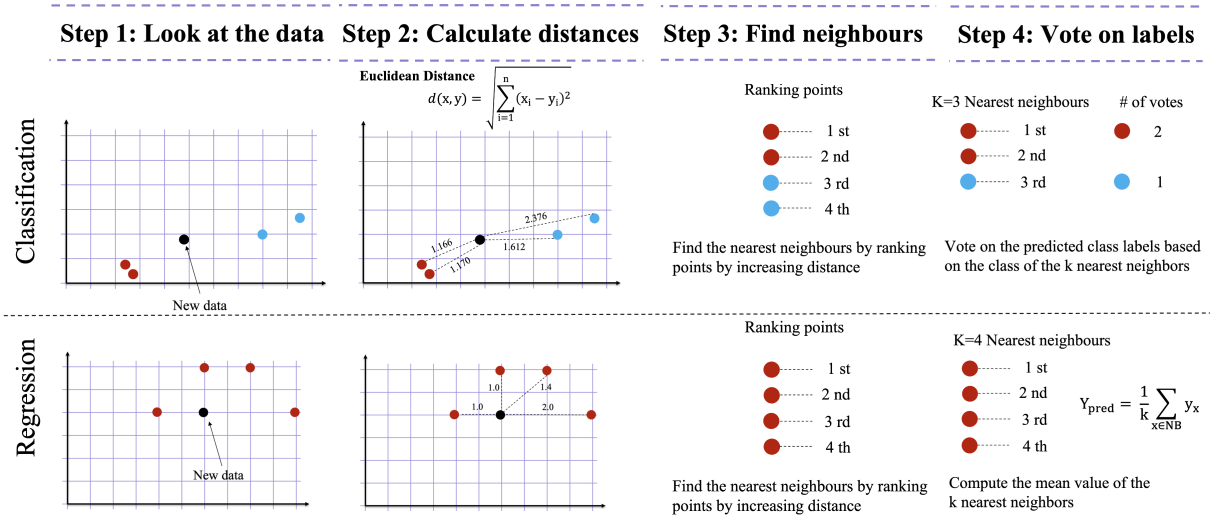
1. Euclidean
2. Chebyshev
3. Manhattan
4. Minkowski

Các độ đo này tính toán dữ liệu dự đoán với các điểm dữ liệu khác được lưu trữ trong mô hình huấn luyện. Từ đó xếp hạng và tìm ra K điểm dữ liệu huấn luyện có kết quả gần với dữ liệu dự đoán nhất. Cuối cùng dựa vào phương pháp biểu quyết của các dữ liệu hàng xóm trong tập huấn luyện để đưa ra kết quả dự đoán.



Hình 1: Mô tả quá trình huấn luyện và dự đoán mô hình KNN.

KNN được sử dụng rộng rãi cho ứng dụng phân loại (Classification) và dự đoán (Regression) được mô tả như hình sau:



Hình 2: Mô tả quá trình huấn luyện và dự đoán mô hình KNN.

Câu hỏi 1 Hàm đánh giá nào trong các hàm sau đây không được sử dụng trong KNN?
a) Manhattan
b) Minkowski
c) Tanimoto
d) ROUGE

Câu hỏi 2 Cho hàm tính khoảng cách sau đây:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Hàm tính độ đo khoảng cách trên được gọi là:
a) Manhattan
b) Minkowski
c) Eucidean
d) Tanimoto

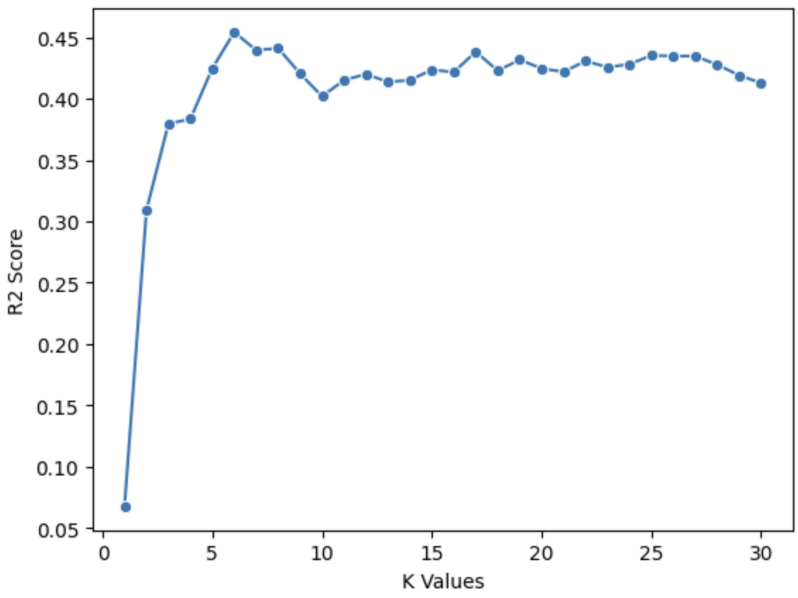
Câu hỏi 3 Hoàn thành đoạn code sau đây để được thứ tự đúng cho bài toán phân loại hoa Iris dử dụng mô hình KNN?

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

# Load the diabetes dataset
iris_X, iris_y = datasets.load_iris(return_X_y=True)

# Split train:test = 8:2
X_train, X_test, y_train, y_test = train_test_split(
```

```
    iris_X,
    iris_y,
    test_size=0.2,
    random_state=42
)

# Scale the features using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Build KNN Classifier
*** Your code here ***

# Predict and Evaluate test set
y_pred = knn_classifier.predict(X_test)
accuracy_score(y_test, y_pred)
```

```
a)
knn_classifier = KNeighborsClassifier(n_neighbors=5)
knn_classifier.fit(X_train, y_train)

b)
knn_classifier = KNeighborsRegessor(n_neighbors=5)
knn_classifier.fit(X_train, y_train)

c)
knn_classifier = KNeighborsModel(n_neighbors=5)
knn_classifier.fit(X_train, y_train)

d)
knn_classifier = KNeighbors(n_neighbors=5)
knn_classifier.fit(X_train, y_train)
```

Câu hỏi 4 Quan sát hình dưới về ảnh hưởng của giá trị K đến độ đo R_2Score.



Hình 3: Ảnh hưởng của giá trị K đến độ đo R_2Score.

Giá trị K nào trong các giá trị sau đây là tốt nhất cho mô hình KNN ở trong hình 5?
a) 10
b) 7
c) 17
d) 4

Câu hỏi 5 Sắp xếp các đoạn code sau đây để được thứ tự đúng cho bài toán dự đoán chỉ số bệnh trên bô dữ liệu *Diabetes* dử dụng mô hình KNN?

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor

Paragraph A:
# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

Paragraph B:
# Scale the features using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

Paragraph C:
# Build KNN model
knn_regressor = KNeighborsRegressor(n_neighbors=5)
knn_regressor.fit(X_train, y_train)

Paragraph D:
# Split train:test = 8:2
X_train, X_test, y_train, y_test = train_test_split(
    diabetes_X,
    diabetes_y,
    test_size=0.2,
    random_state=42
)
```

Thứ tự đúng của các đoạn trên là:
a) D - C - B - A
b) A - D - B - C
c) C - B - A - D
d) B - C - D - A

Câu hỏi 6 Phương pháp nào không được sử dùng để biểu diễn văn bản thành các giá trị vector là:
a) Bag-of-Word
b) TF-IDF
c) One Hot Encoding
d) F-Score

Câu hỏi 7 Hoàn thành đoạn code sau đây cho bài toán phân loại văn bản sử dụng mô hình KNN trên bộ dữ liệu đánh giá phim:

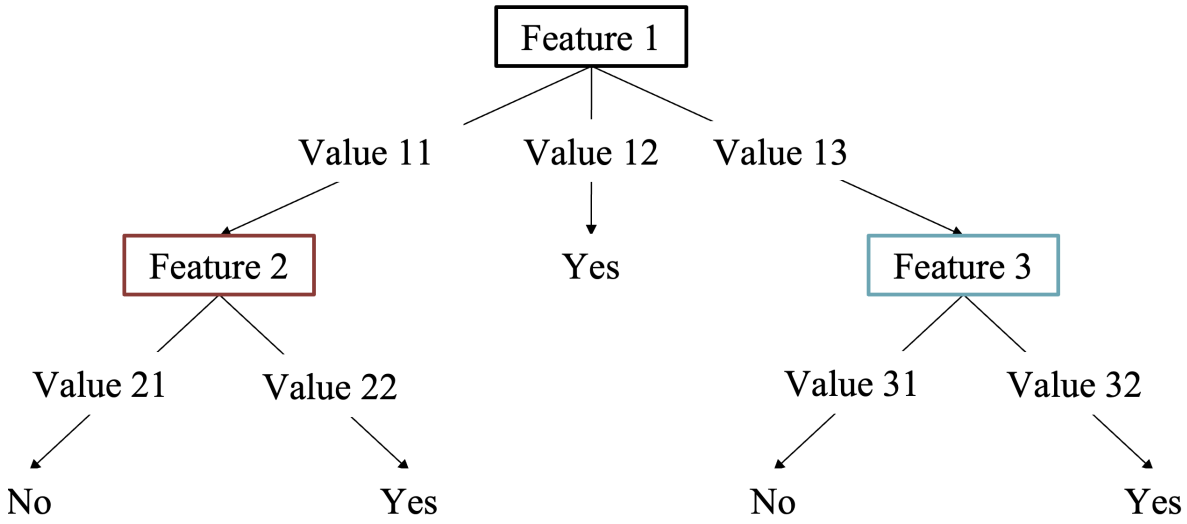
```
# Import library
!pip install -q datasets
import numpy as np
from datasets import load_dataset
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction.text import CountVectorizer

# Load IMDB dataset
imdb = load_dataset("imdb")
imdb_train, imdb_test = imdb['train'], imdb['test']

# Convert text to vector using BoW
*** Your code here ***
vectorizer =
X_train =
X_test =

y_train = np.array(imdb_train['label'])
y_test = np.array(imdb_test['label'])

# Scale the features using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Build KNN Classifier
knn_classifier = KNeighborsClassifier(n_neighbors=1, algorithm='ball_tree')
knn_classifier.fit(X_train, y_train)

# predict test set and evaluate
y_pred = knn_classifier.predict(X_test)
accuracy_score(y_test, y_pred)
```

Chọn đáp án đúng trong các đáp án sau đây:

```
a)
vectorizer = CountVectorizer(max_features=1000)
X_train = vectorizer.fit_transform(imdb_train['text']).toarray()
X_test = vectorizer.transform(imdb_test['label']).toarray()

b)
vectorizer = CountVectorizer(max_features=1000)
X_train = vectorizer.fit_transform(imdb_train['label']).toarray()
X_test = vectorizer.transform(imdb_test['text']).toarray()

c)
vectorizer = CountVectorizer(max_features=1000)
X_train = vectorizer.fit_transform(imdb_train['text']).toarray()
X_test = vectorizer.transform(imdb_test['text']).toarray()

d)
vectorizer = CountVectorizer(max_features=1000)
X_train = vectorizer.fit_transform(imdb_train['text']).toarray()
X_test = vectorizer.transform(imdb_test['label']).toarray()
```

2. Decision Tree

Decison Tree là phương pháp học có giám sát dựa vào ý tưởng xây dựng mô hình dạng cây để xấp xỉ dữ liệu huấn luyện. Cấu trúc cây quyết định được mô tả như hình sau:



Hình 4: Mô hình biểu diễn cây quyết định.

Để xây dựng cây quyết định trên, chúng ta thường sử dụng thuật toán Iterative Dichotomiser (ID3) tính toán độ đo Information Gain để xem xét đặc trưng nào sẽ là đầu tiên. Công thức tính Entropy với tập dữ liệu S ví dụ tương ứng với C lớp được tính như sau:

$$E(S) = -\sum_{c \in C} p_c log_2 p_c$$

Trong đó p_c là phân phối xác suất của lớp c trong bộ dữ liệu S

Informarmation Gain (IG) cho biết độ quan trọng của feature F trên bộ dữ liệu S được tính như sau:

$$Gain(F, S) = E(S) - \sum_{f \in F} \frac{|S_f|}{|S|} E(S_f)$$

Trong đó, S_f là tập hợp các ví dụ của bộ dữ liệu S ứng với đặc trưng F mà có giá trị là f

Câu hỏi 8 Giả sử cho bài toán bộ dữ liệu S gồm các ví dụ được phân loại thành 2 lớp. Độ đo Entropy bằng 1 khi nào?
a) Số dữ liệu trong hai lớp bằng nhau
b) số dữ liệu trong lớp 1 bằng 3 lần số dữ liệu trong lớp 2
c) số dữ liệu trong lớp 1 bằng 2 lần số dữ liệu trong lớp 2
d) số dữ liệu trong lớp 1 bằng 4 lần số dữ liệu trong lớp 2

Cho độ dữ liệu phân loại sau:

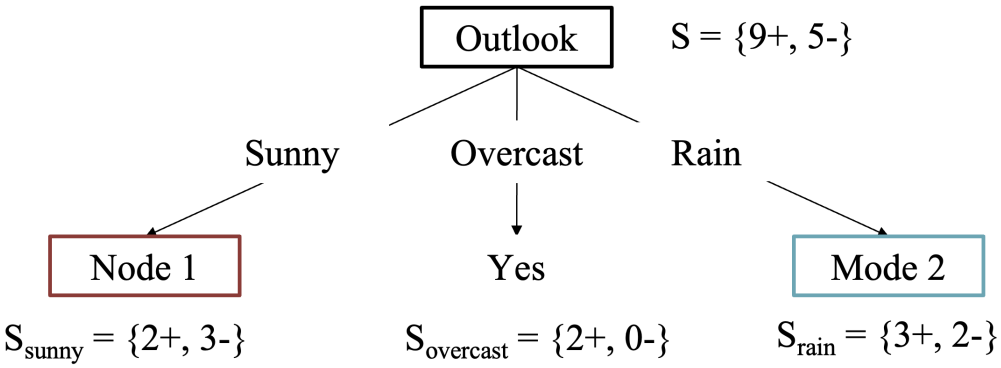
PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Hình 5: Dữ liệu dự đoán khả năng chơi Tennis.

Câu hỏi 9 Dựa vào công thức tính Information Gain, tính giá trị Gain(S, Wind)?
a) 0.048
b) 0.008
c) 0.448
d) 0.408

Câu hỏi 10 Từ bộ dữ liệu trên, node gốc được chọn sẽ là Outlook, vậy để tìm đặc trưng nào tại node 1 trong hình sau chúng ta cần tính giá trị Gain($S_{sunny}, Temperature$), Gain($S_{sunny}, Humidity$), Gain($S_{sunny}, Wind$)



Hình 6: Dữ liệu dự đoán khả năng chơi Tennis.

Các giá trị Gain($S_{sunny}, Temperature$), Gain($S_{sunny}, Humidity$), Gain($S_{sunny}, Wind$) lần lượt là:

a) 0.019, 0.97, 0.57
b) 0.57, 0.97, 0.019
c) 0.019, 0.57, 0.97
d) 0.97, 0.019, 0.97

Câu hỏi 11 Đặc trưng nào phù hợp tại node 1 là
a) Temperate
b) Humidity
c) Wind
d) Không có đặc trưng nào

Câu hỏi 12 Sắp xếp các đoạn code phân loại trên bộ dữ liệu Iris dựa vào Decision Tree:

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier

Paragraph A:
# Scale the features using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

Paragraph B:
# Define model
dt_classifier = DecisionTreeClassifier(max_depth=3, min_samples_leaf=10, random_state
    =1)
dt_classifier.fit(X_train, y_train)

Paragraph C:
# Load the diabetes dataset
iris_X, iris_y = datasets.load_iris(return_X_y=True)
# Split train:test = 8:2
X_train, X_test, y_train, y_test = train_test_split(
    iris_X, iris_y,
    test_size=0.2,
    random_state=42)

Paragraph D:
# Preidct and evaluate
y_pred = dt_classifier.predict(X_test)
accuracy_score(y_test, y_pred)
```

Thứ tự đúng là
a) C - A - B - D
b) A - B - D - C
c) D - C - A - B
d) A - C - D - B

- Hết -