Nguyễn Hoàng Trung

Nguyễn Cảnh Thanh

Đào Huyền Trang

Nguyễn Minh Đức

Nguyễn Thị Thùy

KNN model and mobile price

# **KNN model**

Định nghĩa:

K-nearest neighbor (KNN) là một trong những thuật toán học có giám sát đơn giản nhất trong Machine Learning. Ý tưởng của KNN là tìm ra output của dữ kiệu dựa trên thông tin của những dữ liệu training gần nó nhất.

Áp dụng được với bài toán phân lớp và hồi qui.

Áp dụng với lớp khá rộng các đặc trưng.

Quy trình làm việc của thuật toán KNN:

**Bước 1:** xác định tham số K = số láng giềng gần nhất.

**Bước 2:** tính khoảng cách đối tượng cần phân lớp với tất cả các đối tượng trong training data.

**Bước 3:** sắp xếp khoảng cách theo thứ tự tăng dần và xác định K láng giềng gần nhất với đối tượng cần phân lớp.

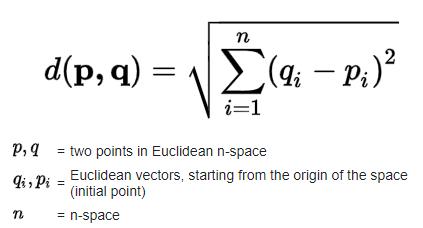
**Bước 4:** lấy tất cả các lớp của K láng giềng gần nhất.

**Bước 5:** dựa vào phần lớn lớp của K để xác định lớp cho đối tượng cần phân lớp.

Tính khoảng cách trong không gian vector:

Trong không gian 2 chiều, khoảng cách giữa hai điểm là trị tuyệt đối giữa hiệu giá trị của hai điểm đó.

Trong không gian nhiều chiều, khoảng cách giữa hai điểm có thể được định nghĩa bằng nhiều hàm số khác nhau. Trong bài tập này, ta sử dụng hàm euclidean-distance:



Lựa chọn thông số:

Sự lựa chọn K tốt nhất phụ thuộc vào dữ liệu.

Các giá trị lớn hơn của K làm giảm ảnh hưởng của nhiễu đối với việc phân loại nhưng có thể khiến cho ranh giới giữa các lớp không rõ ràng.

Thường chọn K là số lẻ.

Chọn K sao cho tỷ lệ lỗi không thấp hơn 2 lần tỷ lệ lỗi bayes.

Độ chính xác của thuật toán KNN có thể bị suy giảm nghiêm trọng do sự hiện diện của các đối tượng nhiễu hoặc không liên quan, hoặc nếu thang đo đối tượng không phù hợp với tầm quan trọng của chúng.

Ưu, nhược điểm của thuật toán KNN:

**Ưu điểm:**

-Dễ sử dụng và cài đặt.

-Việc dự đoán kết quả của dữ liệu mới dễ dàng.

-Độ phức tạp tính toán nhỏ.

**Nhược điểm:**

-Dễ bị ảnh hưởng bởi nhiễu khi K nhỏ.

-Cần thời gian lưu training set, khi dữ liệu training và test tăng lên nhiều sẽ mất nhiều thời gian tính toán.

# **Mobile price**

Dữ liệu train: train.csv

Dữ liệu test: test.csv

Nhập dữ liệu:

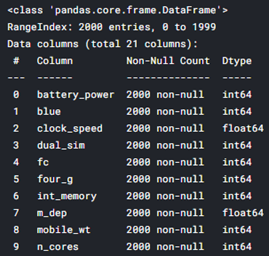
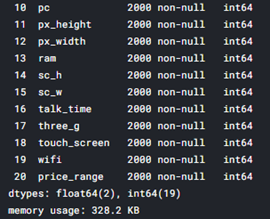


Phân tích dữ liệu:

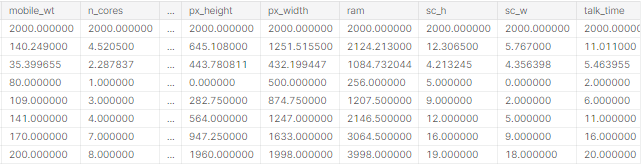
Input:

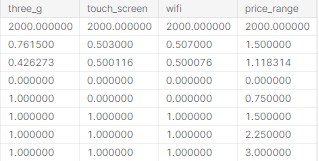


Output:







Chia tập dữ liệu thành hai thành phần test và train:



Áp dụng model KNN (với K = 11) vào tập dữ liệu:



Dự đoán lớp cho dữ liệu thuộc thành phần test:



Tính độ chính xác:

Input:



Output:



# **Áp dụng KNN vào bài toán Mobile price**

Ngôn ngữ lập trình: Python.

Thư viện sử dụng: Numpy, Pandas, sklearn

Tên file: main.py, train\_bang\_ham\_co\_san.py

**Cách chạy chương trình:**

Cài đặt và chạy Jupyter-lab:

Cài đặt anaconda.

Mở Anaconda promt, sử dụng lệnh sau để cài đặt Jupyter lab:

conda install -c conda-forge jupyterlab

Mở Jupyter lab bằng lệnh:

jupyter-lab

Cài đặt thư viện:

Sử dụng những lệnh sau trên Anaconda pompt:

conda install pandas

conda install numpy

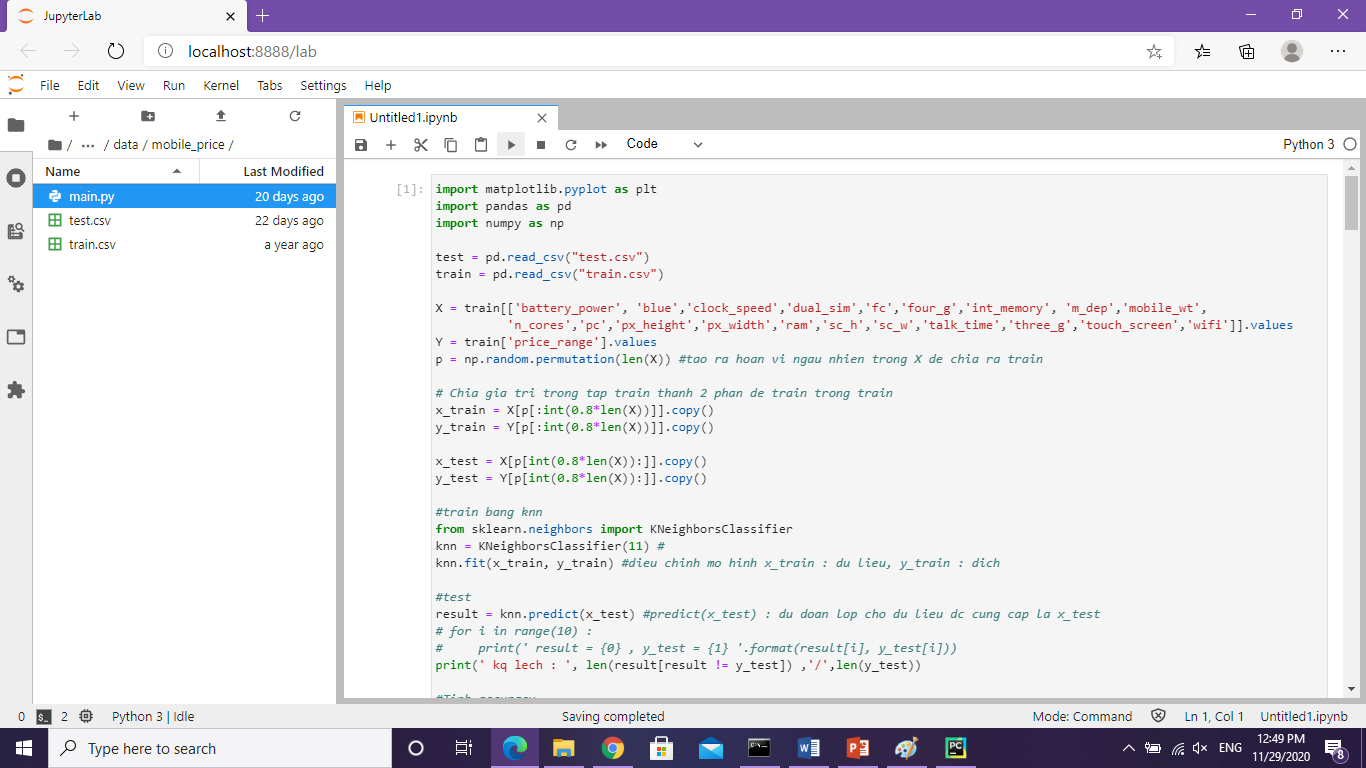
conda install matplotlib

Chạy code:

Mở Jupyter lab

Mở main.py bằng Pycharm hoắc Notepad, copy nội dung trong file vào Jupyter lab

Ấn biểu tượng run.



**Sử dụng hàm tự viết:**

**Bước 1:** xác định tham số K = số láng giềng gần nhất.

**Bước 2:** tính khoảng cách đối tượng cần phân lớp với tất cả các đối tượng trong training data:

Ta xây dựng hàm tính toán khoảng cahcs



**Bước 3:** sắp xếp khoảng cách theo thứ tự tăng dần và xác định K láng giềng gần nhất với đối tượng cần phân lớp.

Ta xây dựng hàm tìm K điểm gần nhất:

def KNNeighbor(data\_train , point, k) :  
 distance = []  
 for item in data\_train :  
 distance.append({  
 "label": item[-1],  
 "value":caclDistances(item, point)  
 })  
 distance.sort(key=lambda x: x["value"])  
 labels = [item["label"] for item in distance]  
 return labels[:k]

**Bước 4:** lấy tất cả các lớp của K láng giềng gần nhất.

**Bước 5:** dựa vào phần lớn lớp của K để xác định lớp cho đối tượng cần phân lớp.

Ta xây dựng hàm tìm lớp xuất hiện nhiều nhất trong K điểm đang xét:

def FindKNN (arr) :  
 labels = set(arr) #truyen arr vao labels  
 max\_mobile = -1  
 for label in labels:  
 mobile = arr.count(label) #tra ve so luong phan tu co gia tri la label nhieu nhat trong arr  
 if mobile > max\_mobile :  
 max\_mobile = mobile  
 res = label  
 return res

Sử dụng hàm:

for item in x\_test:  
 knn = KNNeighbor(x\_train, item, 11)  
 result = FindKNN(knn)  
 ansTrain.append(result)

**Sử dụng hàm của sklearn:**

from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier(11)   
knn.fit(x\_train, y\_train)   
result = knn.predict(x\_test)

**Kết quả:**

Train bằng 80% dữ liệu trong train.csv, dự đoán lớp của 20% dữ liệu còn lại và so sánh với lớp đúng.

Cách tính độ chính xác:

Input:

from sklearn.metrics import accuracy\_score  
E = accuracy\_score(y\_test, ansTrain)  
print(' Accuracy : ', E)  
print('So phep sai : {0}; accuracy: {1}'.format(count, (400-count)/400))

Output:

Accurary : xx

So phep sai : xx

Khi dùng hàm có sẵn:

Kết quả dự đoán lệch: 32/400

Độ chính xác: 0.92

Khi dùng hàm tự viết:

Kết quả dự đoán lệch: 20/400

Độ chính xác: 0.95