

ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Dự án 3:
DỰ ĐOÁN GIÁ NHÀ

Mục lục

1	Thông tin chung	4
1.1	Thành viên	4
1.2	Đề tài: Dự đoán giá nhà	4
2	Phương pháp	4
3	Thực nghiệm	4
3.1	Dataset	5
3.2	Độ đo	6
3.3	Mục đích	6
3.4	Thiết lập	6
3.5	Kết quả	7
3.6	Nhận xét	8
4	Tổ chức chương trình	9
4.1	Hàm chia tệp	9
4.1.1	Tên hàm	9
4.1.2	Mô tả	9
4.1.3	Tham số truyền vào	9
4.1.4	Kết quả trả về	9
4.1.5	Chức năng	9
4.1.6	Mã nguồn	9
4.2	Hàm chuẩn hóa	10
4.2.1	Tên hàm	10
4.2.2	Mô tả	10
4.2.3	Tham số truyền vào	10
4.2.4	Kết quả trả về	10
4.2.5	Chức năng	10
4.2.6	Mã nguồn	10
4.3	Hàm vẽ biểu đồ trước khi chuẩn hóa	10
4.3.1	Tên hàm	10
4.3.2	Mô tả	11
4.3.3	Tham số truyền vào	11
4.3.4	Kết quả trả về	11
4.3.5	Chức năng	11
4.3.6	Mã nguồn	11
4.4	Hàm vẽ biểu đồ sau khi chuẩn hóa	11
4.4.1	Tên hàm	11
4.4.2	Mô tả	11
4.4.3	Tham số truyền vào	12
4.4.4	Kết quả trả về	12
4.4.5	Chức năng	12
4.4.6	Mã nguồn	12
4.5	Hàm KNN	12

4.5.1	Tên hàm	12
4.5.2	Mô tả	13
4.5.3	Tham số truyền vào	13
4.5.4	Kết quả trả về	13
4.5.5	Chức năng	13
4.5.6	Mã nguồn	13
4.6	Hàm RMSE	14
4.6.1	Tên hàm	14
4.6.2	Mô tả	14
4.6.3	Tham số truyền vào	14
4.6.4	Kết quả trả về	14
4.6.5	Chức năng	14
4.6.6	Mã nguồn	15
4.7	Hàm vẽ đồ thị RMSE	15
4.7.1	Tên hàm	15
4.7.2	Mô tả	15
4.7.3	Tham số truyền vào	15
4.7.4	Kết quả trả về	15
4.7.5	Chức năng	15
4.7.6	Mã nguồn	15
A Các thư viện và thuật toán tham khảo		17

1 Thông tin chung

1.1 Thành viên

Vai trò	Họ và tên	Mã số sinh viên
Nhóm trưởng	Ngô Nguyễn Đông Quân	24120505
Thành viên	Hồ Ngọc Lan Anh	24120256
Thành viên	Phan Minh Anh	24120498
Thành viên	Phan Thị Ngọc Khanh	24120071
Thành viên	Huỳnh Hoàng Yến	24120246

1.2 Đề tài: Dự đoán giá nhà

2 Phương pháp

Quy trình thực hiện được xây dựng dựa trên mã nguồn và gồm các bước sau:

1. **Dữ liệu:** Tập dữ liệu ban đầu được đọc từ tệp CSV. Dữ liệu bao gồm các đặc trưng (features) và nhãn (labels), trong đó nhãn là giá trị cuối cùng của mỗi dòng dữ liệu.
2. **Tiền xử lý:**
 - Chia tập dữ liệu thành các tập huấn luyện (`train_data`, `train_labels`) và tập kiểm tra (`test_data`, `test_labels`) bằng hàm `split_train_test`, với tỉ lệ kiểm tra do người dùng chỉ định (mặc định là 10%).
 - Thực hiện chuẩn hóa dữ liệu để đưa các giá trị về cùng miền bằng hàm `normalize_data`. Phương pháp sử dụng là Min-Max Scaling:

$$x_{\text{normalized}} = \frac{x_{\text{raw}} - x_{\min}}{x_{\max} - x_{\min}}$$

3. **Phân tích dữ liệu:** Vẽ biểu đồ tần suất của các đặc trưng trước và sau khi chuẩn hóa, giúp đánh giá sự phân phối của dữ liệu và tác động của việc chuẩn hóa.
4. **Phân lớp:** Áp dụng thuật toán *K-Nearest Neighbors* (KNN):
 - Tính khoảng cách Euclidean giữa các mẫu kiểm tra và mẫu huấn luyện.
 - Chọn ra k hàng xóm gần nhất và tính trung bình nhãn (giá nhà) của các hàng xóm này để dự đoán.
5. **Đánh giá mô hình:**
 - Tính RMSE (Root Mean Square Error) để đánh giá độ chính xác của mô hình.
 - Vẽ biểu đồ RMSE theo các giá trị khác nhau của k để chọn được giá trị tối ưu.

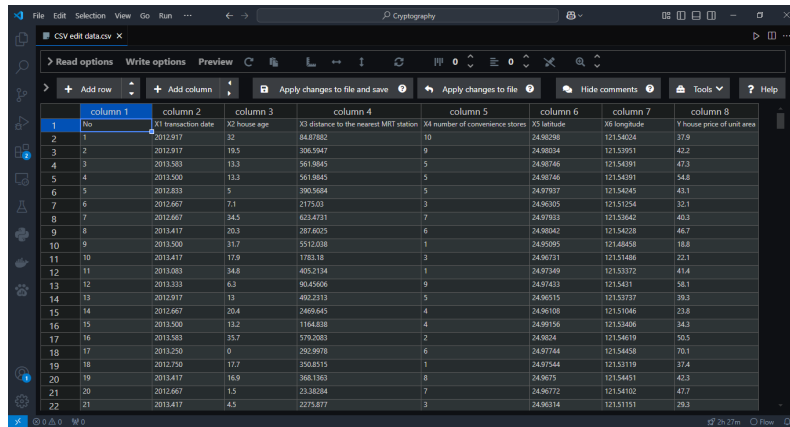
3 Thực nghiệm

Mô hình dự đoán được thực hiện có thông qua các thực nghiệm sau

3.1 Dataset

Tập data được cung cấp là một file .csv lưu những thông tin cần thiết để phân tích giá nhà và nhân với thông tin thống kê chi tiết như sau:

- Tổng có 414 dữ liệu mẫu được cung cấp.
- Mỗi dữ liệu gồm 6 đặc trưng:
 - **X1 transaction date**: Ngày giao dịch (năm tính bằng số thập phân).
 - **X2 house age**: Tuổi của ngôi nhà (năm).
 - **X3 distance to the nearest MRT station**: Khoảng cách đến trạm MRT gần nhất (mét).
 - **X4 number of convenience stores**: Số lượng cửa hàng tiện lợi gần đó.
 - **X5 latitude**: Vĩ độ.
 - **X6 longitude**: Kinh độ.
- Nhân của mỗi dữ liệu là **Y house price of unit area**: Giá nhà trên mỗi đơn vị diện tích (nghìn TWD/m²).



	column 1	column 2	column 3	column 4	column 5	column 6	column 7	column 8
	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
1	1	2012.917	32	84.87802	10	24.90298	121.54024	37.9
2	2	2012.917	19.5	306.5947	9	24.90034	121.53951	42.2
3	3	2013.503	13.3	561.9043	5	24.90746	121.54091	47.3
4	4	2013.500	13.3	561.8943	5	24.90746	121.54391	54.8
5	5	2012.833	5	390.5054	5	24.97937	121.54345	43.1
6	6	2012.667	7.1	2175.03	3	24.96305	121.51254	32.1
7	7	2012.667	34.5	623.4731	7	24.97933	121.53642	40.3
8	8	2013.417	20.3	287.6023	6	24.98042	121.54238	46.7
9	9	2013.500	31.7	553.0303	1	24.95995	121.54058	18.8
10	10	2013.417	17.9	1783.18	3	24.96731	121.51466	22.1
11	11	2013.083	34.8	405.2134	1	24.97349	121.53372	41.4
12	12	2013.333	6.3	90.45006	9	24.97433	121.5431	58.1
13	13	2012.917	13	480.2313	5	24.90515	121.53737	39.3
14	14	2012.667	20.4	2489.643	4	24.96108	121.51046	23.8
15	15	2013.500	18.2	1168.830	4	24.99156	121.53206	34.5
16	16	2013.583	35.7	579.3083	2	24.98024	121.54819	50.5
17	17	2012.250	0	282.9978	6	24.97744	121.54458	70.1
18	18	2012.750	17.7	350.8515	1	24.97544	121.53119	37.4
19	19	2013.417	16.9	368.1363	8	24.9675	121.54451	42.3
20	20	2012.667	1.5	23.38384	7	24.96772	121.54102	47.7
21	21	2013.417	4.5	2275.877	3	24.96314	121.51151	29.8

Hình 1: Dataset

Để thuận tiện cho việc học có giám sát từ dữ liệu có sẵn, nhóm chia tập dữ liệu thành 2 tập train và test.

- Tập train (90%): Giữ lại 90% dữ liệu của tập mẫu để đảm bảo mô hình có thể thực hiện học có giám sát với đầy đủ thông tin, và dữ liệu học tập.
- Tập test (10%): Đảm bảo có đủ thông tin để đánh giá mô hình đã học thành công hay không và đánh giá hiệu suất.

Vì vậy, tỉ lệ 9:1 là phù hợp cho tập dữ liệu được cung cấp, với một tập dữ liệu vừa phải, vừa tối đa hiệu suất và đảm bảo dữ liệu được cung cấp đầy đủ cho mô hình dự đoán.

3.2 Độ đo

Độ đo được tính toán dựa trên dựa trên RSME:

- RMSE (Root Mean Squared Error): là một phương pháp phổ biến để đánh giá độ chính xác của mô hình dự đoán. Nó giúp đo sự khác biệt giữa giá trị thực tế và giá trị dự đoán, sau đó lấy căn bậc hai của trung bình bình phương sai số.
Ví dụ: Nếu $RMSE = 50$ và đơn vị giá nhà là triệu đồng, nghĩa là, trung bình, dự đoán của mô hình sai lệch 50 triệu đồng so với giá thực tế.
- RMSE được tính bằng công thức:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

- Trong đó các thông số như sau:
 - n : Tổng số điểm dữ liệu.
 - y_i : Giá trị thực tế tại mẫu thứ i .
 - \hat{y}_i : Giá trị dự đoán tại mẫu thứ i .

3.3 Mục đích

RMSE được thực hiện nhằm đo lường sự khác biệt giữa giá trị thực tế của giá nhà và giá trị mà mô hình dự đoán được. Chính vì vậy, RMSE có thể được sử dụng để đánh giá hiệu suất của mô hình. Đặc biệt, RMSE rất phổ biến và được tính toán dựa trên cùng đơn vị với giá nhà, do đó có thể được đánh giá một cách trực quan và dễ dàng. Vì vậy, trong mô hình học này, nhóm sử dụng RMSE để đánh giá hiệu suất và tìm ra k tối ưu để hàm KNN trong mô hình trả về kết quả ít sai lệch nhất, cũng như mô hình đạt hiệu quả cao nhất.

3.4 Thiết lập

Thực nghiệm tính toán RMSE tìm k tối ưu được thực hiện dựa trên đoạn code sau.

```
1
2 import matplotlib.pyplot as plt
3
4 def rmse_vs_k(train_data, train_labels, test_data, test_labels, max_k
5 ):
6     rmse_values = []
7     for k in range(1, max_k + 1):
8         predictions = knn(train_data, train_labels, test_data, k)
9         rmse_value = rmse(test_labels, predictions)
10        print(f'RMSE for k={k}: {rmse_value}')
11        rmse_values.append(rmse_value)
12    plt.plot(range(1, max_k + 1), rmse_values)
13    plt.xlabel('k')
14    plt.ylabel('RMSE')
```

Listing 1: Hàm `rmse_vs_k`

RMSE trong mô hình được tính toán với các tham số là các tập **train**, **test** và **max_k**:

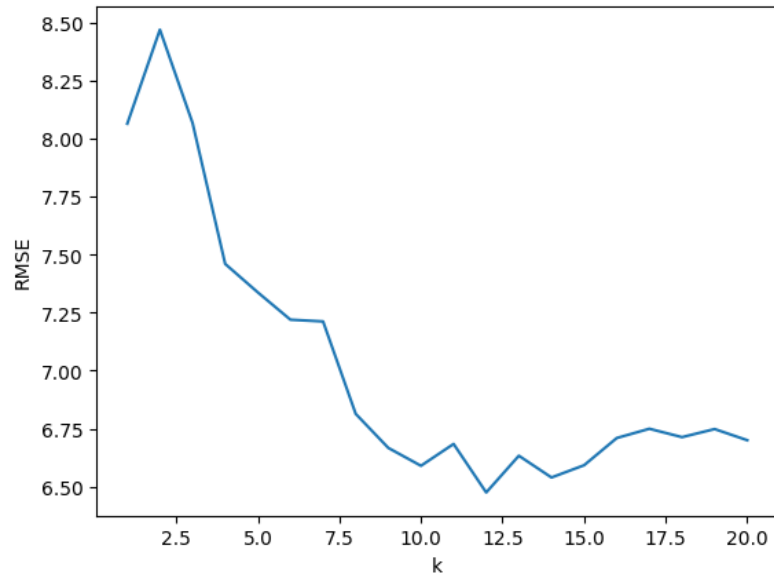
- Giá trị của `k` được xác định trong khoảng tối ưu (1, 21):
 - Đối với tập dữ liệu được cung cấp là một tập nhỏ chỉ với 414 trường hợp, nếu chọn số `k` lớn hơn số lượng mẫu thì sẽ không mang lại giá trị.
 - Đối với giá trị `k` nhỏ, việc kiểm tra và phân tích trở nên trực quan, dễ quản lý hơn và cũng tối ưu hoá thời gian chạy hơn so với giá trị `k` lớn dù vẫn cho ra giá trị tối ưu nhất.
- Sử dụng cả tập **train_data** và **train_labels** là dữ liệu huấn luyện để tìm các láng giềng gần nhất.
- **test_data** là dữ liệu cần dự đoán.
- Dựa trên hai tham số trên, mô hình sử dụng KNN để dự đoán theo `k` láng giềng gần nhất với `k` đang được thử nghiệm, và sử dụng hàm `rmse` để đánh giá hiệu suất của mô hình với `k` đang thử nghiệm. Cuối cùng, kết quả được lưu lại và tập **rmse_values** để phục vụ so sánh và tìm `k` tối ưu.

3.5 Kết quả

Sau khi tiến hành thử nghiệm với các giá trị `k` thuộc (1, 21), hàm trả về tập **rmse_values**, ta tiến hành in các giá trị và dựng đồ thị dựa trên các giá trị được trả về để tiến hành đánh giá trực quan giữa các sai số RMSE mà các giá trị `k` trả về và dễ dàng tìm được `k` tối ưu.

```
RMSE for k=1: 8.063557055106163
RMSE for k=2: 8.468474450797471
RMSE for k=3: 8.067038886860308
RMSE for k=4: 7.459852864628415
RMSE for k=5: 7.3372344601906745
RMSE for k=6: 7.219534145665029
RMSE for k=7: 7.2120010526672464
RMSE for k=8: 6.814069281902806
RMSE for k=9: 6.667275545387243
RMSE for k=10: 6.590348643136813
RMSE for k=11: 6.6844139022687505
RMSE for k=12: 6.475407282240976
RMSE for k=13: 6.633618282561763
RMSE for k=14: 6.539995159486206
RMSE for k=15: 6.592784094478001
RMSE for k=16: 6.70990433763687
RMSE for k=17: 6.749867418432714
RMSE for k=18: 6.713703498654259
```

RMSE for k=19: 6.748538024364782
RMSE for k=20: 6.700413544451318



Hình 2: Biểu đồ thể hiện giá trị RMSE theo k

3.6 Nhận xét

Trong thí nghiệm với tập dữ liệu được cung cấp, ta có thể thấy được đối với các giá trị k được thử nghiệm trong khoảng (1, 21), các giá trị RMSE trả về biến thiên với các đặc trưng sau:

- RMSE có xu hướng giảm khi các giá trị k tăng lên, đặc biệt là trong khoảng k từ 1 đến 10.
- Đối với $k > 10$, các giá trị RMSE trả về có giá trị giao động nhưng không đáng kể.
- Đối với $k = 1$, giá trị RMSE là cao nhất cho thấy mô hình bị over-fitting, chỉ dựa vào láng giềng gần nhất để dự đoán, dẫn đến nhạy cảm với các điểm nhiễu.
- Đối với k tiến dần về 20, mô hình lại bị over-smoothing, do phải dự đoán dựa trên giá trị trung bình của quá nhiều láng giềng.
- Ta nhận được k tối ưu khi k chạy trong khoảng (10, 15), nhỏ nhất tại $k = 12$ và với $k > 15$ biểu đồ cho thấy hiệu suất trả về không được cải thiện gì nhiều hơn.

Do đó, mô hình nhận được $k = 12$ là k tối ưu để thực hiện dự đoán dựa trên KNN với RMSE nhỏ nhất. Từ đó, mô hình áp dụng để dự đoán giá nhà với hiệu suất cao nhất có thể đạt được.

4 Tổ chức chương trình

4.1 Hàm chia tập

4.1.1 Tên hàm

`split_train_test`

4.1.2 Mô tả

Hàm này được sử dụng để chia bộ dữ liệu thành các tập `train_data`, `train_label`, `test_data`, và `test_label` một cách ngẫu nhiên, dựa trên tỉ lệ tập kiểm tra (`test_ratio`).

4.1.3 Tham số truyền vào

- `data`: Ma trận chứa các đặc trưng (features) của tập dữ liệu.
- `labels`: Một vector chứa nhãn (labels) tương ứng với từng hàng dữ liệu trong `data`.
- `test_ratio`: Một số thực (float) trong khoảng (0, 1), biểu diễn tỉ lệ kích thước của tập kiểm tra so với tổng dữ liệu. Mặc định là 0.1.

4.1.4 Kết quả trả về

Hàm trả về 4 giá trị:

- `train_data`: Ma trận đặc trưng của tập huấn luyện.
- `train_label`: Vector nhãn của tập huấn luyện.
- `test_data`: Ma trận đặc trưng của tập kiểm tra.
- `test_label`: Vector nhãn của tập kiểm tra.

4.1.5 Chức năng

Hàm này hỗ trợ chia dữ liệu thành các tập huấn luyện và tập kiểm tra một cách nhanh chóng và thuận tiện, phục vụ cho việc xây dựng và đánh giá mô hình học máy.

4.1.6 Mã nguồn

```
1 def split_train_test(data, labels, test_ratio=0.1):
2     combined_data = np.column_stack((data, labels))
3     np.random.shuffle(combined_data)
4     split_index = int(len(combined_data) * (1 - test_ratio))
5     train_set = combined_data[:split_index]
6     test_set = combined_data[split_index:]
7     return train_set[:, :-1], train_set[:, -1], test_set[:, :-1],
        test_set[:, -1]
```

Listing 2: Hàm chia dữ liệu thành tập huấn luyện và kiểm tra

4.2 Hàm chuẩn hóa

4.2.1 Tên hàm

`normalize_data`

4.2.2 Mô tả

Hàm này chuẩn hóa giá trị của các đặc trưng trong tập dữ liệu vào khoảng $[0, 1]$, sử dụng phương pháp **Min-Max Scaling**.

4.2.3 Tham số truyền vào

- **data**: Ma trận dữ liệu đầu vào cần chuẩn hóa.

4.2.4 Kết quả trả về

Hàm trả về ma trận **data** sau khi đã được chuẩn hóa, trong đó mỗi cột được đưa về khoảng $[0, 1]$.

4.2.5 Chức năng

Hàm chuẩn hóa dữ liệu giúp giảm thiểu sự ảnh hưởng do sự sai khác trong các đơn vị khác nhau, tạo điều kiện thuận lợi cho việc huấn luyện mô hình. Công thức chuẩn hóa sử dụng:

$$x_{\text{normalized}} = \frac{x_{\text{raw}} - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

Trong đó:

- $x_{\text{normalized}}$: Giá trị sau khi chuẩn hóa.
- x_{raw} : Giá trị ban đầu.
- x_{min} : Giá trị nhỏ nhất trong cột của dữ liệu.
- x_{max} : Giá trị lớn nhất trong cột của dữ liệu.

4.2.6 Mã nguồn

```
1 def normalize_data(data):  
2     scaler = MinMaxScaler()  
3     return scaler.fit_transform(data)
```

Listing 3: Hàm chuẩn hóa dữ liệu

4.3 Hàm vẽ biểu đồ trước khi chuẩn hóa

4.3.1 Tên hàm

`data_before_normalization`

4.3.2 Mô tả

Hàm này được sử dụng để vẽ biểu đồ phân phối dữ liệu (histogram) cho bộ dữ liệu `train_data` trước khi chuẩn hóa.

4.3.3 Tham số truyền vào

- `train_data`: Một ma trận dữ liệu đầu vào. Đây là tập dữ liệu cần vẽ biểu đồ trước khi chuẩn hóa.
- `column_names`: Danh sách chứa tên các cột của `train_data`. Phần tử đầu tiên là ID và sẽ được loại bỏ.

4.3.4 Kết quả trả về

Hàm không trả về giá trị nào. Kết quả là một biểu đồ histogram được hiển thị trên màn hình.

4.3.5 Chức năng

Hàm giúp trực quan hóa phân phối dữ liệu trước khi chuẩn hóa, từ đó đánh giá đặc điểm và sự phân bố của dữ liệu gốc.

4.3.6 Mã nguồn

```
1 def data_before_normalization(train_data, column_names):
2     train_data_df = pd.DataFrame(train_data)
3
4     # Loại bỏ cột đầu tiên
5     train_data_df = train_data_df.iloc[:, 1:]
6     train_data_df.columns = column_names[1:]
7
8     # Vẽ biểu đồ histogram
9     train_data_df.hist(bins=20, figsize=(10, 8))
10    plt.suptitle("Histogram before normalization")
11    plt.show()
```

Listing 4: Hàm chuẩn bị dữ liệu trước khi chuẩn hóa

4.4 Hàm vẽ biểu đồ sau khi chuẩn hóa

4.4.1 Tên hàm

`data_after_normalization`

4.4.2 Mô tả

Hàm này được sử dụng để vẽ biểu đồ phân phối dữ liệu (histogram) cho cả tập dữ liệu `train_data` và `test_data` sau khi chuẩn hóa.

4.4.3 Tham số truyền vào

- `train_data`: Một ma trận dữ liệu huấn luyện sau khi chuẩn hóa.
- `test_data`: Một ma trận dữ liệu kiểm tra sau khi chuẩn hóa.
- `column_names`: Danh sách chứa tên các cột. Phần tử đầu tiên là ID và sẽ được loại bỏ.

4.4.4 Kết quả trả về

Hàm không trả về giá trị nào. Kết quả là hai biểu đồ histogram (một cho `train_data` và một cho `test_data`) được hiển thị trên màn hình.

4.4.5 Chức năng

Hàm giúp trực quan hóa phân phối dữ liệu sau khi chuẩn hóa, từ đó đánh giá sự thay đổi của dữ liệu sau quá trình chuẩn hóa.

4.4.6 Mã nguồn

```
1 def data_after_normalization(train_data, test_data, column_names)
2     :
3     train_data_df = pd.DataFrame(train_data)
4     test_data_df = pd.DataFrame(test_data)
5
6     # Loại bỏ cột đầu tiên
7     train_data_df = train_data_df.iloc[:, 1:]
8     test_data_df = test_data_df.iloc[:, 1:]
9
10    # Gán tên cột
11    train_data_df.columns = column_names[1:]
12    test_data_df.columns = column_names[1:]
13
14    # Vẽ biểu đồ histogram cho train data
15    train_data_df.hist(bins=20, figsize=(10, 8))
16    plt.suptitle("Histogram after normalization (Train Data)")
17    plt.show()
18
19    # Vẽ biểu đồ histogram cho test data
20    test_data_df.hist(bins=20, figsize=(10, 8))
21    plt.suptitle("Histogram after normalization (Test Data)")
22    plt.show()
```

Listing 5: Hàm xử lý dữ liệu sau chuẩn hóa

4.5 Hàm KNN

4.5.1 Tên hàm

`knn`

4.5.2 Mô tả

Hàm `knn` được sử dụng để dự đoán giá trị của các điểm dữ liệu mới dựa trên thuật toán K-Nearest Neighbors (KNN). Đây là một thuật toán học máy phi tham số thường được áp dụng cho các bài toán hồi quy hoặc phân loại.

4.5.3 Tham số truyền vào

- `train_data`: Ma trận $n \times d$ chứa n mẫu huấn luyện, mỗi mẫu có d đặc trưng.
- `train_labels`: Mảng 1 chiều chứa nhãn (giá trị mục tiêu) tương ứng của dữ liệu huấn luyện.
- `test_data`: Ma trận $m \times d$ chứa m mẫu cần dự đoán, mỗi mẫu có d đặc trưng.
- `k`: Số lượng hàng xóm gần nhất để xem xét.

4.5.4 Kết quả trả về

Hàm trả về danh sách các giá trị dự đoán cho tất cả các mẫu trong `test_data`.

4.5.5 Chức năng

Hàm `knn` được sử dụng để dự đoán giá trị dựa trên các thông tin dữ liệu huấn luyện. Cụ thể, hàm tính khoảng cách giữa dữ liệu cần dự đoán và các mẫu huấn luyện, xác định k hàng xóm gần nhất, sau đó lấy trung bình giá trị nhãn của k hàng xóm này làm kết quả dự đoán.

4.5.6 Mã nguồn

```
1 def knn(train_data, train_labels, test_data, k):
2
3     predictions = []
4
5     for test_instance in test_data:
6         distances = []
7         # Tính và thêm các khoảng cách từ test_instance đến mỗi
           train_instance
8         for i, train_instance in enumerate(train_data):
9             distance = np.linalg.norm(test_instance -
               train_instance)
10            distances.append((train_instance, train_labels[i],
               distance))
11
12            # Sắp xếp mảng distances theo giá trị khoảng cách
13            distances.sort(key=lambda x: x[2])
14
15            # Lấy nhãn của k mẫu gần nhất
16            neighbors = [x[1] for x in distances[:k]]
17
18            # Tính giá trị dự đoán là trung bình của các nhãn gần nhất
19            prediction = np.mean(neighbors)
20
```

```

21         # Thêm giá trị prediction vào mảng predictions
22         predictions.append(prediction)
23
24     return predictions

```

Listing 6: Hàm KNN

4.6 Hàm RMSE

4.6.1 Tên hàm

`rmse`

4.6.2 Mô tả

Hàm `rmse` được sử dụng để tính toán Root Mean Squared Error (RMSE) – một thước đo phổ biến để đánh giá độ chính xác của mô hình dự đoán.

Công thức toán học:

RMSE được định nghĩa như sau:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Trong đó:

- y_i : Giá trị thực tế tại mẫu thứ i .
- \hat{y}_i : Giá trị dự đoán tại mẫu thứ i .
- n : Tổng số lượng mẫu.

4.6.3 Tham số truyền vào

- `actual`: Danh sách hoặc mảng chứa các giá trị thực tế y_i .
- `predicted`: Danh sách hoặc mảng chứa các giá trị dự đoán \hat{y}_i .

4.6.4 Kết quả trả về

Hàm trả về một giá trị số thực, biểu diễn RMSE giữa các giá trị thực tế và giá trị dự đoán.

4.6.5 Chức năng

Hàm `rmse` được sử dụng để đánh giá độ chính xác của mô hình dự đoán giá nhà. Hàm tính sai số giữa giá trị thực tế và giá trị dự đoán, sau đó trả về căn bậc hai trung bình bình phương sai số (RMSE). Chỉ số RMSE nhỏ cho thấy mô hình dự đoán chính xác hơn.

4.6.6 Mã nguồn

```
1 def rmse(actual, predicted):
2     return np.sqrt(np.mean((np.array(actual) - np.array(predicted)
3                             )) ** 2))
```

Listing 7: Hàm RMSE trong Python

4.7 Hàm vẽ đồ thị RMSE

4.7.1 Tên hàm

rmse_vs_k

4.7.2 Mô tả

Hàm `rmse_vs_k` thực hiện việc tính toán sai số RMSE cho nhiều giá trị k trong thuật toán KNN và vẽ đồ thị mối quan hệ giữa giá trị k và RMSE. Đây là cách giúp lựa chọn giá trị k tối ưu cho mô hình.

4.7.3 Tham số truyền vào

- `train_data`, `train_labels`: Dữ liệu huấn luyện và nhãn huấn luyện.
- `test_data`, `test_labels`: Dữ liệu kiểm tra và nhãn kiểm tra.
- `max_k`: Số lượng giá trị tối đa của k cần kiểm tra.

4.7.4 Kết quả trả về

- Danh sách các giá trị RMSE tương ứng với từng giá trị k .
- Đồ thị thể hiện mối quan hệ giữa k và RMSE, giúp xác định giá trị k tối ưu.

4.7.5 Chức năng

Mục tiêu là tìm ra giá trị k tối ưu sao cho RMSE thấp nhất.

4.7.6 Mã nguồn

```
1 import matplotlib.pyplot as plt
2
3 def rmse_vs_k(train_data, train_labels, test_data, test_labels,
4               max_k):
5     rmse_values = []
6
7     # Tính sai số RMSE cho mỗi giá trị k
8     for k in range(1, max_k + 1):
9         predictions = knn(train_data, train_labels, test_data, k)
10        rmse_value = rmse(test_labels, predictions)
11        print(f'RMSE for k={k}: {rmse_value}')
```

```
11         rmse_values.append(rmse_value)
12
13     # Vẽ đồ thị RMSE theo k
14     plt.plot(range(1, max_k + 1), rmse_values)
15
16     plt.xlabel('k')
17     plt.ylabel('RMSE')
18     plt.show()
```

Listing 8: Hàm `rmse_vs_k`

Phụ lục A Các thư viện và thuật toán tham khảo

Chúng tôi sử dụng các thư viện NumPy [1], Pandas [2], Matplotlib [3], và scikit-learn [4] để hỗ trợ các hàm xử lý. Ngoài ra, thuật toán KNN được tham khảo từ [5], kỹ thuật tiền xử lý từ [6], và phương pháp Cross-Validation từ [7].

Tài liệu

- [1] Harris, C.R., Millman, K.J., Walt, S.J., *et al.*: Array programming with numpy. Nature **585**(7825), 357–362 (2020)
- [2] McKinney, W.: Data structures for statistical computing in python, 51–56 (2010). Austin, TX
- [3] Hunter, J.D.: Matplotlib: A 2d graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007)
- [4] Pedregosa, F., Varoquaux, G., Gramfort, A., *et al.*: Scikit-learn: Machine learning in python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
- [5] Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory **13**(1), 21–27 (1967)
- [6] Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Elsevier, ??? (2011)
- [7] Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, vol. 14, pp. 1137–1145 (1995)