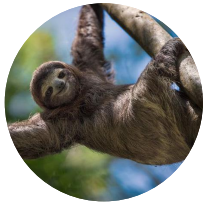
Machine learning project

topic: predict house's price in Hanoi

Project Team



Ngô Anh Kiệt

Đỗ Hữu Đại

Phan Đức Mạnh

TASK OVERVIEW

1. Presentation
2. Crawl Data
3. Preprocessing Data
4. Design models
5. Optimal

PREDICT HOUSE'S PRICE IN HANOI

1. Nguyên nhân

- Người dân di cư đến các thành phố lớn để lập nghiệp và làm việc.
- Sự gia tăng dân số khiến diện tích ngày càng thu hẹp.
- Vị trí của các trung tâm mua sắm và trung tâm giải trí ảnh hưởng đáng kể đến giá nhà.
- Các yếu tố bên ngoài ảnh hưởng đến chất lượng cuộc sống.

2. Mục tiêu

- Có thể dự đoán giá nhà với độ chính xác cao.
- Tối ưu dự đoán.
- Có thể đưa ra dự đoán giá từng mảnh đất, giá nhà dựa trên các yếu tố cho sẵn.

CRAWL DATA:

📌 Dữ liệu được tìm kiếm và thu thập từ nguồn uy tín dựa trên đánh giá của mọi người.

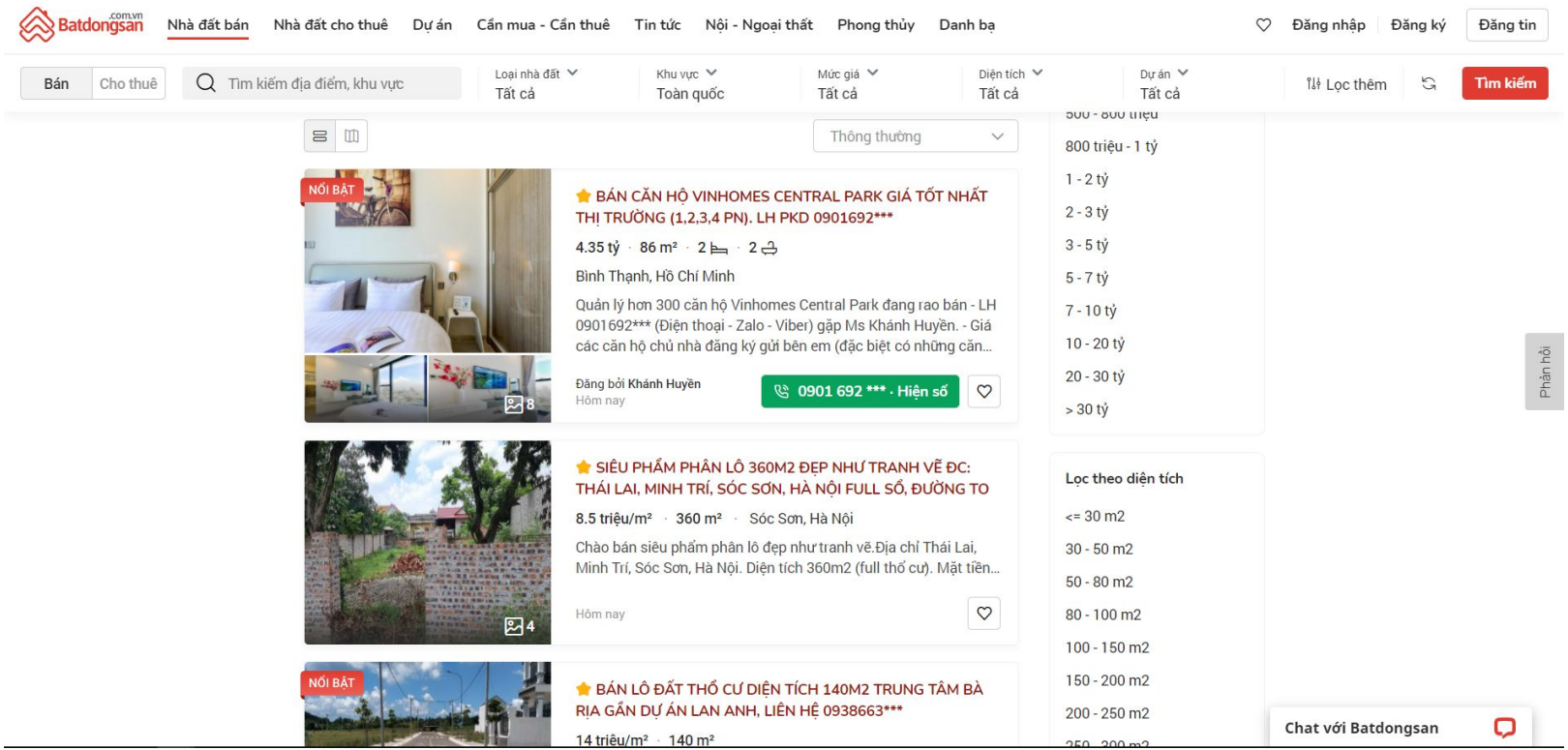
Nguồn: https://batdongsan.com.vn/nha-dat-ban

Data : data link

📌 Dữ liệu về một ngôi nhà có các thuộc tính: Địa chỉ, mặt tiền, số phòng ngủ, số phòng vệ sinh, diện tích, số tầng, số đường vào

📌 Giá nhà luôn được cập nhật theo từng theo gian khác nhau.

WEBSITE: BATDONGSAN.COM.VN



METHOD OF CRAWL DATA:

- Lập trình Bot kéo dữ liệu (Python & BeautifulSoup):
  - Python:
    - Sử dụng Selenium và Chromedriver để truy cập vào website.
    - Sử dụng python-csv để lưu thông tin vào CSV

  - BeautifulSoup: Chiết xuất các thông tin cần thiết từ các thông tin của một ngôi nhà.

=> Kết quả: 3368 data

DATASET:

A	B	C	D	E	F	G	H	I
Date	địa chỉ	mặt tiền	đường vào	số tầng	số phòng ngủ	số phòng toilet	m^2	price
21/08/2021	318/70 Tổ 11, Phố Ngọc Trì, Phường Thạch Bàn, Long Biên, Hà Nội			4	3	3	35.5	76.1 triệu/m²
11/09/2021	198/78 Đường Số 3, Phường 9, Gò Vấp, Hồ Chí Minh	5	6	4	22	22	110	86.4 triệu/m²
13/09/2021	Phố Bạch Mai, Phường Bạch Mai, Hai Bà Trưng, Hà Nội	9	10	3	4	3	60	316.7 triệu/m²
15/09/2021	Phố Cầu Cốc, Phường Tây Mỗ, Nam Từ Liêm, Hà Nội	9	3.5	3	3	3	122	47.5 triệu/m²
16/09/2021	Dự án KVG The Capella Nha Trang, Đường Võ Nguyên Giáp, Xã Vĩnh Thái, Nha Trang, Khánh Hòa	5	14	3			100	39 triệu/m²
18/09/2021	Đường Số 9, Phường Dĩ An, Dĩ An, Bình Dương			3	5		226	84.1 triệu/m²
21/09/2021	Phố Xã Đàn, Phường Nam Đồng, Đống Đa, Hà Nội	5	5	3	3	3	55	87.3 triệu/m²
21/09/2021	Đường Nguyễn Hữu Thọ, Phường Đại Kim, Hoàng Mai, Hà Nội	4.5	10	5	4	4	60	250 triệu/m²
22/09/2021	32c Ngõ 53, Đường Tân Ấp, Phường Phúc Xá, Ba Đình, Hà Nội	4			3	4	40	7.5 tỷ
22/09/2021	Dự án KĐT Linh Đàm, Phường Đại Kim, Hoàng Mai, Hà Nội		15.5	3			254	86.6 triệu/m²
23/09/2021	Ngõ 158, Đường Nguyễn Sơn, Phường Bồ Đề, Long Biên, Hà Nội	4.5	17	5	5	4	68	200 triệu/m²
24/09/2021	Ngõ 68, Đường Phú Diễn, Phường Phú Diễn, Bắc Từ Liêm, Hà Nội	4.7		5	3	4	30	120 triệu/m²
25/09/2021	Dự án La Vida Residences, Đường 3/2, Phường 12, Vũng Tàu, Bà Rịa Vũng Tàu		11	4	4	4	111	81.1 triệu/m²
26/09/2021	Dự án Sudico Mỹ Đình, Đường Phạm Hùng, Phường Mỹ Đình 1, Nam Từ Liêm, Hà Nội	9		4			193.5	154 triệu/m²
26/09/2021	Dự án La Vida Residences, Đường 3/2, Phường 12, Vũng Tàu, Bà Rịa Vũng Tàu			4	4	3	78	67.9 triệu/m²
26/09/2021	Đường Tăng Thiết Giáp, Phường Cổ Nhuế 2, Bắc Từ Liêm, Hà Nội	7	18	2			95	189.5 triệu/m²
26/09/2021	Số 2A, Đường Lý Đạo Thành, Phường 16, Quận 8, Hồ Chí Minh	9	8	3	4	3	62.1	93.4 triệu/m²
27/09/2021	Dự án Sun Group Sầm Sơn, Phường Trung Sơn, Sầm Sơn, Thanh Hóa	10	7	3	5	6	150	73.3 triệu/m²
27/09/2021	Đường Cầu Giấy, Phường Quan Hoa, Cầu Giấy, Hà Nội	7.5	3		43	43	133	214.3 triệu/m²
28/09/2021	Dự án Sun Grand Boulevard, Phường Trung Sơn, Sầm Sơn, Thanh Hóa.	7.5	22	5	12	13	112.5	103.1 triệu/m²
28/09/2021	Đường Đại Từ, Phường Đại Kim, Hoàng Mai, Hà Nội	10		2			45	177.8 triệu/m²
28/09/2021	Phố Chùa Láng, Phường Láng Thượng, Đống Đa, Hà Nội	3.7	5	5	4	4	38	173.7 triệu/m²
28/09/2021	Phố Thái Hà, Phường Trung Liệt, Đống Đa, Hà Nội	4	7	6	5	6	45	266.7 triệu/m²
28/09/2021	Golden Avenue, 45, Đường Hùng Vương, Bắc Giang, Bắc Giang	5	15	5			75	101.3 triệu/m²
29/09/2021	Dự án Vinhomes Ocean Park Gia Lâm, Xã Dương Xá, Gia Lâm, Hà Nội	8	52	4	4		95	210.5 triệu/m²
29/09/2021	Ngõ 388 số nhà 5, Đường Thụy Khuê, Phường Thụy Khuê, Tây Hồ, Hà Nội	6	3	5	4	3	50	112 triệu/m²
29/09/2021	Dự án CityLand Park Hills, Đường Phan Văn Trị, Phường 10, Gò Vấp, Hồ Chí Minh	5	12	5	8	8	100	184 triệu/m²
29/09/2021	23 ngõ 262, Đường Thanh Bình, Phường Mỗ Lao, Hà Đông, Hà Nội	6	4		33	33	130	119.2 triệu/m²
29/09/2021	549/ Đường Lê Văn Thọ, Phường 14, Gò Vấp, Hồ Chí Minh	4	8	5	5	6	68	125 triệu/m²
29/09/2021	Phố Bạch Mai, Phường Bạch Mai, Hai Bà Trưng, Hà Nội	4.5	3	5	4	4	35	102.9 triệu/m²
29/09/2021	Đường Cầu Giấy, Phường Dịch Vọng, Cầu Giấy, Hà Nội	4	5	5	6	5	52	143.3 triệu/m²
30/09/2021	Dự án Louis City Hoàng Mai, Đường Tân Mai, Phường Hoàng Văn Thụ, Hoàng Mai, Hà Nội	5	13.5	5	9		107.5	13.4 tỷ
30/09/2021	Phường Dương Nội, Hà Đông, Hà Nội	4	5	3	3	3	33	40.9 triệu/m²
30/09/2021	Dự án Horizon Bay, Đường Hoàng Quốc Việt, Phường Bãi Cháy, Hạ Long, Quảng Ninh	6.5	22	5			84.5	6.3 tỷ
30/09/2021	Dự án The Manor Central Park, Đường Nghiêm Xuân Yêm, Phường Đại Kim, Hoàng Mai, Hà Nội	6.6	13	4	4	5	75	226.7 triệu/m²
30/09/2021	Phố Nguyễn Phúc Lai, Phường Ô Chợ Dừa, Đống Đa, Hà Nội	3.7	3	4	3	3	53	84.9 triệu/m²
30/09/2021	Đường Ngô Thì Nhậm, Hà Đông, Hà Nội	4	15	4			48	141.7 triệu/m²

DATA PREPROCESSING:

1. Làm sạch và xử lý dữ liệu
2. Splitting Training set and Test set

RAW DATA

```
Index(['Date', 'địa chỉ', 'mặt tiền', 'đường vào', 'số tầng', 'số phòng  ngủ',
       'số phòng toilet', 'm^2', 'price'],
      dtype='object')
```

	Date	địa chỉ	mặt tiền	đường vào	số tầng	số phòng ngủ	số phòng toilet	m^2	price
1037	2021-01-10	Đường Hòa Bình, Phường Yên Nghĩa, Hà Đông, Hà Nội	3.5	3.0	4.0	3.0	4.0	35.0	44.3 triệu/m²
1041	2021-01-10	34, Đường An Dương Vương, Phúc Yên, Vĩnh Phúc	5.0	20.0	3.0	4.0	3.0	53.0	112.3 triệu/m²
3076	2021-01-10	Đường Liên Phường, Phường Phú Hữu, Quận 9, Hồ ...	NaN	NaN	3.0	4.0	5.0	84.0	148.8 triệu/m²
1042	2021-01-10	Đường Nguyễn Văn Cừ, Phường Bồ Đề, Long Biên, ...	4.0	NaN	5.0	3.0	4.0	42.0	92.6 triệu/m²
1043	2021-01-10	82, Đường Yên Lãng, Phường Láng Hạ, Đống Đa...	9.0	NaN	5.0	3.0	5.0	51.0	152.9 triệu/m²
...	...	...	...	...	...	...	...	...	...
1958	2021-10-10	Đường Mễ Trì Hạ, Phường Mễ Trì, Nam Từ Liêm, H...	4.0	3.0	5.0	6.0	5.0	45.0	93.3 triệu/m²
1249	2021-10-10	Đường Lê Quang Đạo, Nam Từ Liêm, Hà Nội.	4.0	3.0	4.0	2.0	4.0	32.0	106.2 triệu/m²
2819	2021-10-10	Đường Yên Xá, Xã Tân Triều, Thanh Trì, Hà Nội	4.0	4.0	7.0	18.0	18.0	73.0	121.9 triệu/m²
2818	2021-10-10	đường Trung Văn, Phường Trung Văn, Nam Từ Liêm...	6.3	2.5	6.0	14.0	14.0	50.0	158 triệu/m²
1484	2021-11-09	198/78 Đường Số 3, Phường 9, Gò Vấp, Hồ Chí Minh	5.0	6.0	4.0	22.0	22.0	110.0	86.4 triệu/m²

3368 rows × 9 columns

CLEAN DATA:

```
# loại bỏ đi data dư thừa
df=df.dropna()
df=df.reset_index()
df.describe
```

```
<bound method NDFrame.describe of       index       Date  ...    m^2          price
0         3 2021-09-10  ...   67.5  214.8 triệu/m²
1         4 2021-09-10  ...   75.0   54.7 triệu/m²
2         6 2021-08-10  ...  117.0   51.3 triệu/m²
3         8 2021-08-10  ...  135.0   74.1 triệu/m²
4        10 2021-08-10  ...   34.0   79.4 triệu/m²
...     ...        ...  ...    ...              ...
1914   3356 2021-09-10  ...   47.0   63.4 triệu/m²
1915   3359 2021-09-10  ...   42.0  114.3 triệu/m²
1916   3361 2021-09-10  ...   32.0   83.1 triệu/m²
1917   3362 2021-09-10  ...   68.0   69.1 triệu/m²
1918   3363 2021-09-10  ...   90.0  122.2 triệu/m²

[1919 rows x 10 columns]>
```

RE-FORMAT PRICE COLUMN:

Price: String to Float

```
import re

# format lại cột Price
for i in range(len(df)):
  price = df['price'][i]
  if price.find('triệu') != -1:
    num = re.findall(r"[-+]?\d*\.\d+|\d+", price)[0]
    df['price'][i] =num
  elif price.find('tỷ') != -1:
    num = float(re.findall(r"[-+]?\d*\.\d+|\d+", price)[0])
    num = (num*(10**9) / (df['m^2'][i]))
    num /= (10**6)
    num=round(num,2)
    df['price'][i] = str(num)
  elif price.find('nghìn'):
    num = float(re.findall(r"[-+]?\d*\.\d+|\d+", price)[0])
    num = (num*(10**3))
    num /= (10**6)
    num=round(num,2)
    df['price'][i] = str(num)
df
```

	index	Date	địa chỉ	mặt tiền	đường vào	số tầng	số phòng ngủ	số phòng toilet	m^2	price
0	3	2021-09-10	Dự án Vinhomes Ocean Park Gia Lâm, Đường Ngọc ...	5.0	40.0	5.0	5.0	5.0	67.5	214.8
1	4	2021-09-10	Dự án Wyndham Thanh Thủy, Đường Tỉnh Lộ 317, X...	5.0	24.0	4.0	3.0	4.0	75.0	54.7
2	6	2021-08-10	Dự án Izumi City, Biên Hòa, Đồng Nai	8.0	60.0	3.0	4.0	4.0	117.0	51.3
3	8	2021-08-10	Dự án Vinhomes Ocean Park Gia Lâm, Gia Lâm, Hà...	6.0	52.0	4.0	3.0	3.0	135.0	74.1
4	10	2021-08-10	Ngõ 1 ngách 22 số 28, Đường Bùi Xương Trạch, P...	3.5	2.0	3.0	3.0	2.0	34.0	79.4
...	...	...	...	...	...	...	...	...	...	...
1914	3356	2021-09-10	Đường Định Công Thượng, Phường Định Công, Hoàn...	4.0	3.0	4.0	4.0	3.0	47.0	63.4
1915	3359	2021-09-10	Phố Khúc Thừa Dụ, Phường Dịch Vọng, Cầu Giấy, ...	3.5	3.2	5.0	5.0	4.0	42.0	114.3
1916	3361	2021-09-10	Đường Hà Trì 2, Phường Hà Cầu, Hà Đông, Hà Nội	3.9	2.5	5.0	3.0	3.0	32.0	83.1
1917	3362	2021-09-10	Đường Thạnh Lộc 29, Phường Thạnh Lộc, Quận 12,...	4.0	8.0	4.0	4.0	5.0	68.0	69.1
1918	3363	2021-09-10	Dự án Vinhomes Ocean Park Gia Lâm, Huyện Gia L...	5.0	30.0	4.0	4.0	3.0	90.0	122.2

1919 rows × 10 columns

- Chiếu vị trí theo địa chỉ
  => locate theo quận

```
[ ] dataframe = df[df['city']== 'Hà Nội']

[ ] listColumn=dataframe.columns
    dataframe =dataframe[listColumn[2::]].reset_index()

[ ] !kaggle datasets download -d smileymask/quanhanoi

    quanhanoi.zip: Skipping, found more recently modified local copy (use --force to force download)

    local_zip = '/content/quanhanoi.zip'
    zip_ref = zipfile.ZipFile(local_zip, 'r')
    zip_ref.extractall()
    zip_ref.close()

[ ] listQuan = pd.read_csv('/content/HaNoi quan - Sheet1.csv')

[ ] danhSachQuan = listQuan['Tên gọi'].tolist()

[ ] listQuan[(listQuan['Tên gọi']== 'Ba Đình')]['Thể loại hành chính'][0]

    'Quận'
```

DATASET:

	index	Date	địa chỉ	mặt tiền	đường vào	số tầng	số phòng ngủ	số phòng toilet	m^2	price	city	locate
0	0	2021-09-10	Dự án Vinhomes Ocean Park Gia Lâm, Đường Ngọc ...	5.0	40.0	5.0	5.0	5.0	67.5	214.8	Hà Nội	Gia Lâm
1	2	2021-08-10	Dự án Vinhomes Ocean Park Gia Lâm, Gia Lâm, Hà...	6.0	52.0	4.0	3.0	3.0	135.0	74.1	Hà Nội	Gia Lâm
2	3	2021-08-10	Ngõ 1 ngách 22 số 28, Đường Bùi Xương Trạch, P...	3.5	2.0	3.0	3.0	2.0	34.0	79.4	Hà Nội	Thanh Xuân
3	4	2021-08-10	34 ngõ 637 Phố Trương Định, Phường Thịnh Liệt,...	3.5	3.0	5.0	3.0	4.0	35.0	94.3	Hà Nội	Hoàng Mai
4	6	2021-08-10	Dự án Vinhomes Green Bay Mễ Trì, Đường Hoàng L...	7.5	13.0	4.0	4.0	5.0	93.5	278.1	Hà Nội	Nam Từ Liêm
...	...	...	...	...	...	...	...	...	...	...	...	...
1211	1835	2021-09-10	Đường Định Công, Phường Định Công, Hoàng Mai, ...	5.0	4.0	2.0	2.0	2.0	80.0	65.0	Hà Nội	Hoàng Mai
1212	1836	2021-09-10	Đường Định Công Thượng, Phường Định Công, Hoàn...	4.0	3.0	4.0	4.0	3.0	47.0	63.4	Hà Nội	Hoàng Mai
1213	1837	2021-09-10	Phố Khúc Thừa Dụ, Phường Dịch Vọng, Cầu Giấy, ...	3.5	3.2	5.0	5.0	4.0	42.0	114.3	Hà Nội	Cầu Giấy
1214	1838	2021-09-10	Đường Hà Trì 2, Phường Hà Cầu, Hà Đông, Hà Nội	3.9	2.5	5.0	3.0	3.0	32.0	83.1	Hà Nội	Hà Đông
1215	1840	2021-09-10	Dự án Vinhomes Ocean Park Gia Lâm, Huyện Gia L...	5.0	30.0	4.0	4.0	3.0	90.0	122.2	Hà Nội	Gia Lâm

1216 rows × 12 columns

- Bảng số liệu theo quận, huyện

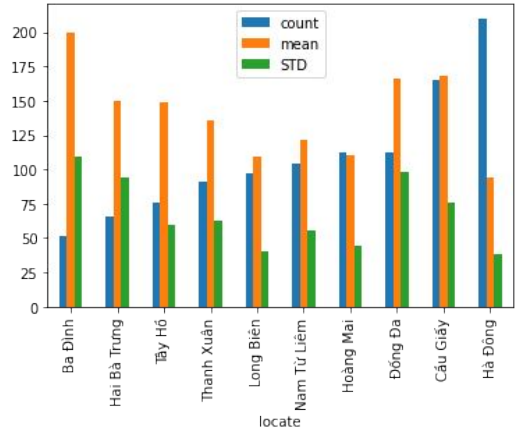
```
count=dataframe.groupby('locate').size().rename('count')
mean= dataframe.groupby('locate').mean()['price'].rename('mean')
var = dataframe.groupby('locate').std()['price'].rename('STD')

# des =pd.merge(pd.merge(count,mean,on='locate'),var,on='locate')

des=pd.concat([count, mean,var], axis=1)
des=des.sort_values(by='count')
des.plot(kind="bar")
des
```

locate	Count	Mean	STD
Quốc Oai	1	45	NaN
Sơn Tây	1	33.3	NaN
Chương Mỹ	2	20.8	8.202439
Thạch Thất	3	29.1	13.286459
Hoàn Kiếm	4	531.15	316.089149
Đan Phượng	5	66.18	9.266175
Đông Anh	6	50.555	17.479638
Thanh Oai	13	54.453846	25.768056
Gia Lâm	17	131.05	53.840261
Bắc Từ Liêm	22	77.018182	26.751549
Hoài Đức	27	73.967037	22.088858
Thanh Trì	30	83.145667	19.585528
Ba Đình	52	199.719231	109.604181
Hai Bà Trưng	66	150.1	94.131335
Tây Hồ	76	148.942105	59.760862
Thanh Xuân	91	135.291758	62.996142
Long Biên	97	109.617835	40.473905
Nam Từ Liêm	104	121.972115	55.23982
Hoàng Mai	112	110.484375	44.397553
Đống Đa	112	165.836607	98.088529
Cầu Giấy	165	168.231091	75.599743
Hà Đông	210	93.892524	37.893072

- Lọc các huyện theo số lượng nhà bán >= 50



	index	Date	địa chỉ	mặt tiền	đường vào	số tầng	số phòng ngủ	số phòng toilet	m^2	price	city	locate
2	3	2021-08-10	Ngõ 1 ngách 22 số 28, Đường Bùi Xương Trạch, P...	3.5	2.0	3.0	3.0	2.0	34.0	79.4	Hà Nội	Thanh Xuân
3	4	2021-08-10	34 ngõ 637 Phố Trương Định, Phường Thịnh Liệt,...	3.5	3.0	5.0	3.0	4.0	35.0	94.3	Hà Nội	Hoàng Mai
4	6	2021-08-10	Dự án Vinhomes Green Bay Mễ Trì, Đường Hoàng L...	7.5	13.0	4.0	4.0	5.0	93.5	278.1	Hà Nội	Nam Từ Liêm
5	9	2021-07-10	Đường Nguyễn Khang, Phường Yên Hòa, Cầu Giấy, ...	6.5	5.0	7.0	5.0	5.0	53.0	292.5	Hà Nội	Cầu Giấy
6	11	2021-06-10	Đường Lê Trọng Tấn, Phường Dương Nội, Hà Đông,...	5.0	3.0	3.0	4.0	3.0	33.0	40.9	Hà Nội	Hà Đông
...	...	...	...	...	...	...	...	...	...	...	...	...
1210	1833	2021-09-10	Đường Tư Đình, Phường Long Biên, Long Biên, Hà...	3.8	2.4	3.0	3.0	3.0	56.0	60.4	Hà Nội	Long Biên
1211	1835	2021-09-10	Đường Định Công, Phường Định Công, Hoàng Mai, ...	5.0	4.0	2.0	2.0	2.0	80.0	65.0	Hà Nội	Hoàng Mai
1212	1836	2021-09-10	Đường Định Công Thượng, Phường Định Công, Hoàn...	4.0	3.0	4.0	4.0	3.0	47.0	63.4	Hà Nội	Hoàng Mai
1213	1837	2021-09-10	Phố Khúc Thừa Dụ, Phường Dịch Vọng, Cầu Giấy, ...	3.5	3.2	5.0	5.0	4.0	42.0	114.3	Hà Nội	Cầu Giấy
1214	1838	2021-09-10	Đường Hà Trì 2, Phường Hà Cầu, Hà Đông, Hà Nội	3.9	2.5	5.0	3.0	3.0	32.0	83.1	Hà Nội	Hà Đông

1085 rows × 12 columns

2. Model

Split train test set

```
import random as rd
y_train = dataUse['price'].to_numpy()

xData=dataUse.drop(columns=['price'])
x_train =xData.to_numpy()
x_test = []
y_test = []

#Split data to train set and test set
for i in range(int(x_train.shape[0] * 0.2)):
  id = rd.randint(0, x_train.shape[0]-1)
  x_test.append(x_train[id])
  y_test.append(y_train[id])

  x_train = np.delete(x_train, id, axis = 0)
  y_train = np.delete(y_train, id)

x_test = np.array(x_test)
y_test = np.array(y_test)

x_test = np.vstack((np.ones(x_test.shape[0], ),x_test.T)).T
x_train = np.vstack((np.ones(x_train.shape[0], ), x_train.T)).T
```

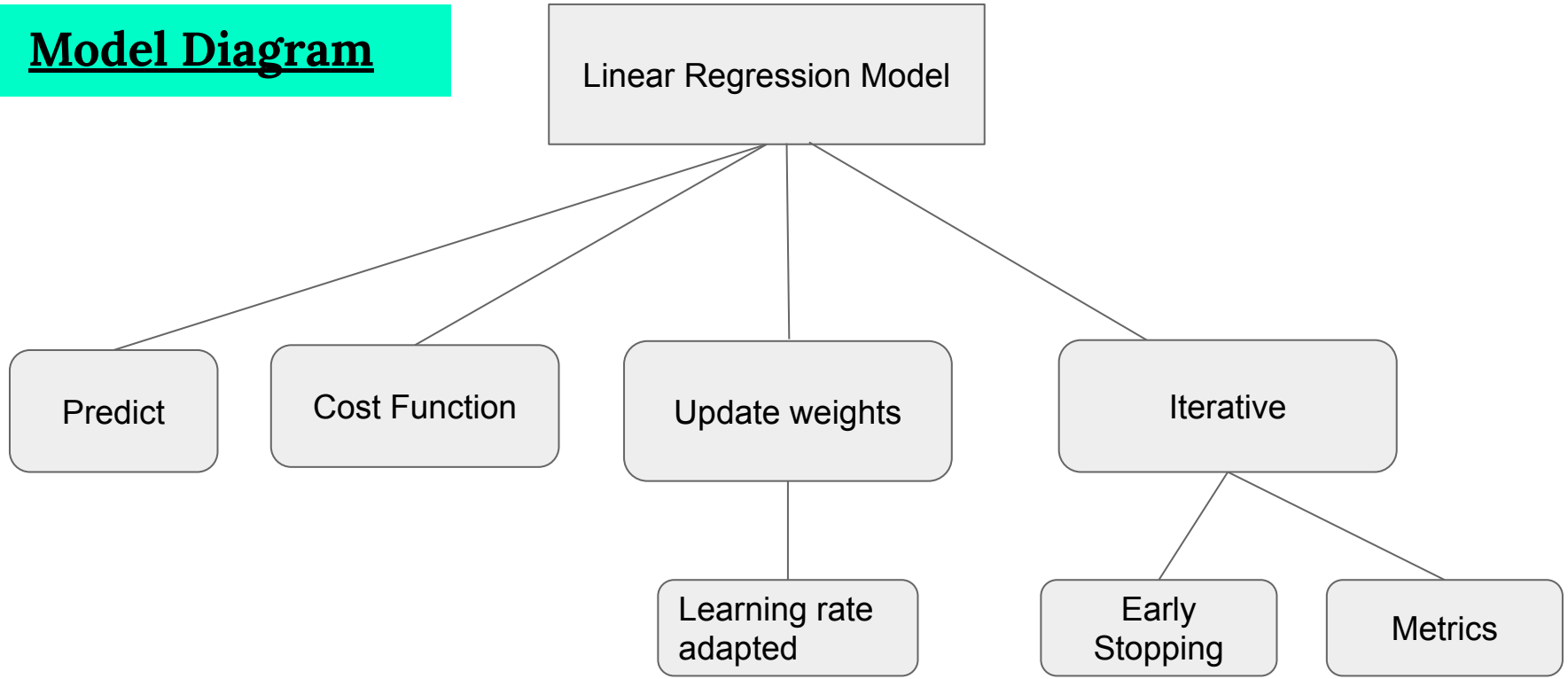
Linear Regression using Gradient Descent

$$\text{Loss(w)} = \frac{1}{2N}\sum_{i=0}^{\text{N}}(y_i - x_i \,.\, w)^2$$

$$\text{Loss(w)'} = \frac{1}{N}\sum_{i=0}^{\text{N}} -x_i(y_i - x_i \,.\, w)$$

$$w_{new} = w_{old} - \text{lr} * \text{Loss}(w_{old})'$$

Model Diagram



```python
def update_weight(self, w):
    m = len(self.X)
    predictions = np.dot(self.X, w)
    error = self.y - predictions
    gradient = np.dot(-self.X.T,  error)
    gradient /= m
    gradient *= self.lr
    gradient += ((self.landa * w)/m)
    w -= gradient
    return w
```

```python
def fit(self, Xc, yc,testData=None,dropCount = 0):
    self.X = Xc
    self.weights = np.zeros((Xc.shape[1], 1))
    self.y = yc.reshape(yc.shape[0], 1)
    count = 0
    for i in range(self.iter):
        loss = self.cost_function(self.X,self.y)
        if(i > 0):
            if((loss - self.hist["lossTrain"][-1] ) > 0.1):
                if(count == dropCount):
                    break
                count += 1
                self.lr /= 10

        self.hist["lossTrain"].append(loss)
        self.hist["accuTrain"].append(self.score(self.y, np.dot(self.X,self.weights)))
        if testData is not None:
            x_test,y_test = testData
            self.hist["lossTest"].append(self.cost_function(x_test,y_test))
            self.hist["accuTest"].append(self.score(y_test, np.dot(x_test,self.weights)))
        self.weights = self.update_weight(self.weights)

    return self.weights
```

```python
def cost_function(self,X,y):
    N = len(y)
    pred = np.dot(X, self.weights)
    sq_error = (pred - y)**2
    p =self.landa *(np.power(self.weights[1::],2).sum())
    return 1.0/(2*N) * sq_error.sum() + (p/2*N)
```

Score

Regression	
'explained_variance'	metrics.explained_variance_score
'max_error'	metrics.max_error
'neg_mean_absolute_error'	metrics.mean_absolute_error
'neg_mean_squared_error'	metrics.mean_squared_error
'neg_root_mean_squared_error'	metrics.mean_squared_error
'neg_mean_squared_log_error'	metrics.mean_squared_log_error
'neg_median_absolute_error'	metrics.median_absolute_error
'r2'	metrics.r2_score
'neg_mean_poisson_deviance'	metrics.mean_poisson_deviance
'neg_mean_gamma_deviance'	metrics.mean_gamma_deviance
'neg_mean_absolute_percentage_error'	metrics.mean_absolute_percentage_error

```python
# Error class
class error():
  def __init__(self, y_true, y_pre):
    self.y_true = y_true
    self.y_pre = y_pre
    self.len = len(y_true)

  def mean_abs_err(self):
    return np.sum(np.abs(self.y_true - self.y_pre)) / self.len

  def max_err(self):
    return np.max(self.y_true - self.y_pre)

  def r2_score(self):
    u = (self.y_true - self.y_pre) ** 2
    v = (self.y_true - np.average(self.y_true)) ** 2
    return np.average(1 - (u.sum(axis=0) / v.sum(axis=0)))

  def explain_cor(self):
    y_diff_avg = np.average(self.y_true - self.y_pre, axis=0)
    u = np.average((self.y_true - self.y_pre - y_diff_avg) ** 2, axis=0)

    y_true_avg = np.average(self.y_true, axis=0)
    v = np.average((self.y_true - y_true_avg) ** 2, axis=0)

    return np.average(1 - (u.sum(axis=0) / v.sum(axis=0)))
from sklearn.metrics import r2_score
```

R2 Score

"...the proportion of the variance in the dependent variable that is predictable from the independent variable(s)."

R^2 is defined as $(1 - \frac{u}{v})$

u is the residual sum of squares `((y_true - y_pred)** 2).sum()`
v is the total sum of squares `((y_true - y_true.mean()) ** 2).sum()`

3. Optimize

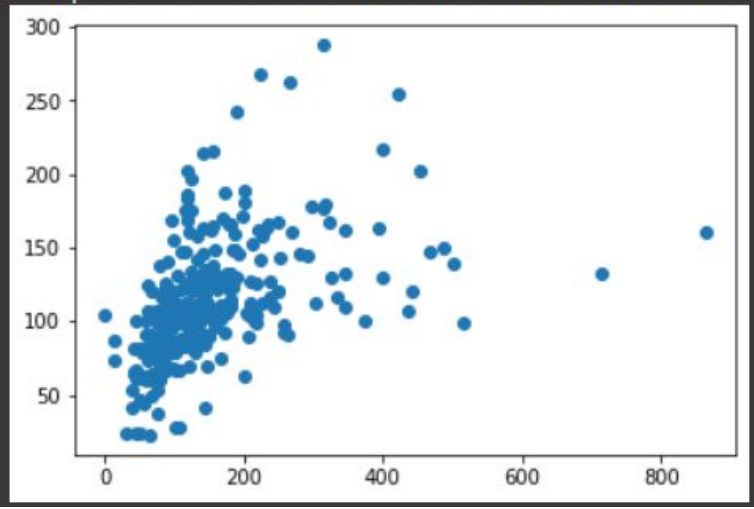
Optimize Step

1. Evaluate
2. Analysis
3. Adjust

Evaluate

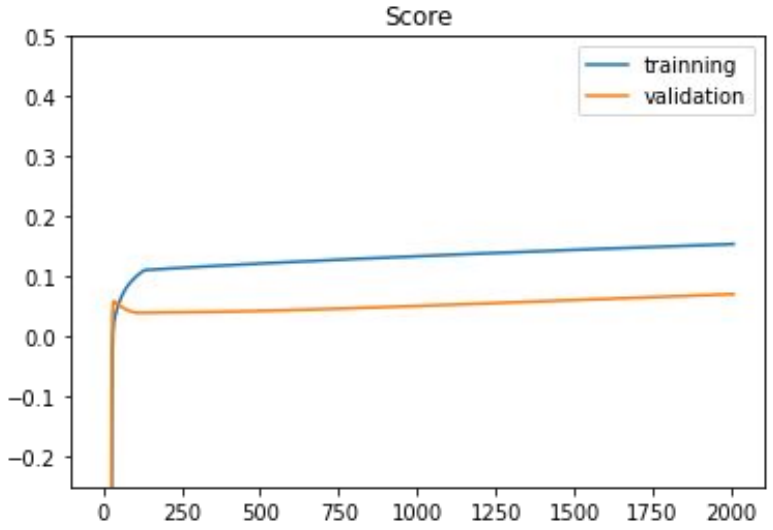
	Coefficient
Floors	17.993847
RoadIn	2.819143
bathrooms	2.443736
roadwidth	0.345596
weights	0.000000
area	-0.097842
bedrooms	-0.847092

```
Correlation of module:  0.5025101311357947
<matplotlib.collections.PathCollection at 0x7fd09253a150>
```

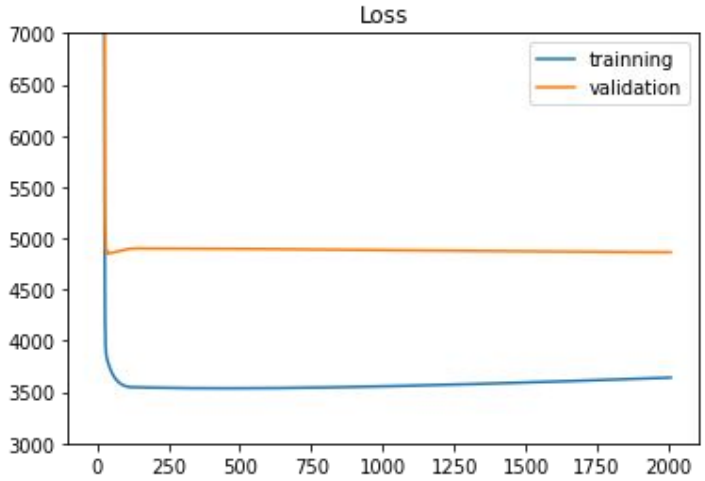


+ Số tầng có trọng số cao nhất (áp đảo)
+ Độ tương quan giá trị dự đoán và giá trị thực của tập test là (0.5)

Đánh giá : không quá tệ



R2 Score



Loss

+ Loss training vẫn cao (high bias)
+ Loss test có high varience , không thể xuống thấp được nữa

=> **Overfit, data xấu**

Regularized linear regression

Overall: Regularization giúp giảm độ lớn của các theta đi nhưng vẫn giữ nguyên feature, làm giảm sức ảnh hưởng của một số feature khiến Module đơn giản hơn, tránh được overfit

---

$$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2\right) + \frac{\lambda}{2m}\left(\sum_{j=1}^{n}\theta_j^2\right)$$

Loss

```
def cost_function(self,X,y):
    N = len(y)
    pred = np.dot(X, self.weights)
    sq_error = (pred - y)**2
    p =self.landa*(
    ... np.power(self.weights[1::],2).sum())
    return 1.0/(2*N) * sq_error.sum() + (p/2*N)
```

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j$$

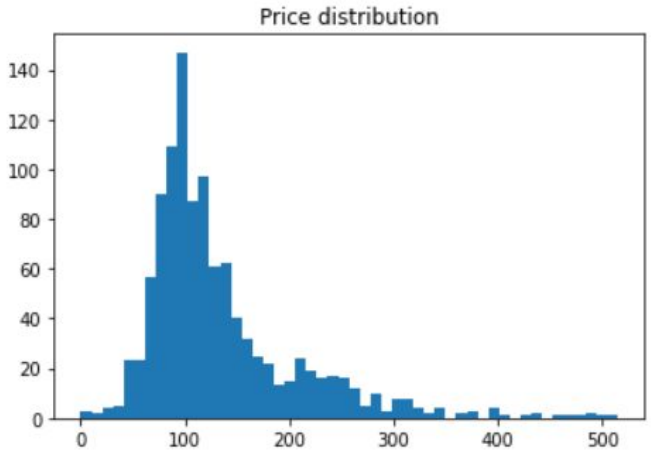
GRADIENT

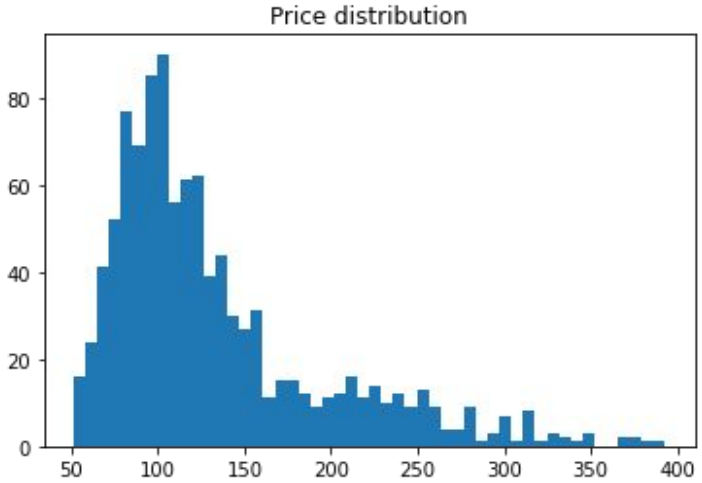
```
def update_weight(self, w):
    m = len(self.X)
    predictions = np.dot(self.X, w)
    error = self.y - predictions
    gradient = np.dot(-self.X.T,  error)
    gradient /= m
    # add landa ()
    gradient *= self.lr
    gradient += ((self.landa * w)/m)
    w -= gradient
    return w
```

Early Stop & Drop learning Rate

+ Trong những biểu đồ đánh giá cũ, Loss và score có dấu hiệu bị bão hòa không tăng lên được nữa dùng drop learning rate để thay đổi chiến thuật học
+ Early stop dùng để tăng tốc độ mô hình khi mà dù có học thêm, accuracy cũng không tăng
+ Drop sẽ được áp dụng trước -> early stop

```
count = 0
    for i in range(self.iter):
      loss = self.cost_function(self.X,self.y)
      if(i > 0):
        if((loss - self.hist["lossTrain"][-1] ) > 0.1):
          if(count == dropCount):
            break
          count += 1
          self.lr /= 10
```






```
dataUse = dataUse[dataUse['price'] < 400][dataUse['price'] > 50]

plt.hist(dataUse['price'], bins=50)
plt.title('Price distribution')
plt.show()
```

Tương quan


price
floors
locateID
front
roadwidth
bedrooms
bathrooms
area
1 0.35 0.42 0.075 0.38 0.19 0.24 0.095
0.35 1 0.19 -0.042 -0.065 0.53 0.58 -0.031
0.42 0.19 1 0.024 -0.07 0.13 0.14 -0.063
0.075 -0.042 0.024 1 0.18 0.074 0.056 0.23
0.38 -0.065 -0.07 0.18 1 0.045 0.055 0.38
0.19 0.53 0.13 0.074 0.045 1 0.92 0.33
0.24 0.58 0.14 0.056 0.055 0.92 1 0.29
0.095 -0.031 -0.063 0.23 0.38 0.33 0.29 1
price
floors
locateID
front
roadwidth
bedrooms
bathrooms
area
1.0
0.8
0.6
0.4
0.2
0.0

Thêm địa điểm?

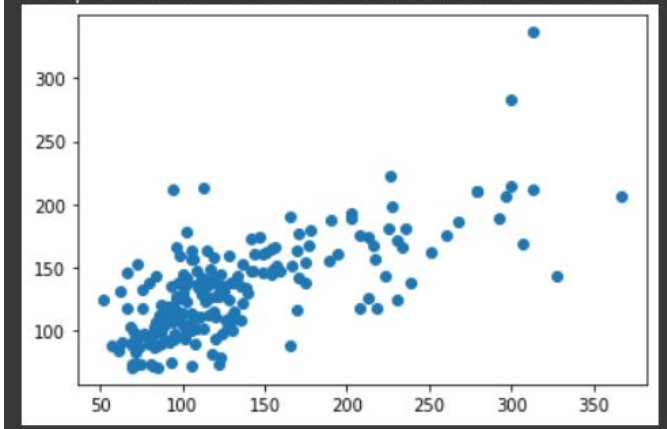
- Đổi địa điểm thành trọng số
- Trọng số được xếp hạng tăng dần theo mean Price của mỗi địa điểm

	price	locateID	roadwidth	floors	bathrooms	bedrooms
0	79.4	5	2.0	3.0	2.0	3.0
1	300.0	5	15.0	8.0	8.0	6.0
2	278.6	5	18.0	8.0	9.0	9.0
3	143.4	5	6.0	5.0	5.0	5.0
5	181.8	5	10.0	7.0	12.0	14.0
...	...	...	...	...	...	...
1080	106.7	7	2.0	4.0	4.0	5.0
1081	92.5	7	2.5	4.0	3.0	4.0
1082	123.5	7	10.0	5.0	4.0	5.0
1083	117.9	7	4.0	7.0	25.0	24.0
1084	76.2	7	3.0	3.0	2.0	3.0

1036 rows × 6 columns

Evaluate

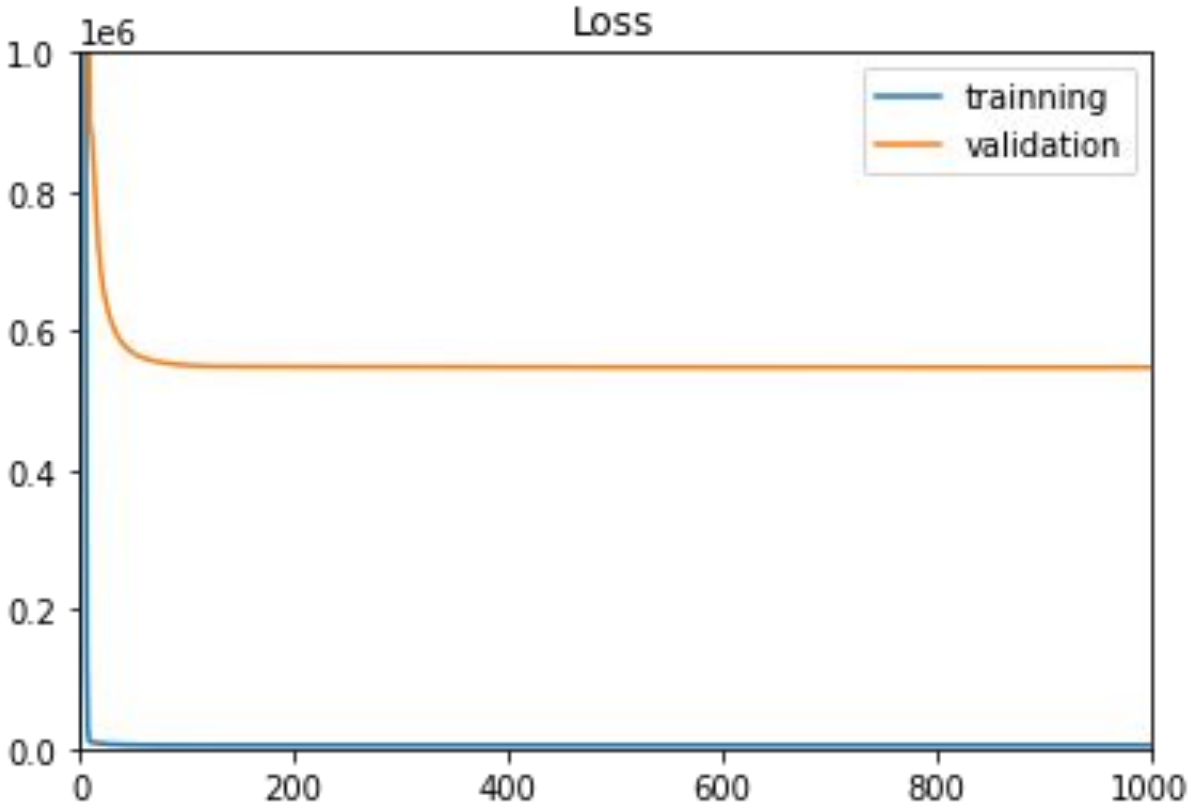
```
Correlation: 0.709159739598992
<matplotlib.collections.PathCollection at 0x7fd08f274250>
```



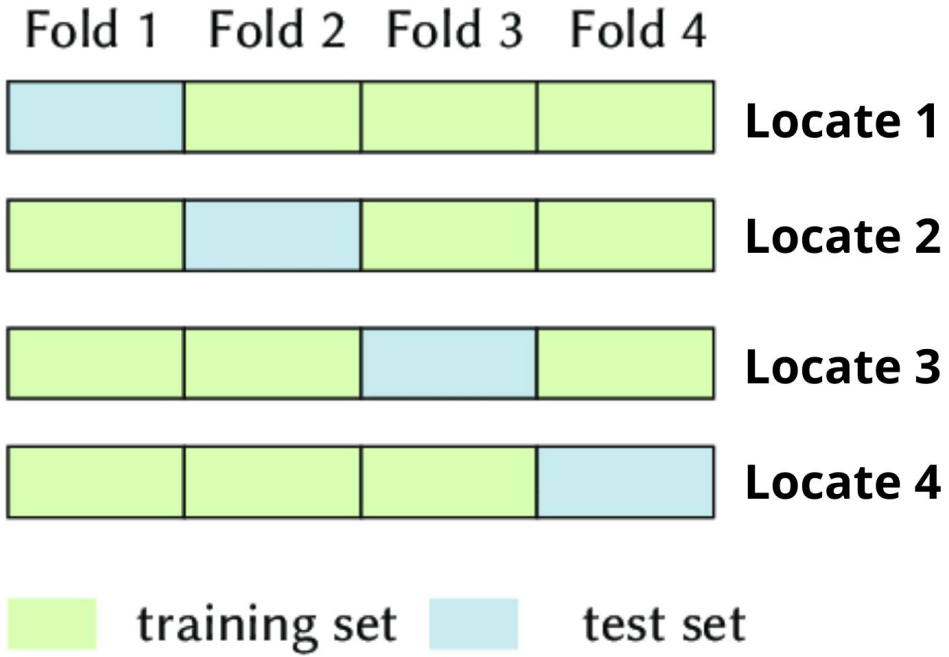
	Coefficient
floors	12.120792
locateID	8.022859
roadwidth	3.060225
weight	1.916188
bathrooms	1.797843
bedrooms	-0.225408

```
Correlation of ourModule 0.709685244341431
Correlation of skLearnModule 0.693981936903701
```

	y_true	y_pre_ourmodel	y_pre_skmodel
0	128.1	120.457354	122.769525
1	81.0	88.140509	89.641461
2	87.1	99.636819	100.780072
3	79.4	79.850652	89.779645
4	180.3	158.114593	162.920613
5	125.6	147.093752	153.956762
6	122.7	142.655767	149.346787
7	300.0	178.154412	196.594587
8	110.0	130.760383	134.508662
9	227.3	192.779316	203.761490


Loss
1e6
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
trainning
validation

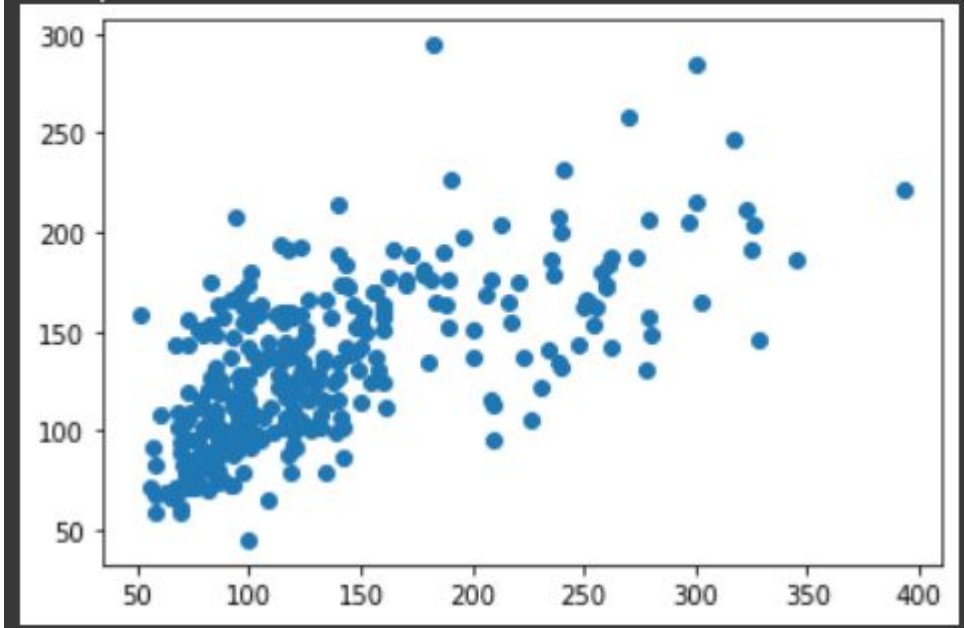
Điều chỉnh Train Test,Split



Evaluate

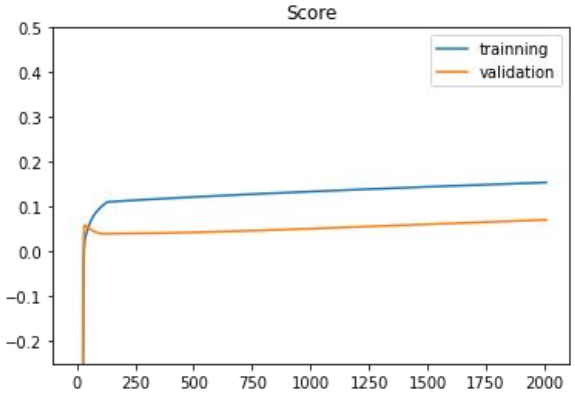
	Coefficient
floors	11.677083
locateID	8.478403
roadwidth	3.116607
weight	1.994058
bathrooms	1.377601
bedrooms	0.256675

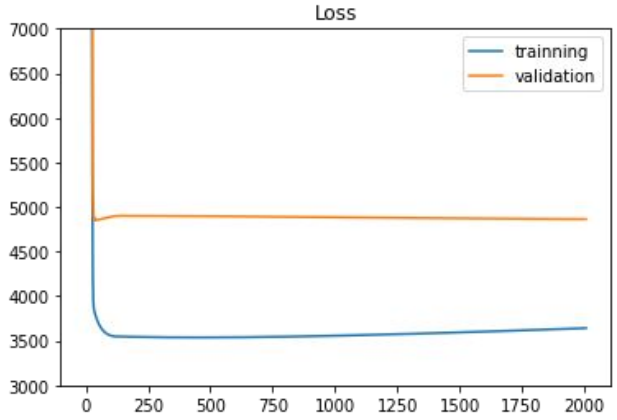
```
[[1.         0.64428781]
 [0.64428781 1.        ]]
<matplotlib.collections.PathCollection at 0x7f809672
```



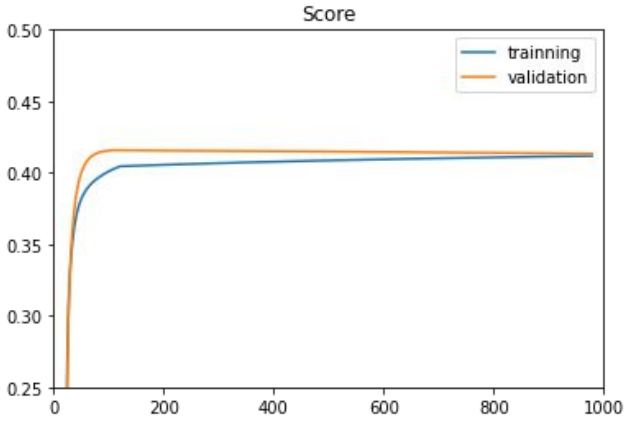
R2 Score:0.41

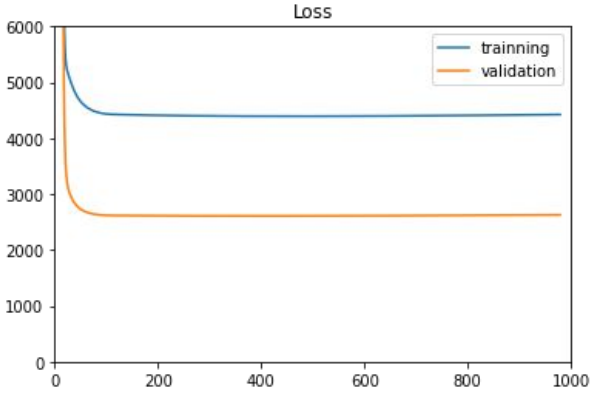
Before





After





Do you have any questions?