

Nguyễn Xuân Anh 20183480 (trưởng nhóm) Nguyễn Minh Đức 20183500 Vũ Quang Đại 20172993 Pathana Peungnthoung 20167995	BTL
--	-----

# Nhóm DataA

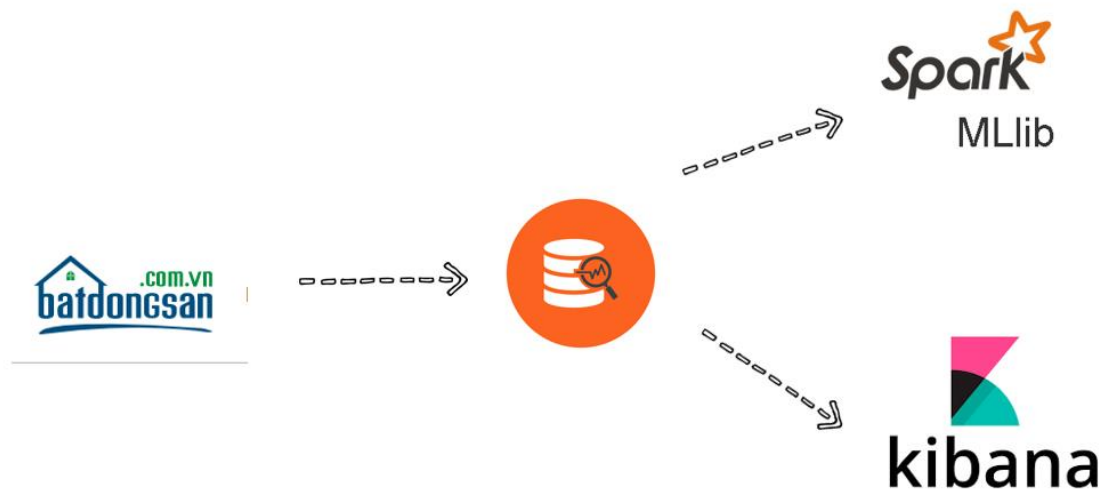
PHÂN TÍCH DATA VÀ TẠO DỰ  
ĐOÁN DỰA VÀO SPARKML

## Mở đầu:

Trong khi chúng ta bước vào kỷ nguyên dữ liệu lớn, nơi chúng ta xử lý lượng lớn dữ liệu mỗi ngày, thế giới đang đồng thời hướng tới Học máy và AI. Trong những năm gần đây, với sự gia tăng số lượng dữ liệu được tạo ra mỗi ngày, học máy đã chứng kiến một sự phát triển vượt bậc về tầm quan trọng và mức độ phổ biến của nó trong hầu hết mọi lĩnh vực. Đặc biệt là trong thời đại ngày nay, hầu hết các doanh nghiệp kinh doanh đang cần xử lý một lượng lớn dữ liệu để hiểu rõ hơn bằng cách chuyển dữ liệu thành kinh doanh thông qua phân tích để tìm ra những hiểu biết có ý nghĩa để đối phó với rủi ro và cơ hội nhằm có lợi thế cạnh tranh việc kinh doanh. Học máy đóng một vai trò quan trọng trong quá trình này.

Các ngành sử dụng Apache Spark được triển khai trên các cụm lớn để phân tích dữ liệu quy mô lớn bằng cách tạo các đường ống học máy thống nhất. Điều này có thể thực hiện được với thư viện MLlib của Spark tạo điều kiện cho việc học máy trên Spark. PySpark MLlib là thư viện học máy của Spark và hoạt động như một trình bao bọc trên lõi PySpark cung cấp một bộ API hợp nhất cho máy học để thực hiện phân tích dữ liệu bằng cách sử dụng phân tán các thuật toán phân chia máy khác nhau như phân loại, hồi quy, phân cụm

## Flow:



# Crawl Data

## Nơi lấy data

Mua Bán Nhà Đất Việt Nam Giá X

batdongsan.com.vn/nha-dat-ban

batdongsan

Nhà đất bánNhà đất cho thuêDự ánCần mua - Cần thuêTin tứcNội - Ngoại thấtPhong thủyDanh bạ

Đài

BánCho thuê

Q Tìm kiếm địa điểm, khu vực

Loại nhà đấtCăn hộ chung cư

Khu vựcToàn quốc

Mức giáTất cả

Diện tíchTất cả

Dự ánTất cả


Mua bán nhà đất toàn quốc

Hiện có 169,069 bất động sản.

Danh sáchBản đồ

Sắp xếp

NỔI BẬT



★ BIỆT THỰ ĐÔI KHU C XANH VILLAS VIEW HỒ THUNG LŨNG NGỌC LINH DT 195M-250M-400M2.GIÁ TỪ 7 TỶ


6 tỷ · 200 m<sup>2</sup> · 5 PN · 4 WC

Thạch Thất, Hà Nội

\* Quý căn hot khu C chính thức ra hàng 19h ngày 19/5/2021.4 Phân khu C9, C10, C11, C12. Liên hệ PKD DA Xanh Villas: 098 116 2\*\*\* - Trực tiếp của PKD DA XANH VILLAS- Cam kết luôn lấy được quý căn...

Hôm nayDa Việt0981 162 \*\*\* · Hiện số

NỔI BẬT



★ BÁN ĐẤT KHU ĐÔ THỊ CIENCO 5 MÊ LINH SỐ ĐỎ GIÁ TỐT CHỈ 1, X TỶ, MẶT ĐƯỜNG 24M, GẦN ĐẠI HỌC TÀI CHÍNH

19.5 triệu/m<sup>2</sup> · 100 m<sup>2</sup>

Mê Linh, Hà Nội

1. Liên kê 16 - Khu A, diện tích 155m2. 2. Liên kê 09 - Khu A, diện tích 100m2. 3. Liên kê 01 - Khu B, diện tích 100m2. 4. Liên kê 02 - Khu B, diện tích 100m2. 5. Liên kê 03 - Khu B, diện tích 100m2. 6. Liên kê 06 ...

Hôm nayĐào Quang 24/70961 225 \*\*\* · Hiện số

< 500 triệu

500 - 800 triệu

800 triệu - 1 tỷ

1 - 2 tỷ

2 - 3 tỷ

3 - 5 tỷ

5 - 7 tỷ

7 - 10 tỷ

10 - 20 tỷ

20 - 30 tỷ

> 30 tỷ

Lọc theo diện tích

<= 30 m2

30 - 50 m2

50 - 80 m2

80 - 100 m2

100 - 150 m2

150 - 200 m2

200 - 250 m2

250 - 300 m2

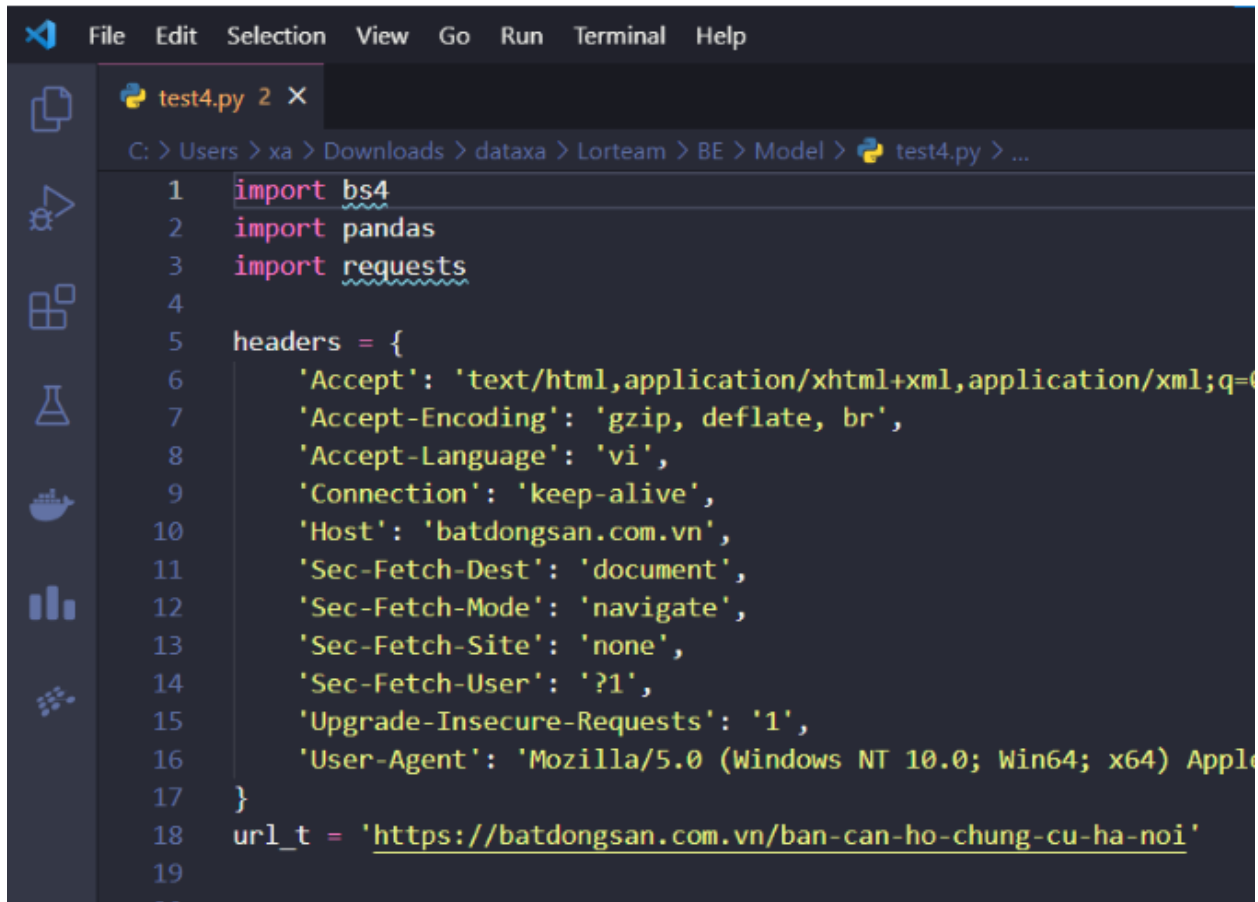
300 - 500 m2

>= 500 m2

Chat với Batdongsan.com.vn

## Thư viện BS4

Nhóm em sử dụng thư viện BeautifulSoup:



```
File Edit Selection View Go Run Terminal Help
test4.py 2 X
C: > Users > xa > Downloads > dataxa > Lorteam > BE > Model > test4.py > ...
1 import bs4
2 import pandas
3 import requests
4
5 headers = {
6     'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
7     'Accept-Encoding': 'gzip, deflate, br',
8     'Accept-Language': 'vi',
9     'Connection': 'keep-alive',
10    'Host': 'batdongsan.com.vn',
11    'Sec-Fetch-Dest': 'document',
12    'Sec-Fetch-Mode': 'navigate',
13    'Sec-Fetch-Site': 'none',
14    'Sec-Fetch-User': '?1',
15    'Upgrade-Insecure-Requests': '1',
16    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4398.96 Safari/537.36'
17 }
18 url_t = 'https://batdongsan.com.vn/ban-can-ho-chung-cu-ha-noi'
19
20
```

Lưu dữ liệu:

```
short_detail_2 = sub_soup.find( 'div', class_ = 'short-detail-2 list2 clearfix' )
time = short_detail_2.findAll('span', class_='sp3')
ngaydang = time[0].text
ngayhet = time[1].text
phone = sub_soup.find('span', class_='phoneEvent').get('raw')
coordinate_div = sub_soup.find('div', class_ = 'map')
if(coordinate_div is not None):
    coordinate_iframe = coordinate_div.find('iframe').attrs['src']
    head = coordinate_iframe.index('q=') + 2
    tail = coordinate_iframe.index('&key')
    coordinate = coordinate_iframe[head:tail].split(",")
    latitude = coordinate[0]
    longitude = coordinate[1]
    print(latitude,longitude)
df = df.append(pandas.DataFrame([
    'title': pandas.Series(title),
    'Địa chỉ': pandas.Series(location),
    'Giá': pandas.Series(price),
    'Diện tích': pandas.Series(area),
    'Số phòng ngủ': pandas.Series(nobed),
    'Số toilet': pandas.Series(sotoilet),
    'Mô tả': pandas.Series(describe),
    'Chủ đầu tư': pandas.Series(chudautu),
    'Quy mô': pandas.Series(quymo),
    'Ngày đăng': pandas.Series(ngaydang),
    'Ngày hết': pandas.Series(ngayhet),
    'Liên hệ': pandas.Series(phone)])))
```

⇒ Bọn em lưu trữ 11.000 bản ghi

## Tiền xử lý dữ liệu

```
1 data.loc[ (data['SoPhongNgu'] ==0) & (data['DienTich'] > 80) & (data['DienTich']<=90), 'SoPhongNgu'] = 2

1 data.loc[ (data['SoPhongNgu'] ==0) & (data['DienTich'] > 90) & (data['DienTich']<=160), 'SoPhongNgu'] = 3

1 data.loc[ (data['SoPhongNgu'] ==0) & (data['DienTich'] < 220), 'SoPhongNgu'] = 4

+ Code + Markdown

1 data.loc[ (data['SoPhongNgu'] ==0) & (data['DienTich'] < 300), 'SoPhongNgu'] = 5

1 data.loc[ (data['SoPhongNgu'] ==0)] = 6

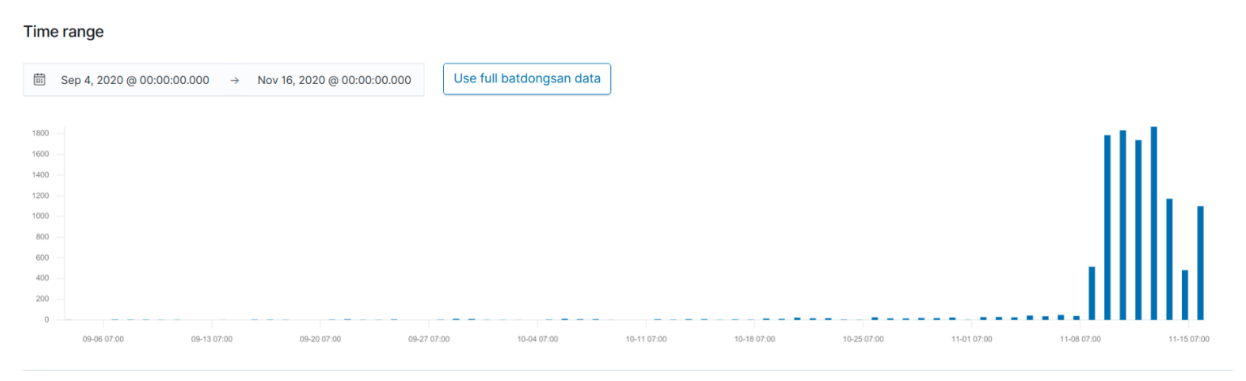
1 data['SoPhongNgu'].value_counts()

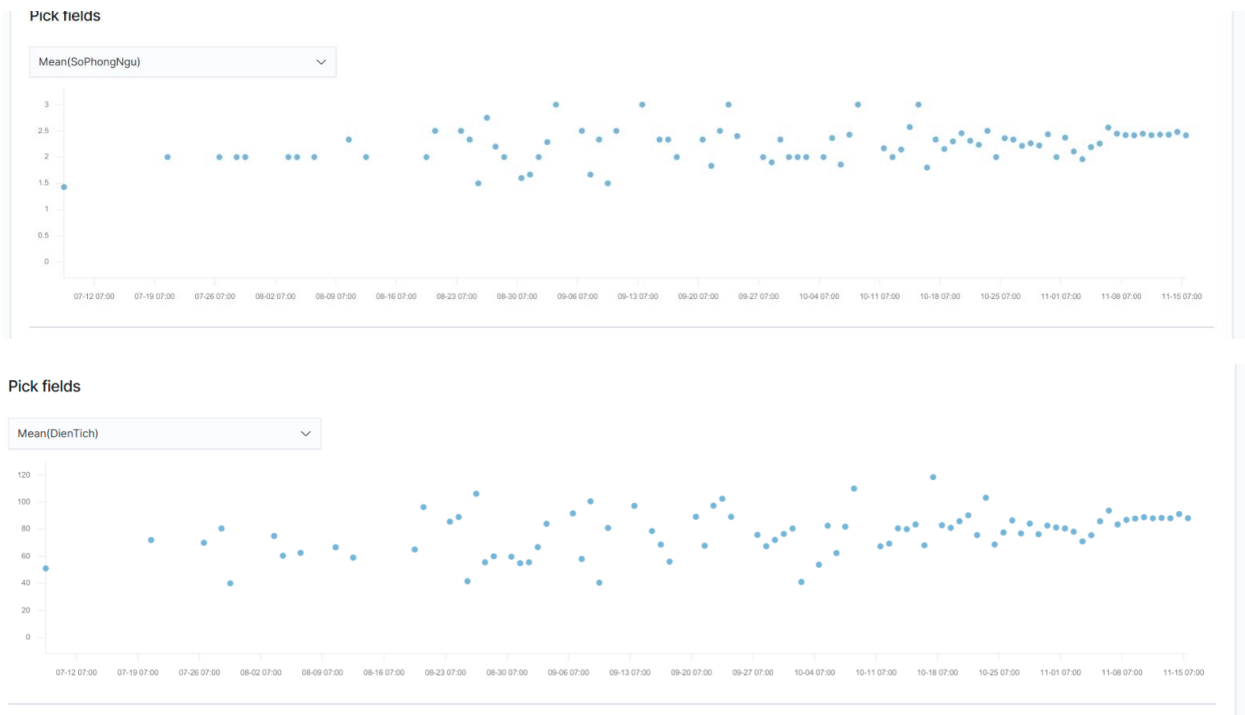
2 5646
3 4289
1 712
```

## Visualize data bằng Kibana

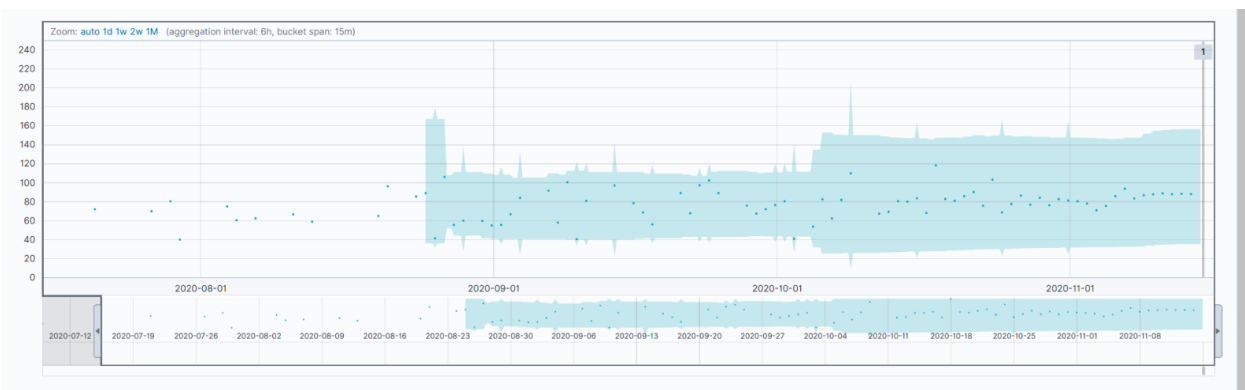
Tổng quan dữ liệu:

Data thu được tập trung nhiều vào tháng 11:

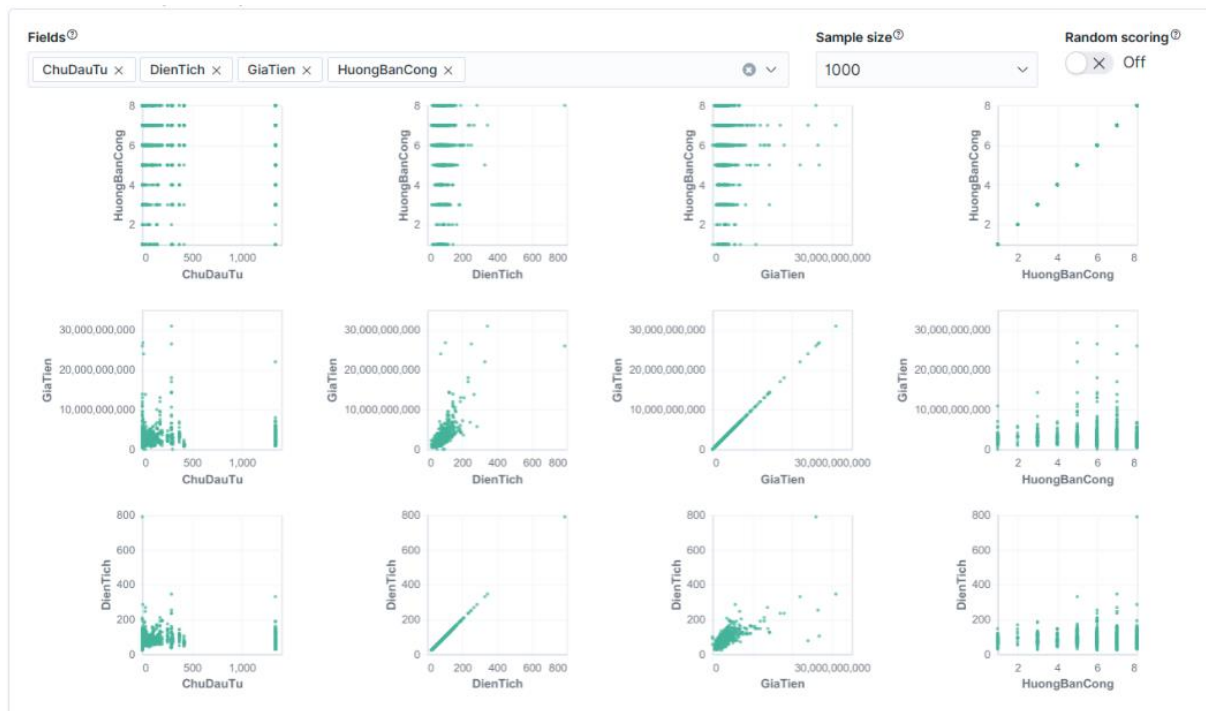




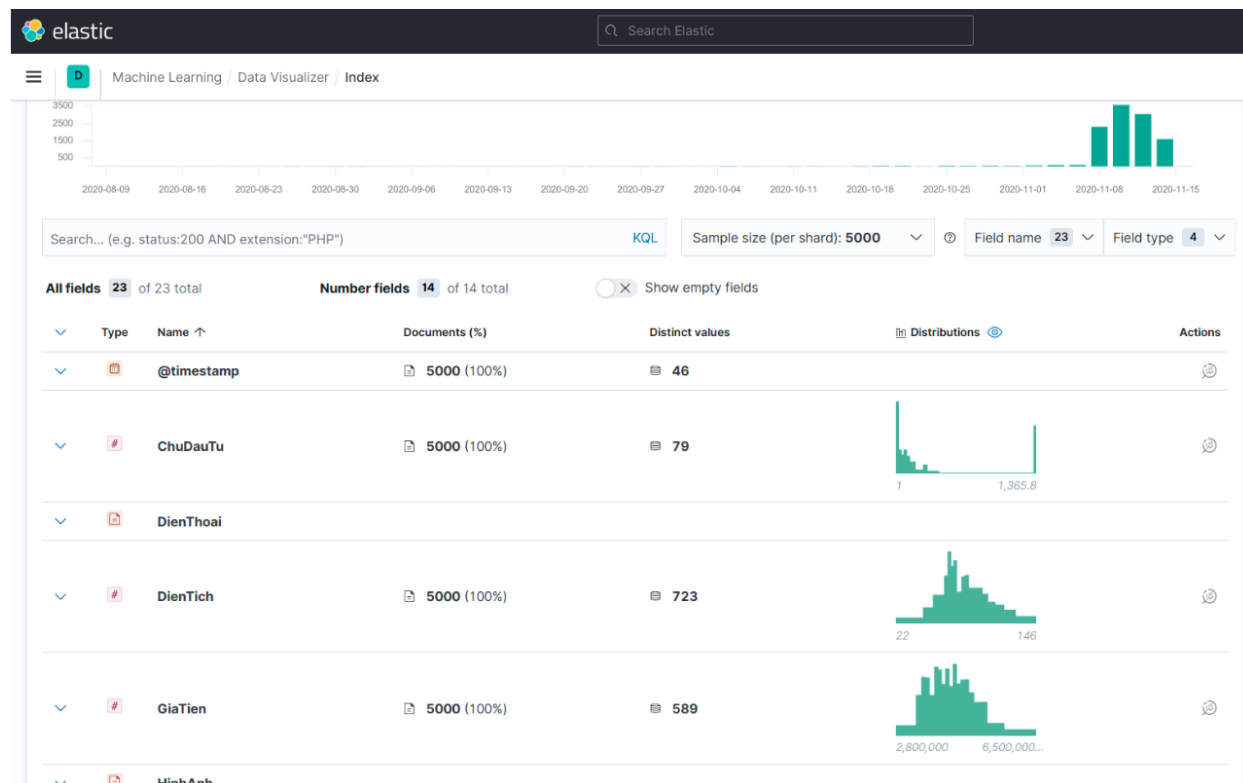
Đi sâu hơn vào Diện Tích:



Tương quan giữa các cột giữa liệu để tìm ra các dữ liệu bị outline ( ảo ).

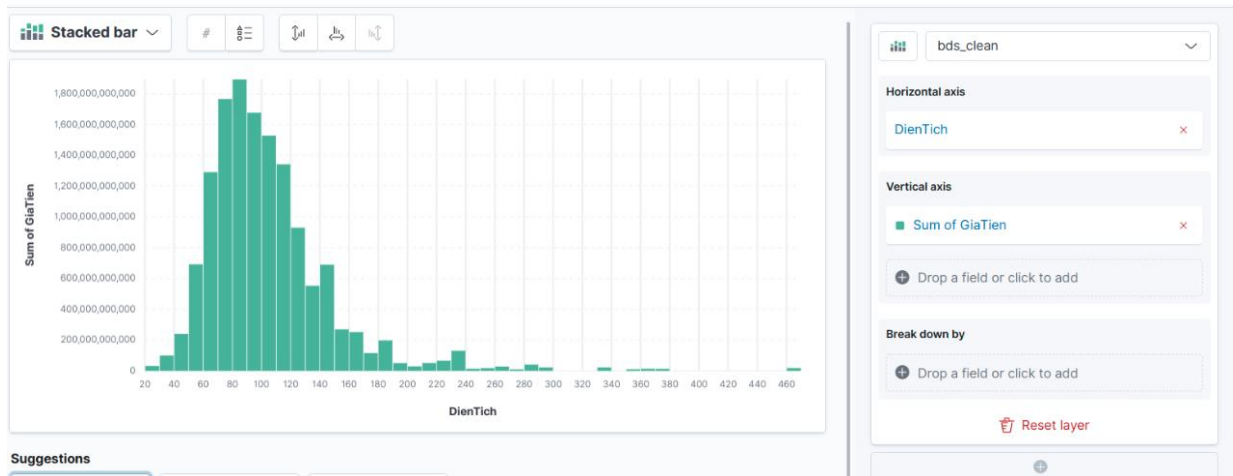


Sau khi làm sạch:



Tìm ra Phân phức diện tích chiếm tỷ trọng cao nhất





Số phòng ngủ dựa trên giá tiền và diện tích



Dùng SparkML

Data exploration and preprocessing

Tạo Spark Context

```
[225] from pyspark.sql import SparkSession

# Build the SparkSession
spark = SparkSession.builder \
    .master("local") \
    .appName("Linear Regression Model") \
    .config("spark.executor.memory", "1gb") \
    .getOrCreate()

sc = spark.sparkContext
```

Bước đầu tiên và cũng là một trong những bước quan trọng nhất là xử lý trước dữ liệu trong đó dữ liệu được chuyển đổi thành dữ liệu sạch và có thể sử dụng được để nó khả thi cho việc phân tích và đào tạo mô hình. Dữ liệu được sử dụng sẽ quyết định đáng kể hiệu suất của mô hình học máy. Do đó, việc thăm dò dữ liệu, làm sạch, xử lý trước và kỹ thuật tính năng giúp cải thiện hiệu suất mô hình.

### Load data csv

Chúng ta có thể tạo trực tiếp dataframe PySpark bằng cách đọc dữ liệu từ CSV bằng phương thức `spark.read.csv()`. `Header = True` để có hàng đầu tiên của CSV làm tiêu đề

```
[226] myDF = spark.read.csv('/content/drive/MyDrive/Tài liệu/20202/Lưu trữ và xử lý dữ liệu lớn/Data_train (1).csv', header=True)
```

Bây giờ dữ liệu đã được tải, chúng ta có thể sử dụng phương thức `printSchema()` để hiểu lược đồ của khung dữ liệu. Bạn cũng có thể lấy tiêu đề bằng cách sử dụng `df.columns` trả về tên cột.

```
▶ myDF.printSchema()
myDF.columns
```

```
root
|-- _c0: string (nullable = true)
|-- Quan: string (nullable = true)
|-- GiaTien: string (nullable = true)
|-- DienTich: string (nullable = true)
|-- SoPhongNgu: string (nullable = true)
|-- SoToilet: string (nullable = true)
|-- HuongNha: string (nullable = true)
|-- HuongBanCong: string (nullable = true)
|-- NoiThat: string (nullable = true)
|-- ChuDauTu: string (nullable = true)
|-- ViDo: string (nullable = true)
|-- KinhDo: string (nullable = true)
|-- LoaiTin: string (nullable = true)

['_c0',
'Quan',
'GiaTien',
'DienTich',
'SoPhongNgu',
'SoToilet',
'HuongNha',
'HuongBanCong',
'NoiThat',
'ChuDauTu',
'ViDo',
'KinhDo',
'LoaiTin']
```

## Chuyển kiểu dữ liệu

Ở đây, bạn có thể thấy rằng tất cả các cột đều có kiểu chuỗi. Nhưng để phân tích hiệu quả hơn, chúng ta sẽ cần chuyển đổi các cột thành kiểu dữ liệu float. Để làm điều đó, chúng ta sẽ tạo một hàm chuyển đổi kiểu dữ liệu của từng cột trong khung dữ liệu thành float. Hàm `withColumn()` được sử dụng để thao tác (đổi tên, thay đổi giá trị, chuyển đổi kiểu dữ liệu) một cột hiện có trong khung dữ liệu hoặc để tạo một cột mới.

```
[228] from pyspark.sql.types import *
      def convertColDatatype(df, colNames, newDatatype):
          for colName in colNames:
              df= df.withColumn(colName, df[colName].cast(newDatatype))
          return df
      columns= myDF.columns
      myDF= convertColDatatype(myDF, columns, FloatType())
      myDF.printSchema()

root
 |-- _c0: float (nullable = true)
 |-- Quan: float (nullable = true)
 |-- GiaTien: float (nullable = true)
 |-- DienTich: float (nullable = true)
 |-- SoPhongNgu: float (nullable = true)
 |-- SoToilet: float (nullable = true)
 |-- HuongNha: float (nullable = true)
 |-- HuongBanCong: float (nullable = true)
 |-- NoiThat: float (nullable = true)
 |-- ChuDauTu: float (nullable = true)
 |-- ViDo: float (nullable = true)
 |-- KinhDo: float (nullable = true)
 |-- LoaiTin: float (nullable = true)
```

Mười giá trị nhà cao nhất cho các khối.

```
print(myDF.count())
myDF.sort('GiaTien',ascending=False)\
.select('GiaTien').distinct().show(10)
```

```
11164
+-----+
|   GiaTien|
+-----+
|3.09900006E9|
|2.87500006E9|
|   3.08E9|
|3.99100006E9|
|   3.12E9|
|2.70499994E9|
|   4.26E9|
|   9.0E8|
|   7.6E8|
|   1.33E9|
+-----+
only showing top 10 rows
```

Kiểm tra giá trị rỗng trong file csv. Bạn có thể thấy rằng chúng tôi không có bất kỳ giá trị null nào trong khung dữ liệu của chúng tôi.

```
[281] from pyspark.sql.functions import *
myDF.select([count(when(isnull(c), c)).alias(c) for c in myDF.columns]).show()
```

_c0	Quan	GiaTien	DienTich	SoPhongNgu	SoToilet	HuongNha	HuongBanCong	NoiThat	ChuDauTu	ViDo	KinhDo	LoaiTin
0	0	0	0	0	0	0	0	0	0	0	0	0

Ở đây, chúng ta sẽ sử dụng phương thức select () để chọn và sắp xếp lại các cột cần thiết trong dataframe.

```
[282] myDF= myDF.select('GiaTien', 'DienTich', 'SoPhongNgu', 'SoToilet', 'NoiThat', 'Quan', 'HuongNha', 'HuongBanCong', 'ChuDauTu', 'ViDo', 'KinhDo', 'LoaiTin')
```

Bây giờ chúng ta đã sắp xếp lại thứ tự các cột trong khung dữ liệu, đã đến lúc chúng ta tách biến mục tiêu (median\_house\_value) là cột đầu tiên trong khung dữ liệu khỏi các biến độc lập (các cột khác với biến mục tiêu).

```
[283] from pyspark.ml.linalg import DenseVector
#x[0] is the target variable (label) and x[1:] are the features
data = myDF.rdd.map(lambda x: (x[0], DenseVector(x[1:])))
#creating a dataframe with columns 'label' (target variable) and #'features'(dense vector of independent variables)
myDF = spark.createDataFrame(data, ['label', 'features'])
myDF.printSchema()

root
 |-- label: double (nullable = true)
 |-- features: vector (nullable = true)
```

Như chúng ta đã thấy rằng phạm vi giá trị trong một vài cột là lớn, chúng ta phải chuẩn hóa dữ liệu bằng cách sử dụng StandardScaler để biến đổi dữ liệu sao cho giá trị trung bình là 0 và độ lệch chuẩn là 1. Ở đây, các cột đầu vào là các tính năng được chia tỷ lệ và cột đầu ra sẽ là các tính năng được chia tỷ lệ kết quả sẽ được đưa vào làm cột thứ ba trong khung dữ liệu.

```
[284] from pyspark.ml.feature import StandardScaler
ss= StandardScaler(inputCol='features', outputCol='scaled_features')
scaler = ss.fit(myDF)
myDF = scaler.transform(myDF)
myDF.printSchema()
myDF.take(1)

root
 |-- label: double (nullable = true)
 |-- features: vector (nullable = true)
 |-- scaled_features: vector (nullable = true)
```

## Model Linear Regression

Chia dữ liệu đã được xử lý trước thành các tập huấn luyện và thử nghiệm bằng cách sử dụng `randomSplit()`.

```
▶ train, test = myDF.randomSplit([.8,.2],seed=1)
```

### Khởi tạo model

```
[286] from pyspark.ml.regression import LinearRegression
      linearReg= LinearRegression(featuresCol= 'features', labelCol='label',maxIter=20)
      #fit the model to the the training data
      model=linearReg.fit(train)
```

### Dự đoán giá nhà

```
| 529.3068985952996|420.0|
| 671.7263938231044|420.0|
| 646.0825154962367|430.0|
```

### Đánh giá mô hình

Chúng ta có thể biết mức độ hiệu quả của mô hình bằng cách xem xét một số chỉ số hiệu suất. Để đánh giá hiệu suất của mô hình hồi quy tuyến tính, chúng tôi sẽ sử dụng RMSE (lỗi bình phương trung bình) cung cấp phép đo tuyệt đối của sai số giữa giá trị thực tế và giá trị dự đoán và điểm R2 (R bình phương) cung cấp thông tin về mức độ tốt của phù hợp của mô hình.

```
R2 score on test set:  0.532547989617213
RMSE on test set:  1529.6158607642722
```

Thông thường R2 score trên 0.5 thì mô hình có mức độ tốt phù hợp.

Ngoài ra chúng ta có thể dùng chỉ số MAE:

```
MAE on test set:  814.4847331496717
```

## Decision tree regression

### Khởi tạo mô hình

```
▶ from pyspark.ml.regression import DecisionTreeRegressor
  dt = DecisionTreeRegressor(maxDepth=5, varianceCol="variance", labelCol='label')
  dt_model = dt.fit(train)

  dt_model.featureImportances

SparseVector(11, {0: 0.6715, 2: 0.0532, 4: 0.1021, 7: 0.0391, 8: 0.063, 9: 0.071})
```

**RMSE và MAE của model:**

MAE on test set: 678.4182341668168

 Spark Master at `spark://192.168.1.2:7077`

```
URL: spark://192.168.1.2:7077
Alive Workers: 2
Cores in use: 8 Total, 8 Used
Memory in use: 21.4 GiB Total, 2.0 GiB Used
Resources in use:
Applications: 1 Running, 2 Completed
Drivers: 0 Running, 0 Completed
Status: AI IVE
```

▼ Workers (2)

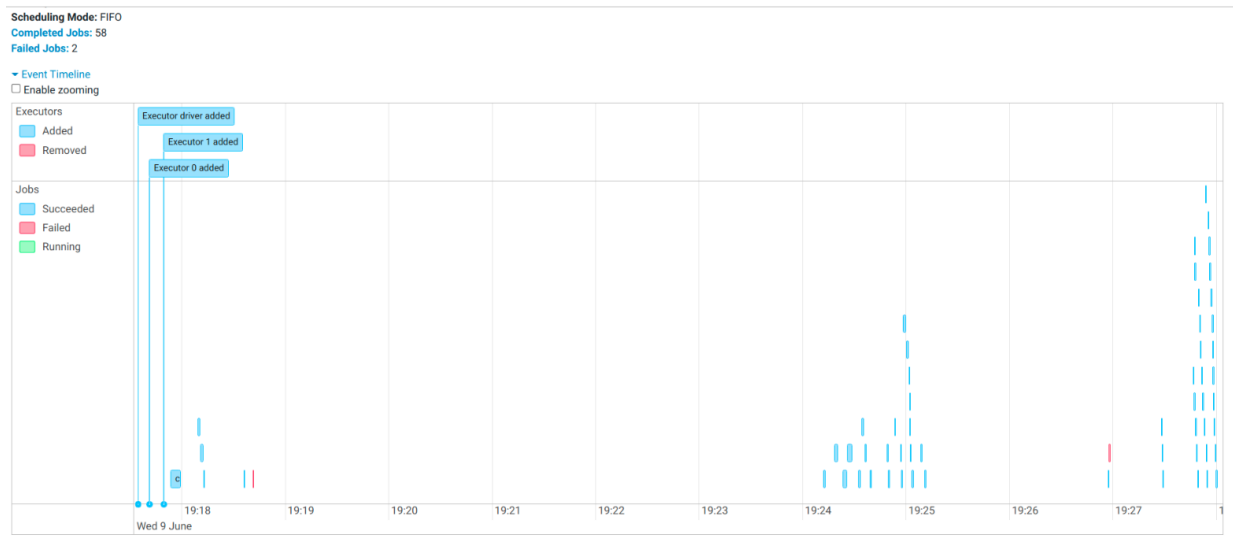
Worker Id	Address	State	Cores	Memory	Resources
worker-20210609184425-192.168.1.2-38423	192.168.1.2:38423	ALIVE	4 (4 Used)	14.5 GiB (1024.0 MiB Used)	
worker-20210609184911-192.168.1.3-56753	192.168.1.3:56753	ALIVE	4 (4 Used)	6.9 GiB (1024.0 MiB Used)	

### ▼ Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210609191734-0002	(kill) DataA	8	1024.0 MiB		2021/06/09 19:17:34	xa	RUNNING	8.4 min

### ▼ Completed Applications (2)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210609185036-0000	DataA	8	1024.0 MiB		2021/06/09 18:50:36	xa	KILLED	25 min
app-20210609191325-0001	DataA	0	1024.0 MiB		2021/06/09 19:13:25	xa	FINISHED	1.9 min



### Spark Jobs (?)

User: xa  
Total Uptime: 12 min  
Scheduling Mode: FIFO  
Completed Jobs: 58  
Failed Jobs: 2

Event Timeline

#### Completed Jobs (58)

Page: 1

1 Pages. Jump to 1. Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
59	treeAggregate at Statistics.scala:58 treeAggregate at Statistics.scala:58	2021/06/09 19:27:59	0.4 s	1/1	1/1 (2 failed)
58	treeAggregate at Statistics.scala:58 treeAggregate at Statistics.scala:58	2021/06/09 19:27:59	0.5 s	1/1	1/1
57	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2021/06/09 19:27:58	0.6 s	1/1	1/1 (1 failed)
56	collectAsMap at RandomForest.scala:663 collectAsMap at RandomForest.scala:663	2021/06/09 19:27:57	0.1 s	2/2	2/2
55	collectAsMap at RandomForest.scala:663 collectAsMap at RandomForest.scala:663	2021/06/09 19:27:57	0.1 s	2/2	2/2
54	collectAsMap at RandomForest.scala:663 collectAsMap at RandomForest.scala:663	2021/06/09 19:27:57	0.2 s	2/2	2/2
53	collectAsMap at RandomForest.scala:663 collectAsMap at RandomForest.scala:663	2021/06/09 19:27:57	0.1 s	2/2	2/2
52	collectAsMap at RandomForest.scala:663 collectAsMap at RandomForest.scala:663	2021/06/09 19:27:56	0.7 s	2/2	2/2 (3 failed)

### Application: DataA

ID: app-20210609191734-0002  
Name: DataA  
User: xa  
Cores: Unlimited (8 granted)  
Executor Limit: Unlimited (2 granted)  
Executor Memory: 1024.0 MiB  
Executor Resources:  
Submit Date: 2021/06/09 19:17:34  
State: RUNNING  
Application Detail UI

#### Executor Summary (2)

ExecutorID	Worker	Cores	Memory	Resources	State	Logs
1	worker-20210609184425-192.168.1.2-38423	4	1024		RUNNING	stdout stderr
0	worker-20210609184911-192.168.1.3-56753	4	1024		RUNNING	stdout stderr