ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

MÔN HỌC: DỮ LIỆU LỚN

Đề tài: DỰ ĐOÁN GIÁ TRỊ VỐN HÓA THỊ TRƯỜNG DỰA TRÊN DOANH SỐ BÁN HÀNG

GVHD: Nguyễn Hồ Duy Trí

1. Nguyễn Thái Bảo	-	20521105
2. Lê Gia Bảo	-	20521101
3. Nguyễn Thị Vân	-	20522144
4. Nguyễn Ngọc Hải Sơn	-	20521846

TP. Hồ Chí Minh, tháng 6/2024

LỜI CẢM ƠN

Nhóm em xin cảm ơn thầy thạc sĩ Nguyễn Hồ Duy Trí (giảng viên môn Dữ liệu lớn) đã tận tình hướng dẫn chúng em trong học kỳ này. Thầy đã truyền đạt hết các công cụ, tài liệu để thực hiện đồ án môn học này.

Trong thời gian thực hiện đề tài, nhóm em đã cố gắng vận dụng những kiến thức nền tảng đã được học đồng thời kết hợp với việc học hỏi và nghiên cứu những kiến thức mới để ứng dụng vào thực hiện đề tài này. Mặc dù đã hết sức cố gắng, nhưng đồ án vẫn còn những thiếu sót. Chính vì vậy, nhóm rất mong nhận được những sự góp ý quý báu từ Thầy nhằm hoàn thiện những kiến thức của và là hành trang để nhóm thực hiện tiếp các đề tài khác trong tương lai.

Xin chân thành cảm ơn thầy!

Nhóm sinh viên thực hiện

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

……, ngày…….....tháng……năm 2024…

Người nhận xét

(Ký tên và ghi rõ họ tên)

MỤC LỤC

DANH MỤC ẢNH

CHƯƠNG 1: MỤC TIÊU ĐỀ TÀI

1.1 Tổng quan đề tài

Hiện nay, chúng ta đang sống trong thời kỳ của cuộc cách mạng công nghiệp lần thứ 4. Đây là thời kỳ gắn với những đột phá về công nghệ cũng như chuyển đổi số trong các doanh nghiệp. Công nghệ thông tin ngày càng phát triển thì đời sống của con người càng cần nhiều ứng dụng để hỗ trợ, phát triển hơn.

Trong bối cảnh thị trường tài chính ngày càng phức tạp và biến động, khả năng dự đoán giá trị vốn hóa thị trường trở thành một yếu tố quyết định quan trọng đối với nhà đầu tư và các doanh nghiệp. Trong nhiều trường hợp, doanh số bán hàng đã được xác định là một trong những yếu tố chính tác động đến giá trị vốn hóa của một công ty. Điều này mở ra cơ hội đối với các nhà nghiên cứu và chuyên gia tài chính để áp dụng các kỹ thuật khoa học dữ liệu và mô hình hóa dự đoán, nhằm hiểu rõ hơn về mối liên quan giữa doanh số bán hàng và giá trị vốn hóa thị trường.

Sự tương quan giữa doanh số bán hàng và giá trị vốn hóa thị trường thường được xem xét như một chỉ báo quan trọng về sức mạnh cạnh tranh, khả năng sinh lời và tiềm năng tăng trưởng của một doanh nghiệp. Mối liên kết này không chỉ thể hiện sự ảnh hưởng của doanh số bán hàng đối với giá trị thị trường của doanh nghiệp mà còn là một bộ chỉ số quan trọng để nhà đầu tư đánh giá rủi ro và cơ hội đầu tư.

Trong đồ án môn học này chúng em sẽ tiến hành tìm hiểu về cách doanh số bán hàng có thể là một yếu tố quyết định đối với giá trị vốn hóa thị trường, và làm thế nào thông tin này có thể được chuyển đổi thành lợi ích cụ thể trong quyết định đầu tư và chiến lược kinh doanh.

1.2 Mục tiêu đề tài

- Hiểu và nắm được dữ liệu bài toán.
- Có thể xử lý, làm sạch dữ liệu ban đầu thành nguồn dữ liệu có giá trị phục vụ giải quyết vấn đề đặt ra.

- Khám phá, phân tích và trực quan hóa tập dữ liệu.
- Tự xây dựng được mô hình Linear Regression để dự đoán giá trị vốn hóa thị trường và xây dựng được mô hình đánh giá hiệu quả, chính xác.
- Tự xây dựng được mô hình Ridge Regression để dự đoán giá trị vốn hóa thị trường
- So sánh, đánh giá được sự hiệu quả, tính chính xác, ưu, nhược điểm giữa các mô hình Linear Regression, Rigde Regression tự xây dựng.

CHƯƠNG 2: NGUỒN VÀ CHI TIẾT TẬP DỮ LIỆU

2.1 Tổng quan dữ liệu

Để có thể thực hiện dự án thì nhóm chúng em chọn 1 tập dữ liệu Dataset thuộc Kaggle được lấy từ cuộc khảo sát của 2000 công ty trên toàn cầu.

- Tên Dataset: Top 2000 Companies Globally
- Kích thước: 137.69 kB
- Loại tệp tin khi tải xuống từ Kaggle: .csv
- Loại tệp tin sau khi giải nén: .csv
- Tần suất cập nhật: Hằng tháng.
- Link Dataset: Link

2.2 Chi tiết dữ liệu

Tập dữ liệu Top 2000 Companies Globally có 1924 dòng từ 2000 công ty lớn trên toàn cầu. Tập dữ liệu gồm 10 thuộc tính bao gồm:

1. Global Rank: Chỉ số xếp hạng trên toàn cầu, dạng Interger (vd: 1, 2, 3...)
2. Company: Tên công ty, dạng String (vd: JPMorgan)
3. Sales ($billion): Doanh thu hoặc doanh số bán hàng, dạng Double (vd: 134.8)
4. Profits ($billlion): Lợi nhuận của công ty, dạng Double (vd: 30.6)
5. Assets ($billion): Giá trị tài sản của công ty, dạng Double (vd: 2359.2)
6. Market Value ($billion): Định giá của công ty trên thị trường, dạng Double (vd: 202.5)
7. Country: Quốc gia nơi công ty đặt trụ sở, dạng String (vd: Vietnam)
8. Continent: Châu lục nơi công ty đặt trụ sở, dạng String (vd: North America)
9. Latitude: Vĩ độ, dạng Double (vd: 35.86166)
10. Longtitude: Kinh độ, dạng Double (vd: 104.195397)

2.2.1 Thêm các thư viện cần thiết.

```
[ ]  1 import matplotlib.pyplot as plt
     2 import seaborn as sns
     3 import pandas as pd
     4 from pandas import *
     5 import numpy as np
     6 import plotly.express as ex
     7 import plotly.graph_objs as go
     8 import plotly.figure_factory as ff
     9 from plotly.subplots import make_subplots
    10 import plotly.offline as pyo
    11 import time
    12 !pip install pyspark
    13 from pyspark.sql import SparkSession
    14 from pyspark.sql.functions import count, when, isnan, col, regexp_replace, length
    15 from pyspark.sql.types import *
```

Hình 1: Thêm thư viện

2.2.2 Đọc dữ liệu.

```
[ ]  1 from google.colab import drive
     2 drive.mount('/content/gdrive')
     3 spark = SparkSession.builder.appName("example").getOrCreate()
     4
     5
     6 file_path = 'gdrive/My Drive/BigData/Final/Top2000CompaniesGlobally.csv'
     7 data = spark.read.format("csv")\
     8          .option("header","true")\
     9          .option("inferSchema","true")\
    10          .load(file_path)
    11 data.count()

Mounted at /content/gdrive
1924
```

Hình 2: Đọc dữ liệu vào

2.2.3 Xem dữ liệu

```
[ ]  1 data.show()
```

```
+-----------+--------------------+---------------+----------------+----------------+----------------------+--------------+-------------+----------+----------+
|Global Rank|             Company|Sales ($billion)|Profits ($billion)|Assets ($billion)|Market Value ($billion)|       Country|    Continent|  Latitude| Longitude|
+-----------+--------------------+---------------+----------------+----------------+----------------------+--------------+-------------+----------+----------+
|          1|                ICBC|          134.8|            37.8|          2813.5|                 237.3|         China|         Asia|  35.86166|104.195397|
|          2|China Constructio...|          113.1|            30.6|          2241.0|                 202.0|         China|         Asia|  35.86166|104.195397|
|          3|      JPMorgan Chase|          108.2|            21.3|          2359.1|                 191.4|           USA|North America|  37.09024| -95.712891|
|          4|    General Electric|          147.4|            13.6|           685.3|                 243.7|           USA|North America|  37.09024| -95.712891|
|          5|         Exxon Mobil|          420.7|            44.9|           333.8|                 400.4|           USA|North America|  37.09024| -95.712891|
|          6|       HSBC Holdings|          104.9|            14.3|          2684.1|                 201.3|            UK|       Europe| 55.378051|  -3.435973|
|          7|   Royal Dutch Shell|          467.2|            26.6|           360.3|                 213.1|The Netherlands|       Europe| 52.132633|   5.291266|
|          8|Agricultural Bank...|          103.0|            23.0|          2124.2|                 150.8|         China|         Asia|  35.86166|104.195397|
|          9|          PetroChina|          308.9|            18.3|           347.8|                 261.2|         China|         Asia|  35.86166|104.195397|
|          9|  Berkshire Hathaway|          162.5|            14.8|           427.5|                 252.8|           USA|North America|  37.09024| -95.712891|
|         11|       Bank of China|           98.1|            22.1|          2033.8|                 131.7|         China|         Asia|  35.86166|104.195397|
|         12|         Wells Fargo|           91.2|            18.9|          1423.0|                 201.3|           USA|North America|  37.09024| -95.712891|
|         13|             Chevron|          222.6|            26.2|           233.0|                 232.5|           USA|North America|  37.09024| -95.712891|
|         14|    Volkswagen Group|          254.0|            28.6|           408.2|                  94.4|       Germany|       Europe| 51.165691|  10.451526|
|         15|     Wal-Mart Stores|          469.2|            17.0|           203.1|                 242.5|           USA|North America|  37.09024| -95.712891|
|         15|               Apple|          164.7|            41.7|           196.1|                 416.6|           USA|North America|  37.09024| -95.712891|
|         17|             Gazprom|          144.0|            40.6|           339.3|                 111.4|        Russia|       Europe|  61.52401| 105.318756|
|         18|                  BP|          370.9|            11.6|           301.0|                 130.4|            UK|       Europe| 55.378051|  -3.435973|
|         19|           Citigroup|           90.7|             7.5|          1864.7|                 143.6|           USA|North America|  37.09024| -95.712891|
|         20|           Petrobras|          144.1|            11.0|           331.6|                 120.7|        Brazil|South America|-14.235004|  -51.92528|
+-----------+--------------------+---------------+----------------+----------------+----------------------+--------------+-------------+----------+----------+
only showing top 20 rows
```

Hình 3: Xem dữ liệu

CHƯƠNG 3: XỬ LÝ DỮ LIỆU

3.1 Kiểm tra Kiểu dữ liệu các cột

```
[ ]  1 data.printSchema()

root
 |-- Global Rank: integer (nullable = true)
 |-- Company: string (nullable = true)
 |-- Sales ($billion): double (nullable = true)
 |-- Profits ($billion): double (nullable = true)
 |-- Assets ($billion): double (nullable = true)
 |-- Market Value ($billion): double (nullable = true)
 |-- Country: string (nullable = true)
 |-- Continent: string (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
```

Hình 4: Kiểm tra dữ liệu các cột

⇨ Kiểu dữ liệu của các cột trùng với kiểu dữ liệu ban đầu từ Nguồn.

3.2 Kiểm tra thông tin các cột

```
[ ]  1 data.describe().show()

+-------+------------------+--------------------+------------------+------------------+------------------+--------------------+--------+-------------+------------------+------------------+
|summary|       Global Rank|             Company|  Sales ($billion)|Profits ($billion)|Assets ($billion)|Market Value ($billion)| Country|    Continent|          Latitude|         Longitude|
+-------+------------------+--------------------+------------------+------------------+------------------+--------------------+--------+-------------+------------------+------------------+
|  count|              1924|                1924|              1924|              1924|              1924|                1924|    1924|         1924|              1924|              1924|
|   mean|997.2328482328483|                NULL|19.265904365904273|1.2260395010395047|79.50779625779606|     19.558160083160104|    NULL|         NULL|34.618746549376155|15.455664304573906|
| stddev|575.5027807871161|                NULL| 34.68391064277543|3.4138308377185993| 261.0987745853911|      32.95702335461456|    NULL|         NULL| 18.25949871288175| 92.63965491779362|
|    min|                 1|                  3M|               0.0|              -24.5|               1.0|                 0.0|Australia|       Africa|         -40.900557|        -106.346771|
|    max|              1999|Zoomlion Heavy In...|             469.2|              44.9|            3226.2|               416.6|  Vietnam|South America|          61.92411|         174.885971|
+-------+------------------+--------------------+------------------+------------------+------------------+--------------------+--------+-------------+------------------+------------------+
```

Hình 5: Kiểm tra thông tin các cột

3.3 Kiểm tra dữ liệu bị thiếu

```
[ ]  1 data.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in data.columns]).show()

+-----------+-------+---------------+-----------------+----------------+--------------------+-------+---------+--------+---------+
|Global Rank|Company|Sales ($billion)|Profits ($billion)|Assets ($billion)|Market Value ($billion)|Country|Continent|Latitude|Longitude|
+-----------+-------+---------------+-----------------+----------------+--------------------+-------+---------+--------+---------+
|          0|      0|              0|                0|               0|                   0|      0|        0|       0|        0|
+-----------+-------+---------------+-----------------+----------------+--------------------+-------+---------+--------+---------+
```

Hình 6: Kiểm tra dữ liệu bị thiếu

⇨ Không có dữ liệu bị thiếu

3.4 Tạo view để truy vấn dữ liệu

```
[ ]   1 data.createOrReplaceTempView("data")
```

Hình 7: Tạo view để truy vấn dữ liệu

3.5 Phân loại các thuộc tính có trong tập dữ liệu

Vì trọng tâm bài toán là dự đoán lợi nhuận dựa trên chi phí nghiên cứu và phát triển thuật toán bằng hồi quy tuyến tính nên nhóm em sẽ tiến hành loại bỏ các trường dữ liệu không cần thiết chỉ giữ lại những trường có giá trị. Cụ thể các trường được giữ lại:

- RD Spend: Chi phí nghiên cứu và phát triển
- Profit: Lợi nhuận

```
[ ]   1 df = spark.sql("SELECT `Sales ($billion)` as Sale,`Market Value ($billion)` as MarketValue FROM data")
      2 df.show(5)

+-----+-----------+
| Sale|MarketValue|
+-----+-----------+
|134.8|      237.3|
|113.1|      202.0|
|108.2|      191.4|
|147.4|      243.7|
|420.7|      400.4|
+-----+-----------+
only showing top 5 rows
```

Hình 8: Tạo dataframe để dùng cho việc dự đoán và đổi tên cột

3.6 Kiểm tra giá trị cột Sales ($billion)

```
[10]  1 df.select('Sale').distinct().orderBy('Sales ($billion)').show(10, False)

+----+
|Sale|
+----+
|0.0 |
|0.2 |
|0.3 |
|0.4 |
|0.5 |
|0.6 |
|0.7 |
|0.8 |
|0.9 |
|1.0 |
+----+
only showing top 10 rows
```

Hình 9: Kiểm tra giá trị cột Sales ($billion)

- Tiến hành xem 10 dòng đầu tiên của cột Sale ➔ không có dữ liệu nào là khác thường

3.7 Kiểm tra giá trị cột Market Value ($billion)

```
[11]  1 df.select('MarketValue').distinct().orderBy('Market Value ($billion)').show(10, False)

+-----------+
|MarketValue|
+-----------+
|0.0        |
|0.1        |
|0.2        |
|0.3        |
|0.4        |
|0.5        |
|0.6        |
|0.7        |
|0.8        |
|0.9        |
+-----------+
only showing top 10 rows
```

Hình 10: Kiểm tra giá trị cột Market Value

- Tiến hành xem 10 dòng đầu tiên của cột Market Value ➔ không có dữ liệu nào là khác thường

CHƯƠNG 4: CHECK OUTLIERS, LÀM SẠCH VÀ KIỂM SOÁT NGOẠI LỆ DỮ LIỆU

4.1 Check outliers, làm sạch dữ liệu cho cột Sale

Bước 1: Lấy các giá trị của Sale.

```
[8]   1 sales = df.rdd.map(lambda p: (p.Sale)).collect()
```

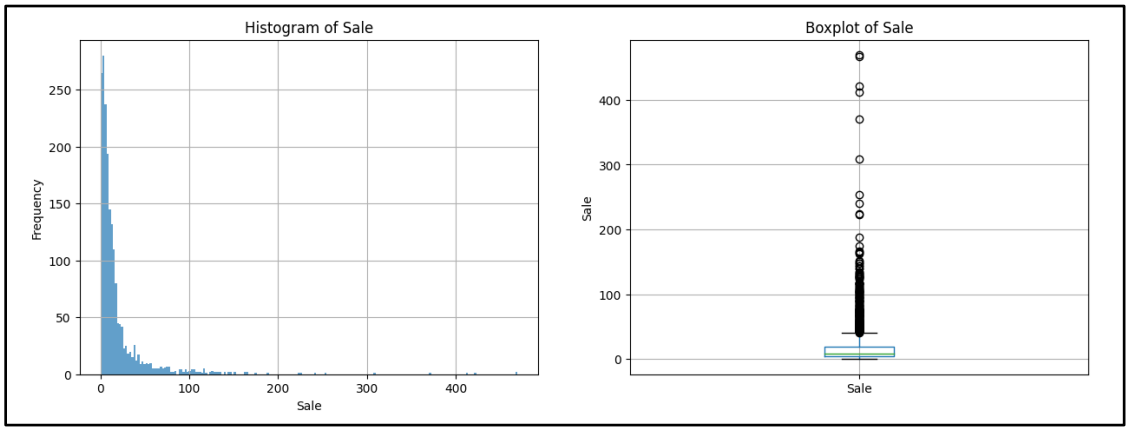
Hình 11: Lấy giá trị của Sale

Bước 2: Đưa các giá trị vừa lấy vào DataFrame.

```
[9]   1 sale_df = DataFrame({'Sale':sales})
```

Hình 12: Đưa giá trị vào df

Bước 3: Trực quan hóa giá trị tổng quan và ngoại lệ của thuộc tính RDSpend.



Hình 13: Biểu đồ giá trị trực quan và ngoại lệ của Sale

⇨ Tập dữ liệu bị nghiêng về một phía bên phải và nằm trong khoảng từ 0 - 100

4.2 Check outliers, làm sạch dữ liệu cho cột Market Value

Bước 1: Lấy các giá trị của Market Value.

```
[ ]   1 market_value = df.rdd.map(lambda p: (p.MarketValue)).collect()
```

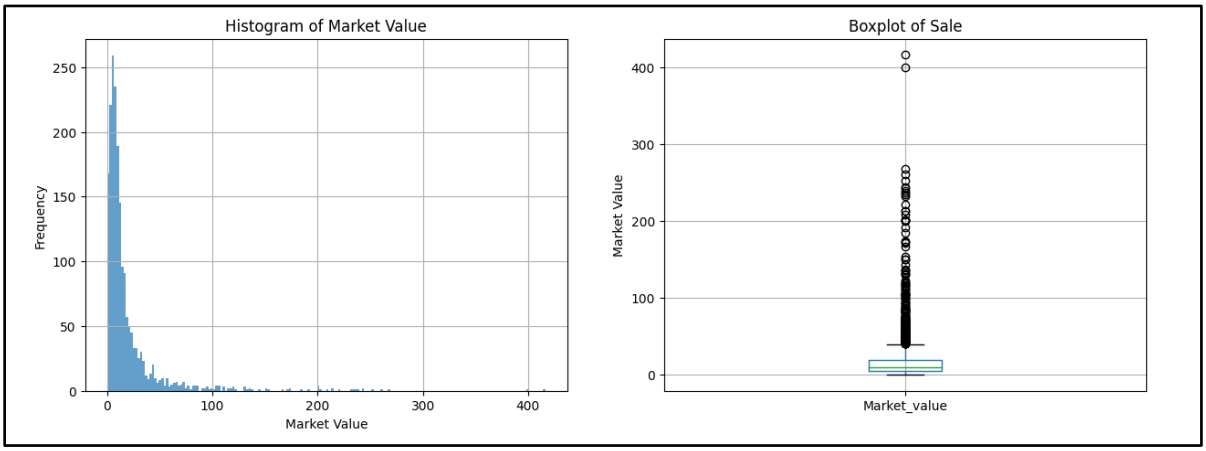
Hình 14: Lấy giá trị của Market Value

Bước 2: Đưa các giá trị vừa lấy vào DataFrame.

```
[ ]   1 market_value_df = DataFrame({'Market_value': market_value})
```

Hình 15: Đưa giá trị vào df

Bước 3: Trực quan hóa giá trị tổng quan và ngoại lệ của Market Value.



Hình 16: Biểu đồ giá trị trực quan và ngoại lệ của Market Value

⇨ Tập dữ liệu bị nghiêng về một phía và năm trong khoảng từ 0 - 100

Bước 4: Sử dụng DataFrame để kiểm tra các giá trị Profit nằm ngoài 300000.

```
[ ]  df.filter((df['Profit'] > 300000)).orderBy('Profit').show(10,True)

+---------+---------+
|  RDSpend|   Profit|
+---------+---------+
|128456.23|333962.19|
|100275.47|413956.48|
|161181.72|476485.43|
+---------+---------+
```

Hình 17: Kiểm tra giá trị ngoại lệ của Profit

⇨ Sử dụng đối chiếu trên 2 DataFrame, ta có thể thấy, khi lợi nhuận càng cao (>300000) thì chi phí bỏ ra để nghiên cứu và phát triển cũng rất cao, điều này là hoàn toàn đúng và hợp logic và có thể xảy ra trong thực tế. Nhóm sẽ xem đây là đặc tính của dữ liệu và không tiến hành xử lí.

CHƯƠNG 5: PHÂN TÍCH DỮ LIỆU

Bước 1: Lấy ra các giá trị của Sales và Market Value.

```
[31]  1 x = df.rdd.map(lambda p: (p.Sale)).collect()
      2 y = df.rdd.map(lambda p: (p.MarketValue)).collect()
```

Hình 18: Lấy các giá trị của Sales và Market Value

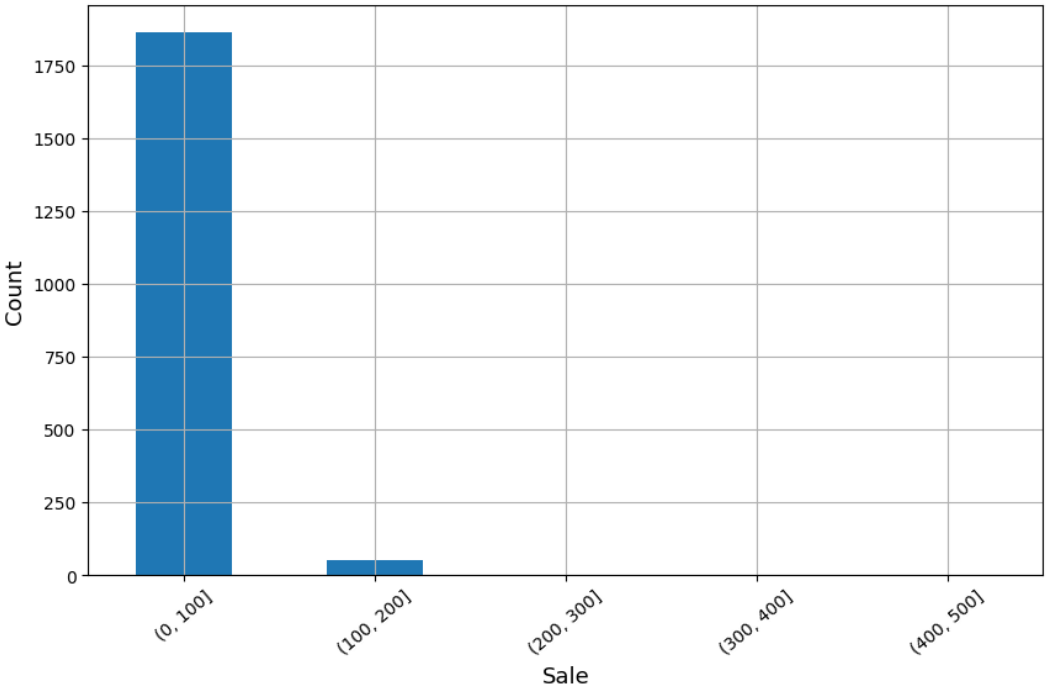
Bước 2: Đưa các giá trị trên vào DataFrame.

```
[32]  1 data_df = DataFrame({'Sale':x, 'MarketValue':y})
```

Hình 19: Đưa các giá trị vào Data Frame

5.1 Trực quan hóa doanh số bán hàng (Sale)

```
[ ]  1 bins = [0, 100, 200, 300, 400, 500]
     2 data_df['Sale'] = pd.cut(data_df['Sale'], bins=bins)
     3 data_df.Sale.value_counts().sort_index().plot(kind='bar', figsize=(10,6))
     4 plt.xlabel('Sale', fontsize= 13)
     5 plt.ylabel('Count', fontsize = 13)
     6 plt.xticks(rotation = 40)
     7 plt.grid()
     8 plt.show()
```

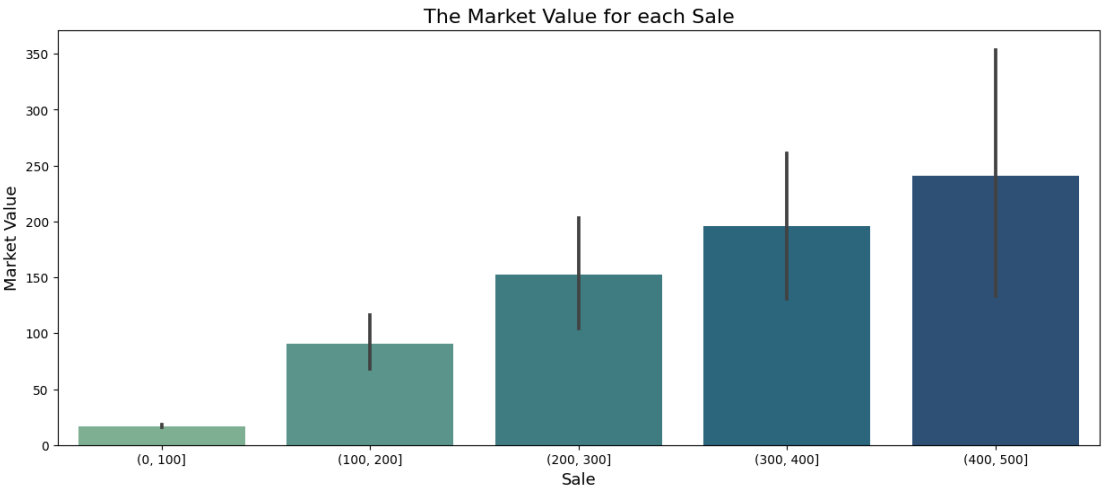


Hình 20: Trực quan hóa các nhóm giá trị trong Sale

- ⇨ Doanh số bán hàng trong nhóm 0 – 100 triệu đô chiếm nhiều nhất, tiếp theo đó là nhóm 100 – 200.
- ⇨ Doanh số bán hàng thấp nhất là trong nhóm 400 – 500 triệu đô.

5.2 Trực quan hóa giữa Doanh số bán hàng (Sale) và Vốn hóa thị trường (Market Value)

```
[ ]  1 plt.figure(figsize=(15, 6))
     2 sns.barplot(data = data_df, x='Sale', y='MarketValue', palette='crest')
     3 plt.title('The Market Value for each Sale', fontsize= 16)
     4 plt.xlabel('Sale', fontsize= 13)
     5 plt.ylabel('Market Value', fontsize = 13)
     6 plt.show();
```



Hình 21: Trực quan hóa giữa Sale và Market Value

⇨ Công ty có giá trị vốn hóa càng cao thì doanh thu bán hàng càng cao, ở đây khi trực quan hóa là nhóm 400 – 500.

CHƯƠNG 6: CHUẨN BỊ DỮ LIỆU

6.1 Phân chia dữ liệu thành 2 phần Feature (X) và Target (Y)

```
[ ]   1 df.show()
```

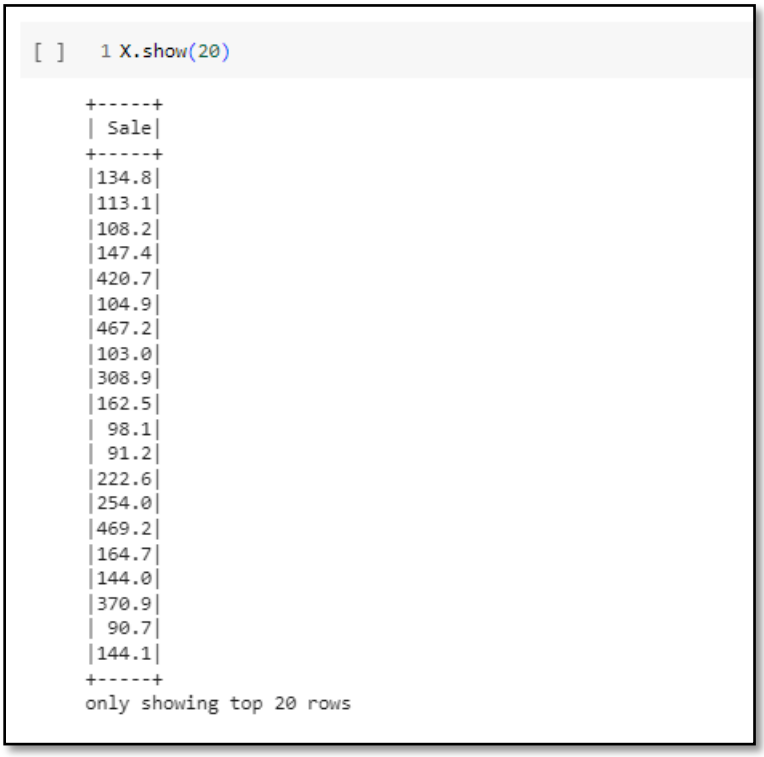
```
+-----+-----------+
| Sale|MarketValue|
+-----+-----------+
|134.8|      237.3|
|113.1|      202.0|
|108.2|      191.4|
|147.4|      243.7|
|420.7|      400.4|
|104.9|      201.3|
|467.2|      213.1|
|103.0|      150.8|
|308.9|      261.2|
|162.5|      252.8|
| 98.1|      131.7|
| 91.2|      201.3|
|222.6|      232.5|
|254.0|       94.4|
|469.2|      242.5|
|164.7|      416.6|
|144.0|      111.4|
|370.9|      130.4|
| 90.7|      143.6|
|144.1|      120.7|
+-----+-----------+
only showing top 20 rows
```

Hình 22: Xem dữ liệu và tiến hành chia dữ liệu thành tập X, Y

```
[ ]   1 df.createOrReplaceTempView("split_data")
      2 X = spark.sql("SELECT DOUBLE(Sale) FROM split_data")
      3 Y = spark.sql("SELECT DOUBLE(MarketValue) FROM split_data")
```

Hình 23: Chia dữ liệu thành tập X, Y

6.1.1 Xem dữ liệu X (Sale)



```
[ ]  1 X.show(20)

+-----+
| Sale|
+-----+
|134.8|
|113.1|
|108.2|
|147.4|
|420.7|
|104.9|
|467.2|
|103.0|
|308.9|
|162.5|
| 98.1|
| 91.2|
|222.6|
|254.0|
|469.2|
|164.7|
|144.0|
|370.9|
| 90.7|
|144.1|
+-----+
only showing top 20 rows
```

Hình 24: Dữ liệu của X

6.1.2 Xem dữ liệu Y (Market Value)

```
[ ]   1 Y.show(20)

+-----------+
|MarketValue|
+-----------+
|      237.3|
|      202.0|
|      191.4|
|      243.7|
|      400.4|
|      201.3|
|      213.1|
|      150.8|
|      261.2|
|      252.8|
|      131.7|
|      201.3|
|      232.5|
|       94.4|
|      242.5|
|      416.6|
|      111.4|
|      130.4|
|      143.6|
|      120.7|
+-----------+
only showing top 20 rows
```

Hình 25: Dữ liệu của y

6.2 Chia dữ liệu Test Train theo theo tỉ lệ Test 30% và Train 70%

Bước 1: Chia tập dữ liệu Test và Train.

```
[ ]   1 # Chia dữ liệu theo tỉ lệ train:test 7:3
      2 X_train, X_test = X.randomSplit([0.7, 0.3], seed = 2)
      3 Y_train, Y_test = Y.randomSplit([0.7, 0.3], seed = 2)
```

Hình 26: Chia dữ liệu thành Test và Train

Bước 2: Show các giá trị X Train.

```
[ ]  1 X_train.describe().show()

+-------+------------------+
|summary|             Sale|
+-------+------------------+
|  count|              1325|
|   mean|18.790113207547183|
| stddev|34.781117916289176|
|    min|               0.0|
|    max|             469.2|
+-------+------------------+
```

Hình 27: Các thông tin của X train

Bước 3: Show các giá trị X Test.

```
[ ]  1 X_test.describe().show()

+-------+-----------------+
|summary|             Sale|
+-------+-----------------+
|  count|              599|
|   mean|20.31836393989984|
| stddev|34.47352379441042|
|    min|              0.3|
|    max|            467.2|
+-------+-----------------+
```

Hình 28: Các thông tin của X test

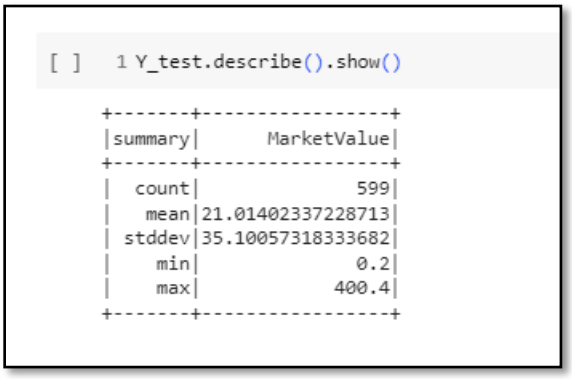
Bước 4: Show các giá trị Y Train.

```
[ ]  1 Y_train.describe().show()

+-------+------------------+
|summary|       MarketValue|
+-------+------------------+
|  count|              1325|
|   mean|18.900000000000002|
| stddev|  31.9327848651619|
|    min|               0.0|
|    max|             416.6|
+-------+------------------+
```

Hình 29: Các thông tin của Y train

Bước 5: Show các giá trị Y Test.



Hình 30: Các thông tin của Y test

CHƯƠNG 7: XÂY DỰNG THUẬT TOÁN LINEAR REGRESSION

7.1 Xây dựng thuật toán Linear Regression (Hướng làm 1)

7.1.1 Tạo mô hình hồi quy tuyến tính (Linear Regression)

Linear Regression là một phương pháp thống kê được sử dụng để mô hình hóa mối quan hệ tuyến tính giữa một biến phụ thuộc (đối tượng cần dự đoán) và một hoặc nhiều biến độc lập (các đặc trưng hoặc biến giải thích). Ý tưởng cơ bản là tìm ra một đường thẳng (hàm tuyến tính) sao cho tổng bình phương sai số giữa giá trị dự đoán và giá trị thực tế là nhỏ nhất.

Ở trong bài toán đối với tập dữ liệu này, đối tượng cần dự đoán là MarketValue và biến độc lập là Sale.

Mô hình hồi quy tuyến tính đơn giản là một đường được xác định bởi các hệ số được ước tính từ dữ liệu huấn luyện. Sau khi các hệ số được ước tính, chúng ta có thể sử dụng chúng để đưa ra dự đoán.

Phương trình đưa ra dự đoán với mô hình hồi quy tuyến tính đơn giản như sau:

$$y = B0 + B1 * x$$

Nhóm sẽ bắt đầu tính các hệ số B0 và B1 cho mô hình hồi quy tuyến tính.

Bước 1: Sau khi đã phân tách dữ liệu ra làm 2 tập train và test, nhóm tiến hành đưa dữ liệu lên trên RDD theo dạng list để tiến hành xử lý.

```
[ ]  # rdd_X_train = X_train.rdd
     rdd_X_train = X_train.rdd.map(list)
     rdd_y_train = y_train.rdd.map(list)

[ ]  # rdd_X_test = X_test.rdd
     rdd_X_test = X_test.rdd.map(list)
     rdd_y_test = y_test.rdd.map(list)
```

Hình 31: Đưa giá trị của 2 tập train test lên RDD theo dạng list

Bước 2: Tính giá trị trung bình và mẫu số B1 của cả biến đầu vào và đầu ra từ dữ liệu huấn luyện.

Giá trị trung bình là trung bình của một danh sách các số

Dưới đây là một hàm có tên là *mean()* thực hiện tính giá trị trung bình cho danh sách các số.

```
[ ]  1 mean_X_train = rdd_X_train_1.mean()
     2 print(mean_X_train)

18.790113207547172
```

```
[ ]  1 mean_Y_train = rdd_Y_train_1.mean()
     2 print(mean_Y_train)

18.900000000000006
```

Công thức tính hệ số B1 (slope):

$$B1 = \frac{\sum_{i=1}^{m}(xi - \bar{x})(yi - \bar{y})}{\sum_{i=1}^{m}(xi - \bar{x})^2}$$

Công thức tính tử số của B1:

$$\sum_{i=1}^{m}(xi - \bar{x})(yi - \bar{y})$$

```
[ ]  1 # Tính tử số B1 (Slope)
     2 def nume_slope(X, mean_X, Y, mean_Y):
     3     result = 0.0
     4     common_rdd = rdd_X_train_1.zip(rdd_Y_train_1)
     5     nume_slope = common_rdd.map(lambda a: (a[0] - mean_X) * (a[1] - mean_Y))
     6     result += nume_slope.reduce(lambda a, b: a + b)
     7     return result
```

Công thức tính mẫu số B1:

$$\sum_{i=1}^{m}(xi - \bar{x})^2$$

```
[ ]  # Tính mẫu số B1 (Slope)
     def deno_slope(data, mean):
         deno_slope = data.map(lambda a: (a - mean) ** 2)
         result = deno_slope.reduce(lambda a, b: a + b)
         return result
```

Hình 32: Hàm tính mẫu số B1 (Slope)

Bước 3: Tính tử số

```
[ ]  1 # Tính tử số B1 (Slope)
     2 def nume_slope(X, mean_X, Y, mean_Y):
     3     result = 0.0
     4     common_rdd = rdd_X_train_1.zip(rdd_Y_train_1)
     5     nume_slope = common_rdd.map(lambda a: (a[0] - mean_X) * (a[1] - mean_Y))
     6     result += nume_slope.reduce(lambda a, b: a + b)
     7     return result

[ ]  1 nume_slope_xy = nume_slope(rdd_X_train_1, mean_X_train, rdd_Y_train_1, mean_Y_train)
     2 print(nume_slope_xy)

     1431582.5799999996
```

Hình 33: Tính tử số của x, y train

Bước 4: Tính mẫu số và Mean(y)

```
[ ]  1 deno_X_train = deno_slope(rdd_X_train_1, mean_X_train)
     2 print(deno_X_train)
```

```
1601677.4404830195
```

Hình 34: Tính mẫu số x train

```
[ ]  1 deno_Y_train = deno_slope(rdd_Y_train_1, mean_Y_train)
     2 print(deno_Y_train)
```

```
1350086.4400000002
```

Hình 35: Tính mẫu số y train

Bước 5: Tính toán hệ số B0 và B1.

$$B0 = \bar{y} - B1 * \bar{x}$$

$$B1 = \frac{\sum_{i=1}^{m}(xi - \bar{x})(yi - \bar{y})}{\sum_{i=1}^{m}(xi - \bar{x})^2}$$

```
[ ]  1 # Tính toán các hệ số B0 và B1
     2 def coefficients(X, Y):
     3     X_mean = X.mean()
     4     Y_mean = Y.mean()
     5
     6     common_rdd = X.zip(Y)
     7
     8     nume = common_rdd.map(lambda xy: (xy[0] - X_mean) * (xy[1] - Y_mean)).sum()
     9     deno = common_rdd.map(lambda xy: (xy[0] - X_mean) ** 2).sum()
    10
    11     b1 = nume / deno
    12     b0 = Y_mean - b1 * X_mean
    13
    14     return [b0, b1]
```

Hình 36: Hàm tính hệ số B0 và B1

Bước 6: Hiển thị hệ số của mô hình.

```
[ ]  1 # Hệ số của mô hình
     2 coef = coefficients(rdd_X_train_1, rdd_Y_train_1)
     3 print(">>>> Hệ số - B0, B1 = " + str(coef))

>>>> Hệ số - B0, B1 = [2.1053582923411227, 0.8938020501607838]
```

Hình 37: Hệ số B0 và B1 tính được

Bước 7: Tiến hành xây dựng hàm dự đoán lợi nhuận dựa trên mô hình Hồi quy tuyến tính đơn giản.

$$y = B0 + B1 * x$$

Dưới đây là hàm có tên *linear_regression_1()* thực hiện phương trình dự đoán để đưa ra dự đoán trên tập dữ liệu thử nghiệm. Nó cũng liên kết với nhau ước tính các hệ số trên dữ liệu đào tạo từ các bước trên.

```
[ ]  1 # Mô hình hồi quy tuyến tính để dự đoán market value
     2 def linear_regression_1(X_train, Y_train, X_test):
     3     coef = coefficients(X_train, Y_train)
     4     b0 = coef[0]
     5     b1 = coef[1]
     6     yhat = X_test.map(lambda a: a * b1 + b0)
     7     return yhat
```

Hình 38: Hàm tính toán dựa trên công thức hồi quy tuyến tính

7.1.2 Ứng dụng mô hình hồi quy tuyến tính trên tập dữ liệu Test

Bước 1: Chuyển RDD Test thành danh sách các giá trị (dạng List).

```
[ ]  # Chuyển rdd_X_test thành danh sách các giá trị
     rdd_X_test_1 = rdd_X_test.map(lambda a: a[0])
```

Hình 39: Chuyển RDD_X_test thành danh sách các giá trị

Bước 2: Hiển thị thời gian dự đoán giá trị vốn hóa.

```
[ ]  1 # Prediction lưu trữ kết quả dự đoán giá trị vốn hóa
     2 start = time.time()
     3 prediction = linear_regression_1(rdd_X_train_1, rdd_Y_train_1, rdd_X_test_1)
     4 end = time.time()
     5 run_time_reg1 = end - start
     6 print(">>>> Run time:" + str(run_time_reg1) + "(s)")

>>>> Run time:1.8029918670654297(s)
```

Hình 40: Dự đoán kết quả và tính toán thời gian chạy

Bước 3: Xuất kết quả dự đoán lợi n trên mô hình hồi quy tuyến tính.

```
[ ]  1 # Xuất kết quả dự đoán market value trên mô hình hồi quy tuyến tính
     2 prediction_result = prediction.collect()
     3 DataFrame({'prediction_result':prediction_result})
```

	prediction_result
0	2.373499
1	2.462879
2	2.552259
3	2.552259
4	2.552259
...	...
594	138.231411
595	147.348191
596	157.805675
597	229.131079
598	419.689676

599 rows × 1 columns

Hình 41: Kết quả dự đoán

Bước 4: Chuyển RDD_Y_test thành danh sách các giá trị.

```
[ ]   1 # Chuyển rdd_y_test thành danh sách các giá trị
      2 rdd_Y_test_1 = rdd_Y_test.map(lambda a: a[0])
```

Hình 42: Chuyển rdd_y_test thành danh sách các giá trị

7.1.3 Tạo và ứng dụng chỉ số đánh giá RMSE - Root Mean Square Error

Để đánh giá được độ chinh xác của mô hình cũng sẽ xây dựng thêm một hàm để quản lý việc đánh giá các dự đoán được gọi là *evaluate_algorithm()* và một hàm khác để ước tính *Sai số toàn phương trung bình căn – Root Mean Square Error* của các dự đoán được gọi là *rmse_metric().*

Công thức:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_1 - y_2)^2}{n}}$$

- y_1 là giá trị ước lượng
- y_2 là biến độc lập
- n = (N – k – 1)
- N: số tổng lượng quan sát
- K: tổng lượng biến

```
[ ]  from math import sqrt
     # Hàm tính toán RMSE
     def rmse_metric(actual, predicted):
         common_rdd = actual.zip(predicted)
         prediction_error = common_rdd.map(lambda a: a[0] - a[1])
         # prediction_error.collect()
         prediction_error_square = prediction_error.map(lambda a: a ** 2)
         sum_error = prediction_error_square.reduce(lambda a, b: a + b)
         # print(sum_error)
         mean_error = sum_error / float(actual.count())
         return sqrt(mean_error)
```

Hình 43: Hàm tính chỉ số RMSE

```
[ ]  # Hàm đánh giá thuật toán hồi quy trên tập dữ liệu huấn luyện
     def evaluate_algorithm(X_train, y_train, X_test, y_test, algorithm):
         predicted = algorithm(X_train, y_train, X_test)
         rmse = rmse_metric(y_test, predicted)
         return rmse
```

Hình 44: Hàm đánh giá thuật toán trên tập train

```
[ ]  1 # Đánh giá thuật toán hồi quy trên tập dữ liệu huấn luyện
     2 evaluate_model_HL1 = evaluate_algorithm(rdd_X_train_1, rdd_Y_train_1, rdd_X_test_1, rdd_Y_test_1, linear_regression_1)
     3 print(">>>> Root Mean Squared Error (RMSE) = " + str(evaluate_model_HL1))

>>>> Root Mean Squared Error (RMSE) = 7.039201229422485
```

Hình 45: Đánh giá thuật toán

7.1.4 Trực quan hóa và so sánh kết quả dự đoán với thực tế

Bước 1: Lấy các giá trị từ tập rdd_Y_test_1

```
[ ]  1 Y_test_origin = rdd_Y_test_1.collect()
```

Hình 46: Giá trị gốc

Bước 2: So sánh giá trị Market Value dự đoán và giá trị gốc

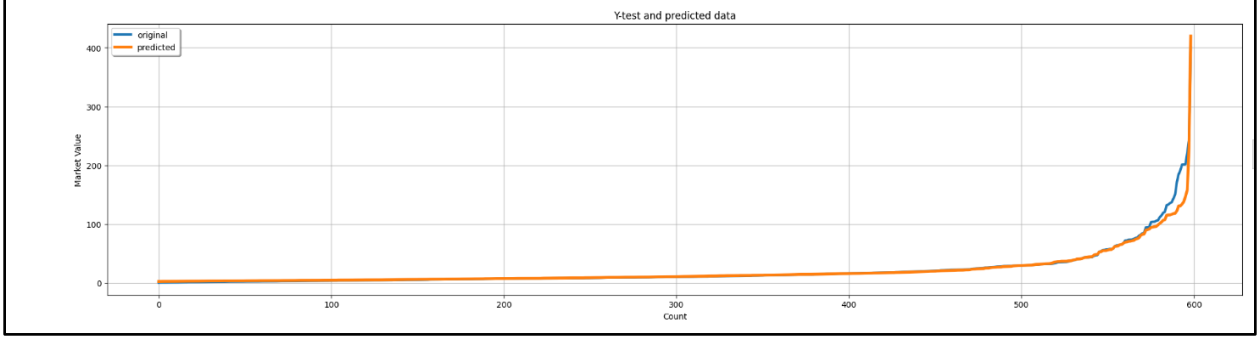
```
[ ]  1 # So sánh giữa giá trị Market Value dự đoán và giá trị gốc
     2 compare = DataFrame({'Origin': Y_test_origin, 'Predict': prediction_result})
     3 compare.head(10)
```

	Origin	Predict
0	0.2	2.373499
1	0.4	2.462879
2	0.4	2.552259
3	0.4	2.552259
4	0.5	2.552259
5	0.5	2.552259
6	0.6	2.641640
7	0.6	2.641640
8	0.6	2.641640
9	0.7	2.641640

Hình 47: So sánh giá trị dự đoán và giá trị gốc

Bước 3: Trực quan hóa so sánh kết quả dự đoán và thực tế

```
[ ]  1 # Trực quan hóa so sánh kết quả trên tập test và kết quả dự đoán
     2 x_ax = range(len(Y_test_origin))
     3 plt.figure(figsize=(25, 6))
     4 plt.plot(x_ax, Y_test_origin, linewidth=3, label="original")
     5 plt.plot(x_ax, prediction_result, linewidth=3.5, label="predicted")
     6 plt.title("Y-test and predicted data")
     7 plt.xlabel('Count')
     8 plt.ylabel('Market Value')
     9 plt.legend(loc='best',fancybox=True, shadow=True)
    10 plt.grid(True)
    11 plt.show()
```



Hình 48: Trực quan giá trị dự đoán và giá trị thực tế

7.2 Xây dựng thuật toán Linear Regression (Hướng làm 2)

7.2.1 Tạo mô hình hồi quy tuyến tính (Linear Regression)

Bước 1: Tạo 2 tập dữ liệu train data và test data.

Tạo ra 1 tập dữ liệu rdd_train và rdd_test bằng hàm zip() bởi 2 tập dữ liệu x (rdd_X_train_1) và tập dữ liệu y (rdd_y_train_1).

rdd_train và rdd_test sẽ có từng dòng dữ liệu sẽ là các cặp khóa giá trị (key – value).

```
[ ]  1 rdd_train = rdd_X_train_1.zip(rdd_Y_train_1)
```

```
[ ]  1 rdd_test = rdd_X_test_1.zip(rdd_Y_test_1)
```

Hình 49: Tạo tập dữ liệu train test

Bước 2: Định nghĩa hàm Tính hệ số ước lượng - estimate_coef()

Đây là hàm tính toán và trả về 2 hệ số ước tính: b_0 và b_1.

Với Phương trình của đường hồi quy được biểu diễn như sau:

$$h(x_i) = \beta_0 + \beta_1 . x_i \quad (1)$$

Chúng ta cần phải tính:

$$\beta_1 = \frac{SS_{xy}}{SS_{xx}} \quad | \quad \beta_0 = \bar{y} - \beta_1 . \bar{x}$$

và

$$SS_{xy} = \sum_{i=1}^{n} y_i x_i - n \,.\, \bar{x} . \bar{y} \quad | \quad SS_{xx} = \sum_{i=1}^{n} (x_i)^2 - n\,(\bar{x}_i)^2$$

Trong đó:

- SS_xy: là tổng các độ lệch chéo (cross-deviations) của y và x:
- SS_xx: là tổng các độ lệch bình phương của x:
- $\bar{y}$: là giá trị trung bình của y.
- $\bar{x}$: là giá trị trung bình của x.

- n: là số lượng các giá trị x hoặc y.

```
[ ]
def estimate_coef(data):
  # Số lượng phần tử trong m
  n = data.count()

  # Trung bình của mảng bao gồm các keys
  sum_x = data.keys().reduce(lambda a, b: a + b)
  mean_x = sum_x/n
  # Trung bình của mảng bao gồm các values
  sum_y = data.values().reduce(lambda a, b: a + b)
  mean_y = sum_y/n
  # Tính toán giá trị cross-deviation (độ lệch chéo) và deviation của x và y
  sum_xy = data.map(lambda x: (x[0],x[0]*x[1])).values().reduce(lambda x,y: x+y)
  SS_xy = sum_xy - n*mean_x*mean_y

  sum_xx = data.map(lambda x: (x[0],x[0]*x[0])).values().reduce(lambda x,y: x+y)
  SS_xx = sum_xx - n*mean_x*mean_x

  # Tính toán hệ số hồi quy
  b_1 = SS_xy / SS_xx
  b_0 = mean_y - b_1*mean_x

  return (b_0, b_1)
```

Hình 50: Hàm tính toán hệ số B0 và B1

Bước 3: Tính hệ số ước lượng B1 và B0

Đầu vào sẽ là tập rdd_train trước đó. Đầu ra sẽ là 2 chỉ số ước lượng của công thức tính phương trình hồi quy.

```
[ ]
# Hệ số ước lượng (estimating coefficients)
b = estimate_coef(rdd_train)
print("Hệ số ước lượng:\nb_0 = {} \nb_1 = {}".format(b[0], b[1]))
```

```
Hệ số ước lượng:
b_0 = 2.1053582923410943
b_1 = 0.8938020501607846
```

Hình 51: Hệ số B0 và B1 tính được

Bước 4: Định nghĩa hàm dự đoán các giá trị y - linear_regression_2()

Hàm linear_regression_2() có đối số đầu vào là 2 tập dữ liệu RDD đại diện 2 tập dữ liệu training data và testing data.

Hàm này sẽ gọi tới hàm estimate_coef() để trả về 2 giá trị b_0 và b_1. Sau đó sẽ thực hiện tính toán với các dữ liệu x của testing data và trả về các giá trị y được dự đoán (tạm gọi là y_predict).

```
[ ]  1 # Hàm dự đoán - trả về các giá trị y
     2 def linear_regression_2(train, test):
     3     b_0, b_1 = estimate_coef(train)
     4     predictions = test.keys().map(lambda x: x*b_1 + b_0)
     5     return predictions
```

Hình 52: Hàm tính toán dựa trên công thức hồi quy tuyến tính

7.2.2 Ứng dụng mô hình dự đoán trên tập Test

Thực hiện việc tính toán tập dữ liệu y_predict

Tại đây, chúng em sẽ chạy hàm linear_regression_2() và trả về tập dữ liệu y_predict cũng như sẽ tính toán thời gian chạy thuật toán.

```
[ ]  1 start = time.time()
     2 prediction = linear_regression_2(rdd_train, rdd_test)
     3 end = time.time()
     4 run_time_HL2 = end - start
     5 print(">>>> Run time:" + str(run_time_HL2) + "(s)")

>>>> Run time:3.0440900325775146(s)
```

Hình 53: Dự đoán kết quả và tính toán thời gian chạy

7.2.3 Tạo và ứng dụng chỉ số đánh giá RMSE - Root Mean Square Error

Bước 1: Định nghĩa hàm tính RMSE – rmse_metric()

Hàm có đối số đầu vào là 2 tập dữ liệu y_actual (y thực tế - là các giá trị y trong tập test data) và và y_predict (các giá trị y được dự đoán từ phương trình hồi quy).

Hàm sẽ trả về độ sai số giữa y_actual và y_predict.

Công thức được sử dụng để tính RMSE:

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

```
[ ]  # Hàm tính toán RMSE
     def rmse_metric(actual, predict):
         rdd_zip = actual.zip(predict)

         predict_error = rdd_zip.map(lambda a: a[0] - a[1])
         predict_error_square = predict_error.map(lambda a: a**2)

         sum_error = predict_error_square.reduce(lambda a,b: a+b)
         #print(sum_error)

         mean_error = sum_error/float(actual.count())

         rmse = math.sqrt(mean_error)
         return rmse
```

Hình 54: Hàm tính chỉ số RMSE

Bước 2: Định nghĩa hàm đánh giá thuật toán - evaluate_algorithm()

Hàm có đối số đầu vào là 2 tập RDD của 2 tập dữ liệu train data và test data.

Trong thân hàm sẽ gọi tới hàm rmse_metric() để lấy và trả về giá trị sai số RMSE.

```
[ ]  # Hàm đánh giá thuật toán hồi quy trên tập dữ liệu huấn luyện
     def evaluate_algorithm(train, test):
         y_test = test.values()
         y_predict = simple_linear_regression(train, test)
         rmse = rmse_metric(y_test, y_predict)
         return rmse
```

Hình 55: Hàm đánh giá thuật toán hồi quy trên tập train

Bước 3: Tính toán chỉ số RMSE

```
[ ]  1 # Đánh giá thuật toán hồi quy trên tập dữ liệu huấn luyện
     2 import math
     3 evaluate_model_2 = evaluate_algorithm(rdd_train, rdd_test)
     4 print("Chỉ số RMSE: {} ".format(evaluate_model_2))

    Chỉ số RMSE: 7.039201229422472
```

Hình 56: Đánh giá thuật toán hồi quy

7.2.4 Trực quan hóa và so sánh kết quả dự đoán với thực tế

Bước 1: Lấy giá trị gốc và dự đoán

```
[ ]  1 Y_test = rdd_test.values().collect()
```

```
[ ]  1 Y_predict = linear_regression_2(rdd_train, rdd_test).collect()
```

Hình 57: Lấy giá trị gốc và dự đoán

Bước 2: So sánh giá trị Market Value dự đoán và giá trị gốc

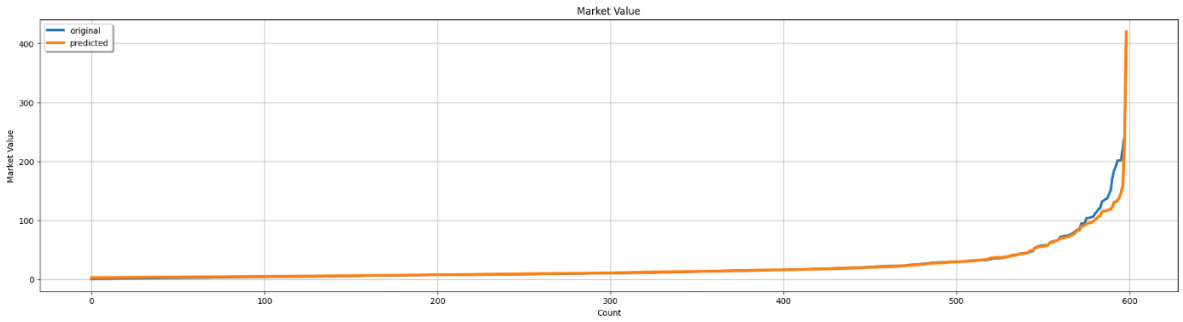
```
[ ]  1 compare = DataFrame({'Origin': Y_test, 'Predict': Y_predict})
     2 compare.head(5)
```

	Origin	Predict
0	0.2	2.373499
1	0.4	2.462879
2	0.4	2.552259
3	0.4	2.552259
4	0.5	2.552259

Hình 58: So sánh giá trị dự đoán với giá trị gốc

Bước 3: Trực quan hóa so sánh kết quả dự đoán và thực tế.

```
[ ]  1 # Trực quan hóa so sánh kết quả trên tập test và kết quả dự đoán
     2 x_ax = range(len(Y_test))
     3 plt.figure(figsize=(25, 6))
     4 plt.plot(x_ax, Y_test, linewidth=3, label="original")
     5 plt.plot(x_ax, Y_predict, linewidth=3.5, label="predicted")
     6 plt.title("Market Value")
     7 plt.xlabel('Count')
     8 plt.ylabel('Market Value')
     9 plt.legend(loc='best',fancybox=True, shadow=True)
    10 plt.grid(True)
    11 plt.show()
```



Hình 59: Trực quan giá trị dự đoán với giá trị gốc

7.3 Xây dựng thuật toán Ridge Regression (Hướng làm 3)

7.3.1 Giới thiệu về Ridge Regressions

Hồi qui Ridge là biến thể của hồi qui tuyến tính mà ở đó chúng ta thay đổi hàm mất mát MSE để kểm soát độ lớn của tham số huấn luyện nhằm giảm thiểu hiện tượng quá khớp trong các bài toán dự báo của học có giám sát.

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^\intercal \mathbf{x}_i) = \frac{1}{N}||\bar{\mathbf{X}}\mathbf{w} - \mathbf{y}||_2^2$$

Hình 60: Công thức lý thuyết của hồi qui Ridge

7.3.2 Chuẩn bị dữ liệu

Lấy giá trị RDD của tập dữ liệu X_train, X_Test, Y_train, Y_test và lưu vào các mảng.

```
[ ]  1 X_train_values = np.array(X_train.rdd.map(lambda x: x[0]).collect())
     2 Y_train_values = np.array(Y_train.rdd.map(lambda x: x[0]).collect())
     3 X_test_values = np.array(X_test.rdd.map(lambda x: x[0]).collect())

[ ]  1 Y_test_values = np.array(Y_test.rdd.map(lambda x: x[0]).collect())
```

Hình 61: Lấy giá trị từ RDD và lưu vào mảng

7.3.3 Tạo mô hình hồi quy Ridge

- Tính hệ số B1

```
[ ]  1 # Hàm tính tử số
     2 def nume_slope(X, mean_X, y, mean_y):
     3     common_array = np.column_stack((X, y))
     4     return np.sum((common_array[:, 0] - mean_X) * (common_array[:, 1] - mean_y))
     5
```

Hình 62: Tính tử số của hệ số B1

```
[ ]  1 def deno_slope(data, mean):
     2     return np.sum((data - mean) ** 2)
     3
```

Hình 63: Tính mẫu số của hệ số B1

- Tính hệ số B0

```
[ ]  1 # Hàm tính hệ số B0, B1 của Ridge Regression
     2 def coefficients(X, y, lambda_param):
     3     X_mean, y_mean = mean(X), mean(y)
     4     b1_nume = nume_slope(X, X_mean, y, y_mean)
     5     b1_deno = deno_slope(X, X_mean) + lambda_param
     6     b1 = b1_nume / b1_deno
     7     b0 = y_mean - b1 * X_mean
     8     return b0, b1
```

Hình 64: Tính hệ số B1 và B0 thông qua hàm coefficients

- Chạy hàm coeficients với lambda = 0.1

```
[ ]  1 lambda_param = 0.1
     2
     3 # Tính hệ số B0, B1 của Ridge Regression
     4 b0, b1 = coefficients(X_train_values, Y_train_values, lambda_param)
```

Hình 65: Chạy hàm coefficients

- Tạo hàm để tính loss

```
[ ]  1 # Hàm tính hàm loss
     2 def ridge_loss(X, y, b0, b1, lambda_param):
     3     y_pred = predict(X, b0, b1)
     4     loss = np.mean((y_pred - y) ** 2) + lambda_param * b1 ** 2
     5     return loss
```

Hình 66: Tạo hàm tính Loss

- Tạo hàm để dự đoán

```
     1 # Hàm dự đoán
     2 def predict(X, b0, b1):
     3     return X * b1 + b0
```

Hình 67: Tạo hàm để dự đoán

- Kết quả sau khi chạy

```
[ ]  1 # Tính hàm loss trên tập train
     2 loss_train = ridge_loss(X_train_values, Y_train_values, b0, b1, lambda_param)

[ ]  1 # Tính hàm loss trên tập test
     2 loss_test = ridge_loss(X_test_values,Y_test_values, b0, b1, lambda_param)

[ ]  1 print(f"Intercept (B0): {b0}")
     2 print(f"Slope (B1): {b1}")
     3 print(f"Loss on training set: {loss_train}")
     4 print(f"Loss on testing set: {loss_test}")

Intercept (B0): 2.10535934090683
Slope (B1): 0.8938019943566647
Loss on training set: 53.31384670733413
Loss on testing set: 49.63025683471783
```

Hình 68: Kết quả khi chạy ra hệ số B1, B0 và hàm Loss

7.3.4 Tạo hàm tính RMSE và đánh giá mô hình

- Hàm tính RMSE

```
[ ]  1 # Hàm tính RMSE
     2 def rmse_metric(actual, predicted):
     3     return np.sqrt(np.mean((actual - predicted) ** 2))
```

Hình 69: Khởi tạo hàm tính RMSE

- Hàm đánh giá mô hình dự đoán

```
[ ]  1 # Hàm đánh giá mô hình dự đoán
     2 def evaluate_model(X, y, b0, b1):
     3     y_pred = predict(X, b0, b1)
     4     rmse = rmse_metric(y, y_pred)
     5     return rmse
     6
```

Hình 70: Khởi tạo hàm tính RMSE

- Kết quả sau khi đánh giá mô hình

```
[ ]  1 # Đánh giá mô hình trên tập test
     2 rmse_test = evaluate_model(X_test_values,Y_test_values, b0, b1)
     3 # Đánh giá mô hình trên tập train
     4 rmse_train = evaluate_model(X_train_values, Y_train_values, b0, b1)
     5 print(f"RMSE on training set: {rmse_train}")
     6 print(f"RMSE on testing set: {rmse_test}")

RMSE on training set: 7.296160531870344
RMSE on testing set: 7.039202272573664
```

Hình 71: Khởi tạo hàm tính RMSE

7.3.5 Tính thời gian chạy và trực quan hóa dữ liệu so với dữ liệu thực tế

- Tính thời gian chạy

```
[ ]  1 start = time.time()
     2 prediction = predict(X_test_values, b0, b1)
     3 end = time.time()
     4 run_time_HL3 = end - start
     5 print(">>>> Run time:" + str(run_time_HL3) + "(s)")

>>>> Run time:0.00018906593322753906(s)
```

Hình 72: Khởi tạo hàm tính RMSE

- Trực quan hóa so với dữ liệu thực tế

```
[ ]  1 # Trực quan hóa so sánh kết quả trên tập test và kết quả dự đoán
     2 x_ax = range(len(X_test_values))
     3
     4 y_pred_values = predict(X_test_values, b0, b1)
     5
     6 plt.figure(figsize=(25, 6))
     7 plt.plot(x_ax, Y_test_values, linewidth=3, label="original")
     8 plt.plot(x_ax, y_pred_values, linewidth=3.5, label="predicted")
     9 plt.title("Trực quan hóa so sánh kết quả trên tập test và kết quả dự đoán")
    10 plt.xlabel('Sale')
    11 plt.ylabel('Market Values')
    12 plt.legend(loc='best',fancybox=True, shadow=True)
    13 plt.grid(True)
    14 plt.show()
```

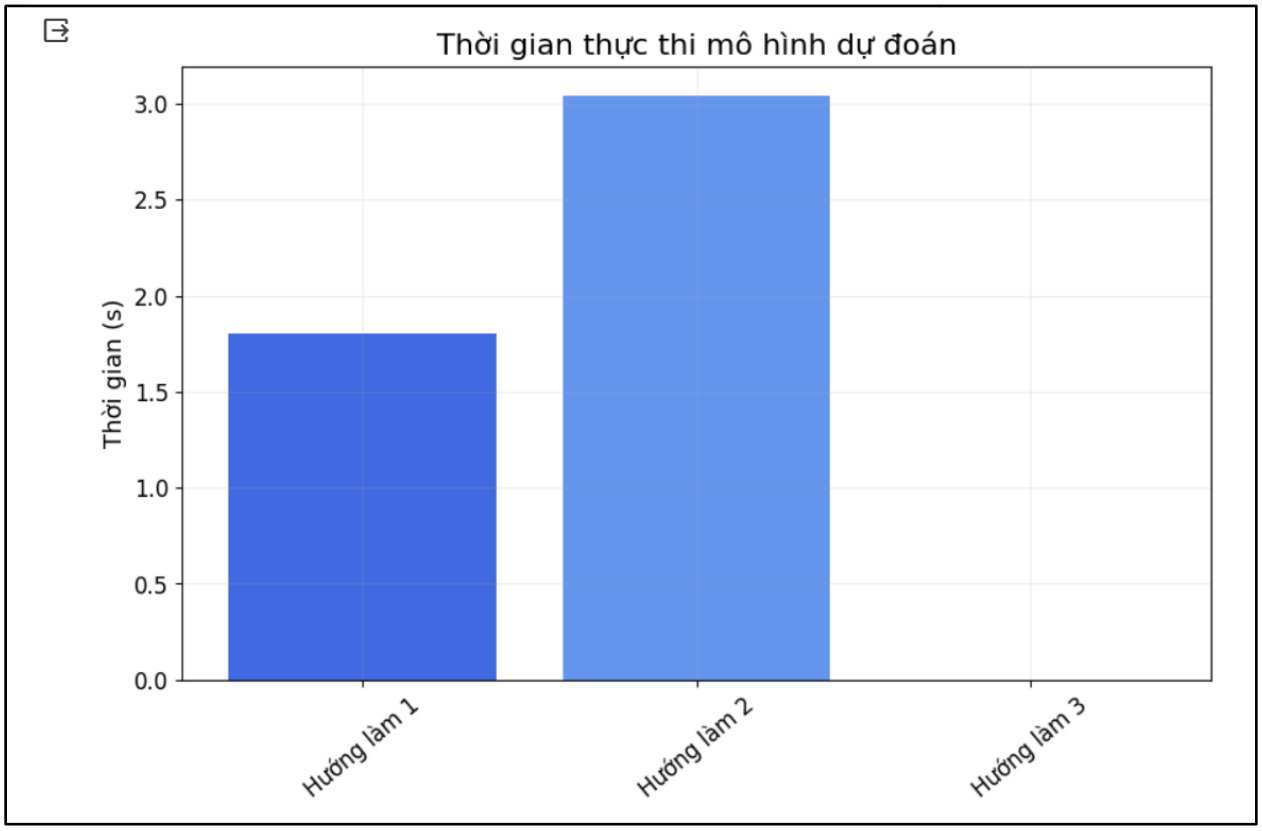
Hình 73: Khởi tạo hàm tính RMSE

CHƯƠNG 8: SO SÁNH THUẬT TOÁN

8.1 So sánh, đánh giá các hướng làm

Nhóm sẽ tiến hành so sánh và đánh giá 3 hướng mô hình làm khác nhau mà nhóm đã thực hiện so với mô hình sử dụng thuật toán Linear Regression và Ridge Regression.

8.1.1 Thời gian thuật toán thực hiện



Hình 74: So sánh thời gian thực hiện

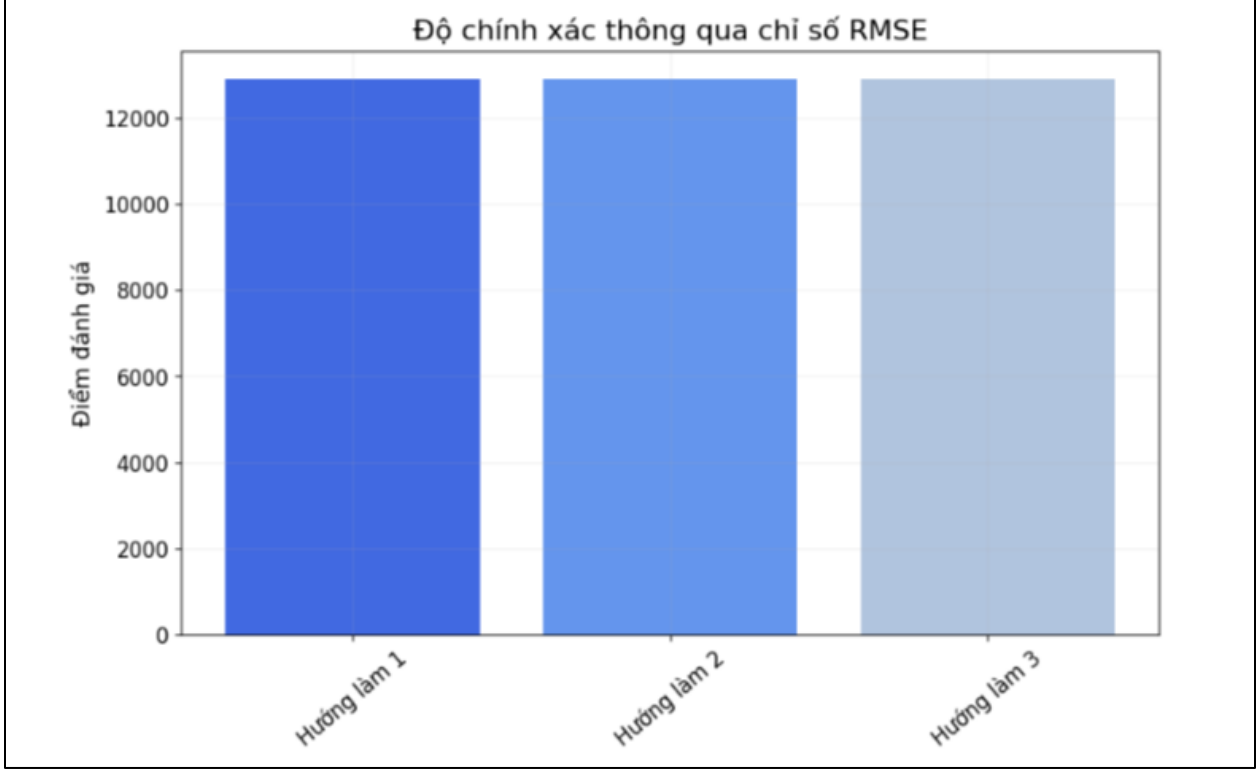
8.1.2 Độ chính xác thông qua chỉ số RMSE

```
[ ]  # create a dataset
     run_time = [evaluate_model_HL1, evaluate_model_HL2, evaluate_model_HL3]
     bars = ('Hướng làm 1', 'Hướng làm 2', 'Hướng làm 3')
     x_pos = np.arange(len(bars))

     # Create bars with different colors
     plt.figure(figsize=(10,6))
     plt.bar(x_pos, run_time, color=['royalblue', 'cornflowerblue', 'lightsteelblue'])

     # Create names on the x-axis
     plt.title('Độ chính xác thông qua chỉ số RMSE', fontsize= 16)
     plt.ylabel("Điểm đánh giá",fontsize=13)
     plt.xticks(x_pos, bars, fontsize=12, rotation = 40)
     plt.yticks(fontsize=12)
     plt.grid(alpha=0.2)
     plt.show();

     # Show graph
     plt.show()
```



Hình 75: So sánh độ chính xác bằng RMSE

CHƯƠNG 9: KẾT LUẬN

9.1 Ưu và nhược điểm giữa các hướng làm

- Thời gian thực thi dự đoán giữa hướng làm 1 và 2 chậm hơn so với hướng làm 3.
- Độ chính xác, đúng đắn giữa giá trị dự đoán và giá trị kiểm thử trên 3 hướng làm gần như tương đồng với nhau.

9.2 Hướng phát triển

- Nhóm sẽ nghiên cứu và mô phỏng lại thuật toán Multiple Regression để hiểu rõ hơn về các yếu tố nào tác động đến giá trị vốn hóa thị trường.
- Nhóm sẽ tìm thêm tài liệu về các mô hình khác như Dicision Tree, NLP để ứng dụng vô tập dữ liệu của bài toán.

CHƯƠNG 10: TÀI LIỆU THAM KHẢO

[1] – spark.apache.org, "RDD Programming Guide", "*https://spark.apache.org/docs/latest/rdd-programming-guide.html*"

[2] – machinelearningmastery.com, "How To Implement Simple Linear Regression From Scratch With Python", "*https://machinelearningmastery.com/implement-simple-linear-regression-scratch-python/*"

[3] – medium.com, " Linear Regression Full Tutorial Guide", "https://medium.com/@rupeshthetech/linear-regression-full-tutorial-guide-f32ac17caa9f"

[4] – geeksforgeeks.org, "Linear Regression (Python Implementation)", "*https://www.geeksforgeeks.org/linear-regression-python-implementation/*"

[5] – spark.apache.org, "Pyspark RDD", "https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.html"

[6] – phamdinhkhanh.github.io, "Hồi qui Ridge", "https://phamdinhkhanh.github.io/deepai-book/ch_ml/RidgedRegression.html"

[5] – machinelearningplus.com, "Pyspark Ridge Regression", "https://www.machinelearningplus.com/pyspark/pyspark-ridge-regression/"