

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN DỮ LIỆU LỚN

Đề tài:

PHÂN TÍCH BỘ DỮ LIỆU VỀ SỰ TƯƠNG TÁC CÁC BÀI ĐĂNG BÁN HÀNG TRÊN FACEBOOK LIVE TẠI THÁI LAN

Giảng Viên Hướng Dẫn: **ThS. Nguyễn Hồ Duy Trí**

Lớp: **IS405.O11**

Nhóm: **10**

Trần Gia Phong	20521748
Nguyễn Hải Đăng	20521158
Dương Ngọc Hải	20521275
Nguyễn Tiến Nhân	20521702

Thành phố Hồ Chí Minh, ngày 2 tháng 6 năm 2024

[illegible]

LỜI CẢM ƠN

Trước tiên, chúng em xin bày tỏ lòng biết ơn sâu sắc và chân thành nhất đến Ban Giám hiệu Trường Đại học Công Nghệ Thông Tin - Đại học Quốc gia Thành phố Hồ Chí Minh và Khoa Hệ Thống Thông Tin vì đã tạo điều kiện hỗ trợ và giúp đỡ chúng em trong suốt quá trình học tập và thực hiện đồ án môn học này.

Tiếp theo, chúng em xin gửi lời cảm ơn chân thành tới giảng viên bộ môn Dữ liệu Lớn, Thạc sĩ Nguyễn Hồ Duy Trí. Chúng em rất biết ơn Thầy đã truyền đạt những vốn kiến thức quý báu, tận tình hướng dẫn, quan tâm và động viên chúng em trong suốt quá trình thực hiện đề tài. Những tài liệu và kinh nghiệm mà Thầy chia sẻ đã là hành trang vô cùng quý giá cho chúng em.

Cuối cùng, chúng em xin gửi lời cảm ơn chân thành đến tất cả các bạn trong nhóm. Cảm ơn các bạn đã cùng nhau chia sẻ công việc, hoàn thành tốt trách nhiệm của cá nhân dưới sự hướng dẫn của Thầy và sự phân công của nhóm trưởng. Các bạn là những nhân tố quan trọng không thể thiếu, là chìa khóa để hoàn thành đề tài.

Mặc dù đã cố gắng hoàn thành đề tài với tất cả nỗ lực, nhưng chúng em vẫn mong nhận được sự thông cảm và những đóng góp, nhận xét quý báu từ Thầy. Những lời góp ý từ Thầy sẽ là hành trang vô cùng quý giá để chúng em vận dụng cho những môn học khác trong tương lai.

Chúng em xin chân thành cảm ơn!
Thành phố Hồ Chí Minh, tháng năm

Nhóm sinh viên thực hiện

MỤC LỤC

NHẬN XÉT CỦA GIÁO VIÊN	2
MỤC LỤC.....	4
DANH MỤC BẢNG.....	8
CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI.....	9
1.1 Lý do chọn đề tài.....	9
1.2 Dữ liệu	9
1.2.1 Bộ dữ liệu	9
1.2.2 Mô tả dữ liệu	9
1.2.3 Thống kê dữ liệu	11
1.2 Mô tả bài toán	17
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	18
2.1 Kỹ thuật tiền xử lý dữ liệu.....	18
2.1.1 Tải và Kiểm Tra Dữ Liệu (Data Loading and Inspection)	18
2.1.2 Chuyển Đổi Kiểu Dữ Liệu (Data Type Conversion)	18
2.1.3 Loại Bỏ Cột (Dropping Columns).....	18
2.1.4 Mã Hóa Nhãn (Label Encoding)	19
2.2 Các thuật toán áp dụng.....	20
2.2.1 Linear Regression.....	20
2.2.1 K-Means	20
2.3 Phương pháp đánh giá thuật toán.....	21
2.3.1 R-Squared (R^2):.....	21
2.3.2 Root Mean Square Deviation (RMSD):	22
2.3.3 So sánh giá trị dự đoán với giá trị thực tế:	22
CHƯƠNG 3: TRIỂN KHAI MÔ HÌNH.....	24
3.1 Tiền xử lý dữ liệu.....	24
3.2 Triển khai mô hình.....	33
CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC	36
4.1 Kết quả chạy thuật toán.....	36

4.2 Đánh giá.....	37
CHƯƠNG 5: KẾT LUẬN	40
5.1 Ưu điểm và hạn chế.....	40
5.1.1 Ưu điểm	40
5.1.2 Hạn chế	40
5.2 Hướng phát triển	40
PHÂN CÔNG CÔNG VIỆC	41
TÀI LIỆU THAM KHẢO	42

DANH MỤC HÌNH ẢNH

Hình 1. Minh họa bộ dữ liệu Facebook Live sellers in Thailand, UCI ML Rep	10
Hình 2. Các giá trị min, max, mean, median của các cột giá trị số	12
Hình 3. Các giá trị phổ biến và ít phổ biến trên các cột dữ liệu kiểu số	13
Hình 4. Các giá trị phổ biến nhất và ít phổ biến nhất trên cột giờ	13
Hình 5. Biểu đồ so sánh số lượng bài viết theo từng loại trạng thái	14
Hình 6. Biểu đồ stack các loại phản ứng trên từng loại bài đăng	14
Hình 7. Biểu đồ stack các loại tương tác trên từng loại bài viết	15
Hình 8. Ma trận tương quan giữa các thuộc tính	16
Hình 9. Code Ví dụ để tải và kiểm tra dữ liệu từ tệp csv	18
Hình 10. Code ví dụ về cách chuyển đổi kiểu dữ liệu từ chuỗi sang số nguyên	18
Hình 11. Code loại bỏ một cột không cần thiết	19
Hình 12. Code ví dụ mã hóa thuộc tính kiểu string	19
Hình 13. Kết quả mã hóa	19
Hình 14. Hồi quy tuyến tính	20
Hình 15. Minh họa hồi quy tuyến tính	21
Hình 16. Công thức tính khoảng cách Euclidean	22
Hình 17. Công thức tính R^2	22
Hình 18. Công thức tính RMSD	22
Hình 19. Ví dụ một hàm cal_accuracy	23
Hình 20. Cài đặt pySpark	24
Hình 21. Khởi tạo spark session	24
Hình 22. Đọc dữ liệu từ file csv	24
Hình 23. Kiểm tra dữ liệu	25
Hình 24. Kiểm tra số dòng dữ liệu	25
Hình 25. Kiểm tra cấu trúc dữ liệu	26
Hình 26. Kiểm tra có tồn tại dữ liệu null không bằng cách đếm dữ liệu không null	26
Hình 27. Kiểm tra số cột dữ liệu	26
Hình 28. Xóa các cột dữ liệu thừa	27
Hình 29. Kiểm tra số cột sau khi xóa	27
Hình 30. Kiểm tra các giá trị duy nhất	27
Hình 31. Lấy dữ liệu của cột "status_published" và chuyển thành danh sách kiểu Python	28
Hình 32. Tách ngày và giờ	28
Hình 33. Định dạng lại kiểu dữ liệu ngày	28
Hình 34. Import các thư viện và khởi tạo hàm cần thiết	29
Hình 35. Tạo một cột mới chứa giờ	29
Hình 36. Lưu giờ vào một list	29
Hình 37. Tạo cột chứa dữ liệu giờ	30
Hình 38. Chuyển kiểu dữ liệu cột "Hour" thành integer	31
Hình 39. Kiểm tra lại cấu trúc của bộ dữ liệu	31
Hình 40. Encoding	32
Hình 41. Triển khai trên cột "status_type"	32
Hình 42. Dữ liệu sau khi tiền xử lý	33
Hình 43. Định nghĩa hàm math_cal	33

Hình 44. Chạy thuật toán Linear Regression	34
Hình 45. Chia train test và ứng dụng mô hình hồi quy tuyến tính	35
Hình 46. Định nghĩa hàm predict	36
Hình 47. Dự đoán trên tập test.....	36
Hình 48. Đổi kiểu dữ liệu	37
Hình 49. Tạo cột mới chứa kết quả dự đoán.....	37
Hình 50. Hàm khởi tạo R Square.....	38
Hình 51. Hàm khởi tạo RMSD	38
Hình 52. Khởi tạo hàm cal_accuracy	38
Hình 53. Dữ liệu gốc và dữ liệu dự đoán	39
Hình 54. Tính R Square và RMSD	39
Hình 55. Tính accuracy	39

DANH MỤC BẢNG

Bảng 1. Mô tả dữ liệu.....	11
Bảng 2. Phân công công việc.....	41

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

1.1 Lý do chọn đề tài

Bán hàng trên mạng xã hội đã trở thành một phương thức kinh doanh vô cùng phổ biến trong những năm gần đây. Các nền tảng như Facebook, Instagram, TikTok và nhiều mạng xã hội khác cung cấp cơ hội tiếp cận với hàng triệu người dùng, mở ra những khả năng mới cho các doanh nghiệp trong việc tiếp thị và bán hàng. Một trong những lợi ích chính của bán hàng trên mạng xã hội là khả năng tiếp cận được với một nhóm khách hàng tiềm năng rộng lớn. Các công cụ quảng cáo và phân segmentation tinh vi trên các nền tảng này giúp doanh nghiệp xác định và nhắm mục tiêu đến đúng nhóm khách hàng mà họ muốn tiếp cận, từ đó tăng hiệu quả của các chiến dịch marketing. Hơn nữa, mạng xã hội còn mang lại cơ hội tương tác trực tiếp và xây dựng mối quan hệ gắn bó với khách hàng. Thông qua các tính năng như bình luận, chia sẻ và nhắn tin, doanh nghiệp có thể lắng nghe phản hồi từ khách hàng, hiểu rõ nhu cầu và mong muốn của họ, qua đó cải thiện chất lượng sản phẩm và dịch vụ.

Trong số các nền tảng mạng xã hội, Facebook được xem là một trong những kênh bán hàng trực tuyến hữu hiệu nhất. Với hàng tỷ người dùng trên toàn cầu, Facebook cung cấp nhiều tính năng như Trang Facebook, Nhóm và Marketplace để doanh nghiệp có thể quảng bá, tiếp thị và thực hiện các giao dịch trực tiếp. Các công cụ quảng cáo nâng cao trên Facebook còn giúp doanh nghiệp tối ưu hóa chiến lược tiếp cận khách hàng.

Sử dụng các giải thuật big data để phân tích dữ liệu liên quan đến bán hàng trên Facebook là một chiến lược vô cùng quan trọng để tối ưu hóa hiệu quả bán hàng trên nền tảng này. Thông qua việc thu thập và phân tích các dữ liệu lớn từ Facebook, như dữ liệu về hành vi người dùng, thông tin nhân khẩu học, tương tác với nội dung và quảng cáo, doanh nghiệp có thể tạo ra những hiểu biết sâu sắc về khách hàng của mình. Các công cụ phân tích dữ liệu tiên tiến sẽ giúp doanh nghiệp phân đoạn khách hàng một cách chính xác, xác định các phân khúc quan trọng và đưa ra các chiến lược tiếp thị phù hợp. Với sự kết hợp giữa big data và các chiến lược bán hàng trên mạng xã hội, doanh nghiệp sẽ có thể đạt được những kết quả kinh doanh ấn tượng trên Facebook.

1.2 Dữ liệu

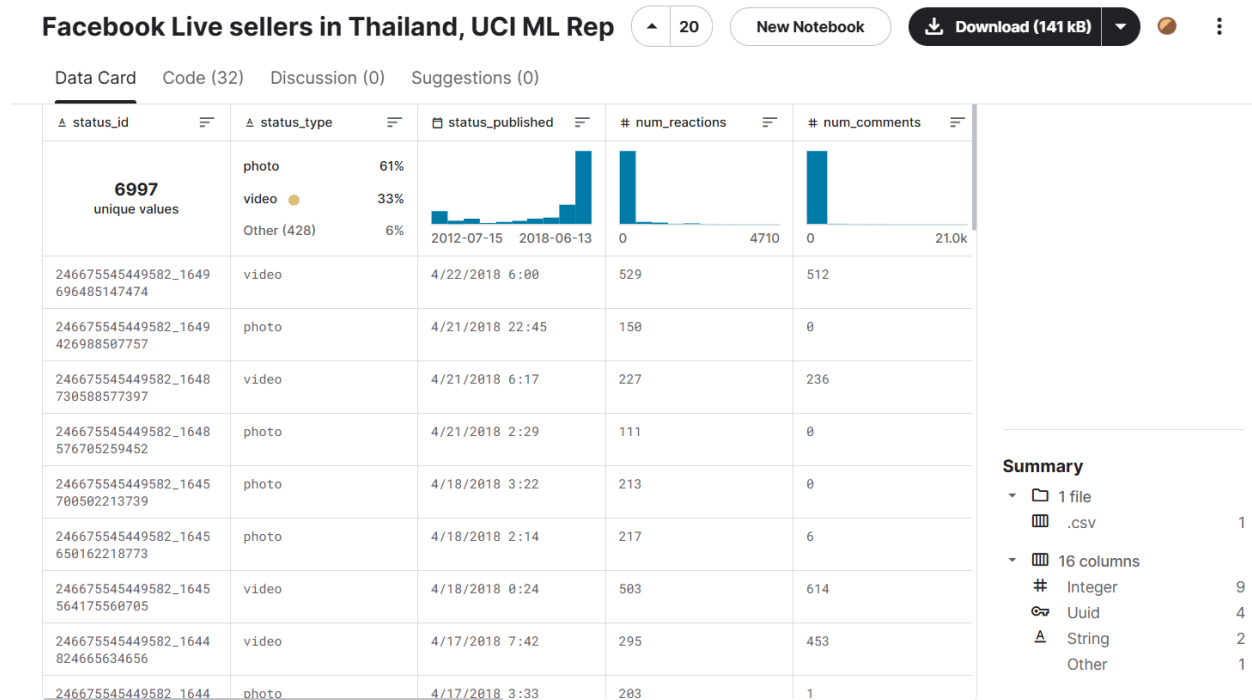
1.2.1 Bộ dữ liệu

Tên: “Facebook Live sellers in Thailand, UCI ML Repo”

Nguồn: [Facebook Live sellers in Thailand, UCI ML Repo \(kaggle.com\)](https://www.kaggle.com/datasets/uciml/facebook-live-sellers-in-thailand)

1.2.2 Mô tả dữ liệu

Dataset theo dõi các post được đăng trên ứng dụng Facebook, các dữ liệu như thời điểm đăng bài, loại bài viết, lượt tương tác, bình luận, chia sẻ, cảm xúc.



Hình 1. Minh họa bộ dữ liệu Facebook Live sellers in Thailand, UCI ML Rep

Dataset là file csv tên là Live.csv, gồm dữ liệu có 16 thuộc tính và 7050 dòng, với mô tả thuộc tính trong bảng sau:

Tên thuộc tính	Kiểu dữ liệu	Mô tả thuộc tính
status_id	integer	Mã định danh của post
status_type	string	Loại post
status_published	string	Thời điểm người dùng đăng bài (%m-%d-%y %h-%m)
num_reactions	integer	Số lượng cảm xúc của post
num_comments	integer	Số lượng bình luận của post
num_shares	integer	Số lượng chia sẻ của post
num_likes	integer	Số lượng thích của post
num_loves	integer	Số lượng yêu thích của post
num_wows	integer	Số lượng wows của post


num_hahas	integer	Số lượng hahas của post
num_sads	integer	Số lượng buồn của post
num_angrys	integer	Số lượng phẫn nộ của post
Column1	string	
Column2	string	
Column3	string	
Column4	string	

Bảng 1. Mô tả dữ liệu

1.2.3 Thống kê dữ liệu

Sử dụng PySpark để trực quan hóa và nhiều tác vụ khác đối với bộ dữ liệu từ đó có thể hiểu rõ hơn để thực hiện phân tích.

Các giá trị Min, Max, mean, median:



```
Column: num_reactions
Min: 0
Max: 4710
Mean: 230.11716312056737
Median: 57.0

Column: num_comments
Min: 0
Max: 20990
Mean: 224.3560283687943
Median: 4.0

Column: num_shares
Min: 0
Max: 3424
Mean: 40.022553191489365
Median: 0.0

Column: num_likes
Min: 0
Max: 4710
Mean: 215.0431205673759
Median: 55.0

Column: num_loves
Min: 0
Max: 657
Mean: 12.728652482269503
Median: 0.0

Column: num_wows
Min: 0
Max: 278
Mean: 1.2893617021276595
Median: 0.0

Column: num_hahas
Min: 0
Max: 157
Mean: 0.6964539007092199
Median: 0.0

Column: num_sads
Min: 0
Max: 51
Mean: 0.24368794326241136
Median: 0.0

Column: num_angrys
Min: 0
Max: 31
Mean: 0.11319148936170213
Median: 0.0
```

Hình 2. Các giá trị min, max, mean, median của các cột giá trị số

Các giá trị phổ biến nhất (mode) và ít phổ biến (least frequent) nhất của các cột kiểu số:

Column: num_reactions
Mode: 1
Least Frequent: 1580

Column: num_comments
Mode: 0
Least Frequent: 3175

Column: num_shares
Mode: 0
Least Frequent: 463

Column: num_likes
Mode: 1
Least Frequent: 1580

Column: num_loves
Mode: 0
Least Frequent: 251

Column: num_wows
Mode: 0
Least Frequent: 31

Column: num_hahas
Mode: 0
Least Frequent: 28

Column: num_sads
Mode: 0
Least Frequent: 28

Column: num_angrys
Mode: 0
Least Frequent: 31

Hình 3. Các giá trị phổ biến và ít phổ biến trên các cột dữ liệu kiểu số

Các giá trị phổ biến nhất (mode) và ít phổ biến (least frequent) nhất của cột giờ (sau khi tách giờ ra khỏi ngày):

status_id	status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys	num_id	time	new_date	hour
246675545449582_1649696485147474	video	529	512	262	432	92	3	1	1	0	1	6:00	2018-01-22	6
246675545449582_1649426988507757	photo	150	0	0	150	0	0	0	0	0	2	22:45	2018-01-21	22
246675545449582_1648730588577397	video	227	236	57	204	21	1	1	0	0	3	6:17	2018-01-21	6
246675545449582_1648576705259452	photo	111	0	0	111	0	0	0	0	0	4	2:29	2018-01-21	2
246675545449582_1645700502213739	photo	213	0	0	204	9	0	0	0	0	5	3:22	2018-01-18	3
246675545449582_1645650162218773	photo	217	6	0	211	5	1	0	0	0	6	2:14	2018-01-18	2
246675545449582_1645564175506705	video	503	614	72	418	70	10	2	0	3	7	0:24	2018-01-18	0
246675545449582_1644824665634656	video	295	453	53	260	32	1	1	0	1	8	7:42	2018-01-17	7
246675545449582_1644655795651543	photo	203	1	0	198	5	0	0	0	0	9	3:33	2018-01-17	3
246675545449582_1638788379571618	photo	170	9	1	167	3	0	0	0	0	10	4:53	2018-01-11	4
246675545449582_1637655039684952	photo	210	2	3	202	7	1	0	0	0	11	1:01	2018-01-10	1
246675545449582_163673006444122	photo	222	4	0	213	5	4	0	0	0	12	2:06	2018-01-09	2
246675545449582_163584603199186	photo	313	4	2	305	6	2	0	0	0	13	5:10	2018-01-08	5
246675545449582_1635730986544024	photo	209	4	0	200	8	1	0	0	0	14	2:23	2018-01-08	2
246675545449582_1632874756829647	photo	346	11	0	335	10	1	0	0	0	15	9:23	2018-01-05	9
246675545449582_1628507150599741	video	332	100	30	303	23	1	5	0	0	16	5:16	2018-01-01	5
246675545449582_1626504134125376	video	135	256	79	117	18	0	0	0	0	17	8:28	2018-01-30	8
246675545449582_1622470701203386	video	150	173	47	132	16	1	0	1	0	18	8:09	2018-01-26	8
246675545449582_1619188648198258	video	221	166	36	192	28	0	1	0	0	19	7:09	2018-01-23	7
246675545449582_161785807831315	photo	152	2	0	149	3	0	0	0	0	20	1:25	2018-01-22	1

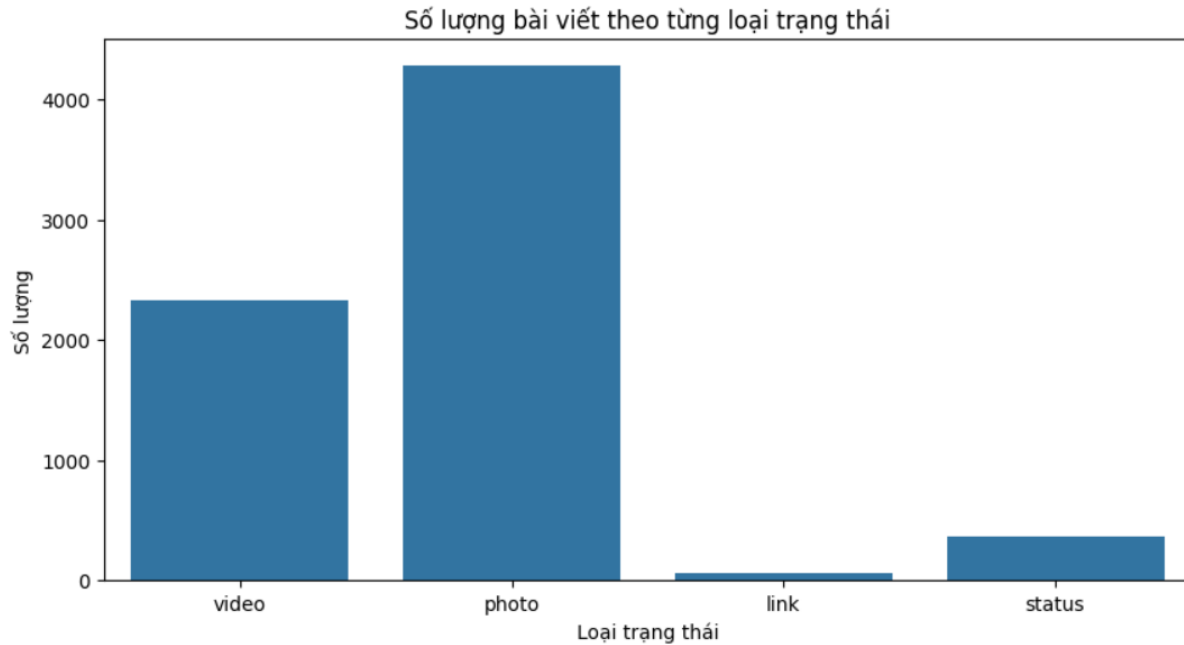
only showing top 20 rows

Giờ phổ biến nhất: 7
Giờ ít phổ biến nhất: 16

Hình 4. Các giá trị phổ biến nhất và ít phổ biến nhất trên cột giờ

Khám phá các thuộc tính

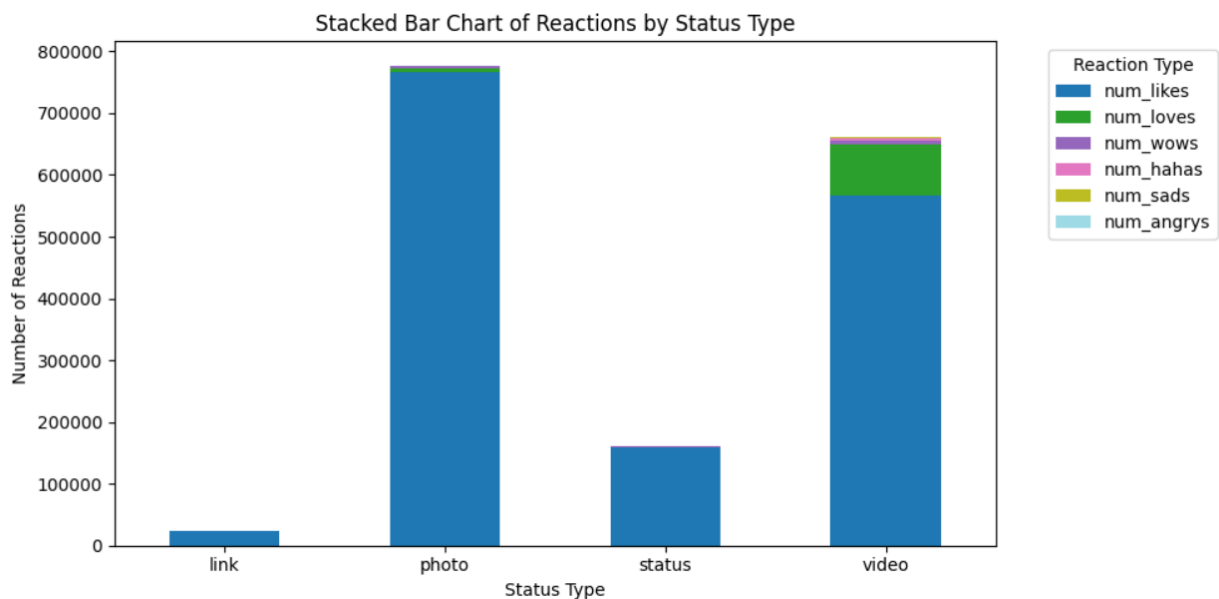
Biểu đồ trực quan hóa số lượng bài viết theo từng loại trạng thái



Hình 5. Biểu đồ so sánh số lượng bài viết theo từng loại trạng thái

Nhận xét: Các bài viết loại video, hình ảnh chiếm số lượng lớn, trong khi loại status thuần (văn bản) và đường dẫn chiếm số lượng rất nhỏ. Điều đó cho thấy người bán thường lựa chọn thêm ảnh và video để thu hút người mua.

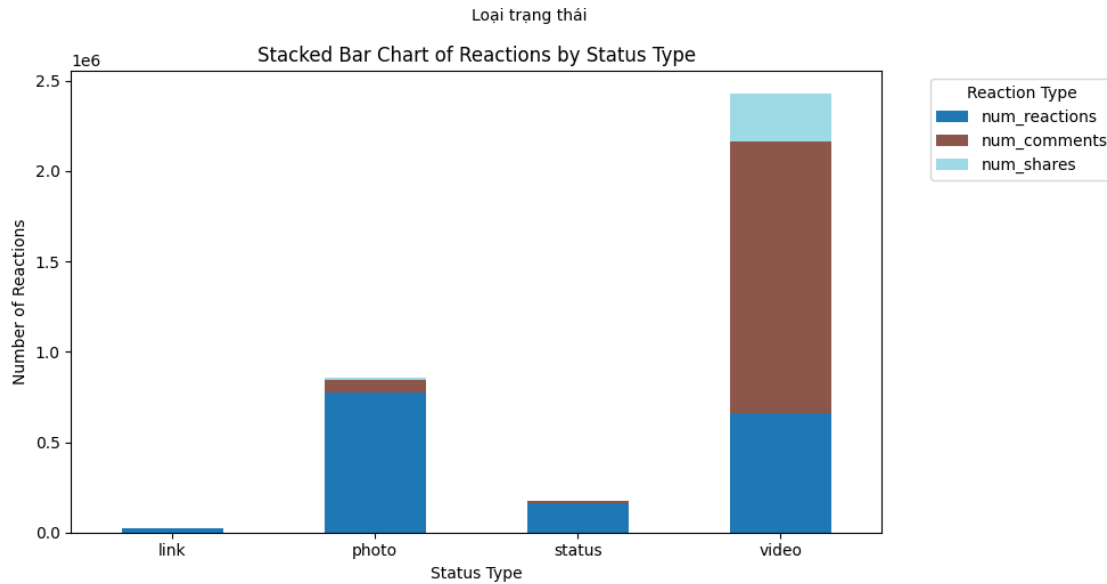
Biểu đồ stack các loại phản ứng (likes, loves, wows, hahas, sads, angrys) để đánh giá mức độ tương tác trên từng loại bài đăng



Hình 6. Biểu đồ stack các loại phản ứng trên từng loại bài đăng

Nhận xét: lượt likes chiếm đa số trên tất cả loại bài đăng, tuy nhiên đối với loại video, lượt yêu thích chiếm tỉ lệ lớn hơn so với trên các loại bài đăng khác. Điều này có thể ảnh hưởng đến quyết định lựa chọn loại bài đăng của người bán.

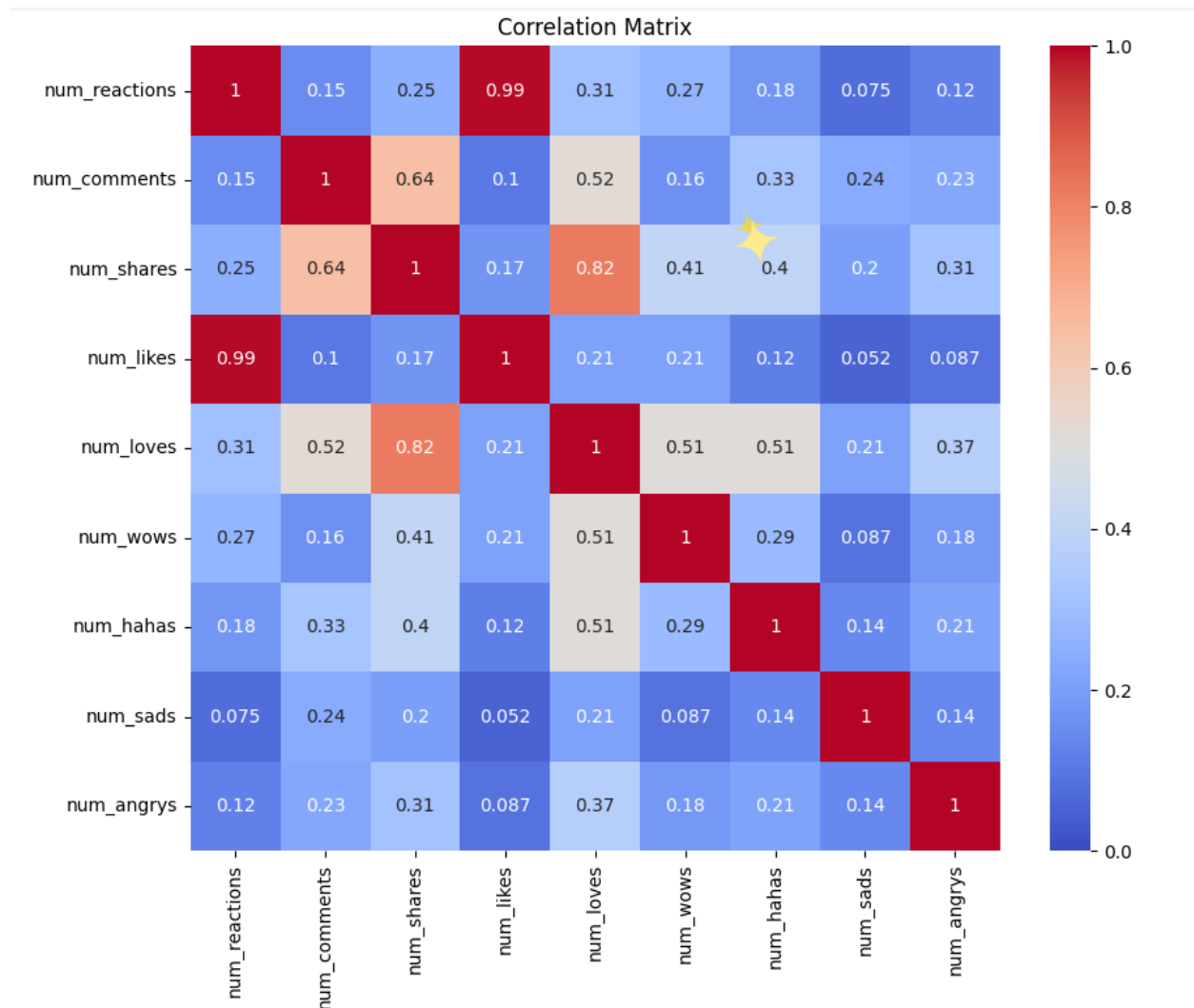
Biểu đồ stack các loại tương tác (cảm xúc, bình luận, chia sẻ) trên từng loại bài viết:



Hình 7. Biểu đồ stack các loại tương tác trên từng loại bài viết

Nhận xét: ngoại trừ loại bài đăng là video có tỉ lệ bình luận rất cao, lượt chia sẻ cũng không ít thì đặc điểm chung của các loại bài đăng còn lại là lượt tương tác cao, còn lượt bình luận và chia sẻ thấp. điều này cho thấy mức độ tương tác có hiệu quả hơn khi đăng bài bán hàng có video.

Vẽ ma trận tương quan giữa các thuộc tính:



Hình 8. Ma trận tương quan giữa các thuộc tính

Nhận xét:

- “num_reaction” và “num_likes” có độ tương quan rất cao. Điều này cho thấy sự gia tăng đồng điệu mạnh mẽ của hai cột này, cụ thể lượt reactions cao dẫn đến lượt likes sẽ cao, độ tương quan gần như bằng 1 (0.99) cho thấy lượt likes cũng chiếm tỉ trọng cao nhất trong tổng số reactions.
- num_share” và “num_comments” có độ tương quan cao với nhau và cả hai đều tương quan khá cao với loves. Điều này cho thấy mức độ yêu thích bài viết có ảnh hưởng lớn đến việc bình luận và chia sẻ bài viết.
- “num_wows” và “num_hahas” có độ tương quan khá cao với “num_loves”. Điều này chứng tỏ mức độ tương quan cao giữa các cảm xúc tích cực.

- “num_sads” và “num_angrys” có độ tương quan rất thấp với các cột khác. Từ đó cho thấy các cảm xúc tiêu cực tăng thì các lượt tương tác, phản ứng tích cực sẽ có xu hướng giảm.

1.2 Mô tả bài toán

Để hiểu rõ hơn về hoạt động của các nhà bán hàng trực tiếp trên Facebook, chúng ta có thể khai thác và phân tích bộ dữ liệu lớn "Facebook Live sellers in Thailand, UCI ML Repo" được chia sẻ trên nền tảng Kaggle. Bộ dữ liệu này chứa thông tin chi tiết về các nhà bán hàng, như hồ sơ, dữ liệu bán hàng, các chỉ số tương tác của người dùng và nhiều đặc điểm khác.

Thông qua phân tích các thông tin về lượt tương tác, bình luận, chia sẻ, cảm xúc, giờ đăng bài của các bài đăng trên bộ dữ liệu này, nhóm sẽ dự đoán loại bài đăng của các nhà bán hàng trên Facebook Live tại Thái Lan. Điều này sẽ giúp chúng ta đưa ra các chiến lược kinh doanh hiệu quả hơn, cũng như hiểu rõ hơn về tiềm năng và thách thức của thị trường bán hàng trực tuyến thông qua nền tảng mạng xã hội ở Thái Lan.

Đề tài được thực hiện trên bộ dữ liệu Real-time Facebook Live sellers, phân tích và dự đoán loại bài đăng dựa trên tương tác của bài đăng.

- Input: giờ đăng bài, lượt bình luận, chia sẻ, tương tác, các loại cảm xúc
- Output: loại bài đăng

Cách triển khai bài toán:

- Thu thập dữ liệu trên Kaggle, sử dụng bộ dữ liệu Real-time Facebook Live sellers
- Tiền xử lý dữ liệu
- Ứng dụng giải thuật huấn luyện, xây dựng mô hình

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Kỹ thuật tiền xử lý dữ liệu

2.1.1 Tải và Kiểm Tra Dữ Liệu (Data Loading and Inspection)

Tải dữ liệu từ các nguồn bên ngoài là bước đầu tiên trong quá trình xử lý dữ liệu. Các nguồn dữ liệu có thể bao gồm tệp CSV, cơ sở dữ liệu, API, v.v. Sau khi dữ liệu được tải, việc kiểm tra dữ liệu giúp hiểu rõ cấu trúc, các loại dữ liệu và phát hiện các giá trị thiếu hoặc không hợp lệ.

Ví dụ để tải và kiểm tra dữ liệu từ tệp csv có tên “dataset_Facebook” ta có thể sử dụng đoạn code sau:

```
data = spark.read.csv("dataset_Facebook.csv", header=True, inferSchema=True)
data.show(5)
```

Hình 9. Code Ví dụ để tải và kiểm tra dữ liệu từ tệp csv

2.1.2 Chuyển Đổi Kiểu Dữ Liệu (Data Type Conversion)

Chuyển đổi kiểu dữ liệu là quá trình biến đổi các giá trị của một cột trong dữ liệu từ một kiểu dữ liệu sang kiểu dữ liệu khác. Điều này thường được thực hiện để chuẩn bị dữ liệu cho các phân tích hoặc xử lý tiếp theo, hoặc để đảm bảo tính chính xác của dữ liệu. Trong nhiều trường hợp, các thuật toán và phép toán yêu cầu dữ liệu có kiểu dữ liệu cụ thể.

Trong Python, khi làm việc với dữ liệu trong PySpark DataFrame, ta có thể cần chuyển đổi kiểu dữ liệu của các cột sang kiểu dữ liệu mong muốn. Dưới đây là một ví dụ về cách chuyển đổi kiểu dữ liệu từ chuỗi sang số nguyên:

```
from pyspark.sql.types import IntegerType
data = data.withColumn("num_reactions", data["num_reactions"].cast(IntegerType()))
```

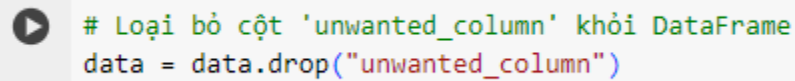
Hình 10. Code ví dụ về cách chuyển đổi kiểu dữ liệu từ chuỗi sang số nguyên

Trong ví dụ trên, **cast()** được sử dụng để chuyển đổi kiểu dữ liệu của cột 'num_reactions' từ chuỗi sang số nguyên. Điều này giúp làm cho dữ liệu trở nên dễ dàng thao tác và phân tích hơn.

2.1.3 Loại Bỏ Cột (Dropping Columns)

Loại bỏ cột là quá trình loại bỏ các cột không cần thiết hoặc không mong muốn khỏi dữ liệu. Điều này có thể cần thiết để làm cho dữ liệu gọn gàng hơn, giảm kích thước của DataFrame, hoặc tập trung vào các cột quan trọng hơn cho phân tích.

Trong một số trường hợp, khi làm việc với dữ liệu, ta có thể muốn loại bỏ một hoặc nhiều cột không cần thiết.



```
# Loại bỏ cột 'unwanted_column' khỏi DataFrame
data = data.drop("unwanted_column")
```

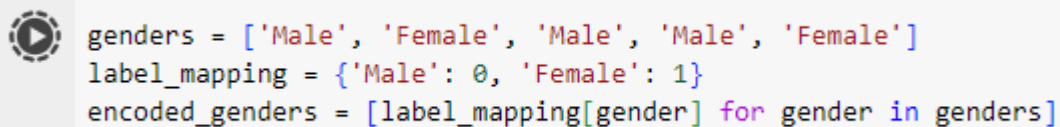
Hình 11. Code loại bỏ một cột không cần thiết

Trong ví dụ trên, **drop()** được sử dụng để loại bỏ cột có tên là “*unwanted_column*” khỏi DataFrame. Điều này giúp làm cho dữ liệu trở nên gọn gàng hơn và tập trung vào các cột quan trọng hơn cho phân tích hoặc xử lý tiếp theo.

2.1.4 Mã Hóa Nhãn (Label Encoding)

Mã hóa nhãn là kỹ thuật chuyển đổi các giá trị của biến hạng mục thành các số nguyên, giúp các thuật toán máy học có thể xử lý chúng một cách hiệu quả hơn. Các thuật toán máy học thường làm việc tốt hơn với dữ liệu số, do đó, việc chuyển đổi các giá trị hạng mục thành số là bước quan trọng trong quá trình tiền xử lý dữ liệu.

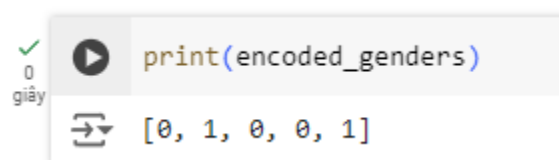
Giả sử ta có một danh sách các giá trị hạng mục như giới tính ('Male' và 'Female') và muốn chuyển đổi chúng thành các giá trị số. Ta có thể thực hiện mã hóa bằng cách:



```
genders = ['Male', 'Female', 'Male', 'Male', 'Female']
label_mapping = {'Male': 0, 'Female': 1}
encoded_genders = [label_mapping[gender] for gender in genders]
```

Hình 12. Code ví dụ mã hóa thuộc tính kiểu string

Kết quả thu được sẽ là:



```
print(encoded_genders)
```

[0, 1, 0, 0, 1]

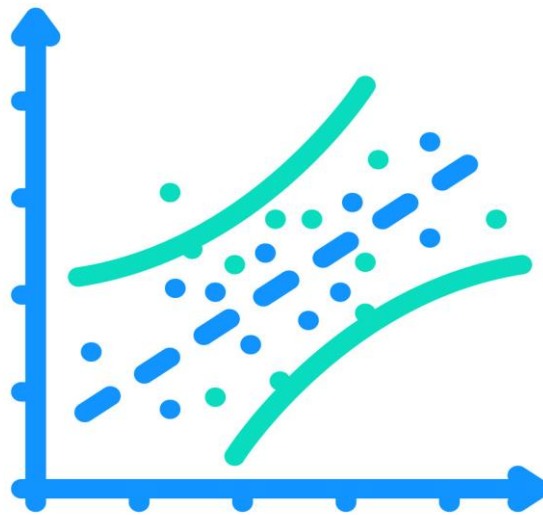
Hình 13. Kết quả mã hóa

Như vậy **genders** đã được chuyển đổi thành một mảng số nguyên. Việc này giúp đơn giản hóa quá trình xử lý dữ liệu và làm cho các thuật toán máy học có thể hoạt động hiệu quả hơn với dữ liệu số.

2.2 Các thuật toán áp dụng

2.2.1 Linear Regression

Hồi quy tuyến tính (Linear Regression) là một trong những phương pháp quan trọng nhất trong học máy và thống kê, được sử dụng rộng rãi để dự đoán giá trị của một biến phụ thuộc dựa trên một hoặc nhiều biến độc lập. Thuật ngữ "**tuyến tính**" xuất phát từ việc mối quan hệ giữa các biến được mô tả bằng một đường thẳng. Trong báo cáo này, chúng em sẽ khám phá lý thuyết đằng sau hồi quy tuyến tính và cách nó được áp dụng trong PySpark.

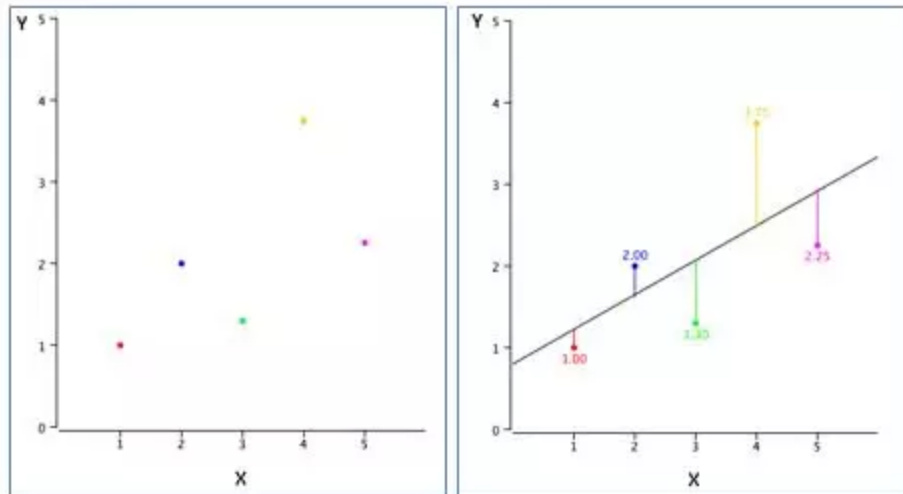


Hình 14. Hồi quy tuyến tính

Mô hình hồi quy tuyến tính giả định rằng có một mối quan hệ tuyến tính giữa biến phụ thuộc y và các biến độc lập X . Cụ thể, mô hình được biểu diễn dưới dạng:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n + \epsilon$$

Trong khi sử dụng hồi quy tuyến tính, mục tiêu của chúng ta là để làm sao một đường thẳng có thể tạo được sự phân bố gần nhất với hầu hết các điểm. Do đó làm giảm khoảng cách (sai số) của các điểm dữ liệu cho đến đường đó.



Hình 15. Minh họa hồi quy tuyến tính

Như hình minh họa, ở các điểm ở hình trên (trái) biểu diễn các điểm dữ liệu khác nhau và đường thẳng (bên phải) đại diện cho một đường gần đúng có thể giải thích mối quan hệ giữa các trục x & y . Thông qua, hồi quy tuyến tính chúng ta cố gắng tìm ra một đường như vậy. Ví dụ, nếu chúng ta có một biến phụ thuộc Y và một biến độc lập X - mối quan hệ giữa X và Y có thể được biểu diễn dưới dạng phương trình sau:

$$Y = B_0 + B_1 * X$$

Trong đó:

Y = Biến phụ thuộc

X = biến độc lập

B_0 = Hằng số

B_1 = Hệ số mối quan hệ giữa X và Y

2.3 Phương pháp đánh giá thuật toán

2.3.1 R-Squared (R^2):

R-Squared là một phép đo thường được sử dụng để đánh giá mức độ phù hợp của một mô hình hồi quy tuyến tính với dữ liệu thực tế.

Giá trị R^2 nằm trong khoảng từ 0 đến 1. Giá trị càng gần 1 cho thấy mô hình giải thích một phần lớn sự biến động của dữ liệu. Ngược lại, giá trị càng gần 0 cho thấy mô hình không giải thích được sự biến động của dữ liệu.

Công thức tính R^2 thường được biểu diễn như sau:

$$R^2 = 1 - \frac{ESS}{TSS}$$

Hình 16. Công thức tính R^2

Trong đó:

- ESS là viết tắt của Residual Sum of Squares, tức là tổng các độ lệch bình phương của phần dư.
- TSS là viết tắt của Total Sum of Squares, tức là tổng độ lệch bình phương của toàn bộ các nhân tố nghiên cứu.

2.3.2 Root Mean Square Deviation (RMSD):

RMSD là một phép đo để đánh giá độ chính xác của một mô hình dự đoán. Nó biểu thị sự chênh lệch trung bình giữa giá trị dự đoán và giá trị thực tế.

➔ Giá trị RMSD càng thấp thì mô hình càng chính xác.

Công thức tính RMSD thường được biểu diễn như sau:

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

Trong đó:

- n là số lượng mẫu trong tập dữ liệu.
- y_i là giá trị thực tế.
- \hat{y}_i là giá trị dự đoán của mô hình cho mẫu thứ i .

Hình 17. Công thức tính RMSD

R-Squared và RMSD là hai phép đo quan trọng để đánh giá hiệu suất của một mô hình hồi quy tuyến tính trên dữ liệu thực tế. R-Squared giúp hiểu được tỷ lệ biến động dữ liệu mà mô hình có thể giải thích, trong khi RMSD cho biết mức độ chính xác trung bình của dự đoán so với giá trị thực tế.

2.3.3 So sánh giá trị dự đoán với giá trị thực tế:

Trong quá trình triển khai thuật toán với Pyspark, chúng em đã đối mặt với hạn chế là không được phép sử dụng các thư viện bên ngoài như sklearn.metrics để đánh giá mô hình. Để giải quyết vấn đề này, nhóm của chúng em đã xây dựng một hàm để tính độ chính xác một cách đơn giản nhưng hiệu quả.

Ví dụ một hàm **cal_accuracy** :

```
[ ] def cal_accuracy(df,actual, predict):  
    actual_values = df.select(actual).rdd.flatMap(lambda x: x).collect()  
    predict_values = df.select(predict).rdd.flatMap(lambda x: x).collect()  
    val = 0  
    for i in range(len(actual_values)):  
        if str(actual_values[i]) == str(predict_values[i]):  
            val= val + 1  
    return val/len(actual_values)*100
```

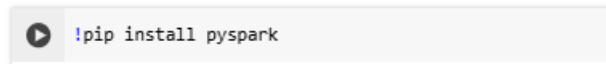
Hình 18. Ví dụ một hàm *cal_accuracy*

Hàm **cal_accuracy** nhận vào một DataFrame **df** chứa dữ liệu cần đánh giá, cùng với tên của cột chứa giá trị thực tế (**actual**) và tên của cột chứa giá trị dự đoán (**predict**). Sau đó, nó tính toán tỷ lệ phần trăm của các dự đoán chính xác bằng cách so sánh giữa giá trị thực tế và giá trị dự đoán.

CHƯƠNG 3: TRIỂN KHAI MÔ HÌNH

3.1 Tiền xử lý dữ liệu

- Cài đặt pySpark



Hình 19. Cài đặt pySpark

- Khởi tạo spark session

```
[ ] from pyspark.sql import SparkSession

# Initialize Spark session
spark = SparkSession.builder \
    .appName("Facebook Live sellers in Thailand") \
    .getOrCreate()
```

Hình 20. Khởi tạo spark session

- Đọc dữ liệu từ file csv

```
[ ] from re import MULTILINE
file_path = 'Live_20210128.csv'
spark_df = spark.read.option('header', True).option('inferSchema', True).csv(file_path)
#
# .option("quote", "\"")\
# .option("escape", "\\")\
```

Hình 21. Đọc dữ liệu từ file csv

- Kiểm tra dữ liệu


```
spark_df.show(vertical=True)
```

-RECORD 0-----	
status_id	1
status_type	video
status_published	4/22/2018 6:00
num_reactions	529
num_comments	512
num_shares	262
num_likes	432
num_loves	92
num_wows	3
num_hahas	1
num_sads	1
num_angrys	0
Column1	NULL
Column2	NULL
Column3	NULL
Column4	NULL
-RECORD 1-----	
status_id	2
status_type	photo
status_published	4/21/2018 22:45
num_reactions	150
num_comments	0
num_shares	0
num_likes	150
num_loves	0
num_wows	0
num_hahas	0
num_sads	0
num_angrys	0
Column1	NULL
Column2	NULL
Column3	NULL
Column4	NULL
-RECORD 2-----	

Hình 22. Kiểm tra dữ liệu

- Kiểm tra số dòng dữ liệu(7050)

```
[ ] spark_df.count()
```

7050

Hình 23. Kiểm tra số dòng dữ liệu

- Kiểm tra cấu trúc dữ liệu

```
[ ] spark_df.printSchema()

root
|-- status_id: integer (nullable = true)
|-- status_type: string (nullable = true)
|-- status_published: string (nullable = true)
|-- num_reactions: integer (nullable = true)
|-- num_comments: integer (nullable = true)
|-- num_shares: integer (nullable = true)
|-- num_likes: integer (nullable = true)
|-- num_loves: integer (nullable = true)
|-- num_wows: integer (nullable = true)
|-- num_hahas: integer (nullable = true)
|-- num_sads: integer (nullable = true)
|-- num_angrys: integer (nullable = true)
|-- Column1: string (nullable = true)
|-- Column2: string (nullable = true)
|-- Column3: string (nullable = true)
|-- Column4: string (nullable = true)
```

Hình 24. Kiểm tra cấu trúc dữ liệu

- Kiểm tra có tồn tại dữ liệu null không bằng cách đếm dữ liệu không null

```
# checking null data
from pyspark.sql.functions import col, count, isnan, lit, sum

def count_not_null(c, nan_as_null=False):
    """Use conversion between boolean and integer
    - False -> 0
    - True -> 1
    """
    pred = col(c).isNotNull() & (~isnan(c) if nan_as_null else lit(True))
    return sum(pred.cast("integer")).alias(c)

spark_df.agg([count_not_null(c) for c in spark_df.columns]).show()
```

status_id	status_type	status_published	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys	Column1	Column2	Column3	Column4
7050	7050	7050	7050	7050	7050	7050	7050	7050	7050	7050	7050	0	0	0	0

Hình 25. Kiểm tra có tồn tại dữ liệu null không bằng cách đếm dữ liệu không null

- Kiểm tra số cột dữ liệu

```
spark_df.columns

['status_id',
 'status_type',
 'status_published',
 'num_reactions',
 'num_comments',
 'num_shares',
 'num_likes',
 'num_loves',
 'num_wows',
 'num_hahas',
 'num_sads',
 'num_angrys',
 'Column1',
 'Column2',
 'Column3',
 'Column4']
```

Hình 26. Kiểm tra số cột dữ liệu

- Xóa các cột dữ liệu thừa ‘Column1’, ‘Column2’, ‘Column3’, ‘Column4’ vì các cột này không có giá trị tính toán

```
[ ] # Xóa các cột dữ liệu dư thừa
drop_cols=[]
spark_df = spark_df.drop('Column1','Column2','Column3','Column4')
spark_df.show()
```

	status_id	status_type	status_published	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys
1	video	4/22/2018 6:00	529	512	262	432	92	3	1	1	0	
2	photo	4/21/2018 22:45	150	0	0	150	0	0	0	0	0	
3	video	4/21/2018 6:17	227	236	57	204	21	1	1	0	0	
4	photo	4/21/2018 2:29	111	0	0	111	0	0	0	0	0	
5	photo	4/18/2018 3:22	213	0	0	204	9	0	0	0	0	
6	photo	4/18/2018 2:14	217	6	0	211	5	1	0	0	0	
7	video	4/18/2018 0:24	503	614	72	418	70	10	2	0	3	
8	video	4/17/2018 7:42	295	453	53	260	32	1	1	0	1	
9	photo	4/17/2018 3:33	203	1	0	198	5	0	0	0	0	
10	photo	4/11/2018 4:53	170	9	1	167	3	0	0	0	0	
11	photo	4/10/2018 1:01	210	2	3	202	7	1	0	0	0	
12	photo	4/9/2018 2:06	222	4	0	213	5	4	0	0	0	
13	photo	4/8/2018 5:10	313	4	2	305	6	2	0	0	0	
14	photo	4/8/2018 2:23	209	4	0	200	8	1	0	0	0	
15	photo	4/5/2018 9:23	346	11	0	335	10	1	0	0	0	
16	video	4/1/2018 5:16	332	100	30	303	23	1	5	0	0	
17	video	3/30/2018 8:28	135	256	79	117	18	0	0	0	0	
18	video	3/26/2018 8:28	150	173	47	132	16	1	0	1	0	
19	video	3/23/2018 7:09	221	166	36	192	28	0	1	0	0	
20	photo	3/22/2018 1:25	152	2	0	149	3	0	0	0	0	

only showing top 20 rows

Hình 27. Xóa các cột dữ liệu thừa

- Kiểm tra số cột sau khi xóa(12)

```
[ ] len(spark_df.columns)
```

12

Hình 28. Kiểm tra số cột sau khi xóa

- Kiểm tra các giá trị duy nhất của cột ‘status_type’

```
[ ] spark_df.select('status_type').distinct().show()
```

```
+-----+
|status_type|
+-----+
|link|
|status|
|video|
|photo|
+-----+
```

Hình 29. Kiểm tra các giá trị duy nhất

- Lấy dữ liệu của cột “status_published” và chuyển thành danh sách kiểu Python và in ra


```

#tokenizer text
from pyspark.sql.functions import row_number, monotonically_increasing_id
from pyspark.sql.window import Window
from pyspark.sql import functions as func
from pyspark.sql.types import StringType, FloatType, DateType

times_udf = func.udf( \
    lambda indx: time[indx-1], StringType())
df1 = spark_df.withColumn( \
    "num_id", row_number().over( \
        Window.orderBy(monotonically_increasing_id())))

```

Hình 33. Import các thư viện và khởi tạo hàm cần thiết

- Tạo một cột mới tên là “time” trong spark_df DataFrame bằng hàm “times_udf”. Drop cột “num_id” sau khi đã hoàn tất và hiển thị kết quả

```

# Create a new column by calling the user defined function
spark_df = df1.withColumn('time', \
    times_udf('num_id'))
spark_df.drop('num_id').show()

```

status_id	status_type	status_published	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys	time
1	video	4/22/2018 6:00	529	512	262	432	92	3	1	1	0	6:00
2	photo	4/21/2018 22:45	150	0	0	150	0	0	0	0	0	22:45
3	video	4/21/2018 6:17	227	236	57	204	21	1	1	0	0	6:17
4	photo	4/21/2018 2:29	111	0	0	111	0	0	0	0	0	2:29
5	photo	4/18/2018 3:22	213	0	0	204	9	0	0	0	0	3:22
6	photo	4/18/2018 2:14	217	6	0	211	5	1	0	0	0	2:14
7	video	4/18/2018 0:24	503	614	72	418	70	10	2	0	3	0:24
8	video	4/17/2018 7:42	295	453	53	260	32	1	1	0	1	7:42
9	photo	4/17/2018 3:33	203	1	0	198	5	0	0	0	0	3:33
10	photo	4/11/2018 4:53	170	9	1	167	3	0	0	0	0	4:53
11	photo	4/10/2018 1:01	210	2	3	202	7	1	0	0	0	1:01
12	photo	4/9/2018 2:06	222	4	0	213	5	4	0	0	0	2:06
13	photo	4/8/2018 5:10	313	4	2	305	6	2	0	0	0	5:10
14	photo	4/8/2018 2:23	209	4	0	200	8	1	0	0	0	2:23
15	photo	4/5/2018 9:23	346	11	0	335	10	1	0	0	0	9:23
16	video	4/1/2018 5:16	332	100	30	303	23	1	5	0	0	5:16
17	video	3/30/2018 8:28	135	256	79	117	18	0	0	0	0	8:28
18	video	3/26/2018 8:28	150	173	47	132	16	1	0	1	0	8:28
19	video	3/23/2018 7:09	221	166	36	192	28	0	1	0	0	7:09
20	photo	3/22/2018 1:25	152	2	0	149	3	0	0	0	0	1:25

only showing top 20 rows

Hình 34. Tạo một cột mới chứa giờ

- Lấy dữ liệu giờ từ cột “time” đưa vào một list

```

[ ] time_values = spark_df.select("time").rdd.flatMap(lambda x: x).collect()

time = []
for i in range(len(time_values)):
    val = time_values[i].split(':')
    time.append((val[0]))
print(time)
print(type(time))

```

```

['6', '22', '6', '2', '3', '2', '0', '7', '3', '4', '1', '2', '5', '2', '9', '5', '8', '8', '7', '1', '8', '7', '1', '1', '0', '22', '8', '7', '5', '7', '5', '13', '6']
<class 'list'>

```

Hình 35. Lưu giờ vào một list

- Thêm các hàm cần thiết để thêm cột “Hour” chứa dữ liệu giờ vào bộ dữ liệu và hiển thị kết quả. Sau đó drop cột “time” vì không còn cần thiết cho việc dự đoán

```

times_udf = func.udf( \
    lambda indx: time[indx-1])
df1 = spark_df.withColumn( \
    "num_id", row_number().over( \
        Window.orderBy(monotonically_increasing_id()))))

# Create a new column by calling the user defined function
spark_df = df1.withColumn('Hour', \
    times_udf('num_id'))
spark_df= spark_df.drop('num_id')
spark_df.show()

```

status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys	time	Hour
video	529	512	262	432	92	3	1	1	0	6:00	6
photo	150	0	0	150	0	0	0	0	0	22:45	22
video	227	236	57	204	21	1	1	0	0	6:17	6
photo	111	0	0	111	0	0	0	0	0	2:29	2
photo	213	0	0	204	9	0	0	0	0	3:22	3
photo	217	6	0	211	5	1	0	0	0	2:14	2
video	503	614	72	418	70	10	2	0	3	0:24	0
video	295	453	53	260	32	1	1	0	1	7:42	7
photo	203	1	0	198	5	0	0	0	0	3:33	3
photo	170	9	1	167	3	0	0	0	0	4:53	4
photo	210	2	3	202	7	1	0	0	0	1:01	1
photo	222	4	0	213	5	4	0	0	0	2:06	2
photo	313	4	2	305	6	2	0	0	0	5:10	5
photo	209	4	0	200	8	1	0	0	0	2:23	2
photo	346	11	0	335	10	1	0	0	0	9:23	9
video	332	100	30	303	23	1	5	0	0	5:16	5
video	135	256	79	117	18	0	0	0	0	8:28	8
video	150	173	47	132	16	1	0	1	0	8:28	8
video	221	166	36	192	28	0	1	0	0	7:09	7
photo	152	2	0	149	3	0	0	0	0	1:25	1

only showing top 20 rows

```


[ ] spark_df = spark_df.drop('time')

```

Hình 36. Tạo cột chứa dữ liệu giờ

- Chuyển kiểu dữ liệu cột “Hour” thành integer

```
[ ] from pyspark.sql.types import (IntegerType, DoubleType, StringType)
    spark_df = spark_df.withColumn('Hour', spark_df.Hour.cast(IntegerType()))
    spark_df.show()
```




status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys	Hour
video	529	512	262	432	92	3	1	1	0	6
photo	150	0	0	150	0	0	0	0	0	22
video	227	236	57	204	21	1	1	0	0	6
photo	111	0	0	111	0	0	0	0	0	2
photo	213	0	0	204	9	0	0	0	0	3
photo	217	6	0	211	5	1	0	0	0	2
video	503	614	72	418	70	10	2	0	3	0
video	295	453	53	260	32	1	1	0	1	7
photo	203	1	0	198	5	0	0	0	0	3
photo	170	9	1	167	3	0	0	0	0	4
photo	210	2	3	202	7	1	0	0	0	1
photo	222	4	0	213	5	4	0	0	0	2
photo	313	4	2	305	6	2	0	0	0	5
photo	209	4	0	200	8	1	0	0	0	2
photo	346	11	0	335	10	1	0	0	0	9
video	332	100	30	303	23	1	5	0	0	5
video	135	256	79	117	18	0	0	0	0	8
video	150	173	47	132	16	1	0	1	0	8
video	221	166	36	192	28	0	1	0	0	7
photo	152	2	0	149	3	0	0	0	0	1

only showing top 20 rows

Hình 37. Chuyển kiểu dữ liệu cột “Hour” thành integer

- Kiểm tra lại cấu trúc của bộ dữ liệu

```
[ ] spark_df.printSchema()
```



```
root
|-- status_type: string (nullable = true)
|-- num_reactions: integer (nullable = true)
|-- num_comments: integer (nullable = true)
|-- num_shares: integer (nullable = true)
|-- num_likes: integer (nullable = true)
|-- num_loves: integer (nullable = true)
|-- num_wows: integer (nullable = true)
|-- num_hahas: integer (nullable = true)
|-- num_sads: integer (nullable = true)
|-- num_angrys: integer (nullable = true)
|-- Hour: integer (nullable = true)
```

Hình 38. Kiểm tra lại cấu trúc của bộ dữ liệu

- Import các thư viện cần thiết và thực hiện định nghĩa hàm label_encode trên một cột dữ liệu, chuyển đổi các giá trị trên cột sang các giá trị số nguyên tuần tự

```
[ ] from pyspark.sql.types import (IntegerType, DoubleType, StringType)
    from pyspark.sql.functions import col, udf, lit, expr, collect_set
    import pyspark.sql.functions as F
    from pyspark.sql.functions import (avg, col, countDistinct, split, isnan, when, count, udf, round as sqlround)
    import math
    import numpy as np

def label_encode(df, input_col, output_col):
    distinct_values = df.select(input_col).distinct().collect()
    mapping = {row[input_col]: index for index, row in enumerate(distinct_values)}
    mapping_broadcast = spark.sparkContext.broadcast(mapping)
    def label_encode_udf(value):
        return mapping_broadcast.value.get(value, None)
    spark.udf.register("label_encode_udf", label_encode_udf, DoubleType())
    df = df.withColumn(output_col, lit(None).cast(DoubleType()))
    for value, index in mapping.items():
        condition = col(input_col) == lit(value)
        df = df.withColumn(output_col, when(condition, lit(index).cast(DoubleType())).otherwise(col(output_col)))
    df = df.drop(input_col).withColumnRenamed(output_col, input_col)
    return df
```

Hình 39. Encoding

- Thực hiện hàm label_encode trên cột “status_type” và hiển thị kết quả

```
[ ] df = spark_df
```

```
[ ] df = label_encode(df, 'status_type', 'type')
```

```
[ ] df.printSchema()
```

```
root
 |-- num_reactions: integer (nullable = true)
 |-- num_comments: integer (nullable = true)
 |-- num_shares: integer (nullable = true)
 |-- num_likes: integer (nullable = true)
 |-- num_loves: integer (nullable = true)
 |-- num_wows: integer (nullable = true)
 |-- num_hahas: integer (nullable = true)
 |-- num_sads: integer (nullable = true)
 |-- num_angrys: integer (nullable = true)
 |-- Hour: integer (nullable = true)
 |-- status_type: double (nullable = true)
```

```
[ ] df.select("status_type").distinct().show()
```

```
+-----+
|status_type|
+-----+
|         0.0|
|         1.0|
|         3.0|
|         2.0|
+-----+
```

Hình 40. Triển khai trên cột “status_type”

- Hiển thị dữ liệu sau khi đã hoàn tất tiền xử lý

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

	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys	Hour	status_type
	529	512	262	432	92	3	1	1	0	6	2.0
	150	0	0	150	0	0	0	0	0	22	3.0
	227	236	57	204	21	1	1	0	0	6	2.0
	111	0	0	111	0	0	0	0	0	2	3.0
	213	0	0	204	9	0	0	0	0	3	3.0
	217	6	0	211	5	1	0	0	0	2	3.0
	503	614	72	418	70	10	2	0	3	0	2.0
	295	453	53	260	32	1	1	0	1	7	2.0
	203	1	0	198	5	0	0	0	0	3	3.0
	170	9	1	167	3	0	0	0	0	4	3.0
	210	2	3	202	7	1	0	0	0	1	3.0
	222	4	0	213	5	4	0	0	0	2	3.0
	313	4	2	305	6	2	0	0	0	5	3.0
	209	4	0	200	8	1	0	0	0	2	3.0
	346	11	0	335	10	1	0	0	0	9	3.0
	332	100	30	303	23	1	5	0	0	5	2.0
	135	256	79	117	18	0	0	0	0	8	2.0
	150	173	47	132	16	1	0	1	0	8	2.0
	221	166	36	192	28	0	1	0	0	7	2.0
	152	2	0	149	3	0	0	0	0	1	3.0

only showing top 20 rows

Hình 41. Dữ liệu sau khi tiền xử lý

Tiền xử lý dữ liệu đến đây là kết thúc, đến đây có thể bắt đầu thực hiện các giải thuật dự đoán trên bộ dữ liệu.

3.2 Triển khai mô hình

- Định nghĩa hàm `math_cal`. Hàm này được sử dụng để tính toán các giá trị cần thiết cho việc thực hiện hồi quy tuyến tính

```
[ ] def math_cal(lines,index,i):
    return (index, float(lines[i]) * float(lines[index]))
```

Hình 42. Định nghĩa hàm `math_cal`

- Chạy thuật toán

```

def linear_regression(data_train, X_train, y_train):
    value_X = []
    value_y = []

    data_train.createOrReplaceTempView("data")
    columns_data = ""

    for i in range(len(X_train)):
        columns_data = columns_data + X_train[i] + ","
    columns_data = columns_data + y_train[0]

    sql_data_train = spark.sql("SELECT " + columns_data + " FROM data")

    data_train_rdd = sql_data_train.rdd
    n = data_train_rdd.count()
    value_X.append(float(n))

    map_X_sum = data_train_rdd.map(lambda lines: [(str(index), float(lines[index])) for index in range(len(lines))])
    flatmap_X_sum = map_X_sum.flatMap(lambda lines: lines)
    reduce_X_sum = flatmap_X_sum.reduceByKey(lambda a,b: a+b)
    data_X_sum=reduce_X_sum.collect()

    for i in range(len(data_X_sum)):
        if(i == len(data_X_sum)-1):
            value_y.append(data_X_sum[i][1])
        else:
            value_X.append(data_X_sum[i][1])

    for i in range (len(data_X_sum)-1):
        map_X_vol = data_train_rdd.map(lambda lines:[math_cal(lines, index, i) for index in range(len(lines))])
        flatmap_X_vol = map_X_vol.flatMap(lambda lines: lines)
        reduce_X_vol = flatmap_X_vol.reduceByKey(lambda a,b: a+b)
        data_X_vol = reduce_X_vol.collect()

        value_X.append(data_X_vol[i][1])

        for j in range(len(data_X_vol)):
            if(j == len(data_X_vol) - 1):
                value_y.append(data_X_vol[j][1])
            else:
                value_X.append(data_X_vol[j][1])

    vector_X = np.asarray(value_X, dtype=np.float32)
    vector_X_convert=np.reshape(vector_X, (len(X_train)+1, len(X_train)+1))

    vector_y = np.asarray(value_y, dtype=np.float32)
    vector_y_convert=np.reshape(vector_y, (len(X_train)+1, 1))

    coefficient = np.linalg.lstsq(vector_X_convert, vector_y_convert)[0]
    return coefficient

```

Hình 43. Chạy thuật toán Linear Regression

Mô tả thuật toán:

1. Chuẩn bị dữ liệu:

- value_X và value_y là hai danh sách để lưu trữ các giá trị đầu vào (X) và đầu ra (y) của mô hình.
- Tạo một view tạm thời "data" từ dataframe data_train.
- Tạo chuỗi columns_data chứa tên các cột X và y.
- Sử dụng Spark SQL để truy vấn dữ liệu từ view "data" và lưu vào sql_data_train.
- Chuyển sql_data_train thành RDD (data_train_rdd).

2. Tính tổng các giá trị X:
 - Sử dụng `map()` để tạo một RDD mới, với mỗi phần tử là một danh sách chứa các cặp (tên cột, giá trị) (`map_X_sum`).
 - Sử dụng `flatMap()` để làm phẳng RDD, thành một danh sách các cặp (tên cột, giá trị) (`flatMap_X_sum`).
 - Sử dụng `reduceByKey()` để tính tổng các giá trị cho từng cột (`reduce_X_sum`).
 - Lưu trữ các tổng các giá trị X vào `value_X` và tổng giá trị y vào `value_y`.
3. Tính ma trận X và vector y:
 - Sử dụng `map()` để tính các giá trị cho ma trận X, sử dụng hàm `math_cal()` (không được hiển thị trong đoạn code).
 - Lặp lại các bước trước để tính tổng các giá trị cho ma trận X.
 - Lưu trữ các giá trị vào `value_X` và `value_y`.
4. Tính hệ số của mô hình hồi quy tuyến tính:
 - Chuyển `value_X` và `value_y` thành các mảng NumPy (`vector_X` và `vector_y`).
 - Reshape các mảng thành ma trận và vector.
 - Sử dụng `np.linalg.lstsq()` để tìm hệ số của mô hình hồi quy tuyến tính.
 - Trả về hệ số tìm được.
 - Huấn luyện trên bộ dữ liệu
- Chia dữ liệu train test và sử dụng hàm `linear_regression()` để xây dựng mô hình hồi quy tuyến tính

```
[ ] X_train = df.columns
    X_train.remove('status_type')

    y_train = ['status_type']

[ ] data_train = df.limit(int(df.count() * 0.8))
    data_test = df.subtract(data_train)
    coefficient = linear_regression(\
                                data_train\
                                ,X_train\
                                ,y_train)
```

Hình 44. Chia train test và ứng dụng mô hình hồi quy tuyến tính

CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC

4.1 Kết quả chạy thuật toán

- Định nghĩa hàm predict

```
[ ] def predict(data_test, coefficient):
    y=coefficient[0].item()
    for i in range(len(data_test.columns)-1):
        y = y + data_test[i+1]*coefficient[i+1].item()
    return y
```

Hình 45. Định nghĩa hàm predict

- Dự đoán kết quả chạy thuật toán trên tập test và hiển thị kết quả

```
[ ] df_predict = data_test.withColumn("type_predict",predict(data_test,coefficient))
df_predict.show(30)
```

num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys	Hour	status_type	type_predict
305	1415	418	186	117	1	1	0	0	7	2.0	-43.52376432905294
241	938	282	134	102	0	5	0	0	7	2.0	-26.836906689095485
56	4	0	51	0	0	5	0	0	7	3.0	2.995450316564529
125	4	0	125	0	0	0	0	0	7	3.0	2.7765260491432855
224	2792	113	191	21	6	4	1	1	11	2.0	-95.67140223258684
78	36	0	73	0	0	1	0	4	22	3.0	2.5096177871600958
394	2195	403	269	122	1	2	0	0	1	2.0	-71.5093588890195
35	3	0	34	1	0	0	0	0	6	3.0	2.823142660112353
63	5	0	58	0	0	2	2	1	23	1.0	3.668319278775016
164	53	0	152	2	10	0	0	0	6	3.0	1.3723463700152934
34	8	0	32	2	0	0	0	0	1	1.0	2.454931895597838
196	2	3	195	1	0	0	0	0	5	3.0	2.7789757982500305
300	1532	281	212	82	4	0	2	0	1	2.0	-49.08624768844675
1918	1	10	1917	0	1	0	0	0	11	3.0	2.7036426884515095
180	2176	314	142	33	4	0	0	1	10	2.0	-73.50752428278065
109	25	5	107	1	1	0	0	0	9	3.0	2.186363904420432
90	4	0	90	0	0	0	0	0	6	1.0	2.7374997473380063
105	56	0	97	3	3	2	0	0	15	3.0	1.5652615709259408
117	5	0	111	1	0	0	5	0	11	3.0	3.15851049804769
120	2773	89	106	10	1	2	1	0	10	2.0	-95.69271595155806
15	1	0	14	1	0	0	0	0	8	3.0	2.988406520074932
11	2	1	10	1	0	0	0	0	0	3.0	2.596196350586979
154	43	6	147	4	3	0	0	0	8	3.0	1.6612739295887877
27	3	3	27	0	0	0	0	0	1	3.0	2.56563704958171
5	24	1	5	0	0	0	0	0	11	2.0	2.2644219022949983
57	366	13	49	5	0	3	0	0	9	2.0	-9.776757901967358
132	9	1	132	0	0	0	0	0	3	3.0	2.4173601330003294
42	3	0	39	0	0	0	0	3	1	3.0	2.680730266190949
70	0	0	62	0	0	0	8	0	11	1.0	3.4308507719251793
36	0	0	36	0	0	0	0	0	12	3.0	3.1624079136527143

only showing top 30 rows

Hình 46. Dự đoán trên tập test

- Chuyển đổi kết quả dự đoán về cùng kiểu dữ liệu với cột “status_type” để tiến hành đánh giá

```
[ ] predict_values = df_predict.select('type_predict').rdd.flatMap(lambda x: x).collect()

[ ] print(predict_values)

[-43.52376432905294, -26.836906689095485, 2.995450316564529, 2.7765260491432855, -95.67140223258684, 2.5096177871600958, -71.5093588890195, 2.823142660112353, 3.668319278775016, 1.37234637800152934,

[ ] val_count = df.select("status_type").distinct().count()
alter_predict = []
for i in range(len(predict_values)):
    val = abs(float(round(predict_values[i] % val_count)))
    alter_predict.append(val)
print(alter_predict)

[0.0, 1.0, 3.0, 3.0, 0.0, 3.0, 0.0, 3.0, 4.0, 1.0, 2.0, 3.0, 3.0, 3.0, 2.0, 2.0, 3.0, 2.0, 3.0, 0.0, 3.0, 3.0, 2.0, 3.0, 2.0, 2.0, 2.0, 3.0, 3.0, 3.0, 3.0, 4.0, 2.0, 3.0, 2.0, 3.0, 3.0, 3.0, 1.0, 2.0]
```

Hình 47. Đổi kiểu dữ liệu

- Tạo cột “type_predict” chứa các giá trị kết quả

```
predicts_udf = func.udf( \
    lambda indx: alter_predict[indx-1])
df_predict = df_predict.withColumn( \
    "num_id", row_number().over( \
        Window.orderBy(monotonically_increasing_id())))

# Create a new column by calling the user defined function
df_predict = df_predict.withColumn('type_predict', \
    predicts_udf('num_id'))
df_predict = df_predict.drop('num_id')
df_predict.show()
```

num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys	Hour	status_type	type_predict
305	1415	418	186	117	1	1	0	0	7	2.0	0.0
241	938	282	134	102	0	5	0	0	7	2.0	1.0
56	4	0	51	0	0	5	0	0	7	3.0	3.0
125	4	0	125	0	0	0	0	0	7	3.0	3.0
224	2792	113	191	21	6	4	1	1	11	2.0	0.0
78	36	0	73	0	0	1	0	4	22	3.0	3.0
394	2195	403	269	122	1	2	0	0	1	2.0	0.0
35	3	0	34	1	0	0	0	0	6	3.0	3.0
63	5	0	58	0	0	2	2	1	23	1.0	4.0
164	53	0	152	2	10	0	0	0	6	3.0	1.0
34	8	0	32	2	0	0	0	0	1	1.0	2.0
196	2	3	195	1	0	0	0	0	5	3.0	3.0
300	1532	281	212	82	4	0	2	0	1	2.0	3.0
1918	1	10	1917	0	1	0	0	0	11	3.0	3.0
180	2176	314	142	33	4	0	0	1	10	2.0	2.0
109	25	5	107	1	1	0	0	0	9	3.0	2.0
90	4	0	90	0	0	0	0	0	6	1.0	3.0
105	56	0	97	3	3	2	0	0	15	3.0	2.0
117	5	0	111	1	0	0	5	0	11	3.0	3.0
120	2773	89	106	10	1	2	1	0	10	2.0	0.0

only showing top 20 rows

Hình 48. Tạo cột mới chứa kết quả dự đoán

4.2 Đánh giá

- Định nghĩa các độ đo

```
[ ] def R_square(df_predict):
    avg_type = df_predict.select('status_type').summary('mean').collect()[0][1]
    df_predict_avg= df_predict.withColumn("type_avg_log", (col("status_type")-avg_type)*(col("status_type")-avg_type))
    avg_type_avg_log=df_predict_avg.select('type_avg_log').summary('mean').collect()[0][1]
    sst=float(avg_type_avg_log)*float(df_predict_avg.select('type_avg_log').count())

    df_predict_avg_type= df_predict.withColumn("type_predict_avg_log", (col("status_type")-col("type_predict"))*(col("status_type")-col("type_predict")))
    avg_type_predict_avg_log = df_predict_avg_type.select('type_predict_avg_log').summary('mean').collect()[0][1]
    sse = float(avg_type_predict_avg_log)*float(df_predict_avg_type.select('type_predict_avg_log').count())

    R_square=1-(sse/sst)

    return R_square
```

Hình 49. Hàm khởi tạo R Square

```
[ ] def RMSD(df_predict, n):

    df_predict_avg_type= df_predict.withColumn("type_predict_avg_log", (col("status_type")-col("type_predict"))*(col("status_type")-col("type_predict")))
    avg_type_predict_avg_log = df_predict_avg_type.select('type_predict_avg_log').summary('mean').collect()[0][1]
    sse = float(avg_type_predict_avg_log)*float(df_predict_avg_type.select('type_predict_avg_log').count())

    return math.sqrt(sse/n)
```

Hình 50. Hàm khởi tạo RMSD

```
[ ] def cal_accuracy(df,actual, predict):
    actual_values = df.select(actual).rdd.flatMap(lambda x: x).collect()
    predict_values = df.select(predict).rdd.flatMap(lambda x: x).collect()
    val = 0
    for i in range(len(actual_values)):
        if str(actual_values[i]) == str(predict_values[i]):
            val= val + 1
    return val/len(actual_values)*100
```

Hình 51. Khởi tạo hàm cal_accuracy

- Hiển thị hai cột “status_type” và “type_predict” để tiến hành so sánh kết quả

```
[ ] df_predict.select('status_type','type_predict').show()
```

status_type	type_predict
2.0	0.0
2.0	1.0
3.0	3.0
3.0	3.0
2.0	0.0
3.0	3.0
2.0	0.0
3.0	3.0
1.0	4.0
3.0	1.0
1.0	2.0
3.0	3.0
2.0	3.0
3.0	3.0
2.0	2.0
3.0	2.0
1.0	3.0
3.0	2.0
3.0	3.0
2.0	0.0

only showing top 20 rows

Hình 52. Dữ liệu gốc và dữ liệu dự đoán

- Tính giá trị các độ đo

```
[ ] print("R_square: "+ str(R_square(df_predict)))
    print("RMSD: "+str(RMSD(df_predict,data_test.count())))
```

```
R_square: -1.7610793131643003
RMSD: 1.179918553209412
```

Hình 53. Tính R Square và RMSD

- Hệ số xác định (R-squared) có giá trị âm là -1.7610793131643003. Điều này cho thấy mô hình không phù hợp với dữ liệu, nghĩa là mô hình không giải thích được sự biến thiên của dữ liệu thực tế.
- Căn Trung Bình Bình Phương Sai số (RMSD) có giá trị là 1.179918553209412. Với giá trị RMSD này, có thể nói rằng mô hình dự đoán chưa đạt được độ chính xác mong muốn.
- Tính giá trị accuracy

```
[ ] print(cal_accuracy(df_predict,'status_type','type_predict'))
```

```
42.857142857142854
```

Hình 54. Tính accuracy

CHƯƠNG 5: KẾT LUẬN

5.1 Ưu điểm và hạn chế

5.1.1 Ưu điểm

Có khả năng nhận diện, phát biểu bài toán khai thác dữ liệu lớn và áp dụng được các giải thuật khai thác dữ liệu Linear Regression vào tập dữ liệu lớn.

Dựa trên dữ liệu phân tích để có cái nhìn sâu sắc hơn về tập dữ liệu, rút ra các nhận xét về xu hướng bán hàng qua mạng xã hội, cụ thể là Facebook Live, một kênh bán hàng qua mạng lớn.

Đề tài phân tích dữ liệu bán hàng có tính thực tiễn cao, có thể áp dụng trực tiếp vào chiến lược kinh doanh của nhà cung cấp nhỏ lẻ, giúp nâng cao hiệu suất, doanh thu bán hàng.

5.1.2 Hạn chế

Nhóm còn nhiều thiếu sót về kiến thức liên quan đến khai thác phân tích dữ liệu lớn.

Dữ liệu về Facebook Live Sellers có thể có nhiều tính chất phức tạp, như mối quan hệ không tuyến tính, biến ẩn, dữ liệu bị thiếu hoặc lỗi

Linear Regression có thể không đủ mạnh để mô hình hóa các mối quan hệ phức tạp trong dữ liệu Facebook Live Sellers.

5.2 Hướng phát triển

Trao đổi thêm kiến thức và kinh nghiệm trong lĩnh vực dữ liệu lớn, tối ưu hóa các thuật toán và kết hợp các phương pháp, giải thuật khác để tăng độ tin cậy

Tìm kiếm bộ dữ liệu phù hợp hơn với mục tiêu của nhóm để thực hiện khai thác tốt hơn.

PHÂN CÔNG CÔNG VIỆC

Công việc \ Tên	<u>Phong</u>	Hải	Nhân	Đăng
Chọn đề tài, dataset	X	X	X	X
Mô tả bài toán	X	X	X	X
Mô tả dataset			X	X
Cơ sở lý thuyết về tiền xử lý		X		
Tìm hiểu thuật toán áp dụng		X	X	
Tìm hiểu phương pháp đánh giá		X		
Tiền xử lý dữ liệu	X	X	X	X
Triển khai mô hình	X			
Trực quan hóa kết quả			X	
Đánh giá kết quả		X	X	X
Kết luận				X
Chỉnh sửa báo cáo		X		X
Viết và chỉnh sửa slide	X	X	X	X
Demo	X			
Thuyết trình	X	X	X	X
Đánh giá	100%	100%	100%	100%

Bảng 2. Phân công công việc

TÀI LIỆU THAM KHẢO

- [1] A. GAURAV, "Facebook Live sellers in Thailand, UCI ML Repo," 2018. [Online]. Available: <https://tinyurl.com/2ndxydd5>. [Accessed 2024].
- [2] scikit-learn, "LinearRegression," 2024. [Online]. Available: <https://tinyurl.com/2tubjzsz>. [Accessed 2024].

Link:

https://vi.wikipedia.org/wiki/H%E1%BB%93i_quy_tuy%E1%BA%BFn_t%C3%ADnh

<https://viblo.asia/p/giai-thich-va-ung-dung-cua-pysparksqlwindow-trong-xu-ly-du-lieu-phan-tan-gwd43M73LX9>